# DIFFERENCE SETS AND THE SYMMETRIC DIFFERENCE PROPERTY

---

A Thesis

Presented to

The Faculty of the Department of Mathematics and Statistics

Sam Houston State University

---

In Partial Fulfillment

of the Requirements for the Degree of

Master of Science

---

by

Matthew W. Williams

May, 2022

# DIFFERENCE SETS AND THE SYMMETRIC DIFFERENCE PROPERTY

by

Matthew W. Williams

---

## DEDICATION

This thesis/dissertation is dedicated to the Department of Mathematics and Statistics, which is resided in the College of Science and Engineering Technology at Sam Houston State University.

## ABSTRACT

Williams, Matthew W., *Difference sets and the symmetric difference property*. Master of Science (Mathematics), May 2022, Sam Houston State University, Huntsville, Texas.

With the intent to discover block designs from groups of order 256 and higher that have the symmetric difference property, we analyze these block designs through products of groups that give smaller block designs with the symmetric difference property (SDP). This research expands upon the knowledge of SDP designs by looking at designs that come from groups of extremely high orders, which we analyze by looking at the difference sets of these groups. This will give way to new SDP designs that can be analyzed and studied in the near future.

KEY WORDS: Block designs, Groups, Difference sets, SDP.

## ACKNOWLEDGEMENTS

I would like to thank the members of my committee who have helped shape my thesis after all this time, especially Dr. Ken Smith, who has been a major influence to the construction of the thesis and the contents that are within it. I could not have accomplished all of this work without him. Also, I would like to thank Dr. Edward Swim who has helped provide me the information I needed on how to create a thesis in the SHSU style. Finally, I would like to thank my friends and family, who have supported and enabled me to do my best at everything that I do, and helped me in this part of my journey by being there for me when I needed them the most.

# PREFACE

This paper is a study of block designs and difference sets. Specifically, it looks into whether or not these block designs and difference sets possess the Symmetric Difference Property (SDP). In order to study these properties and topics, several other mathematical topics must be introduced, including blocks, incidence matrices, group rings, hyperplanes, Hadamard matrices and various other minor topics. We define difference sets, block designs, and codes in order to study SDP designs that comes from groups of order $2^{2m}$.

# TABLE OF CONTENTS

# CHAPTER I

# Introduction

## SDP for difference sets

Designs and difference sets, and their relation to the Symmetric Difference Property (or the SDP) is the focus of this research. Therefore, we will introduce topics and ideas needed to understand difference sets and the SDP, including blocks, group rings, and hyperplanes. **Designs** are ordered pairs that consists of a set (the **points** of a design) and a collection of subsets of that set (where each subset is called a **block**) that follows certain parameters. Furthermore, we say that a design has the Symmetric Difference Property, if the symmetric difference of three blocks is a block or the complement of a block. A **group ring** is a set of formal sums of products of an element in a commutative ring with identity and an element in a finite group. Finally, a **hyperplane** is a subspace of a vector space $V$ of dimension one less than the dimension of $V$. All of these mathematical topics will be explored since they give way to the climax of the thesis: the construction of 6 non-isomorphic SDP designs, which were formed by looking closely at a specific type of group (semi-direct products of two groups with a specific order).

## Literature review

This section goes over the sources used in the bibliography.

*A course in combinatorics*, by J.H. van Lint and R.M. Wilson [16], provides basic information about designs, group rings, and difference sets. The text is also helpful by stating the definition of codes and the many applications of codes in a clear and understandable way.

*Combinatorial theory*, by Marshall Hall Jr. [10], introduces difference sets and block designs.

*Abstract Algebra*, by Dummit and Foote [8], is a professional, well-written text that can be used to instruct higher-level algebra classes in well-recognized campuses. It gives definitions and provides basic information about group rings, the mathematical topic that is used in this thesis to expand on difference sets and the SDP.

*Symplectic groups, symmetric designs, and line ovals*, by W. M. Kantor, is a mathematical article in the Journal of Algebra [12] that talks about symmetric designs, and also introduces the Symmetric Difference Property (SDP).

*Applied Linear Algebra*, by P. J. Olver and Shakiban [14], is a college-level linear algebra text. It provides a definition for vector spaces, which is ultimately used to set up hyperplanes and the hyperplane construction of difference sets.

*Exponential numbers of two-weight codes, difference sets and symmetric designs*, by W. M. Kantor [13], illustrates the lower bounds on the parameters of the $(v,k,\lambda)$ symmetric designs and the codes that are derived from them. It accomplishes this by looking at a vector space $V$ over the field $\{0,1\}$ and a group and then derives certain parameters early on and then uses those parameters to define the rank of codes and incidence matrices.

This research follows the ideas described in the following three papers: *Designs with the symmetric difference property on 64 points and their groups*, by C. Parker, E. Spence, and V. D. Tonchev [15], *On symmetric and quasi-symmetric designs with the symmetric difference property and their codes*, by D. Jungnickel and V. D. Tonchev [11], and *Abelian difference sets with the symmetric difference property*, by J. A. Davis, J. J. Hoo, C. Kissane,

Z. Liu, C. Reedy, K. Sharma, K. Smith, and Y. Sun [5]. *Designs with the symmetric difference property on 64 points and their groups* is an article that looks over symmetric and non-symmetric $(v, k, \lambda)$ designs, and puts some emphasis on the designs/groups of size 64. It also looks at the binary codes and minimum weights of these designs/groups. *On symmetric and quasi-symmetric designs with the symmetric difference property and their codes* gives parameters needed for a design to have the SDP and gives theorems that help derive the SDP designs of order 256 or higher. *Abelian difference sets with the symmetric difference property* gives various theorems and proofs that are connected to difference sets and designs with the Symmetric Difference Property.

Finally, the free software, *GAP-Groups, Algorithms, and Programming, Version 4.11.1* [9], helped find SDP difference sets and SDP designs very quickly and efficiently.

# CHAPTER II

# Symmetric Block Designs

## $(v,k,\lambda)$ symmetric block designs

Block designs are an important field in discrete mathematics and statistics because it is used to create error correcting codes. Block designs have applications to finite geometry, and design of experiments in statistics.

**Definition**: Let $X$ be a finite set and $\mathscr{B}$ a family of subsets of $X$. Call the elements of $X$ points and the elements of $\mathscr{B}$ blocks. The pair $(X,\mathscr{B})$ is a block design, if each of the blocks has $k$ points, every point in $X$ appears in $r$ blocks, and any pair of two distinct points appear together in exactly $\lambda$ blocks.

A block design is often called a design. The cardinality of $X$ is denoted by $v$; the cardinality of $\mathscr{B}$ is denoted by $b$. The definition of a block design implies certain conditions on the parameters, $(v,b,r,k,\lambda)$.

**Lemma 1.** *If $(v,b,r,k,\lambda)$ are the parameters of a block design $(X,\mathscr{B})$, then $bk = vr$ and $r(k-1) = \lambda(v-1)$.*

*Proof.* Consider a set of ordered pairs $\{(x,B) : x \in B\} \subseteq X \times \mathscr{B}$. There are $v$ choices for the point $x$, and each point is contained in $r$ blocks. There are $vr$ elements in this set. Since there are $b$ choices for $B$ and every block has $k$ elements, this set has $bk$ elements. Thus, $vr = bk$.

Consider a fixed point $y \in X$ and the set $\{(x,B) : x \neq y, x,y \in B\} \subseteq X \times \mathscr{B}$. There are $v-1$ distinct other points to choose for $x$. Since $x,y$ are distinct points, they appear $\lambda$ times in $B$. Thus, there are $\lambda(v-1)$ elements in this set. Furthermore, $y$ appears in $r$ blocks, and in each of those blocks, it can be paired with $k-1$ points. Thus, there are $r(k-1)$ elements in this set, which implies that $r(k-1) = \lambda(v-1)$. $\square$

A block design is *symmetric* if $v = b$. If a design is symmetric, then $k = r$. These designs are called $(v,k,\lambda)$ designs.

**Example 1**: Consider the sets $X$ and $\mathscr{B}$, where $X = \{0,1,2,3,4,5,6\}$ and $\mathscr{B} = \{B_0,B_1,B_2,B_3,B_4,B_5,B_6\}$, where the blocks $B_i$ are listed below:

$$B_0 = \{0,1,3\}$$

$$B_1 = \{1,2,4\}$$

$$B_2 = \{2,3,5\}$$

$$B_3 = \{3,4,6\}$$

$$B_4 = \{4,5,0\}$$

$$B_5 = \{5,6,1\}$$

$$B_6 = \{6,0,2\}$$

This is an example of a symmetric design $(X,\mathscr{B})$ with parameters $(7,7,3,3,1)$ or parameters $(7,3,1)$.

**Example 2**: Consider the set $Y = \{1,3,4,5,6,7,9,10,11\}$ and the set $\mathscr{B} = \{B_0,B_1,B_2,...,B_{11}\}$, where the elements of $\mathscr{B}$ are shown below:

$$B_0 : \{1,3,9\}$$

$$B_1 : \{1,4,10\}$$

$$B_2 : \{3,5,11\}$$

$$B_3 : \{3,4,6\}$$

$$B_4 : \{4,5,7\}$$

$$B_5 : \{5,6,1\}$$

$$B_6 : \{6,7,9\}$$

$$B_7 : \{7,10,3\}$$

$$B_8 : \{9,11,4\}$$

$$B_9 : \{9,10,5\}$$

$$B_{10} : \{10,11,6\}$$

$$B_{11} : \{11,1,7\}$$

This is an example of a design $(Y, \mathscr{B})$ with parameters $(9,12,4,3,1)$.

**Definition**: Two block designs, $(X, \mathscr{B})$ and $(Y, \mathscr{A})$, are **isomorphic** if there exists a one-to-one and onto mapping $\alpha : X \mapsto Y$ of points, such that for every block $B$ in $\mathscr{B}$, there is a block $A$ in $\mathscr{A}$ where $\alpha(B) = A$.

In Example 1, the mapping $\alpha : i \mapsto i+1$ sends $B_i$ to $B_{i+1}$. This is an automorphism of the design $(X, \mathscr{B})$ with parameters $(7,7,3,3,1)$.

**Example 3**: Consider the sets $X = \mathbb{Z}_{21}$ and $Y = \mathbb{Z}_7 \times \mathbb{Z}_3$. Consider the two sets $\mathscr{B} = \{B_0, B_1, ..., B_{20}\}$ and $\mathscr{A} = \{A_0, A_1, ..., A_{20}\}$, where the elements of both sets are shown below.

$$B_0 : \{0,3,4,9,11\} \quad A_0 : \{(0,0),(3,0),(4,1),(2,0),(4,2)\}$$

$$B_1 : \{1,4,5,10,12\} \quad A_1 : \{(1,1),(4,1),(5,2),(3,1),(5,0)\}$$

$$B_2 : \{2,5,6,11,13\} \quad A_2 : \{(2,2),(5,2),(6,0),(4,2),(6,1)\}$$

$$B_3 : \{3,6,7,12,14\} \quad A_3 : \{(3,0),(6,0),(0,1),(5,0),(0,2)\}$$

$$B_4 : \{4,7,8,13,15\} \quad A_4 : \{(4,1),(0,1),(1,2),(6,1),(1,0)\}$$

$$B_5 : \{5,8,9,14,16\} \quad A_5 : \{(5,2),(1,2),(2,0),(0,2),(2,1)\}$$

$$B_6 : \{6,9,10,15,17\} \quad A_6 : \{(6,0),(2,0),(3,1),(1,0),(3,2)\}$$

$$B_7 : \{7,10,11,16,18\} \quad A_7 : \{(0,1),(3,1),(4,2),(2,1),(4,0)\}$$

$$B_8 : \{8,11,12,17,19\} \quad A_8 : \{(1,2),(4,2),(5,0),(3,2),(5,1)\}$$

$$B_9 : \{9,12,13,18,20\} \quad A_9 : \{(2,0),(5,0),(6,1),(4,0),(6,2)\}$$

$$B_{10} : \{0,10,13,14,19\} \quad A_{10} : \{(0,0),(3,1),(6,1),(0,2),(5,1)\}$$

$$B_{11} : \{1,11,14,15,20\} \quad A_{11} : \{(1,1),(4,2),(0,2),(1,0),(6,2)\}$$

$$B_{12} : \{0,2,12,15,16\} \quad A_{12} : \{(0,0),(2,2),(5,0),(1,0),(2,1)\}$$

$$B_{13} : \{1,3,13,16,17\} \quad A_{13} : \{(1,1),(3,0),(6,1),(2,1),(3,2)\}$$

$$B_{14} : \{2,4,14,17,18\} \quad A_{14} : \{(2,2),(4,1),(0,2),(3,2),(4,0)\}$$

$$B_{15} : \{3,5,15,18,19\} \quad A_{15} : \{(3,0),(5,2),(1,0),(4,0),(5,1)\}$$

$$B_{16} : \{4,6,16,19,20\} \quad A_{16} : \{(4,1),(6,0),(2,1),(5,1),(6,2)\}$$

$$B_{17} : \{0,5,7,17,20\} \quad A_{17} : \{(0,0),(5,2),(0,1),(3,2),(6,2)\}$$

$$B_{18} : \{0,1,6,8,18\} \quad A_{18} : \{(0,0),(1,1),(6,0),(1,2),(4,0)\}$$

$$B_{19} : \{1,2,7,9,19\} \quad A_{19} : \{(1,1),(2,2),(0,1),(2,0),(5,1)\}$$

$$B_{20} : \{2,3,8,10,20\} \quad A_{20} : \{(2,2),(3,0),(1,2),(3,1),(6,2)\}$$

There exists an isomorphism between these two sets. This isomorphism is the mapping $x \mapsto (a,b)$, where $a$ is $x$ modulo 7 and $b$ is $x$ modulo 3. Then, each of the blocks of $\mathscr{B}$ will be mapped to a block of $\mathscr{A}$. Thus, $(X,\mathscr{B})$ and $(Y,\mathscr{A})$ are isomorphic designs.

**Bruck-Ryser Chowla Theorem**

In addition to Lemma 1, there is one more known condition on the parameters of a symmetric design, which is known as the Bruck-Ryser Chowla Theorem.

**Theorem 2.** *(Bruck-Ryser Chowla Theorem): Suppose the design $(X,\mathscr{B})$ is a $(v,k,\lambda)$ symmetric design. Then:*

*(i) if $v$ is even, then $k-\lambda$ is a square.*

*(ii) if $v$ is odd, then the equation $z^2 = (k-\lambda)x^2 + (-1)^{\frac{v-1}{2}}\lambda y^2$ has a solution in integers $x, y, z$, where not all of them are zero.*

A proof can be found on [16], pg. 230.

**Example 4(a):** Assume that $v = 22, k = 7, \lambda = 2$. Then, $7(6) = 2(22-1) = 42$. However, even through $v$ is even, $k - \lambda = 7 - 2 = 5$ is not a square since the square root of 5 is an irrational number. Thus, by the Bruck-Ryser Chowla Theorem, there does not exist a $(22,7,2)$ symmetric design.

**Example 4(b):** Assume that $v = 43, k = 7, \lambda = 1$. Then, $7(6) = 1(43-1)$. Then, the equation $z^2 = 6x^2 - y^2$ could determine if a $(43,7,1)$ symmetric design does not exist. This equation can be rewritten as $z^2 + y^2 = 6x^2$. If there is a solution to this equation, assume that $x, y, z$ are non-negative integers, and not all zero. By the Law of Well Ordering, the solution can be chosen so that $x$ has the smallest possible value.

Furthermore, $z^2 + y^2 = 6x^2 \equiv 0 \pmod 3$. Therefore, $z \equiv y \equiv 0 \pmod 3$. Therefore, $3 \mid z$ and $3 \mid y$. Then, $z = 3a$ and $y = 3b$, where $a, b \in \mathbb{Z}$. Thus, $(3a)^2 + (3b)^2 = 6x^2$ implies that $3a^2 + 3b^2 = 2x^2$. Then, $2x^2 \equiv 0 \pmod 3$. Therefore, $3 \mid x^2$, which implies that $3 \mid x$. Thus, $x = 3c$ for some $c \in \mathbb{Z}$. Then, we have that $3a^2 + 3b^2 = 18c^2$, which means that $a^2 + b^2 = 6c^2$. However, $c < x$, which is a contradiction. Thus, $z^2 + y^2 = 6x^2$ has no integer solutions. By the Bruck-Ryser Chowla Theorem, there does not exist a $(43,7,1)$ symmetric design.

**Example 4(c):** Assume that $v = 111, k = 11, \lambda = 1$. Then, $11(10) = 111 - 1$. Then, by looking at the equation $z^2 = 10x^2 - y^2$, it can be concluded that the ordered triple $(1,1,3)$ is a solution to the equation. Thus, we cannot conclude from this whether or not there is a $(111,11,1)$ symmetric design with these parameters.

**Examples of symmetric block designs**

**Example 5:** Consider the set $X = \{0, 1, 2, 3, ..., 10\}$, and the set $B = \{B_0, B_1, B_2, ..., B_{10}\}$, where the elements of $B$ are shown below:

$$B_0 : \{0, 2, 3, 4, 8\}$$

$$B_1 : \{1, 3, 4, 5, 9\}$$

$$B_2 : \{2, 4, 5, 6, 10\}$$

$$B_3 : \{3, 5, 6, 7, 0\}$$

$$B_4 : \{4, 6, 7, 8, 1\}$$

$$B_5 : \{5, 7, 8, 9, 2\}$$

$$B_6 : \{6, 8, 9, 10, 3\}$$

$$B_7 : \{7, 9, 10, 0, 4\}$$

$$B_8 : \{8, 10, 0, 1, 5\}$$

$$B_9 : \{9, 0, 1, 2, 6\}$$

$$B_{10} : \{10, 1, 2, 3, 7\}$$

This is a $(11, 5, 2)$ symmetric design since the number of elements in $X$ and number of blocks in $B$ is 11, every block has 5 points and every point in $X$ appears in 5 blocks, and any pair of two distinct points appear together in 2 blocks.

**Example 6:** Consider the set $X = \{0, 1, 2, 3, ..., 12\}$ and the set $B = \{B_0, B_1, B_2, ..., B_{12}\}$ where the elements of $B$ are shown below:

$$B_0 : \{0, 1, 3, 9\}$$

$$B_1 : \{1, 2, 4, 10\}$$

$$B_2 : \{2, 3, 5, 11\}$$

$$B_3 : \{3,4,6,12\}$$

$$B_4 : \{4,5,7,0\}$$

$$B_5 : \{5,6,8,1\}$$

$$B_6 : \{6,7,9,2\}$$

$$B_7 : \{7,8,10,3\}$$

$$B_8 : \{8,9,11,4\}$$

$$B_9 : \{9,10,12,5\}$$

$$B_{10} : \{10,11,0,6\}$$

$$B_{11} : \{11,12,1,7\}$$

$$B_{12} : \{12,0,2,8\}$$

This is an example of a design with parameters $(13,4,1)$ since the number of elements in $X$ and number of blocks in $B$ is 13, every block has 4 points and every point in $X$ appears in 4 blocks, and any pair of two distinct points appear together in 1 block.

**Incidence matrices for symmetric designs**

The definition of an incidence matrix will be illustrated below:

**Definition:** For a $(v,b,r,k,\lambda)$ symmetric design, an incidence matrix, $M$, is a $b \times v$ $(0,1)-$matrix, where $m_{ij} = 1$ if the point $x_j \in B_i$ and $m_{ij} = 0$ if $x_j$ is not in $B_i$, where $0 \leq i \leq b$ and $0 \leq j \leq v$.

**Example 7(a):** Refer to Example 1. The set $X = \{0,1,2,3,4,5,6\}$ and blocks $B_0, B_1, B_2, B_3, B_4, B_5, B_6$ can be used to create an incidence matrix, where each column represents an element of $X$ (numbered numerically from left to right) and the rows represent the blocks (where the $n^{th}$ row represents the $B_{n-1}$ block). This gives the table below:

| $X$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| $B_0$ | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| $B_1$ | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| $B_2$ | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| $B_3$ | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| $B_4$ | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| $B_5$ | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| $B_6$ | 1 | 1 | 0 | 1 | 0 | 0 | 0 |

As a result, the $7 \times 7$ incidence is:

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

This $7 \times 7$ incidence matrix is formed by having the ones in the matrix appear under the numbers that appears in a block. The first row represents $B_0$, and 1's were placed under the second, third and fifth columns (which represents $1,2$ & $4$ respectively). The second row represents $B_1$, and 1's were placed under the third, fourth and sixth column (which represents $2,3$ & $5$ respectively). The remaining rows are formed in a similar manner.

**Example 7(b):** Refer to Example 5. Then, $X = \{0,1,2,...,10\}$ and the blocks $B_0, B_1, B_2, ..., B_{10}$

can be used to create an incidence matrix, similar to how it was done in Example 7(a). This results in the $11 \times 11$ incidence matrix shown below.

$$\begin{pmatrix}
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1
\end{pmatrix}$$

Now, consider the following results that will be useful later on.

**Lemma 3.** *Let $M$ be a $v \times v$ incidence matrix of a symmetric $(v,k,\lambda)$ design. Then,*

*1). $MJ = kJ$ and $M^tM = (k-\lambda)I + \lambda J$, where $I$ is the identity matrix and $J$ is the $v \times v$ matrix where all of the entries are 1.*

*2) $M$ is invertible.*

*3) $M^tM = MM^t$*

*Proof.* Let $M$ be a $v \times v$ incidence matrix of a symmetric $(v,k,\lambda)$ design.

1) We want to show that $MJ = kJ$. Since $J$ is the $v \times v$ matrix where all of the entries are 1, each entry of $MJ$ is $\sum_{k=1}^{v} a_{ik}$, where $i = 1,2,\ldots,v$. Since $M$ is an incidence matrix

and each row has $k$ entries equal to 1, then $\sum_{k=1}^{v} a_{ik} = k$ for all entries of $MJ$, which is the matrix $J$ multiplied by the scalar $k$.

Thus, $MJ = kJ$.

Furthermore, each entry of $M^tM$ is $\sum_{k=1}^{v} b_{ik}c_{kj}$, where $b_{ik}$ is the $(i,k)-$entry of $M^t$ and $c_{kj}$ is the $(k,j)-$entry of $M$, where $i = 1,...,v$ and $j = 1,...,v$. If $i = j$ for all $k$, then that entry of $M^tM$ is $\sum_{k=1}^{v} b_{ik}c_{kj} = k$. If $i \neq j$ for all $k$, then that entry of $M^tM$ is $\sum_{k=1}^{v} b_{ik}c_{kj} = \lambda$. Therefore, this will result in all of the diagonal entries of $M^tM$ to be $k$ and all other entries of $M^tM$ to be $\lambda$. Therefore, $M^tM$ can be rewritten as $(k-\lambda)I + \lambda J$, since $(k-\lambda)I$ is the matrix with $k-\lambda$ in all of its diagonal entries and zeroes everywhere else, and $\lambda J$ is the matrix where all of the entries are $\lambda$. Thus, $M^tM = (k-\lambda)I + \lambda J$.

2) Consider the matrix

$$B = \frac{1}{k-\lambda}M^t - \frac{\lambda}{k(k-\lambda)}J.$$

Then,

$$BM = \left(\frac{1}{k-\lambda}M^t - \frac{\lambda}{k(k-\lambda)}J\right)M = \frac{1}{k-\lambda}M^tM - \frac{\lambda}{k(k-\lambda)}JM$$

$$= \frac{1}{k-\lambda}[(k-\lambda)I + \lambda J] - \frac{\lambda}{k(k-\lambda)}kJ = I.$$

Similarly, $MB = I$ by performing very similar computations as above. Thus, $BM = MB = I$. Thus, $M$ is invertible and $M^{-1} = B = \dfrac{1}{k-\lambda}M^t - \dfrac{\lambda}{k(k-\lambda)}J$.

3) $M^tM = (k-\lambda)I + \lambda J$. Then,

$$MM^tMM^{-1} = M[(k-\lambda)I + \lambda J]M^{-1}.$$

Then,

$$MM^tMM^{-1} = M[(k-\lambda)I]M^{-1} + M[\lambda J]M^{-1}.$$

Therefore, since $JM^{-1} = \frac{1}{k}J$, then

$$MM^tMM^{-1} = MM^t = (k-\lambda)I + \lambda J = M^tM.$$

□

**Lemma 4.** *If $M_1, M_2$ are the incidence matrices of isomorphic $(v,k,\lambda)$ designs, then there are $v \times v$ permutation matrices $P$ and $Q$ such that*

$$M_2 = PM_1Q$$

This lemma can be found in [4] on pg. 786.

**Corollary 5.** *Any pair of distinct blocks in a $(v,k,\lambda)$ design intersect in $\lambda$ points.*

*Proof.* Assume there is a $(v,k,\lambda)$ design with an incidence matrix $M$. Let $B_1, B_2$, be two distinct blocks in this design. Then, $B_1, B_2$ correspond to two distinct rows in the matrix. For $M^t$, these blocks correspond to two distinct columns. These columns will be called the $i^{th}$ column and the $j^{th}$ column, where $i \neq j$. Thus, the $(i,j)$-entry of $MM^t$ is $\lambda$. Since the rows of $M$ and the columns of $M^t$ correspond to the blocks, any pair of distinct block in in a $(v,k,\lambda)$ design intersect in $\lambda$ points.

□

# CHAPTER III

# Difference Sets

## Definition and examples

In Example 1, the blocks $B_i$, where $i = 0, 1, ...6$, have an interesting property. All of the blocks that are listed are sets of residues modulo 7 and the differences between any pair of two distinct elements from this set is also a residue modulo 7. These blocks are a special type of set called a difference set.

**Definition**: Let $G$ be a group of order $v$ written in multiplicative notation. If $D$ is a subset of $G$ of size $k$, then $D$ is a $(v, k, \lambda)$ difference set if, for all non-identity elements $g \in G$, there exists exactly $\lambda$ pairs of elements $d_1, d_2$ in $D$ such that $d_1 d_2^{-1} = g$.

It is important to note that there are several different types of difference sets. A difference set $D$ for some group $G$ is also called "cyclic", "abelian", or "non-abelian" if the group $G$ is cyclic, abelian, or non-abelian, respectively.

Furthermore, there are sets that can be derived from difference sets. These sets are called translates.

**Definition:** Let $D$ be a $(v, k, \lambda)$ difference set in $G$. The translates of $D$ are all the sets of the form $gD := \{gd \mid d \in D\}$, where $g$ varies throughout $G$.

The next two examples illustrate these definitions.

**Example 8(a):** Let $G = \mathbb{Z}_7$ be a group under addition, and $D = \{1, 2, 4\}$. This is a $(7, 3, 1)$ difference set because $|\mathbb{Z}_7| = 7$, there are 3 elements in the set and each nonzero element of the group appears 1 time in the subtraction table below:

| - | 1 | 2 | 4 |
|---|---|---|---|
| 1 | 0 | 6 | 4 |
| 2 | 1 | 0 | 5 |
| 4 | 3 | 2 | 0 |

**Example 8(b):** In Example 8, there was the $D = \{1,2,4\}$ difference set in the group $\mathbb{Z}_7$, which is a group under addition. Then, this set can be used to form its translates by taking an element of $\mathbb{Z}_7$, $j$, and adding it to each element of the set $D$ modulo 7. Thus,

$$1+D = \{1+1,2+1,4+1\} = \{2,3,5\}$$

$$2+D = \{1+2,2+2,4+2\} = \{3,4,6\}$$

$$3+D = \{1+3,2+3,4+3\} = \{4,5,0\}$$

$$4+D = \{1+4,2+4,4+4\} = \{5,6,1\}$$

$$5+D = \{1+5,2+5,4+5\} = \{6,0,2\}$$

$$6+D = \{1+6,2+6,4+6\} = \{0,1,2\}$$

$$0+D = D.$$

These are the 7 translates of $D$.

By looking at Example 8(b), we see that each translate of $D$ gives a block that was given in Example 1.

**Example 9:** Another example is the set $D = \{1,3,4,5,9\}$ contained in the additive cyclic group $\mathbb{Z}_{11}$. $D$ is a $(11,5,2)$ difference set. $D$ is a subset of $\mathbb{Z}_{11}$. There are exactly 5 elements in $D$. Every nonzero element of the cyclic group $\mathbb{Z}_{11}$ appears twice in the table below, implying $\lambda = 2$.

| - | 1 | 3 | 4 | 5 | 9 |
|---|---|---|---|---|---|
| 1 | 0 | 9 | 8 | 7 | 3 |
| 3 | 2 | 0 | 10 | 9 | 5 |
| 4 | 3 | 1 | 0 | 10 | 6 |
| 5 | 4 | 2 | 1 | 0 | 7 |
| 9 | 8 | 6 | 5 | 4 | 0 |

The design illustrated in Example 5 can be recreated by taking as points the elements of $\mathbb{Z}_{11}$ and as blocks the translates of $D$.

The observations shown in Example 8(b) and Example 9 are a result of the following lemma(s):

**Lemma 6.** *(a) If $D$ is a $(v,k,\lambda)$ difference set in a group $G$ of order $v$, then so is $gD$, where $g \in G$.*

*(b) If $D$ is a $(v,k,\lambda)$ difference set in a group $G$ of order $v$ and $\phi$ is an automorphism of $G$, then $\phi(D)$ is a difference set.*

*Proof.* (a) Assume $D$ is a $(v,k,\lambda)$ difference set in a group $G$ of order $v$. Then, consider the set $hD := \{hd \mid d \in D\}$, where $h \in G$ and $h \neq 1_G$, where $1_G$ is the identity of $G$. Then, $|hD| = k$ and all of the elements of $hD$ are distinct since all of the elements of $D$ are distinct and $|D| = k$. Consider the multiset $X_h = \{(d_i,d_j) \mid d_i,d_j \in D, d_id_j^{-1} = h\}$, where $|X_h| = \lambda$. Also, consider the multiset $Y_h = \{(d_i,d_j) \mid gd_i,gd_j \in gD, gd_id_j^{-1}g^{-1} = h\}$, where $g \in G$. If $(d_i,d_j) \in Y_h$, $gd_id_j^{-1}g^{-1} = h$ implies that $d_id_j^{-1} = g^{-1}hg$, where $d_i,d_j \in D$. Since $g,h \in G$ and $h \neq 1_G$, then $g^{-1}hg \in G$ and $g^{-1}hg \neq 1_G$. Thus, $Y_h = X_{g^{-1}hg}$. Then, $|Y_h| = |X_{g^{-1}hg}|$.

Furthermore, $g^{-1}hg = h_1$, where $h_1 \in G$ and $h_1 \neq 1_G$. Thus, $X_{g^{-1}hg} = X_{h_1}$. Then, $|Y_h| = |X_{g^{-1}hg}| = |X_{h_1}| = \lambda$. Therefore, $gD$ is a difference set.

(b) Assume $D$ is a $(v,k,\lambda)$ difference set in a group $G$ of order $v$ and $\phi$ is an automorphism of $G$. Since $D \subset G$ and $\phi$ is bijective, $\phi(D)$ will send each element of $D$ to an element of $G$. Then, each element of $\phi(D)$ is distinct and since $|D| = k$, $|\phi(D)| = k$. Consider the multiset $X_g = \{(d_i,d_j) \mid d_i,d_j \in D, d_i d_j^{-1} = g\}$, where $|X_g| = \lambda$. Also, consider the multiset $Y_g = \{(d_i,d_j) \mid \phi(d_i),\phi(d_j) \in \phi(D), \phi(d_i)\phi(d_j)^{-1} = g\}$. If $(d_i,d_j) \in Y_h$, $\phi(d_i)\phi(d_j)^{-1} = g$ implies that $\phi(d_i d_j^{-1}) = g$, where $d_i,d_j \in D$. Then, $d_i d_j^{-1} = \phi^{-1}(g)$. Since $g \in G$ and $g \neq 1_G$, then $\phi^{-1}(g) \in G$ and $\phi^{-1}(g) \neq 1_G$. Thus, $Y_g = X_{\phi^{-1}(g)}$. Furthermore, $\phi^{-1}(g) = h$, where $h \in G$ and $h \neq 1_G$. Thus, $X_{\phi^{-1}(g)} = X_h$. This implies that $|Y_g| = |X_{\phi(h)}| = |X_h| = \lambda$. Thus, $\phi(D)$ is a difference set.

□

**Lemma 7.** *The translates of a difference set $D$ in a group $G$ of order $v$ gives a $(v,k,\lambda)$ symmetric design on the point set $G$.*

*Proof.* Assume there is a difference set $D$ in a group $G$ of order $v$. Then, there are $v$ distinct translates for $D$. By Lemma 6, all translates are difference sets. Therefore, there are $k$ distinct elements in each of the translates. Let $g_1,g_2 \in G$ such that $g_1 \neq g_2$. Consider the set $g_1D \cap g_2D = \{g_1d_1 \mid d_1 \in D \text{ and } \exists d_2 \in D \text{ such that } g_1d_1 = g_2d_2\}$. Then, $g_1d_1 = g_2d_2$ if and only if $d_1d_2^{-1} = g_1^{-1}g_2 = g$, where $g \in G$ and $g$ is not the identity $1_G$. Since $g_1D$ and $g_2D$ are difference sets and $d_1,d_2 \in D$ such that $d_1d_2^{-1} = g$, $|g_1D \cap g_2D| = \lambda$. Furthermore, if we treat the group $G$ as a point set and the translates as blocks, then it gives a $(v,k,\lambda)$ symmetric design on the points set $G$ since there are $v$ blocks and points, each block has $k$ elements and by Corollary 5, each of these blocks intersect at $\lambda$ points. □

**Multiplier conjecture**

In special cases, difference sets can be constructed using the following conjecture.

**Conjecture 8.** *Suppose that there is an abelian group $G$ of order $v$ written additively. If $G$ has a $(v,k,\lambda)$ difference set with parameters $(v,k,\lambda)$ and $p$ is a prime dividing $k-\lambda$, but not dividing $v$, then there is a difference set $D$ such that $pD = D$.*

This conjecture essentially states that a difference set can be formed by applying a guess and check method to possible values of elements that are in $D$. To illustrate what this means, consider the following examples.

**Example 10:** Assume there is a $(v,k,\lambda)$ difference set with specific parameters that needs to be found, such as $(15,7,3)$. Since 7-3=4, then the only prime $p$ that divides 4 is $p=2$. Since 2 does not divide 7, Conjecture 8 can be applied. If $1 \in D$, then, by multiplying 1 by 2 repeatedly under modulo 15, it implies that $2,4,8$ are also in $D$.

If $5 \in D$, then $5,10$ are also in $D$. Thus, one choice for $D$ might be $\{1,2,4,8\} \cup \{5,10\} \cup \{0\} = \{0,1,2,4,5,8,10\}$, which is a set with 7 elements. It is easy to see that every nonzero element of the cyclic group $\mathbb{Z}_{15}$ would appear three times as a difference of elements in $D$. Thus, $D = \{0,1,2,4,5,8,10\}$ is a $(15,7,3)$ difference set.

**Example 11:** Consider the group $\mathbb{Z}_{21}$. The goal here is to find the $(21,5,1)$ difference set $D$. Since $5-1=4$, then the only prime that divides 4 (but not 7) is $p=2$. If $3 \in D$, then $3,6,12$ are in $D$. If $7 \in D$, then $7,14$ are in $D$. Then, one choice for $D$ might be $\{3,6,7,12,14\}$. This set can be verified as a $(21,5,1)$ difference set by using the previous methods shown in earlier examples.

**Group rings**

The definition of a group ring is shown below.

**Definition:** Fix a commutative ring $R$ with the multiplicative identity $1_R \neq 0_R$ and let $G$ be a finite group of order $n$ with a group operation written multiplicatively. The group ring $R[G]$ is the set of all of the following sums:

$$\sum_{i=1}^{n} a_i g_i = a_1 g_1 + a_2 g_2 + a_3 g_3 + \cdots + a_n g_n,$$

where $a_i \in R$, $g_i \in G$ and $1 \leq i \leq n$. [8], pg. 236.

Furthermore, some of the basic properties for the group rings are shown below (note that $a_i, b_i, b_j \in R$ and $1 \leq i, j \leq n$).

**Addition:** $\sum_{i=1}^{n} a_i g_i + \sum_{j=1}^{n} b_i g_i = \sum_{i=1}^{n} (a_i + b_i) g_i$.

**Scalar Multiplication:** $c \sum_{i=1}^{n} a_i g_i = \sum_{i=1}^{n} (c a_i) g_i$.

**Multiplication:**$(\sum_{i=1}^{n} a_i g_i) \cdot (\sum_{j=1}^{n} b_j g_j) = \sum_{i=1}^{n} a_i g_i (\sum_{j=1}^{n} b_j g_j)$. [16], pg.383.

The definition of difference sets can also be written in group ring notation.

**Definition:** Given a group ring $R[G]$, any difference set $D$ in $G$ can be written as the following in $R[G]$:

$$\sum_{d \in D} d$$

where $D \subset G$ and $|D| = k$.

Group ring notation, where the coefficients of $D$ are in $\{0, 1\}$, can be used to verify that $D$ is a difference set if and only if:

$$DD^{(-1)} = (k - \lambda) 1_G + \lambda G,$$

where $G$ is the group, $1_G$ is the identity of $G$, $DG = kG$, and $D^{(-1)}$ is the sum of the inverses of each of the elements in $D$. For every element $d \in D$, there is a $d' \in D^{(-1)}$ such that $dd' = g$, where $g$ is a non-identity element in $G$. Since there are $\lambda$ pairs of elements $d \in D$ and $d' \in D^{(-1)}$ such that $dd' = g$, then $\lambda G$ appears in $DD^{(-1)}$.

Furthermore, $\forall d \in D$, $\exists\, d^{-1} \in D$ such that $dd^{-1} = 1_G$. Since there are $k$ elements in $D$, this results in $k(1_G)$ to appear in $DD^{(-1)}$. Furthermore, there are $\lambda$ pairs of elements, $d \in D$ and $d' \in D^{(-1)}$ that do not give the identity $1_G$. Thus, $(k-\lambda)1_G$ is in $DD^{(-1)}$. Hence, $DD^{(-1)} = (k-\lambda)1_G + \lambda G$.

This is useful since it is a tool that verifies which sets in a given group are difference sets. For example, it can be used to prove Lemma 6.

*Proof.* (Lemma 6 (a) ): Assume $D$ is a $(v,k,\lambda)$ difference set in a group $G$ of order $v$. We will prove that $gD$ is also a $(v,k,\lambda)$ difference set, where $g \in G$. Then,

$$gDD^{(-1)}g^{-1} = g((k-\lambda)1_G + \lambda G)g^{-1}$$

$$= g((k-\lambda)1_G)g^{-1} + g(\lambda G)g^{-1} = gg^{-1}(k-\lambda)1_G + \lambda[g(G)g^{-1}] = (k-\lambda)1_G + \lambda G$$

(since $g(G)g^{-1} = G$). Thus, $gD$ is a difference set.

(Lemma 6 (b) ): Assume $D$ is a $(v,k,\lambda)$ difference set in a group $G$ of order $v$ and $\phi$ is an automorphism. We will prove that $\phi(D)$ is also a $(v,k,\lambda)$ difference set. Then,

$$\phi(D)\phi(D^{(-1)}) = \phi(DD^{(-1)}) = \phi((k-\lambda)1_G) + \lambda G) = \phi((k-\lambda)1_G) + \phi(\lambda G)$$

since $\phi$ is an automorphism of $G$. Then,

$$\phi((k-\lambda)1_G)+\phi(\lambda G)=(k-\lambda)\phi(1_G)+\lambda\phi(G).$$

Then,

$$(k-\lambda)\phi(1_G)+\lambda\phi(G)=(k-\lambda)1_G+\lambda G$$

since $\phi$ is an isomorphism and bijection. Thus, $\phi(D)$ is a difference set. $\square$

**Example 12:** Consider Example 9. $D=\{x,x^3,x^4,x^5,x^9\}$ is a difference set in the multiplicative group $C_{11}=<x:x^{11}=1>$. Then, $D$ becomes

$$D:=x+x^3+x^4+x^5+x^9,$$

if it is written in group ring notation. It can also be confirmed as a difference set since $DD^{(-1)}:=(5-2)1_G+2G=31_G+2G$, where $k-\lambda=3$ and $\lambda=2$.

**Abelian and non-abelian difference sets**

We will give examples of difference sets in abelian groups.

**Example 13(a):** Consider the difference set $D=\{1,2,4\}$ in Example 7(a). The difference set $D$ can be written in group ring notation, where

$$D:=x+x^2+x^4.$$

Then, $D$ is a $(7,3,1)$ difference set in $C_7=<x:x^7=1>$.

**Example 13(b):** Recall Example 13(a), where there is the difference set $D:=x+x^2+x^4$ in the group $C_7$. Now, $D$ needs to be verified that this is a difference set using the equation

$DD^{(-1)} = (k-\lambda)1_G + \lambda G$. Therefore, $D^{(-1)}$ is the sum of all of the inverses of the elements in $D$, thus $D^{(-1)} := x^3 + x^5 + x^6$. Thus,

$$DD^{(-1)} = (x+x^2+x^4)(x^3+x^5+x^6) = 3+x+x^2+x^3+x^4+x^5+x^6$$

$$= (3-1)1_G + (1+x+x^2+x^3+x^4+x^5+x^6) = (k-\lambda)1_G + \lambda G$$

(where $\lambda = 1$ and $k = 3$). Thus, this once again verifies that $D$ is a difference set in the group $C_7$.

**Example 14:** The set $D = \{x^2, x^3, x^{10}, x^{13}, x^{15}, x^{19}\}$ is a $(31,6,1)$ difference set of the group $G = C_{31} =< x : x^{31} = 1 >$, which can be easily checked using methods previously discussed. However, this can also be checked by using the group ring notation. Thus, $D$ can be written as

$$D := x^2 + x^3 + x^{10} + x^{13} + x^{15} + x^{19},$$

where $x^{31} = 1$. The sum of the inverses of the elements of $D$ is $D^{(-1)} := x^{29} + x^{28} + x^{21} + x^{18} + x^{16} + x^{12}$.

Then,

$$DD^{(-1)} = (x^2+x^3+x^{10}+x^{13}+x^{15}+x^{19})(x^{29}+x^{28}+x^{21}+x^{18}+x^{16}+x^{12})$$

$$= 5 + \sum_{i=0}^{30} x^i = (6-1)1_G + C_{31} = (k-\lambda)1_G + \lambda G.$$

This verifies that $D$ is a $(31,6,1)$ difference set of $C_{31}$.

**Example 15:** Consider the abelian group $C_8 \times C_2 = \langle x, y : x^8 = y^2 = 1, xy = yx \rangle$. Then,

$$D := x + xy + x^2 + x^3 + x^6 + x^7 y$$

is a $(16, 6, 2)$ abelian difference set since it belongs to a group has order 16, there are six terms in $D$ (or six element in $D$ when $D$ is written as a set), and every element in $C_8 \times C_2$ appears twice in the division table below.

| $\div$ | $x$ | $xy$ | $x^2$ | $x^3$ | $x^6$ | $x^7y$ |
|---|---|---|---|---|---|---|
| $x$ | $1$ | $y$ | $x^7$ | $x^6$ | $x^3$ | $x^2y$ |
| $xy$ | $y$ | $1$ | $x^7y$ | $x^6y$ | $x^3y$ | $x^2$ |
| $x^2$ | $x$ | $xy$ | $1$ | $x^7$ | $x^4$ | $x^3y$ |
| $x^3$ | $x^2$ | $x^2y$ | $x$ | $1$ | $x^5$ | $x^4y$ |
| $x^6$ | $x^5$ | $x^5y$ | $x^4$ | $x^3$ | $1$ | $x^7y$ |
| $x^7y$ | $x^6y$ | $x^6$ | $x^5y$ | $x^4y$ | $xy$ | $1$ |

Now that examples of difference sets in abelian groups have been given, an example of a difference set in a non-abelian group will be shown.

**Example 16:** Consider the group of order 16 that is defined as the following:

$$C_4 \rtimes_{-1} C_4 = < x, y \mid x^4 = y^4 = 1, yx = xy^3 > .$$

This is a non-abelian group (since $yx \neq xy$). A difference set in this group is

$$D := x + x^3 + y + y^3 + xy + x^3y^3.$$

This is a $(16,6,2)$ difference set since it belongs to a group has order 16, there are six terms in $D$, and every element in $C_4 \rtimes_{-1} C_4$ appears twice in the division table below, where the operation of the table is defined below.

| $\div$ | $a$ |
|---|---|
| $b$ | $ba^{-1}$ |

| $\div$ | $x$ | $x^3$ | $y$ | $y^3$ | $xy$ | $x^3y^3$ |
|---|---|---|---|---|---|---|
| $x$ | $1$ | $x^2$ | $xy^3$ | $xy$ | $y$ | $x^2y^3$ |
| $x^3$ | $x^2$ | $1$ | $x^3y^3$ | $x^3y$ | $x^2y$ | $y^3$ |
| $y$ | $x^3y^3$ | $xy^3$ | $1$ | $y^2$ | $x^3$ | $xy^2$ |
| $y^3$ | $x^3y$ | $xy$ | $y^2$ | $1$ | $x^3y^2$ | $x$ |
| $xy$ | $y^3$ | $x^2y^3$ | $x$ | $xy^2$ | $1$ | $x^2y^2$ |
| $x^3y^3$ | $x^2y$ | $y$ | $x^3y^2$ | $x^3$ | $x^2y^2$ | $1$ |

# CHAPTER IV

# Hadamard Difference Sets

## Basic definitions

**Definition:** A Hadamard matrix $H$ of order $m$ is a $m \times m$ matrix where the entries of the matrix are either 1 or -1 and $HH^T = mI$, where $I$ is the identity matrix.

**Example 17:** The following matrices below are Hadamard matrices.

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \begin{pmatrix} -1 & -1 \\ -1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & -1 \\ -1 & -1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.$$

Two Hadamard matrices of different orders can be used to create another, larger Hadamard matrix, where the order of this larger matrix is the product of the orders of the two smaller matrices.

**Theorem 9.** *If $H_m$ and $H_n$ are Hadamard matrices of order $m$ and $n$ respectively, $H_m \otimes H_n$ (Kronecker product of $H_m$ and $H_n$) is a Hadamard matrix of order $mn$.*

A proof of this theorem can be found in [16] on page 201.

**Example 18:** Consider the following Hadamard matrices of order 4,

$$H_1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}, H_2 = \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix}.$$

Then, by the previous theorem, $H_1 \otimes H_2 =$

$$\begin{pmatrix}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\
-1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
-1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
-1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \\
-1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
-1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1
\end{pmatrix}$$

is a Hadamard matrix of order 16.

With the basic definition of Hadamard matrices established, a particular type of difference set will be introduced. Then, a connection between these new difference sets and Hadamard matrices will be established.

**Definition of Hadamard difference Sets**

The definition of a Hadamard difference sets is shown below.

**Definition:** A difference set with parameters $(v,k,\lambda)$, where $v = 4(k-\lambda)$ is called a Hadamard difference set.

If $D$ is a Hadamard difference set, it is useful to consider a $\pm 1$ version of a difference set described below.

**Definition:** The $\pm 1$ version of a Hadamard difference set is $\underline{D} := G - 2D$.

**Lemma 10.** *Let $D = \sum a_g g \in \mathbb{Z}[G]$ be a group ring element where $a_g \in \{0,1\}$. Let $v = |G|$ and $k$ be the number of coefficients $a_g$ equal to 1. Fix $h \neq 1_G$ in $G$ and set $\lambda$ equal to the number of times $a_g$ and $a_{h^{-1}g}$ are both equal to 1. Set $\underline{D} := G - 2D$. Then, $D$ is a Hadamard $(v,k,\lambda)$ difference set iff $\underline{DD}^{(-1)} = v1_G$.*

*Proof.* Assume the hypothesis.

$(\rightarrow)$: Suppose $D$ is a Hadamard $(v,k,\lambda)$ difference set. Then, the $\pm 1$ version of $D$ is $\underline{D} := G - 2D$. Then,

$$\underline{DD}^{(-1)} := (G-2D)(G-2D^{(-1)}) = GG - 2DG - 2GD^{(-1)} + 4DD^{(-1)}.$$

Since $DD^{(-1)} = (k-\lambda)1_G + \lambda G$, $DG = GD^{(-1)} = kG$ and $GG = vG$, then

$$GG - 2DG - 2GD^{(-1)} + 4DD^{(-1)}$$

$$= vG - 2kG - 2kG + 4[(k-\lambda)1_G + \lambda G]$$

$$= vG - 4kG + 4(k-\lambda)1_G + 4\lambda G.$$

Furthermore,

$$-4kG+4\lambda G=(4k-4\lambda)1_G(-G)=4(k-\lambda)1_G(-G)=-vG.$$

Therefore,

$$vG-4kG+4(k-\lambda)1_G+4\lambda G=vG-vG+4(k-\lambda)1_G=4(k-\lambda)1_G.$$

Thus, $\underline{DD}^{(-1)}=4(k-\lambda)1_G=v1_G$.

($\leftarrow$): Suppose $\underline{DD}^{(-1)}=v1_G$.Choose a nonidentity element $h\in G$. The coefficient of $h$ in $\underline{DD}^{(-1)}$ is

$$0=\sum a_g a_{h^{-1}g}.$$

If $(a_g,a_{h^{-1}g})$ are equal to $-1$ $\lambda$ times, then there are $k-\lambda$ times that $a_g=-1$ and $a_{h^{-1}g}=1$ and $k-\lambda$ times that $a_g=1$ and $a_{h^{-1}g}=-1$. There are $v-2k+\lambda$ times that both $a_g$ and $a_{h^{-1}g}$ are equal to 1. Then,

$$v-2k+\lambda-2(k-\lambda)=v-4(k-\lambda).$$

This forces $\lambda$ to be constant, independent of $h$ and $v=4(k-\lambda)$. Then, $DG=kG$ and

$$DD^{(-1)}=\left(\frac{G-\underline{D}}{2}\right)\left(\frac{G-\underline{D}^{(-1)}}{2}\right).$$

Then,

$$DD^{(-1)}=\left(\frac{(G-\underline{D})(G-\underline{D}^{(-1)})}{4}\right).$$

Then,

$$DD^{(-1)} = \left( \frac{(GG - \underline{D}G - G\underline{D}^{(-1)} + \underline{DD}^{(-1)})}{4} \right).$$

Since $GG = vG$, $\underline{D}G = (G-2D)G = vG - 2kG$ and $G\underline{D}^{(-1)} = G(G-2D^{(-1)}) = vG - 2kG$,

$$\left( \frac{(GG - \underline{D}G - G\underline{D}^{(-1)} + \underline{DD}^{(-1)})}{4} \right) = \left( \frac{vG - 2vG + 4kG + v1_G}{4} \right).$$

Then,

$$\left( \frac{vG - 2vG + 4kG + v1_G}{4} \right) = \left( \frac{-vG + 4kG + v1_G}{4} \right).$$

Since $v = 4(k-\lambda)$,

$$\left( \frac{-vG + 4kG + v1_G}{4} \right) = \left( \frac{-4(k-\lambda)G + 4kG + 4(k-\lambda)1_G}{4} \right).$$

Therefore,

$$\left( \frac{-4(k-\lambda)G + 4kG + 4(k-\lambda)1_G}{4} \right) = -(k-\lambda)G + kG + (k-\lambda)1_G.$$

Then,

$$-(k-\lambda)G + kG + (k-\lambda)1_G = (k-\lambda)1_G + \lambda G.$$

Therefore, $DD^{(-1)} = (k-\lambda)1_G + \lambda G$. Since $v = 4(k-\lambda)$, $D$ is a Hadamard $(v,k,\lambda)$ difference set.

□

This lemma appears in [3].

An example of how this definition is applied to a difference set is shown below:

**Example 19:** Consider Example 15. Then, the difference set in the group $G = C_8 \times C_2 =< x, y : x^8 = y^2 = 1, xy = yx >$ is

$$D := x + x^2 + x^7y + xy + x^3 + x^6.$$

Then, $G$ is the sum of all of its elements and

$$\underline{D} := G - 2D = 1 + x^4 + x^5 + x^7 + y + x^2y + x^3y + x^4y + x^5y + x^6y - (x + x^2 + x^7y + xy + x^3 + x^6)$$

is the $\pm 1$ version of the difference set $D$. Furthermore, $\underline{DD}^{(-1)} := 16$ in the group ring $\mathbb{Z}G$. Since $|G| = 16 = 4(6-2)$, then $D$ is a Hadamard difference set.

If $D$ is a Hadamard difference set, then a Hadamard matrix can be formed by interchanging the entries of the incidence matrix of $D$, replacing the 1's with $-1$ and the 0's with 1. Thus, Hadamard difference sets yields Hadamard matrices.

**Example 20:** Consider the trivial Hadamard difference set $T = \{1\}$ in the group $C_4 = \{x : x^4 = 1\}$. Then, the incidence matrix of $T$ is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Then, by interchanging all of the 1's with -1 and 0's with 1, this results in the following matrix:

$$\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix},$$

which is a Hadamard matrix of order 4.

**Hyperplane construction**

Finding constructions of difference sets and the various ways to manipulate these constructions is important when going deeper into difference sets and groups. To expand on this concept, one mathematical topic must be introduced to obtain these new difference sets: hyperplane construction.

**Definition:** A subspace of a vector space $V$ of dimension one less than the dimension of $V$ is called a hyperplane.

In this thesis, all vector spaces will be over the field $\mathbb{F}_2 = \{0, 1\}$, where the arithmetic for this field is mod 2. One source for the necessary linear algebra is [14]. Vector spaces are of interest because hyperplanes are a part of vector spaces.

A group $\mathscr{H}$ isomorphic to $\mathbb{Z}_2^{d+1}$ is viewed as a vector space of dimension $d+1$ over $\mathbb{Z}_2$ and so call its subgroups of order $2^d$ "hyperplanes." The number of such hyperplanes is $2^{d+1}-1$.

Let $H_i, H_j$ be hyperplanes of $\mathscr{H}$ in $\mathbb{Z}_2^{d+1}$. Some properties of the hyperplanes of $\mathscr{H}$ include:

1. $H_i = H_i^{(-1)}$

2. $H_i^2 = 2^d H_i$

3. Since $H_i, H_j$ are distinct hyperplanes, then $H_iH_j = 2^d\mathscr{H}$.

4. Every non-identity element of $\mathscr{H}$ is in $2^d - 1$ hyperplanes and so $\sum_i H_i = 2^d + (2^d - 1)\mathscr{H}$.

Now, the theorem for hyperplane construction is shown below.

**Theorem 11.** *Suppose G is a group of order $2^{2d+2}$ with normal subgroup $\mathscr{H}$ isomorphic to $\mathbb{Z}_2^{d+1}$. Label the hyperplanes of $\mathscr{H}$, $H_1, H_2, ..., H_{2^{d+1}-1}$. Then, there is a choice of coset representatives of $\mathscr{H}$, $\{1, g_1, g_2, ..., g_{2^{d+1}-1}\}$, such that $D = \sum_{i=1}^{2^{d+1}-1} g_iH_i$ is a $(2^{2d+2}, 2^d(2^{d+1}-1), 2^d(2^d-1))$ difference set.*

For a proof of this theorem, see Corollary 1.10, page 5 of [1].

We will illustrate this construction in the following example.

**Example 21:** Consider the group $C_2 \times C_8 =< x, y : x^2 = y^8 = 1, xy = yx >$. and $\mathscr{H} = \{1, y^4, x, xy^4\}$.

Then, there are three different hyperplanes in $\mathscr{H}$, called $H_1, H_2, H_3$. They are (in group ring notation):

$$H_1 := 1 + x,$$

$$H_2 := 1 + y^4,$$

$$H_3 := 1 + xy^4.$$

Then, $\{1, y, y^2, y^3\}$ is a choice of coset representatives of $\mathscr{H}$. The coset representatives $y, y^2, y^3$ gives

$$D := y^2H_1 + yH_2 + y^3H_3 = y^2 + xy^2 + y + y^5 + y^3 + xy^7.$$

Then, $D$ is a difference set by Theorem 11.

The exact same strategy that is used for abelian groups can also be applied to non-abelian groups. However, when this action is performed, it must be done so with extreme caution, since the elements in these groups may not commute. An example of a non-abelian group difference set is given below.

**Example 22:** Referring back to Example 16, with the group $C_4 \rtimes_{-1} C_4 =< x,y : x^4 = y^4 = 1, yx = xy^3 >$. Consider $\mathscr{H} = \{1, x^2, y^2, x^2y^2\}$. Then,

$$H_1 := 1 + x^2,$$

$$H_2 := 1 + y^2,$$

$$H_3 := 1 + x^2y^2.$$

are the three hyperplanes of $\mathscr{H}$. A choice for the cosets representatives of $\mathscr{H}$ is $C = \{1, x, y, xy\}$. Then, the coset representatives $x, y, xy$ gives

$$D := xH_1 + yH_2 + xyH_3 = x + x^3 + y + y^3 + xy + x^3y^3$$

This set is a difference set in this group.

**Product construction**

Difference sets will become more complex in groups of higher orders. One of the ways that a difference set can be found in groups of higher order is by using a product construction.

**Definition:** Given subgroups $G_1, G_2$ of $G$, where $G_1 \cdot G_2 = G$ and $G_1 \cap G_2 = \{1\}$, a

Hadamard difference set $D_1$ of $G_1$ of order $v_1$, and a Hadamard difference set $D_2$ of $G_2$ of order $v_2$, the product construction of a difference set $D$ in $G$ is the following:

$$D := (D_1 D_2^C) + (D_1^C D_2).$$

Here, $D_1^C$ is the complement of the difference set $D_1 \in G_1$ and $D_2^C$ is the complement of the difference set $D_2 \in G_2$. The $\pm 1$ version of this construction is the following:

$$\underline{D} = \underline{D_1} \cdot \underline{D_2}.$$

$\underline{D}$ is a difference set since

$$\underline{DD}^{(-1)} = \underline{D_1}(\underline{D_2 D_2}^{(-1)})\underline{D_1}^{(-1)} = \underline{D_1}(4(k_2 - \lambda_2)1_G)\underline{D_1}^{(-1)} = 4(k_2 - \lambda_2)1_G \underline{D_1 D_1}^{(-1)}$$

$$= 4(k_2 - \lambda_2)1_G 4(k_1 - \lambda_1)1_G = v_1 v_2 1_G = |G| 1_G,$$

where $(v_1, k_1, \lambda_1)$ are the parameters for $D_1$ and $(v_2, k_2, \lambda_2)$ are the parameters for $D_2$.

The product construction is equivalent to a modification of the tensor products of incidence matrices. The following theorem will illustrate what this means.

**Theorem 12.** *Suppose $M$ is the incidence matrix for a $(v_1, k_1, \lambda_1)$ symmetric design $D_1$, where $v_1 = 4(k_1 - \lambda_1)$ and $N$ is the incidence matrix for a $(v_2, k_2, \lambda_2)$ symmetric design $D_2$, where $v_2 = 4(k_2 - \lambda_2)$. Then,*

*1) Assume a new matrix was created by replacing all entries equal to 1 in $M$ with $J - N$ (where $J$ is the all ones matrix) and all entries equal to 0 with $N$. Then, this new matrix,* *called Prod(M,N), is an incidence matrix of a* $(V,K,\Lambda)$ *design where*

$$V = v_1v_2, K = v_2k_1 + v_1k_2 - 2k_1k_2, \Lambda = v_2k_1 + v_1\lambda_2 - 2k_1k_2.$$

*2) If* $M_1, M_2$ *are incidence matrices of the two isomorphic designs and* $N_1, N_2$ *are incidence matrices of two other isomorphic designs, then the design with incidence matrix Prod(*$M_1, N_1$*) is isomorphic to the design with incidence matrix Prod(*$M_2, N_2$*).*

*Proof.* 1) Assume the hypothesis. Then, the incidence matrix $M$ has $v_1$ rows and columns, $k_1$ 1's in each row and $\lambda_1$ is the number in which any two rows of the matrix both have a one in a column. Thus, $N$ has $v_2$ rows and columns, $k_2$ 1's in each row and $\lambda_2$ is the number in which any two rows of the matrix both have a one in a column and $J-N$ has $v_2$ rows and columns, $v_2 - k_2$ 1's in each row and $v_2 - 2k_2 + \lambda_2$ is the number in which any two rows of the matrix both have a one in a column. Thus, we creating this new matrix using the method given, each entry of $M$ is being replaced with a $v_2 \times v_2$ matrix. Since $M$ is a $v_1 \times v_1$, then this new matrix, Prod($M$,$N$), is a $v_1v_2 \times v_1v_2$ matrix. Then, for each row of Prod($M$,$N$), the rows were created by replacing all entries of 1 with $J-N$ and all entries of 0 with $N$, where there are $k_1$ 1's and $v_1 - k_1$ 0's in each row of $M$. Since each row of $N$ has $k_2$ 1's and each row of $J-N$ has $v_2 - k_2$ 1's, then each row of Prod($M$,$N$) will have $K$ 1's, where,

$$K = k_1(v_2 - k_2) + (v_1 - k_1)k_2 = v_1k_2 + v_2k_1 - 2k_1k_2.$$

Furthermore, look at one row of $M$. Recall that each row of $M$ has entries that consist of either the matrix $N$ (which replaced the original entry of 0) or $J-N$ (which replaced the original entry of 1). Then, there are $v_1 - k_1$ entries in which $N$ appears in that row of $M$. Pick two different rows of this matrix formed by $J - N$ and N. Since number in which any two (different) rows of the matrix $N$ both have a one in a column is $\lambda_2$, then the number in which any two (different) rows of each matrix $N$ in that row of $M$ both have a one in a column in that row of $M$ is $(v_1 - k_1)\lambda_2$. Furthermore, there are $k_1$ entries in which $J - N$ appears and the number in which any two (different) rows of the matrix $N$ both have a one in a column is $v_2 - 2k_2 + \lambda_2$. Then, the the number in which any two (different) rows of each matrix $J - N$ in that row of $M$ both have a one in a column is $(v_2 - 2k_2 + \lambda_2)k_1$. Then, the number in which any two rows of the matrix inside that row of $M$ where both have a one in a column is

$$\Lambda = (v_1 - k_1)\lambda_2 + (v_2 - 2k_2 + \lambda_2)k_1 = v_1\lambda_2 + v_2k_1 - 2k_12k_2.$$

Now, consider two different rows of $M$. The number in which any two rows of the matrix both have a one in a column is $\lambda_1$. In each of those rows, they both contain a matrix formed by replacing the entries of 0 and 1 with $N$ and $J - N$ respectively. Then, in each of these matrices, pick a row such that they are both the $i^{th}$ row of their respective matrix. Then, by using a argument similar to the argument above (but slightly modified to fit this case), then the number in which any two different rows of $M$ (where we pick the $i^{th}$ row of the matrices in both rows of $M$) both rows have a one in a column is

$$\Lambda_1 = (v_2 - k_2)\lambda_1 + 2(k_1 - \lambda_1) + (v_1 - 2k_1 + \lambda_1)k_2 = v_2\lambda_1 + v_1k_2 - 2k_1k_2.$$

Furthermore, we have that

$$v_1\lambda_2 + v_2k_1 - 2k_1 2k_2 = v_2\lambda_1 + v_1k_2 - 2k_1k_2$$

only when

$$v_2(k_1 - \lambda_1) = v_1(k_2 - \lambda_2).$$

Since $v_1 = 4(k_1 - \lambda_1)$ and $v_2 = 4(k_2 - \lambda_2)$, then $v_2(k_1 - \lambda_1) = v_1(k_2 - \lambda_2)$ becomes

$$4(k_1 - \lambda_1)(k_2 - \lambda_2) = 4(k_1 - \lambda_1)(k_2 - \lambda_2)$$

Therefore,

$$\Lambda_1 = \Lambda.$$

Finally, consider two different rows of $M$ and in each of those rows (where each row contains a different matrix formed by replacing the entries of 0 and 1 with $N$ and $J - N$ respectively), pick a row such that they are both not the $i^{th}$ row of their respective matrix. Then, by an argument similar to the previous two arguments (but modified for this case), this results in the number in which any two different rows of $M$ (where we pick two rows that are both not the $i^{th}$ row of the matrices in both rows of $M$) both rows have a one in a column is

$$\Lambda_2 = (v_2 - 2k_2 + \lambda_2)\lambda_1 + 2(k_1 - \lambda_1)(k_2 - \lambda_2) + (v_1 - 2k_1 + \lambda_1)\lambda_2 = v_2\lambda_1 + v_1k_2 - 2k_1k_2.$$

Then, this equation will become

$$v_1\lambda_2 + v_2\lambda_1 + 2k_1k_2 - 4k_1\lambda_2 - 4k_1\lambda_2 + 4\lambda_1\lambda_2.$$

Since $2k_1k_2 = -2k_1k_2 + 4k_1k_2$, then the above equation can be transformed into

$$v_1\lambda_2 + v_2\lambda_1 - 2k_1k_2 + 4(k_1k_2 - k_1\lambda_2 - k_1\lambda_2 + \lambda_1\lambda_2).$$

Furthermore, by taking $v_2\lambda_1 = v_2k_1 - v_2(k_1 - \lambda_1)$, this implies that

$$v_1\lambda_2 + v_2k_1 - v_2(k_1 - \lambda_1) - 2k_1k_2 + 4(k_1 - \lambda_1)(k_2 - \lambda_2).$$

Then, this equation equals

$$v_1\lambda_2 + v_2k_1 - 2k_1k_2 - v_2(k_1 - \lambda_1) + 4(k_1 - \lambda_1)(k_2 - \lambda_2) = \Lambda + (k_1 - \lambda_1)[-v_2 + 4(k_2 - \lambda_2)].$$

Since $v_2 = 4(k_2 - \lambda_2)$, then $(k_1 - \lambda_1)[-v_2 + 4(k_2 - \lambda_2)] = 0$, implying that

$$\Lambda_2 = \Lambda.$$

Therefore, this new matrix, Prod($M$,$N$) is an incidence matrix of a $(V,K,\Lambda)$ design where

$$V = v_1v_2, K = v_2k_1 + v_1k_2 - 2k_1k_2, \Lambda = v_2k_1 + v_1\lambda_2 - 2k_1k_2.$$

2) Assume the hypothesis. If there are permutation matrices $P,Q,R,S$ such that $M_2 = PM_1Q$ and $N_2 = RN_1S$ (by Lemma 4), then $(P \otimes R)$Prod$(M_1,N_1)(Q \otimes S)$ =Prod$(M_2,N_2)$. $\square$

The simplest application of Theorem 12 occurs when $G$ is an internal direct product of two subgroups $G_1$ and $G_2$. In this case, if $G_1$ and $G_2$ contain Hadamard difference sets $D_1$ and $D_2$ with incidence matrices $M$ and $N$, respectively, then the incidence matrix of the difference set created by the product construction will be Prod$(M,N)$.

**Example 23:** Let $G = C_4 \times C_4 = \langle a,b : a^4 = b^4 = 1, ab = ba \rangle$. Let $D_1 = \{1\}$ in $\langle a \rangle$ and $D_2 = \{1\}$ in $\langle b \rangle$.

Then,

$$\underline{D_1} := -1 + a + a^2 + a^3$$

and

$$\underline{D_2} := -1 + b + b^2 + b^3.$$

Then, $\underline{D} = \underline{D_1} \cdot \underline{D_2} = (-1 + a + a^2 + a^3)(-1 + b + b^2 + b^3)$ yields a difference set in $G$.

Overall, this is important because this introduces another method that can used to find difference sets within groups of a specific order.

# CHAPTER V

# Designs with the Symmetric Difference Property

## SDP designs

When looking into designs that come from groups of a specific order, (64,256, etc.), one concept that is of particular interest is whether or not a design and/or difference set has what will be called the Symmetric Difference Property, or the SDP.

**Definition:** The symmetric difference of sets $A$ and $B$, denoted $A \triangle B$, is $A \triangle B = (A - B) \cup (B - A)$.

**Definition:** A $(v,k,\lambda)$ symmetric design has the Symmetric Difference Property if the symmetric difference of any three blocks is either a block or the complement of a block.[5]

This property can occur in Hadamard $(v,k,\lambda)$ difference sets with parameters

$$(2^{2m}, 2^{2m-1} - 2^{m-1}, 2^{2m-2} - 2^{m-1}).$$

This particular design has been defined and studied in [13], [15] and [11].

These particular difference sets are the difference sets we are interested in because these difference sets hold some interesting properties, which will be discussed later on.

## Examples of SDP and non-SDP difference sets

Below are some examples of SDP difference sets and difference sets that do not have SDP.

**Example 24:** Consider the abelian group $C_2 \times C_8$ in Example 21. Then, a difference set is $y + y^2 + y^3 + xy^2 + y^5 + xy^7$, which was found using Theorem 11.

Then, the incidence matrix of the difference set is:

$$\begin{pmatrix}
0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0
\end{pmatrix}$$

One may verify this design has the SDP by seeing if any three rows of this matrix added together forms another row of the matrix or its complement. This is equivalent to finding the symmetric difference of three blocks since each row of the matrix represents a block. When adding two rows together modulo 2, there will be $\lambda$ 1's that add up to zero. This is equivalent to how the symmetric difference "throws out" the $\lambda$ elements shared by the blocks. The sum of three rows modulo 2 is equivalent to the symmetric difference of three blocks.

**Example 25:** Consider the non-abelian group that was given in Example 16. Then, $x+x^3+y+y^3+xy+x^3y^3$ is a difference set in $C_4 \rtimes_{-1} C_4 =< x,y : x^4 = y^4 = 1, yx = xy^3 >$. This difference set has the SDP since any three rows added together from its incidence matrix either gives another row of the matrix or its complement.

For example, consider the first four rows of the incidence matrix that are shown below. The first three rows added together gives the fourth row of the matrix.

0 1 0 1 1 0 1 1 0 0 0 0 0 0 0 1

1 0 1 0 0 0 1 1 0 1 1 0 1 1 0 0

0 1 0 1 0 0 0 0 0 1 1 0 1 1 0 0

1 0 1 0 1 0 0 0 0 0 0 0 1 1 0 1

**Example 26:** Consider the group $C_8 \times C_2$. This group has the difference set

$$D = x+x^2+x^3+x^5+x^2y+x^7y,$$

The incidence matrix for this design is:

$$\begin{pmatrix}
0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

This difference set does not have the SDP since the sum of the second, third and fourth rows modulo 2 is (0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 ), which is not a row in the matrix or its complement.

**Rank of a matrix**

The definition of a row space and its rank is shown below.

**Definition:** The **row space** of a matrix is the subspace generated by rows of the matrix or, for a $m \times n$ matrix $A$,

$$\text{row space} = < \bar{x}^T A \mid \bar{x} \in \mathbb{F}^m >,$$

where $\mathbb{F}$ is a field.

**Definition:** The **rank** of a matrix is the dimension of the row space.

**Claim:** If the dimension of the row space for a matrix $A$ is $r$ and the field is $\mathbb{F}_2$, the row space will have $2^r$ elements.

This claim is true because each element of the row space is a linear combination of the members of the basis. Since each scalar is either 0 or 1, there are only $2^r$ ways we can construct an element of the row space.

**Example 27(a):** Consider the matrix

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix},$$

which is the incidence matrix for the difference set $D = \{1,2,4\}$ in the group $\mathbb{Z}_7$. Then, the above matrix can be row reduced to the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The rank of this matrix is 4 and the row space has the basis:

$$\{(1\,0\,0\,0\,1\,1\,0), (0\,1\,0\,0\,0\,1\,1), (0\,0\,1\,0\,1\,1\,1), (0\,0\,0\,1\,1\,0\,1)\}.$$

So, the row space has $2^4 = 16$ elements.

The rank of a matrix is important because it is essential to a theorem about difference sets with the SDP. This theorem will be discussed later on.

**Codes of a design**

Below is the definition of a linear code.

**Definition:** A linear code is a subspace of a vector space over the field $\mathbb{F}_2$. A **codeword** is an element of the subspace. The code has the parameter $[n, r]$, where $n$ is the length of a codeword and $r$ is the dimension of the linear code. The number of codewords that a $[n, r]$ linear code has is $2^r$.

**Definition:** The **distance** between two codewords, $x, y$ in the code $C$ is the number of positions in which the codewords differ, that is,

$$d(x,y) := |\{i : 1 \leq i \leq n, x_i \neq y_i\}|,$$

where $x_i$ and $y_i$ are the $ith$ positions of the codewords $x$ and $y$ respectively.[16]

**Definition:** The **minimum distance** of the code $C$ is $d = min\{d(x,y) : \mathbf{x} \in C, \mathbf{y} \in C, \mathbf{x} \neq \mathbf{y}\}$.[16]

**Example 27(b):** Consider the result that was found in Example 27(a). Then,

$$\{1000110, 0100011, 0010111, 0001101\}.$$

can be used for the basis of the code $C$. Therefore,

$$C = \{0000000, 1000110, 0100011, 0010111, 0001101, 1100101, 1010001, 1001011,$$

$$0110100, 0101110, 0011010, 1110010, 1101000, 1011100, 0111001, 1111111\}.$$

Assume that $x = 1010001$ and $y = 1101000$. Then, the distance is $d(x,y) = 4$ since the codewords $x$ and $y$ differ in four positions (for example, $x$ has a zero in its $2^{nd}$ position, while $y$ has a one in its $2^{nd}$ position, $x$ has a one in its third position while $y$ has a zero in its third position, etc). The minimum distance for $C$ is equal to 3 because the smallest distance that can be obtained from any two codewords is 3.

**Example 28:** Consider Example 21, where there was the group $C_2 \times C_8$ and the difference set $y + y^2 + y^3 + xy^2 + y^5 + xy^7$. With this difference set, an incidence matrix was created in Example 24. Each row of the incidence matrix can be treated as a codeword for a code $C \subset \mathbb{F}_2^{16}$.

Then, after a lot of calculations, the following results can be produced based on the incidence matrix for the difference set $y + y^2 + y^3 + xy^2 + y^5 + xy^7$:

Rank of matrix: 6

Minimum distance: 6

Number of Codewords (in $C$): 64

Number of Codewords in $C$ with weight $0$ : 1

Number of Codewords in $C$ with weight $6$ : 16

Number of Codewords in $C$ with weight $8$ : 30

Number of Codewords in $C$ with weight $10$ : 16

Number of Codewords in $C$ with weight $16$ : 1

Does the Difference Set have the SDP?: Yes

We were able to reach these conclusions by utilizing the following theorems.

**Theorem 13.** *If a design $D$ comes from a $(2^{2m}, 2^{2m-1} - 2^{m-1}, 2^{2m-2} - 2^{m-1})$ difference set (where $m \geq 1$), then $\overline{1}$ is in the code of that design.*

*Proof.* Let $C$ be the code of the design. Since the rank of the code is $r > 0$, the number of codewords, $2^r$, is even. The group $G$ of order $2^{2m}$ acts as an automorphism group of this code, (if $B$ is a block in the code, then $gB$ is also in the code). Consider the orbit of codewords under left multiplication by elements of $G$, or Orbit($c$)=$\{gc \mid g \in G\}$. Then, by the Orbit-Stabilizer Theorem (page 114 of [8]), the size of an orbit of a codeword $c$ gives the index of the stabilizer of $c$. Consider the orbits that have size 1 (where $gc = c$ for all

$g \in G$). Since the code is linear, $\overline{0} \in C$. Given any two coordinates $g_1, g_2$, there is a group element $g = g_1 g_2^{-1}$ that moves $g_1$ to $g_2$. This means if a codeword $c$ is in an orbit by itself, all entries of $c$ are the same. However, the number of codewords must be even. This implies that there is some codeword other than $\overline{0}$ that has all entries be the same. Thus, $\overline{1}$, is in the code of that design. $\square$

The proof was communicated by John Dillon to Ken Smith, [7].

**Theorem 14.** *A symmetric design D coming from a difference set with parameters* $(2^{2m}, 2^{2m-1} - 2^{m-1}, 2^{2m-2} - 2^{m-1})$ *has the SDP if and only if the rank of the associated code is* $2m+2$ *(*$m \geq 2$*).*

*Proof.* Assume the hypothesis.

($\rightarrow$): Assume the symmetric design has the SDP property. Then, by Theorem 13, $\overline{1}$ is in the code of the design $C$. Furthermore, there are $2^{2m}$ blocks that are codewords in the code $C$ since $|\mathscr{B}| = 2^{2m}$ for the set of blocks $\mathscr{B}$. Furthermore, if $B_0$ is a fixed block and $B$ is any block in the design, then the set,

$$\{B_0 + B : B \in \mathscr{B}\},$$

generates $2^{2m}$ blocks that are also codewords in the code $C$. Furthermore, $\mathscr{B} \cap \{B_0 + B : B \in \mathscr{B}\} = \varnothing$ since all of the codewords of the blocks in $\mathscr{B}$ have weight $2^{2m-1} - 2^{m-1}$ and most of the codewords given by $\{B_0 + B : B \in \mathscr{B}\}$ have weight $2^{2m-1}$ (the only element in this set that does not given a codeword of weight $2^{2m-1}$ is the all zeroes vector, $\overline{0}$, which gives a codeword of weight zero). Then, consider the set,

$$\{B+\overline{1} : B \in \mathscr{B}\}.$$

This set generates $2^{2m}$ in $C$, where

$$\mathscr{B} \cap \{B+\overline{1} : B \in \mathscr{B}\} = \varnothing$$

and

$$\{B+\overline{1} : B \in \mathscr{B}\} \cap \{B_0+B : B \in \mathscr{B}\} = \varnothing$$

since all the codewords of the blocks given by $\{B+\overline{1} : B \in \mathscr{B}\}$ have a weight of $2^{2m} - (2^{2m-1} - 2^{m-1})$. Finally, consider the set (where $B_0$ is a fixed block),

$$\{B_0+B+\overline{1} : B \in \mathscr{B}\}.$$

Then,

$$\mathscr{B} \cap \{B_0+B+\overline{1} : B \in \mathscr{B}\} = \varnothing$$

and

$$\{B+\overline{1} : B \in \mathscr{B}\} \cap \{B_0+B+\overline{1} : B \in \mathscr{B}\} = \varnothing$$

since most of the codewords of the blocks given by $\{B_0+B+\overline{1} : B \in \mathscr{B}\}$ have weight $2^{2m-1}$ (the only element in this set that does not given a codeword of weight $2^{2m-1}$ is $\overline{1}$, which gives a codeword of weight 16). However, the weight of all of the codewords of the blocks in

$$\{B_0 + B : B \in \mathscr{B}\},$$

also have weight $2^{2m-1}$. Thus, assume there is an element in $\{B_0 + B : B \in \mathscr{B}\}$ and an element in $\{B_0 + B + \overline{1} : B \in \mathscr{B}\}$ such that

$$B_0 + B = B_0 + B + \overline{1}.$$

By adding $B_0$ to both sides, this becomes

$$B = B + \overline{1}.$$

The codeword of $B$ has weight $2^{2m-1} - 2^{m-1}$ while the codeword of $B + \overline{1}$ has weight $2^{2m} - (2^{2m-1} - 2^{m-1})$. This is a contradiction. Therefore,

$$\{B_0 + B : B \in \mathscr{B}\} \cap \{B_0 + B + \overline{1} : B \in \mathscr{B}\} = \varnothing.$$

Thus, the blocks given in all four sets are distinct. Since the design has the SDP, then it can be concluded that there are no other codewords that can be given outside of these four sets. Therefore, there are $4(2^{2m}) = 2^{2m+2}$ codewords in $C$, which means that the rank of $C$ is $2m + 2$.

($\leftarrow$): Assume that the rank of the code $C$ of the symmetric design $D$ is $2m + 2$. Therefore, all of the codewords of $C$ are given by the blocks of the following four sets, where each set has a cardinality of $2^{2m}$:

$$\mathscr{B}, \{B_0 + B : B \in \mathscr{B}\}, \{B + \overline{1} : B \in \mathscr{B}\}, \{B_0 + B + \overline{1} : B \in \mathscr{B}\},$$

where $\mathscr{B}$ is the set of the blocks of the given design.

Then, consider $B_0 + B_1 + B_2$, where $B_0, B_1, B_2 \in \mathscr{B}$. Since $B_0 + B_1 \in \{B_0 + B : B \in \mathscr{B}\}$, then assume $B_0 + B_1 = B$, where $B \in \{B_0 + B : B \in \mathscr{B}\}$. Then, $B_0 + B_1 + B_2 = B + B_2$. Then, $B + B_2$ gives back a codeword (of a block) with weight $2^{2m-1} - 2^{m-1}$ or $2^{2m} - (2^{2m-1} - 2^{m-1})$, which means that $B_2 + B \in \mathscr{B}$ or $B_2 + B \in \{B + \overline{1} : B \in \mathscr{B}\}$. Therefore, $B_0 + B_1 + B_2 \in \mathscr{B}$ or $B_0 + B_1 + B_2 \in \{B + \overline{1} : B \in \mathscr{B}\}$. Therefore, the design that is given has the SDP. $\square$

**Product construction theorem of difference sets**

One result that is also of interest is determining whether or not symmetric designs exist in groups of order 256. This is difficult since there are lots of groups of order 256 or higher. One tool has been found that could clarify what groups of order 256 or higher give SDP designs. The following theorem is the tool that helped discovered some of these designs:

**Theorem 15.** *Let $D_1 \subset G_1$ be a $(2^{2m+2}, 2^{2m+1} - 2^m, 2^{2m} - 2^m)$ difference set, where $D_1$ has the SDP. Then, the product construction $D := \underline{D_1}(-1 + x + x^2 + x^3)$ will be a SDP difference set in the group $G_1 \times C_4$.[5]*

*Proof.* Let $D_1 \subset G_1$ be a $(2^{2m+2}, 2^{2m+1} - 2^m, 2^{2m} - 2^m)$ difference set, where $D_1$ has the SDP. By the given parameters, $D = \underline{D_1}(-1 + x + x^2 + x^3)$ is a difference set in the group $G_1 \times C_4$. Suppose $A_1$ is an incidence matrix for $D_1$. The complement of that matrix, $A_1^c$ is $A_1^c = A_1 + J$, where $J$ is a $2^{2m+2} \times 2^{2m+2}$ matrix that has 1 in all of its entries. Now, order the elements of $G_1 \times C_4$ such that all of the elements of the coset $G_1 \times \{1\}$ appear first and are ordered using the same ordering that was used in $G_1$ to create $A_1$. Then, this coset will be followed by the elements of the cosets $G_1 \times \{x\}$, $G_1 \times \{x^2\}$, $G_1 \times \{x^3\}$ respectively and each of these cosets are ordered based on the the ordering used for $G_1$ to construct $A_1$. Thus, the incidence matrix $\mathbf{A}$ for $D$ will be:

$$\mathbf{A}= \begin{pmatrix} A_1^c & A_1 & A_1 & A_1 \\ A_1 & A_1^c & A_1 & A_1 \\ A_1 & A_1 & A_1^c & A_1 \\ A_1 & A_1 & A_1 & A_1^c \end{pmatrix}$$

Now, consider any three rows from the matrix $\mathbf{A}$, which will be called $R_c, R_d$, and $R_e$, where $0 \leq c,d,e \leq 2m+2$. For $l \in \{c,d,e\}$, $R_l$ is a concatenation of the row $r_l$ from the matrix $A_1$. More specifically, $R_l = \begin{pmatrix} r_l^c & r_l & r_l & r_l \end{pmatrix}$ and $r_l^c$ is the complement of the row $r_l$ or $r_l^c = r_l + \mathbf{1}_{2m+2}$, where $\mathbf{1}_{2m+2}$ is the $1 \times (2m+2)$ vector consisting of all ones. Since $A_1$ has the SDP, then for any rows $r_c, r_d, r_e$ in $A_1$ then $r_c + r_d + r_e$ equals $r_u$ or $r_u^c$ for some $1 \leq u \leq 2^{2m+2}$. Then,

$$\begin{aligned} R_c + R_d + R_e &= \begin{pmatrix} r_c^c & r_c & r_c & r_c \end{pmatrix} + \begin{pmatrix} r_d^c & r_d & r_d & r_d \end{pmatrix} + \begin{pmatrix} r_e^c & r_e & r_e & r_e \end{pmatrix} \\ &= \begin{pmatrix} r_c + r_d + r_e + \mathbf{1}_{2m+2} & r_c + r_d + r_e & r_c + r_d + r_e & r_c + r_d + r_e \end{pmatrix}, \text{ where} \end{aligned}$$

$\begin{pmatrix} r_c + r_d + r_e + \mathbf{1}_{2m+2} & r_c + r_d + r_e & r_c + r_d + r_e & r_c + r_d + r_e \end{pmatrix} \in \{R_u, R_u^c\}$ where $1 \leq u \leq 2^{2m+2}$.

Thus, the sum of any three rows in $\mathbf{A}$ will give either another row of this matrix or its complement. Therefore, $D$ is an SDP difference set of the group $G_1 \times C_4$. □

The proof of this theorem can also be found in [5].

Below are a few examples of the theorem being used for more difficult and higher ordered groups, including some groups that have order 256.

**Example 29**: Consider the group that was given in Example 25, where there was the group $C_4 \rtimes_{-1} C_4 =< x, y : x^4 = y^4 = 1, yx = xy^3 >$. It had the SDP difference set $D = x + x^3 + y + y^3 + xy + x^3y^3$. By Theorem 15, $\underline{D}(-1 + x + x^2 + x^3)$ will be an SDP difference set of $(C_4 \rtimes_{-1} C_4) \times C_4$.

**Example 30**: Consider a SDP difference set $D_1 \subset C_8 \times C_4 \times C_2$. By Theoreom 15, $\underline{D_1}(-1 + x + x^2 + x^3)$ is an SDP difference set in the group $C_8 \times C_4 \times C_4 \times C_2$.

Theorem 14 tells us that designs from difference sets of order 256 will have the SDP if the rank of code is equal to 10.

Therefore, this theorem can help give $(256, 120, 56)$ difference sets in groups of order 256. This theorem can also be used to find $(2^{2m+4}, 2^{2m+3} - 2^{m+1}, 2^{2m+2} - 2^{m+1})$ SDP difference sets. As a result, $(2^{2m+4}, 2^{2m+3} - 2^{m+1}, 2^{2m+2} - 2^{m+1})$ SDP designs can be found. However, the theorem only can only do this for groups of the form $G \times C_4$ and does not account for groups that are not direct products.

**Motivation and Climax**

While Theorem 15 is useful, it does not give us a lot of new information in regards to the existence of SDP designs that come from groups of order $2^{2m+4}$. There are two reasons for this: the first reason is that the theorem needs special prerequisites in order to work, which includes the existence of a $(2^{2m+2}, 2^{2m+1} - 2^m, 2^{2m} - 2^m)$ SDP difference set in a group of order $2^{2m+2}$. It is easy to see that not all difference sets fit this condition. The second reason is that this theorem only applies to difference sets of groups of the form $G \times C_4$, where $G$ is a group. Thus, this theorem can only look at groups of the form $G \times C_4$, which cannot account for all groups. Therefore, in order to identify non-isomorphic SDP designs, other types of groups must be looked at, such as semi-direct products.

The reason semi-direct are of interest is because in some cases, a semi-direct product of two SDP difference sets produces another SDP difference set. However, this is not always true and can be seen when analyzing semi-direct products.

Regardless, semi-direct products gives us more designs and difference sets to look in order to determine the number of non-isomorphic SDP designs with parameters $(v,k,\lambda)$.

There is one SDP design on 16 points.

There are four non-isomorphic SDP designs on 64 points. These may be found in [15]. All four can be constructed from difference sets in groups of order 64. All four can be constructed as a semi-direct product of a normal subgroup of order 16 and $C_4$, using the SDP design on 16 points.

By Theorem 15, there are four new $(256,120,56)$ SDP designs that are given by the direct product of one of these semi-direct products of order 64 and $C_4$.

We also seek non-isomorphic $(256,120,56)$ SDP designs in semi-direct products of two groups of order 16. The reason the semi-direct products of two groups with order 16 is of main interest is because this is a very large pool of groups that can yield significant results. This is also of interest because this can be used as a starting point for future research on other groups that give SDP designs with parameters $(256,120,56)$.

Therefore, using the information that has been discussed up to this point, a major result was established that answers this question.

## Difference sets in larger groups

Now, the major result of this research will be illustrated.

By using the GAP Program, six non-isomorphic semi-direct products of groups of order 16 were found. The search was exhaustive over all possible groups of order 256 that are semi-direct products of two groups of order 16. Given such a group, the search explored all possible semi-direct products. The order that the products are given (from top to bottom) is the order they appeared in the program. The action of righthand group onto the normal subgroup on the left is by automorphisms. Thus, there are six main designs with $v = 256$ that come from semi-direct products. The first and third semi-direct products listed give entirely new designs. The other four SDP designs that the other four semi-direct products give can also be formed by taking a direct product from an SDP difference set in a group of order 64 (given in [15]) by $C_4$. The groups that gave these designs are the following (written as semi-direct products and in terms of the GAP Programming):

$$(C_4 \times C_2 \times C_2) \rtimes (C_4 \times C_4) \text{ (SmallGroup(256, 542))}$$

$$(C_4 \times C_2 \times C_2) \rtimes (C_4 \rtimes_{-1} C_4) \text{ (SmallGroup(256, 545))}$$

$$(C_4 \times C_4) \rtimes (C_8 \times C_2) \text{ (SmallGroup(256, 841))}$$

$$(C_4 \times C_4) \rtimes (C_8 \times C_2) \text{ (SmallGroup(256, 853))}$$

$$(C_4 \times C_4) \rtimes (C_8 \times C_2) \text{ (SmallGroup(256, 853))}$$

$$(C_4 \times C_4) \rtimes ((C_4 \times C_2) \rtimes C_2) \text{ (SmallGroup(256, 3704)).}$$

The difference set for each of the groups is shown below. The difference sets are written in terms of GAP Programming. This means that each element in these difference sets below is a number that is stored in GAP and used to represent a specific element in the group.

Difference Set of SmallGroup(256, 542): [12, 39, 44, 51, 102, 189, 13, 40, 45, 54, 105, 192, 14, 41, 46, 56, 107, 193, 28, 64, 74, 85, 141, 216, 29, 65, 75, 87, 143, 217, 32, 68, 78, 90, 146, 218, 49, 99, 109, 120, 176, 237, 50, 100, 110, 122, 178, 238, 53, 103, 113, 125, 181, 239, 84, 139, 149, 159, 210, 247, 9, 17, 21, 26, 27, 37, 62, 63, 73, 138, 16, 38, 42, 47, 48, 58, 97, 98, 108, 173, 31, 59, 66, 76, 77, 89, 131, 132, 144, 204, 34, 60, 69, 79, 80, 92, 134, 135, 147, 207, 36, 61, 71, 81, 82, 93, 136, 137, 148, 208, 195, 220, 230, 235, 236, 240, 248, 249, 253, 256].

Difference Set of SmallGroup(256, 545): [ 129, 38, 18, 59, 97, 140, 206, 224, 72, 137, 251, 213, 61, 220, 68, 133, 248, 71, 88, 123, 153, 160, 197, 247, 84, 119, 149, 29, 194, 152, 34, 55, 80, 9, 127, 83, 199, 96, 65, 130, 171, 209, 203, 227, 210, 242, 252, 212, 17, 164, 19, 60, 221, 21, 28, 49, 74, 5, 120, 76, 8, 11, 13, 24, 26, 33, 45, 47, 54, 114, 207, 176, 222, 70, 73, 138, 102, 232, 250, 179, 89, 112, 121, 151, 154, 161, 186, 189, 196, 237, 35, 46, 53, 78, 81, 90, 113, 116, 125, 190, 163, 191, 57, 82, 218, 93, 117, 239, 128, 193, 204, 43, 98, 67, 211, 243, 105, 108, 173, 182].

Difference Set of SmallGroup(256, 841): [8, 15, 26, 33, 58, 157, 18, 42, 59, 64, 108, 201, 20, 177, 61, 68, 233, 206, 21, 101, 62, 69, 179, 207, 29, 122, 75, 84, 197, 215, 35, 14, 81, 90, 57, 218, 65, 41, 130, 139, 107, 242, 66, 10, 131, 140, 43, 243, 87, 50, 152, 159, 123, 247, 142, 106, 202, 209, 183, 255, 9, 11, 13, 24, 27, 34, 45, 48, 55, 115, 22, 94, 99, 60, 63, 70, 164, 167, 176, 222, 31, 44, 49, 74, 77, 86, 109, 112, 121, 186, 36, 116, 125, 78, 82, 91, 190, 193, 198, 239, 89, 111, 120, 150, 154, 161, 185, 189, 196, 237, 148, 165, 174, 205, 208, 213, 220, 224, 231, 249].

Difference Set of SmallGroup(256, 853): [8, 15, 26, 33, 58, 157, 18, 10, 59, 64, 43, 201, 20, 177, 61, 68, 233, 206, 29, 50, 75, 84, 123, 215, 30, 12, 76, 85, 52, 216, 35, 14, 81, 90, 57, 218, 65, 106, 130, 139, 183, 242, 66, 42, 131, 140, 108, 243, 71, 100, 136, 145, 178, 245, 142, 41, 202, 209, 107, 255, 9, 11, 13, 24, 27, 34, 45, 48, 55, 115, 22, 94, 99, 60, 63, 70, 164, 167, 176, 222, 31, 111, 120, 74, 77, 86, 185, 189, 196, 237, 36, 116, 125, 78, 82, 91, 190, 193, 198, 239, 73, 166, 175, 134, 138, 147, 221, 225, 232, 250, 162, 187, 194, 214, 217, 219, 235, 238, 240, 254].

Difference Set of SmallGroup(256, 853): [8, 15, 26, 33, 58, 157, 18, 10, 59, 64, 43, 201, 20, 177, 61, 68, 233, 206, 21, 101, 62, 69, 179, 207, 29, 50, 75, 84, 123, 215, 35, 14, 81, 90, 57, 218, 65, 106, 130, 139, 183, 242, 66, 42, 131, 140, 108, 243, 87, 122, 152, 159, 197, 247, 142, 41, 202, 209, 107, 255, 9, 11, 13, 24, 27, 34, 45, 48, 55, 115, 22, 94, 99, 60, 63, 70, 164, 167, 176, 222, 31, 111, 120, 74, 77, 86, 185, 189, 196, 237, 36, 116, 125, 78, 82, 91, 190, 193, 198, 239, 89, 44, 49, 150, 154, 161, 109, 112, 121, 186, 148, 165, 174, 205, 208, 213, 220, 224, 231, 249].

Difference Set of SmallGroup(256, 3704): [9, 16, 27, 34, 82, 126, 18, 43, 59, 64, 130, 181, 21, 179, 62, 69, 136, 253, 22, 102, 63, 70, 137, 231, 30, 124, 76, 85, 152, 240, 37, 15, 83, 92, 158, 125, 66, 42, 131, 140, 202, 180, 67, 10, 132, 141, 203, 103, 89, 51, 154, 161, 217, 194, 144, 108, 204, 211, 244, 234, 7, 11, 13, 14, 24, 32, 45, 46, 78, 113, 20, 94, 99, 100, 60, 68, 164, 165, 133, 220, 29, 44, 49, 50, 74, 84, 109, 110, 149, 184, 35, 118, 127, 128, 79, 90, 192, 193, 155, 239, 88, 112, 121, 123, 151, 160, 186, 188, 215, 236, 148, 166, 175, 177, 207, 213, 221, 223, 245, 248 ].

This is the main result in our research. This was obtained with the aid of three GAP programs: 1) Programs that Creates Semi-Direct Products using $(16,6,2)$ Difference Sets, 2) Program that Analyzes Difference Sets where $v = 256$ and 3) Program that gives the Non-Isomorphic SDP designs where $v = 256$. The first program took 30 hours to run and give all semi-direct products of groups of order 16 that had difference sets. The second program took 6 hours to run and found all semi-direct products that had the SDP. The last program took two weeks to run and determined which of the SDP semi-direct products given in the last program are non-isomorphic. The code for each of these programs is shown in the Appendix.

This illustrates that there are only six non-isomorphic designs that come from semi-direct products of two groups of order 16. This also eliminates a very large pool of groups that can be looked at for the existence of SDP designs.

## Future research

Finally, this section illustrates some open questions that could be looked into for future research purposes and projects. The questions are shown below:

1. What are all of the non-isomorphic designs (where $v = 1024$) could we find using semi-direct products (using a similar GAP code that we used to find the six non-isomorphic designs given above)?

2. How many different SDP designs occur as difference sets in groups of order 256? (Note: SDP designs on $2^{2n}$ points correspond to bent functions on variables. See Dillon and Schatz text, [6]. The thesis of Bending gives all bent functions on 8 variables. [2]).

3. Can we find SDP designs from groups $H_{64} \rtimes C_4$, where is a normal subgroup of order 64 that contains a Hadamard difference set?

4. Is there a special structure for designs with a rank $2n+3$ (which can be called "almost SDP" designs?)

## REFERENCES

[1] T. Applebaum, J. Clikeman, J. Davis, J. Dillon, J. Jedwab, T. Rabbani, K. Smith, and W. Yolland, *Constructions of difference sets in nonabelian 2-groups*, March 2020, pp. 1–8.

[2] T. D. Bending, *Bent functions, sdp designs and their automorphism groups*, Ph.D. thesis, Queen Mary and Westfield College, 11 1993.

[3] C. Bhattacharya and K. Smith, *Factoring (16,6,2) difference sets*, Electron J. Comb. **15** (2008), 1–16.

[4] C. J. Colbourn and J. H. Dinitz, *Handbook of combinatorial designs*, second ed., Chapman & Hall CRC Press; Taylor & Francis Group, 2007.

[5] J. A. Davis, J. J. Hoo, C. Kissane, Z. Liu, C. Reedy, K. Sharma, K. Smith, and Y. Sun, *Abelian difference sets with the symmetric difference property*, Des. Codes Cryptogr. **89** (2021), no. 3, 517–523. MR 4220826

[6] J. F. Dillon and J. R. Schatz, *Block designs with the symmetric difference property*.

[7] J. F. Dillon, personal communication via Ken Smith, July 2021.

[8] D. S. Dummit and R. M. Foote, *Abstract algebra*, Wiley & Sons, Inc., Hoboken, NJ, 2004. MR 1138725

[9] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.11.1*, 2021.

[10] M. Hall, Jr., *Combinatorial theory*, second ed., Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons, Inc., New York, 1986, A Wiley-Interscience Publication. MR 840216

[11] D. Jungnickel and V. D. Tonchev, *On symmetric and quasi-symmetric designs with*

*the symmetric difference property and their codes*, Journal of Combinatorial Theory (1992), 40–50. MR 1129323

[12] W. M. Kantor, *Symplectic groups, symmetric designs, and line ovals*, J. Algebra **33** (1975), 43–58. MR 363934

[13] ______, *Exponential numbers of two-weight codes, difference sets and symmetric designs*, Discrete Math. **46** (1983), no. 1, 95–98. MR 708168

[14] P. J. Olver and Shakiban, *Applied linear algebra*, Springer International Publilshing AG., 2006. MR 1138725

[15] C. Parker, E. Spence, and V. D. Tonchev, *Designs with the symmetric difference property on* 64 *points and their groups*, J. Combin. Theory Ser. A **67** (1994), no. 1, 23–43. MR 1280597

[16] J. H. van Lint and R. M. Wilson, *A course in combinatorics*, second ed., Cambridge University Press, Cambridge, 2001. MR 1871828

## APPENDIX

All of these codes were written by the supervisor, Dr. Kenneth W. Smith. The codes that listed in this Appendix are the following:

**Three Procedures Program**

**Rank of Matrix Program**

**Generalized Product Procedures**

**Difference Set Procedure**

**Programs that Creates Semi-Direct Products using (16,6,2) Difference Sets**

**Program that Analyzes Difference Sets where v=256**

**Program that gives the Non-Isomorphic SDP designs where v=256**

These codes are essential to this thesis because they were used to verify or check the examples used. They also helped establish the major result of this research, specifically the last five programs that are listed above.

**Three Procedures Program:** This is a separate code that is used to find the rank of an incidence matrix. The first part of the code shown below takes a design (a set of sets) and creates an isomorphic design on the point set [1..v]. The second part of the code here reads in a group and a subset 'sublist' of the group and creates the collection of all images of 'sublist' under the action LEFT multiplication. The third and final part of the code here takes a design (a set of sets) on [1..v] and creates an incidence matrix. This function works on designs that incorporate integers or other symbols.

```
PointsToInterval := function( design )

local blocks, v, k, points, intDesign, i, j, s;

   blocks := Size( design );

   points := Set( Union( design ) );
   v      := Size( points );


   intDesign := [ ];
   for i in [1..blocks] do
      Add(intDesign, ShallowCopy( design[i] ) );
   od;


   for i in [1..blocks] do
      k := Size( design[i] );
      for j in [1..k] do
         for s in [1..v] do
            if design[i][j] = points[s] then
               intDesign[i][j] := s;  fi;
         od;
      od;
   od;
   return intDesign;
end;
```

The second part of the code here.

```
LeftTranslates_Nums := function( group, sublist )
local k, elements, g, design, i, block;
   k        := Size( sublist );
   elements := Elements( group );
   design   := [ ];


   for g in elements do

      block := [ ];


      for i in [ 1..k ] do

         Add( block,  g*elements[sublist[i]] );

      od;


      Add( design, block );

   od;


   return design;

end;
```

The third and final part of the code here.

```
IncidenceMatrix := function( design )

local blocks, v, k, incMat, intDesign, i, j;

   intDesign := PointsToInterval(design); #PointsToInterval comes from the

   first line in first part of code.

   blocks    := Size( intDesign );

   v         := Size( Set( Flat( intDesign ) ) );

   k         := Size( intDesign[1] );
   incMat := NullMat(blocks,v);
   for i in [1..blocks] do
      for j in [1..k] do
         incMat[i][intDesign[i][j]]:=1;
      od;
   od;
   return incMat;
end;
```

**Rank of Matrix Program:** This is a program that is used to directly find the rank of an incidence matrix obtained from the previous code above (Three Procedures Program).

```
lt:=LeftTranslates_Nums(g, gap_difset); #LeftTranslates_Num comes from
the first line of the second part of the Three Procedures code.
design:=PointsToInterval( lt ) ; #PointsToInterval comes from the first
line in first part of Three Procedures code.
mat:=IncidenceMatrix( design ) ; #IncidenceMatrix comes from the first
line in the third part of Three Procedures code.
snf:=Collected(DiagonalOfMat(SmithNormalFormIntegerMat(mat)));
```

**Generalized Product Procedures:** This code uses a number of procedures to find difference sets in groups via a product construction. The first procedures takes a group and returning a set of lists. Second procedure takes a group and a list of PTAs (Perfect Ternary Array or Hadamard difference set in group ring notation) of length 4 and combines them to create products of size 16. Third procedure takes a group and a list of pairs of PTAs of size 4 that gave a product of size 16 and cleans the list up. Fourth procedure takes a group and runs through the list of sets of size 16 until it finds a pair so that the group is the product of this pair of lists. Fifth procedure takes a group and a list of sets of size 4 that form a PTA of size 4 and hunts for Hadamard subgroups *h* of size 64 so that the group is the product of *h* and that list. Sixth procedure takes a group and a set of size sixteen equal to the support of the product of two PTAs of size 4. The last procedure takes a group and two PTA's consisting of indices in the group where the lists have coefficients $\pm 1$. Then, it multiplies the two elements in the group ring.

```
#################################################################
#   This procedure takes a group and returns a set of lists.
# Each list has size four, of the form [1, i1, i2, i3 ] where
# the elements 'identity', e[i1], e[i2] and e[i3] form the support
# of a PTA of length 4.
#################################################################
PTA4 := function( group )
local e, foursets, v, i1, i2;
   e := Elements(group);
   v := Size(e);
   foursets := [ ];
   for i1 in [2..v] do for i2 in [i1+1..v] do
     if Order(e[i1])= 2 and e[i1]*e[i2]=e[i2]*e[i1] then Add(foursets, [1,
     i1, i2, Position(e, e[i1]*e[i2])]); fi;
     if Order(e[i1])= 4 and Order(e[i2])= 4 then
```

```
        if IdGroup(Group(e[i1],e[i2]))=[8,4] then Add(foursets, [1, i1, i2,
        Position(e, e[i1]*e[i2])]); fi;
      fi;
    od; od;
    foursets := Set(foursets);
    return foursets;
end;
#################################################################
#   This procedure takes a group and a list of PTAs of length 4
# and combines them to create products of size 16. It returns
# a list of lists; each list is a pair  [i1, i2, [pta4[i1], pta4[i2]]].
#################################################################
PTA16 := function( group, pta4 )
local e, v, sizepta4, i1, i2, pairsofpta4;
    e := Elements(group);
    v := Size(e);
    sizepta4 := Size( pta4 );
    pairsofpta4 := [ ];
    for i1 in [1..sizepta4] do for i2 in [i1+1..sizepta4] do
      if Size(Intersection(pta4[i1], pta4[i2])) = 1 then
        Add( pairsofpta4 , [i1, i2, [pta4[i1], pta4[i2]]]);
      fi;
    od; od;
```

```
   return pairsofpta4;
end;
#################################################################
#   This procedure takes a group and a list of pairs of PTAs of
# size 4 that gave a product of size 16 and then cleans up this
# list in two ways" (1) making sure that the list of size 16 has
# no repetitions and so is a genuine PTA of length 16 and (2)
# removing duplicates from the final list of genuine PTA-16s.
# The output should then be a list of lists, each list being
# a set of size 16 that is the support for a genuine PTA 16.
#################################################################
CleanUpPTA16 := function( group, pairsofpta4  )
local e, v, cleansixteensets, sizepta16, index,pta16, list1, list2,
finallist, i1, i2;
   e := Elements(group);
   v := Size(e);
   cleansixteensets := [ ];
   sizepta16 := Size( pairsofpta4 );
   for index in [1..sizepta16] do
     pta16 := pairsofpta4[index];
     list1 := pta16[3][1];
     list2 := pta16[3][2];
     finallist := [1, list1[2], list1[3], list1[4], list2[2],
     list2[3], list2[4]];

     for i1 in [2..4] do for i2 in [2..4] do

       Add(finallist, Position(e, e[list1[i1]]*e[list2[i2]]));

     od; od;

    finallist := Set(finallist);

     if Size(finallist) = 16 then Add(cleansixteensets, finallist); fi;

   od;

   cleansixteensets := Set(cleansixteensets); # There could be duplicate,

   different products that give the same 16-set.

   return cleansixteensets;

end;

#################################################################

#   This procedure takes a group and a list of sets of size 16 and

# hunts for a pair, 'list1' and 'list2' whose supports factor the

# group g, so that g = list1*list2

#################################################################

GeneralizedProduct256 := function( group, cleansixteensets )

local e, solution, sizepta16, min, max, breakflag, i1, i2, list1, list2,

inter, product, j1, j2;

  e := Elements(group);

  solution := [ ];

  sizepta16 := Size(cleansixteensets);

  min := 16; max := 0; breakflag := 0;
```

```
   for i1 in [1..sizepta16] do for i2 in [1..sizepta16] do

     list1 := cleansixteensets[i1]; list2 := cleansixteensets[i2];

     inter :=Size(Intersection(list1, list2));

     if inter < min then min:=inter;

     Print("Minimum of ", min," at ",i1," & ", i2,"\n"); fi;

     if inter = 1 then

##########

# Here we check that the product does not collapse:

       product := [ ];

       for j1 in [1..Size(list1)] do for j2 in [1..Size(list2)] do

         Add( product, ( Position( e, e[list1[j1]]*e[list2[j2]] ) ) );

       od; od;

       product := Set(product);

##########

       if Size(product) > max then max := Size(product);

       Print("Found product of size ", max,".\n"); fi;

       if Size(product) = 256 then

         Print("   *** Solution!\n");

         Add(solution, [[i1,i2],[list1, list2]]); breakflag := 1; break;

       fi;

     fi;

   od;

 if breakflag = 1 then break; fi;
```

```
  od;
  return solution;
end;
#################################################################
#   This procedure takes a group identified by a catalogue number 'cn'
# and a list of sets of size 4 that would form a PTA4 and
# hunts for Hadamard subgroups h of size 64 so that
# group g, so that g = h*list.
#    This procedure takes a fraction of a second up to 10 seconds or more
per group
# depending on the number of subgroups of the group.
# It worked through 3780 groups (left to check from the earlier product
# constructions) in fifteen hours.
#################################################################
GeneralizedProduct_H64xT:= function( cn, pta4 )
 local solution, breakflag, intersectionsize, forbidden_64, g, e, sub,
 size, max, i1, h1, eh, x, fourset, product, j1, j2;
 solution := [ ];
 breakflag := 0;
 forbidden_64 := [1, 38, 47, 50, 52, 53, 54, 186]; # These are catalogue
 numbers of groups of order 64 without difference sets
 g:=SmallGroup(256, cn); e :=Elements(g);
 sub := AllSubgroups(g); sub:=SortedList(sub);
```

```
  size := Size(sub); # We construct group and subgroups
   breakflag := 0;
  for i1 in [2..size] do # We hunt for a good group of order 64.
    h1 := sub[i1];
    if Size(h1) = 64 then
    max := 99;
      if not (IdGroup(h1)[2] in forbidden_64) then # This group is good!
# Print("Good group found! ", IdGroup(h1)[2],"\n");
        eh := [ ]; for x in Elements(h1) do Add(eh, Position(e, x)); od;
         for fourset in pta4 do
intersectionsize :=Size(Intersection( eh, fourset ));
# if intersectionsize < max then max := intersectionsize; Print(" ",max,"\n");fi;
           if intersectionsize = 1 then # This looks promising
#Print("Good intersection size at ", fourset,"\n");
##########
# Here we check that the product does not collapse:
             product := [ ]; for j1 in [1..64] do for j2 in [1..4] do
               Add( product, ( Position( e, e[eh[j1]]*e[fourset[j2]] ) ) );
             od; od;
             product := Set(product);
##########
#            if Size(product) > max then max := Size(product);
Print("Found product of size ", max,".\n"); fi;
```

```
             if Size(product) = 256 then
               Print("   *** Solution! ", [i1, fourset], " ", IdGroup(h1),"\n");
# Replace               Add(solution, [i1,fourset]);
solution := [i1, fourset ];
               breakflag := 1; break;
             fi; # if Size(product) ...
            fi; # if Size(Intersection...
if breakflag = 1 then break; fi;
          od; # for fourset in ...
        fi; # if not...
      fi; # if Size(h1) = ...
if breakflag = 1 then break; fi;
    od; # for i1 in ...
  return solution;
 end;
 #################################################################
 #################################################################
 # GeneralizedProduct_H16xT1xT2( cn, cleansixteensets );
 #################################################################
 #   This procedure takes a group identified by a catalogue number 'cn'
 # and a list of sets of size 4 that would form a PTA4 and
 # hunts for Hadamard subgroups h of size 64 so that
 # group g, so that g = h*list
 # This procedure takes a fraction of a second up to 10 seconds or more per group
 # depending on the number of subgroups of the group.
 # It worked through 875 groups (left to check from the earlier product
 # constructions) in ?? hours.
 #################################################################
 GeneralizedProduct_H16xT1xT2:= function( cn, cleansixteensets )
  local solution, breakflag, pta16, intersectionsize, forbidden_16, forbidden_64,


  solution := [ ];
  breakflag := 0;
  forbidden_16 := [1, 7]; # These are catalogue numbers of groups of order
  16 without difference sets
  g:=SmallGroup(256, cn); e :=Elements(g);
  sub := AllSubgroups(g); sub := SortedList(sub);
  size := Size(sub); # We construct group and subgroups
   breakflag := 0;
  for i1 in [2..size] do # We hunt for a good group of order 16.
    h1 := sub[i1];
    if Size(h1) = 16 then
    max := 99;
      if not (IdGroup(h1)[2] in forbidden_16) then # This group is good!
# Print("Good group found! ", IdGroup(h1)[2],"\n");
        eh := [ ]; for x in Elements(h1) do Add(eh, Position(e, x)); od;
```

```
         for pta16 in cleansixteensets do
intersectionsize :=Size(Intersection( eh, pta16 ));
 if intersectionsize < max then max := intersectionsize;
# Print(" ",max,"\n");
fi;
           if intersectionsize = 1 then # This looks promising
 Print("Good intersection size at ", pta16,"\n");
##########
# Here we check that the product does not collapse:
             product := [ ]; for j1 in [1..16] do for j2 in [1..16] do
               Add( product, ( Position( e, e[eh[j1]]*e[pta16[j2]] ) ) );
             od; od;
             product := Set(product);
##########
#            if Size(product) > max then max := Size(product);
Print("Found product of size ", max,".\n"); fi;
             if Size(product) = 256 then
               Print("   *** Solution! ", [i1, pta16 ], " ", IdGroup(h1),"\n");
#               Add(solution, [i1,pta16]);
               solution := [i1,pta16];
               breakflag := 1;
               break;
             fi; # if Size(product) ...
```

```
            fi; # if Size(Intersection...
if breakflag = 1 then break; fi;
          od; # for fourset in ...
        fi; # if not...
      fi; # if Size(h1) = ...
if breakflag = 1 then break; fi;
    od; # for i1 in ...
  return solution;
 end;
 #################################################################
 #################################################################
 # ReverseSixteen( groupid, sixteenset );
 #################################################################
 #   This procedure takes a group identified by a catalogue number 'cn'
 # and a set of size sixteen equal to the support of the product
 # of two PTA4s.  It then recovers the (not necessarily unique)
 # PTA4s used to create the product.
 #################################################################
 ReverseSixteen := function( groupid , sixteenset )
  local v, cn ,g, e, foursets, size, breakflag, i1, i2, j1, j2, list1,
  list2, finallist,
    solution, dsneg, dspos;
    v := groupid[1]; cn := groupid[2]; g:=SmallGroup(v,cn); e:=Elements(g);
```

```
foursets := PTA4( g );

size := Size(foursets);

breakflag := 0;

for i1 in [1..size] do

  for i2 in [i1+1..size] do

    list1 := foursets[i1]; list2 := foursets[i2];

    finallist := [1, list1[2], list1[3], list1[4], list2[2], list2[3],

    list2[4]];

    for j1 in [2..4] do for j2 in [2..4] do

      Add(finallist, Position(e, e[list1[j1]]*e[list2[j2]]));

    od; od;

    finallist := Set(finallist);

    if finallist = sixteenset then

      solution := [i1, i2];

      dsneg := [list1[2], list1[3], list1[4], list2[2], list2[3], list2[4]];

      Sort(dsneg);

      dspos := finallist; SubtractSet(dspos, dsneg);

      breakflag := 1;

      break;

    fi;

  od;

  if breakflag=1 then break; fi;

od;
```

```
    return [solution, [dspos,dsneg]];
  end;


# sixteenset :=  [ 1, 2, 5, 6, 12, 13, 26, 28, 47, 49, 76, 79, 111, 114,
150, 185 ];
# groupid := [256, 323];
# rev := ReverseSixteen( groupid, sixteenset );


 #################################################################
 #   This procedure takes a group identified by a 'GroupId' (of the
 # form [v, cn]) and two PTAs, each given by a list [ pos1, neg1 ] or
 # [ pos2, neg2 ] consisting of indices in the group where pta1 or pta2
 # has coefficient +1 and indices where the pta has coefficient -1.
 #   The procedure then multiplies the two elements in the group
 # ring and returns the results in the form [ newpos, newneg ].
 #   At this time there is NO error checking!
 #################################################################
 ProductPTAs := function( groupid, pta1, pta2 )
  local v, cn, g, e, pos1, pos2, neg1, neg2, newpos, newneg, index1, index2;
   v := groupid[1]; cn := groupid[2]; g:=SmallGroup(v,cn); e:=Elements(g);
    pos1 := pta1[1]; neg1 := pta1[2];
    pos2 := pta2[1]; neg2 := pta2[2];
    newpos := [ ]; newneg := [ ];
   for index1 in  pos1 do
     for index2 in pos2 do
       Add(newpos, Position(e, e[index1]*e[index2]));
     od;
     for index2 in neg2 do
       Add(newneg, Position(e, e[index1]*e[index2]));
     od;
   od;
   for index1 in  neg1 do
     for index2 in pos2 do
       Add(newneg, Position(e, e[index1]*e[index2]));
     od;
     for index2 in neg2 do
       Add(newpos, Position(e, e[index1]*e[index2]));
     od;
   od;
   return [newpos,newneg];
 end;
```

**Difference Set Procedure:** This code contains a large collection of procedures. This code is used in later programs listed in the Appendix. The first and second procedures takes a group $G$, a sorted list of elements from $G$ and a function of those elements to give a $(v,k,\lambda)$ design. Third procedure uses the same group, a sorted list of elements of the group, and a subgroup to find the coset representatives of the subgroup. Fourth procedure turns an integer base into a list. Fifth procedure turns a list of positive integers into one integer. The next six procedures are used to creates an incidence matrix of a design and finds its rank. The twelve procedure takes the incidence matrix of the design and creates a list of triple intersections involving the first row. The thirteenth procedure takes the incidence matrix of a design and creates a list of triple intersections and quadruple intersections involving the first row. The fourth procedure finds all the hyperplanes of the group. The fifteenth procedure finds a new difference set using hyperplane construction. The sixteenth procedure creates a new incidence matrix of the new difference set by switching the entries of the old incidence matrix. The seventeenth procedure creates a list quadruple intersections formed from the rows of the new incidence matrix. The eighteenth procedure finds the design obtained from new difference set. The nineteenth procedure attempts to takes a subgroup acting transitively on a set $[1..2^m]$ and find all 2-groups acting regularly on the set.

```
################################################################
 # 1. ConvolutionTable_f( group, elements, func )               #
 ################################################################
 #   This procedure takes a group, a sorted list of the elements
 # of the group and 'func', a function previously defined on
 # the elements of the group.
 #   It returns a table whose (i,j) entry is the
 # positive integer k such that e[i] * f( e[j] ) = e[k].
 #   This is especially useful for difference sets if
 # the function 'f' was defined by
 #                      f:= x-> x^-1;
 # Example:

 # > g:=SmallGroup(4,1); e:= Elements(g);

 # > CL := ConvolutionTable_f(g, e, x->x);

 # [ [ 1, 2, 3, 4 ], [ 2, 3, 4, 1 ], [ 3, 4, 1, 2 ], [ 4, 1, 2, 3 ] ]

 #

 # > CL_inv := ConvolutionTable_f(g, e, x->x^-1);

 # [ [ 1, 4, 3, 2 ], [ 2, 1, 4, 3 ], [ 3, 2, 1, 4 ], [ 4, 3, 2, 1 ] ]

 #################################################################

 ConvolutionTable_f := function( group, elements, func )

 local order, CL, i0, i1, i2;

    order := Size( group );

    CL    := [ ];

    for i0 in [1..order] do CL[i0] := [ ]; od;

    for i0 in [1..order] do

       for i1 in [1..order] do

          for i2 in [1..order] do

             if elements[i1]*func(elements[i2]) = elements[i0] then

                CL[i1][i2] := i0;  fi;

          od;

       od;

    od;

    return CL;

 end;
```

```
#################################################################
# Example:
# > g:=SmallGroup(4,1); e:= Elements(g); CL := ConvolutionTable_f(g, e, x->x);
# [ [ 1, 2, 3, 4 ], [ 2, 3, 4, 1 ], [ 3, 4, 1, 2 ], [ 4, 1, 2, 3 ] ]
#
# > CL_inv := ConvolutionTable_f(g, e, x->x^-1);
# [ [ 1, 4, 3, 2 ], [ 2, 1, 4, 3 ], [ 3, 2, 1, 4 ], [ 4, 3, 2, 1 ] ]
#################################################################
# 2. Convolution( MT, list1, list2 )                          #
#################################################################
#   This procedure takes table MT (created, say, by
# ConvolutionTable_f( g, e, f ) and two lists, list1 and list2,
# (Members of the integer group ring Zg here are represented as
# a list of coefficients, so for example, the set e[1]+2*e[3]
# in Z(C_5) will appear as [1,0,2,0,0].)
#   It then uses the convolution list MT to quickly compute
# list1*list2 in Zg and returns that answer as a list.
#################################################################
Convolution := function( MT, list1, list2 )
local sizeMT, conprod, i1, i2, i3;
   sizeMT  := Size( MT );
   conprod := ListWithIdenticalEntries( sizeMT, 0 ); # This will be
                                                     # the final answer.
```

```
    for i1 in [1..sizeMT] do for i2 in [1..sizeMT] do

      i3 := MT[i1][i2];

      conprod[i3] := conprod[i3]+list1[i1]*list2[i2];

    od; od;


   return conprod;

end;

#################################################################

# > g:=SmallGroup(4,1); e:= Elements(g);

# > CL := ConvolutionTable_f(g, e, x->x);

# [ [ 1, 2, 3, 4 ], [ 2, 3, 4, 1 ], [ 3, 4, 1, 2 ], [ 4, 1, 2, 3 ] ]

#################################################################

# 3. LeftTransversalList( group, elements, subgroup )           #

#################################################################

#   This procedure takes a group, a sorted list of elements

# of the group, a subgroup, and returns a list of left coset

# representatives of the  subgroup in the group.  The coset representatives

# are positive integers; if the integer i0 is listed then e[i0]

# is in that coset.

#################################################################

LeftTransversalList := function( group, elements, subgroup )

local rt, lt_list, t;

  rt := RightTransversal( group, subgroup);
  lt_list := [ ];
  for t in rt do Add(lt_list, Position(elements,t^-1)); od;
  return lt_list;
end;
#################################################################
# Example:
# > g:=SmallGroup(8,3); e:= Elements(g); h:=Group(e[2]);
# lt := LeftTransversalList(g,e,h);
#[ 1, 3, 4, 7 ]
#################################################################
# 3B. LeftCosetList( group, elements, subgroup )           #
#################################################################
#   This procedure takes a group, a sorted list of elements
# of the group, a subgroup, and returns a list of left cosets of
# the  subgroup in the group.  The elements of the cosets
# are positive integers; if the integer i0 is listed then e[i0]
# is in that coset.
#################################################################
LeftCosetList := function( group, elements, subgroup )
local lt, leftcosets, j, coset, x;
  lt := LeftTransversalList( group, elements, subgroup );
  leftcosets := [ ];
  for j in lt do
```

```
    coset := [ ];
    for x in subgroup do
      Add(coset, Position(elements, elements[j]*x));
    od;
    Add(leftcosets, coset);
  od;
  return leftcosets;
end;
#################################################################
# Example:
# > g:=SmallGroup(8,3); e:= Elements(g); h:=Group(e[2]);
# lt := LeftTransversalList(g,e,h);
#[ 1, 3, 4, 7 ]
#################################################################
#################################################################
# The procedure 'ListToInteger' turns a positive integer into a
# list of a certain length.
#    Using base 3, the integer 14 becomes [1,1,2] with length 3
# or [0,0,0,0,0,1,1,2] with length 8.
# The procedure 'IntegerToList' reverses that process.
#################################################################
#  4. IntegerToList( intIn, length, base )                        #
#################################################################
```

```
# This procedure turns an integer base 'base' into a list
# Convert decimal integer to list
#  'intIn' is the integer we read in;
#  'base' is the base for the computation.
#  'length' is the length of the output vector
#################################################################
IntegerToList := function( intIn, length, base )
local vector, i, minLength, quotient, remainder;
   minLength := LogInt( intIn, base ) + 1;
   if minLength > length then minLength := length; fi;
   vector := ListWithIdenticalEntries( length, 0 );
   for i in [1 .. minLength] do
      quotient  := QuoInt( intIn, base );
      remainder := RemInt( intIn, base );
      intIn     := quotient;
      vector[length + 1 - i] := remainder;
   od;
   return  vector;
end;
#################################################################
#  5. ListToInteger( vector, base )                             #
#################################################################
#   This procedure turns a string (as a list) of positive integers
# into an integer.  This undoes the IntegerToList function.
#################################################################
ListToInteger := function( vector, base )
local int_val, i, length;
   int_val := 0;
   length  := Size( vector );
   for i in [1 .. length ] do
      int_val := int_val + vector[i] * base ^ ( length - i );
   od;
   return int_val;
end;
#################################################################
# Example:
# > IntegerToList( 14, 4, 3 );
#[ 0, 1, 1, 2 ]
# IntegerToList( 6, 3, 2 );
#[ 1, 1, 0 ]
# ListToInteger( [1,2,3,4], 10 );
# 1234
# ListToInteger( [0,1,1,2], 3 );
# 14
#################################################################
# This next five procedures are used to create incidence matrices from
# the output of Dylan Peifer's DifSet package.
# They require > LoadPackage("difset");
#################################################################
# 6. PointsToInterval( design )                                   #
#################################################################
#   This procedure takes a design (a set of sets) and creates
# an isomorphic design on the point set [1..v].
# i, j, and s are iterates.
#################################################################
PointsToInterval := function( design )
local blocks, v, k, points, intDesign, i, j, s;
   blocks := Size( design );
   points := Set( Union( design ) );
   v      := Size( points );
   intDesign := [ ];
   for i in [1..blocks] do
      Add(intDesign, ShallowCopy( design[i] ) );
   od;
   for i in [1..blocks] do
      k := Size( design[i] );
      for j in [1..k] do
         for s in [1..v] do
            if design[i][j] = points[s] then
```

```
              intDesign[i][j] := s;  fi;
          od;
       od;
   od;
   return intDesign;
end;
#################################################################
# 7. LeftTranslates_Nums( group, sublist )                      #
#################################################################
# This procedure reads in a group and a subset 'sublist'
# of integers representing members of the group and creates the
# collection of all images of 'sublist' under the action LEFT multiplication.
#      I've added a line that allows one to work from the numbers [1..v];
#################################################################
LeftTranslates_Nums := function( group, sublist )
local k, elements, g, design, i, block;
   k        := Size( sublist );
   elements := Elements( group );
   design   := [ ];
   for g in elements do
      block := [ ];
      for i in [ 1..k ] do Add( block,  g*elements[sublist[i]] ); od;
      Add( design, block );
   od;
   return design;
end;
#################################################################
# 8. IncidenceMatrix( design )                                  #
#################################################################
#   This procedure takes a design (a set of sets) on [1..v]
# and creates an incidence matrix.  This function works on
# designs that incorporate integers or other symbols.
#################################################################
IncidenceMatrix := function( design )
local blocks, v, k, incMat, intDesign, i, j;
   intDesign := PointsToInterval(design);
   blocks    := Size( intDesign );
   v         := Size( Set( Flat( intDesign ) ) );
   k         := Size( intDesign[1] );
   incMat := NullMat(blocks,v);
   for i in [1..blocks] do
      for j in [1..k] do
         incMat[i][intDesign[i][j]]:=1;
      od;
   od;
   return incMat;
```

```
 end;

 #################################################################

# Example:

# > g:=SmallGroup(16,2);

# > ds := [ 1, 2, 3, 4, 8, 15 ];

# > design := LeftTranslates_Nums( g, ds );

# > mat := IncidenceMatrix( design ); # mat is the incidence matrix

# > snf:=Collected(DiagonalOfMat(SmithNormalFormIntegerMat(mat)));

# The 2-rank, the dimension of the row space of 'mat' over GF(2) is

# > snf[1][2];



#     design1 := LeftTranslates_Nums( g1, ds );

#     pti1 := PointsToInterval( design1 );

# This is an unsorted list of the blocks.

#     blocklist1 := [ ];

#     for list in pti1 do Sort(list); Add(blocklist1, list); od;

# We sort the blocks.

#     SBIBD1 := BlockDesign( 64, blocklist1 );


 #################################################################

 # 9. CodeRank( mat, p )                                         #

 #################################################################
```

```
#   This procedure takes a matrix 'mat' and finds its rank
# in characteristic 'p'.
#################################################################
CodeRank := function( mat, p )
local rank, x, snf, r;
  rank := 0;
  snf:=Collected(DiagonalOfMat(SmithNormalFormIntegerMat(mat)));
  for x in snf do
    r := x[1] mod p;
    if not (r=0) then rank := rank + x[2]; fi;
  od;
  return rank;
end;
#################################################################
# 10. IncidenceMatToDesign( mat )                                  #
#################################################################
#   This procedure takes a (0,1)-matrix 'mat' and builds a design
# (a set of subsets of [1..v]) corresponding to the matrix.
#################################################################
IncidenceMatToDesign := function( mat )
local v, blocklist, row, col, list;
  v := Size(mat);
  blocklist := [ ];
  for row in [1..v] do
    list := [ ];
    for col in [1..v] do
      if mat[row][col] = 1 then Add(list, col); fi;
    od;
    Sort(list);
    Add(blocklist, list);
  od;
  return blocklist;
end;
#################################################################
# 11. IncidenceMatrixOfDifferenceSet( idgroup, ds );            #
#################################################################
IncidenceMatrixOfDifferenceSet := function( idgroup, ds )
 local g, design, mat;
 g:=SmallGroup( idgroup[1], idgroup[2]);
 design := LeftTranslates_Nums( g, ds );
 mat := IncidenceMatrix( design ); # mat is the incidence matrix
 return mat;
end;
#################################################################
# 12. IntersectionTriples( mat )                                    #
#################################################################
```

```
#   This procedure takes the (0,1) incidence matrix of a design
# and creates the list of triple intersections involving the first row.
#################################################################
IntersectionTriples := function( mat )
local v, row1, triples, i2, i3, row2, row3, sum, j;
   v := Size(mat);
   row1 := mat[1];
   triples := [ ];
   for i2 in [2..v] do for i3 in [i2+1..v] do
     row2 := mat[i2]; row3 := mat[i3];
     sum := 0;
     for j in [1..v] do sum := sum+row1[j]*row2[j]*row3[j]; od;
     Add(triples, sum);
   od;od;
   return Collected(triples);
 end;
#################################################################
# 13. IntersectionQuadruples( mat )                                    #
#################################################################
#   This procedure takes the (0,1) incidence matrix of a design
# and creates the list of triple intersections and quadruple intersections
# involving the first row.
#################################################################
```

```
IntersectionQuadruples := function( mat )
local v, row1, quadruples, i2, i3, i4, row2, row3, row4, sum, j;
    v := Size(mat);
    row1 := mat[1];
    quadruples := [ ];
    for i2 in [2..v] do for i3 in [i2+1..v] do for i4 in [i3+1..v] do
      row2 := mat[i2]; row3 := mat[i3]; row4 := mat[i4];
      sum := 0;
      for j in [1..v] do sum := sum+row1[j]*row2[j]*row3[j]*row4[j]; od;
      Add(quadruples, sum);
    od;od;od;
  return Collected(quadruples);
end;
#################################################################
# 14A. CentralInvolutions( g )                                  #
#################################################################
#   This finds all central involutions.  The central involutions,
# along with the identity, form a central elementary abelian 2-subgroup
# If Z(G) is the center of G then this subgroup is Omega_1(Z(G)).
# But this is not Omega(G):
https://en.wikipedia.org/wiki/Omega_and_agemo_subgroup
# The central involutions are returned as a list of integers,
# representing their position in the list Elements(g).
```

```
#################################################################

CentralInvolutions := function( g )

local e, inv, cent, x;

    e := Elements(g); inv := [ ]; cent := Center(g);

    for x in e do if Order(x)=2 and x in cent then Add(inv, Position(e,x));

  fi; od; return inv;

end;

#################################################################

# 14B. CentralInvolutions_Subgroup( g )                         #

#################################################################

#   This finds all central involutions.  The central involutions,

# along with the identity, form a central elementary abelian 2-subgroup

# If Z(G) is the center of G then this subgroup is Omega_1(Z(G)).

# But this is not Omega(G):

https://en.wikipedia.org/wiki/Omega_and_agemo_subgroup

# The central involutions are returned as a list of integers,

# representing their position in the list Elements(g).

#################################################################

CentralInvolutions_Subgroup := function( g )

local e, inv, cent, x;

    e := Elements(g); inv := [ ]; cent := Center(g);

    for x in e do if Order(x)=2 and x in cent then Add(inv, x); fi; od;

  return Group(inv);
```

```
 end;
 #################################################################
 # 15. CosetDistribution( group, subgroup, subsetlist )             #
 #################################################################
 #   This procedure finds the intersection numbers of the subset
 # of 'group' corresponding to the indices in 'subgrouplist'
 # with the cosets of 'subgroup' in a group 'group'.
 #################################################################
 CosetDistribution := function( group, subgroup, subsetlist )
 local e, cosets, cosetdistribution, cos;
   e := Elements(group);
#   list := [ ]; for i in subsetlist do Add(list, e[i]); od;
   cosets := LeftCosetList( group, e, subgroup );
   cosetdistribution := [ ];
   for cos in cosets do
     Add(cosetdistribution, Size(Intersection(cos, subsetlist)));
   od;
   return Collected(cosetdistribution);
 end;
 #################################################################
 # 16. Switching( list, mat )                                   #
 #################################################################
 #   This procedure 'switches' the entries of 'mat' on rows
 # identified in 'list'.
 ################################################################
 Switching := function( list, mat )
 local size, m, switchingset, errormessage, verifyflag, col, row, sum,
  newmat;
   size := Size(mat[1]);
   m := Size(list);
   switchingset := [ ];
   errormessage := [ ];
   # Below we decide which columns require complementation.
   # These columns are the 'switchingset', with sum = m/2
   # We also check that all column sums are 0, m/2 or m.
   # (If not, the column is placed in 'errormessage'.
   for col in [1..size] do
     sum := 0; for row in list do sum := sum + mat[row][col]; od;
     verifyflag := 0;
     if sum = m/2 then verifyflag := 1; Add(switchingset, col); fi;
     if sum = m or sum = 0 then verifyflag := 1; fi;
     if verifyflag = 0 then Add(errormessage, col); fi;
# Print(col," ", sum, "\n");
   od;
   newmat := [ ];
   if errormessage = [ ] then
```

```
     for row in [1..Size(mat)] do Add(newmat, ShallowCopy(mat[row]) ); od;
     for col in switchingset do for row in list do
       newmat[row][col] := 1-newmat[row][col];
     od; od;
# Print(switchingset,"\n");
     else Print("Errors at ", errormessage,"\n");
   fi;
   return newmat;
 end;
 #################################################################
 # 17. SpecialQuadruples( idgroup, difset )                        #
 #################################################################
 SpecialQuadruples := function( idgroup, difset  )
 local v, mat, quadrupleslist, row2, row3, row4, sum;
   v := idgroup[1];
   mat := IncidenceMatrixOfDifferenceSet( idgroup, difset );
   quadrupleslist := [[ ],[ ]];
   for row2 in [2..v] do for row3 in [row2+1..v] do for row4 in [row3+1..v] do
     sum :=  (mat[1]+mat[row2]+mat[row3]+mat[row4]) mod 2;
     if Collected(sum)=[[0,64]] then Add(quadrupleslist[1],
     [1,row2, row3, row4]); fi;
     if Collected(sum)=[[1,64]] then Add(quadrupleslist[2],
     [1,row2, row3, row4]); fi;
```

```
  od; od; od;

  return quadrupleslist;

end;

#################################################################

# 18. AutomorphismGroupOfDifferenceSetDesign( idgroup, ds );            #

#################################################################

AutomorphismGroupOfDifferenceSetDesign := function( idgroup, ds )

 local g, design, pti, blocklist, list, symmetricdesign, aut;

 g:=SmallGroup(idgroup[1], idgroup[2]);

 design := LeftTranslates_Nums( g, ds );

 pti := PointsToInterval( design ); # This is an unsorted list of the blocks.

 blocklist := [ ];

 for list in pti do Sort(list); Add(blocklist, list); od; # We sort the blocks.

 symmetricdesign := BlockDesign( Size(blocklist), blocklist );

#  mat := IncidenceMatrix( design ); # mat is the incidence matrix

 aut := AutomorphismGroup( symmetricdesign );

 return aut;

end;


# design1 := LeftTranslates_Nums( g1, ds );

#    ########################################

#################################################################

# DEMO:
# LoadPackage("design");

# aut := AutomorphismGroupOfDifferenceSetDesign( [16,2], [ 1, 2, 3, 4, 8, 15 ]);

# path := "gap4r8/KensGap/2021/DifSetProcedures/"; filename0 :=

Concatenation(path,"RegularAutomorphismGroup.txt");

# Read(filename0);

# LoadPackage("design");

##################################################################

 # 19A. RegularSubgroup( set,aut, max )                #

##################################################################

#   This procedure attempts to take a subgroup 'aut' acting transitively

# on a set [1..2^m] and find all 2-groups acting regularly on the set.

# The procedure assumes that we seek groups of order 2^m, for some m

#   The parameter max gives an upper bound for the exponent on 2 in

# the order of the Sylow-2 subgroup of 'aut'.  If the exponent is

# higher than 'max', the procedure gives up and returns a message

# "upper bound exceeded".

##################################################################

#######################################################################

#######################################################################

 local syl, target, m, currentexp, solutions, level1, level2,g,

 maxsubs, size1, size2, h;

  syl:=SylowSubgroup(aut,2);

  target := Collected(Factors(Size(set)))[1][2];
  m:=Collected(Factors(Size(syl)))[1][2];
  currentexp := m;
  Print("Working on a group of order ", Size(aut), " with Sylow 2-subgroup
  of size ", Size(syl)," = 2^", m,".\n");

 solutions := [ ];
 level1 := [ syl ];
 level2 := [ ];
 currentexp := Collected(Factors(Size(syl)))[1][2];

if currentexp <= max then
###################################
# While loop begins here.
###################################
 while currentexp > target do
   for g in level1 do
     maxsubs := MaximalNormalSubgroups(g);
     level2 := Concatenation(level2, maxsubs);
# level2 := Set(level2);
   od; # for g in level1 do
   size1 := Size(level2);
   level2 := Set(level2);
   size2 := Size(level2);
   level1 := [ ]; for h in level2 do if IsTransitive(h, set) then
   Add(level1, h); fi; od;

  Print("   Completing level ", currentexp," with ", Size(level1), " new
  subgroups.\n");
  level2 := [ ];
  currentexp := currentexp-1;
od; # while...

 for g in level1 do Add(solutions, IdGroup(g)[2]); od;
 solutions := Set(solutions);
 else solutions := "Upper bound exceeded.";
fi;
#  Print(solutions,"\n");
 return solutions;
end;
######################################################################

################################################################
# 19B. RegularSubgroup_SpaceSaver( set,aut, max )               #
################################################################
#   This procedure attempts to take a subgroup 'aut' acting transitively
# on a set [1..2^m] and find all 2-groups acting regularly on the set.
# The procedure assumes that we seek groups of order 2^m, for some m
#   The parameter max gives an upper bound for the exponent on 2 in
# the order of the Sylow-2 subgroup of 'aut'.  If the exponent is
# higher than 'max', the procedure gives up and returns a message
# "upper bound exceeded".
#   This program adds a line 'level2 := Set(level2);' after a
# 'Concatenation' command to move up the removal of duplicate groups.
# This presumably slows the program down but preserves space.
#################################################################
######################################################################
RegularSubgroup_SpaceSaver := function( set, aut, max )
######################################################################
 local syl, target, m, currentexp, solutions, level1, level2,g,
 maxsubs, size1, size2, h;
  syl:=SylowSubgroup(aut,2);
  target := Collected(Factors(Size(set)))[1][2];
  m:=Collected(Factors(Size(syl)))[1][2];
  currentexp := m;
  Print("Working on a group of order ", Size(aut), " with Sylow 2-subgroup
  of size ", Size(syl)," = 2^", m,".\n");

 solutions := [ ];
 level1 := [ syl ];
```

```
  level2 := [ ];

  currentexp := Collected(Factors(Size(syl)))[1][2];


 if currentexp <= max then

 ###################################

 # While loop begins here.

 ###################################

  while currentexp > target do

    for g in level1 do

      maxsubs := MaximalNormalSubgroups(g);

      level2 := Concatenation(level2, maxsubs);

#

    level2 := Set(level2);

    od; # for g in level1 do

    size1 := Size(level2);

    level2 := Set(level2);

    size2 := Size(level2);

    level1 := [ ]; for h in level2 do if IsTransitive(h, set) then

    Add(level1, h); fi; od;


   Print("Completing level ", currentexp," with ", Size(level1), " new

   subgroups.\n");

   level2 := [ ];
```

```
    currentexp := currentexp-1;
  od; # while...


   for g in level1 do Add(solutions, IdGroup(g)[2]); od;
   solutions := Set(solutions);
   else solutions := "Upper bound exceeded.";
fi;
  #  Print(solutions,"\n");
   return solutions;
  end;
######################################################################
   #################################################################
  # DEMO:
  # LoadPackage("design");
  # aut := AutomorphismGroupOfDifferenceSetDesign( [16,2], [ 1, 2, 3, 4, 8, 15 ]);
  # The program on 'aut' above crashes after five full garbage collections,
  # brk> Size(level1); # 40760 at size 2^12?
  # brk> Size(level2); # 849532
```

**Programs that Creates Semi-Direct Products using (16,6,2) Difference Sets:** This file reads in a list of 16 by 16 semidirect products, previously created, and creates, from them, (256, 120, 56) difference sets using the product construction.

```
   LoadPackage("difset");  # Load Peifer's 'difset' package which
                           # verifies difference sets
```

```
  filename1 := Concatenation(path,"/GeneralizedProductProcedures_2021_10.txt");

  #GeneralizedProductProcedure code is shown above.

  Read(filename1);

  filename2 := Concatenation(path, "/0_Data_SemidirectProducts16in256.txt");

  #0_Data_SemidirectProducts16in256.txt is a GAP code that contains all

  semi-direct products of two groups with order 16.

  Read(filename2);

  outputfile := Concatenation(path, "/1_Output_Difsets_16by16.txt") #This

  is the output file of the results obtained from this code.


  Print("We use semidirect products,

  16 by 16 (first subgroup is normal) to create\n");

  Print("difference sets in groups of order 256.\n\n");


SDP16 := [

[ ],

[ [ 1, 2, 3, 4, 8, 15 ], [ 1, 2, 3, 4, 10, 14 ], [ 1, 2, 3, 8, 9, 11 ] ],

[ [ 1, 2, 3, 4, 8, 15 ], [ 1, 2, 3, 4, 10, 14 ], [ 1, 2, 3, 5, 7, 15 ],

[ 1, 2, 3, 7, 10, 11 ] ],

[ [ 1, 2, 3, 4, 8, 15 ], [ 1, 2, 3, 4, 10, 14 ], [ 1, 2, 3, 5, 7, 15 ] ],

[ [ 1, 2, 3, 4, 8, 15 ] ],

[ [ 1, 2, 3, 4, 8, 15 ], [ 1, 2, 3, 8, 9, 11 ] ],

[ ],
[ [ 1, 2, 3, 4, 7, 10 ], [ 1, 2, 3, 4, 10, 14 ] ],
[ [ 1, 2, 3, 4, 7, 10 ], [ 1, 2, 3, 4, 8, 9 ] ],
[ [ 1, 2, 3, 4, 5, 16 ] ],
[ [ 1, 2, 3, 4, 8, 15 ], [ 1, 2, 3, 4, 8, 15 ], [ 1, 2, 3, 5, 7, 15 ] ],
[ [ 1, 2, 3, 4, 8, 15 ] ],
[ [ 1, 2, 3, 4, 8, 15 ] ],
[ [ 1, 2, 3, 4, 5, 16 ] ]
];

 difsetlist := [ ];

 for cn in [1..56092] do
   semidplist := semidirectproducts16in256[cn]; #Reads the list of
   semi-direct products given in 0_Data_SemidirectProducts16in256.txt

   if Size(semidplist)>0 then
     v := 256; g:= SmallGroup(v, cn); e:= Elements(g);
     norm := NormalSubgroups(g); subs := AllSubgroups(g);
     norm := SortedList(norm); subs := SortedList(subs);

     for x in semidplist do
       k := norm[x[1][1]]; vk := x[1][2][1]; idk := x[1][2][2];
       h := subs[x[2][1]]; vh := x[2][2][1]; idh := x[2][2][2];
```

```
groupk:=SmallGroup( vk, idk ); ek:=Elements(groupk);

fk:=IsomorphismGroups(groupk, k);

grouph:=SmallGroup( vh, idh ); eh:=Elements(grouph);

fh:=IsomorphismGroups(grouph, h);


dsk := SDP16[idk];

dsh := SDP16[idh];


for ds1 in dsk do for ds2 in dsh do

  Print(ds1," ",ds2,"\n");

  negk := [1..vk]; SubtractSet(negk, ds1);

  negh := [1..vh]; SubtractSet(negh, ds2);

  neg1:=[ ]; neg2:=[ ]; pos1:=[ ]; pos2:=[ ];

  for index in negk do Add(neg1, Position(e,Image(fk, ek[index]))); od;

  for index in ds1 do  Add(pos1, Position(e,Image(fk, ek[index]))); od;

  for index in negh do Add(neg2, Position(e,Image(fh, eh[index]))); od;

  for index in ds2 do  Add(pos2, Position(e,Image(fh, eh[index]))); od;

  prod := ProductPTAs( [256,cn], [neg1, pos1], [neg2, pos2] );

  ds := prod[2];

  dstest := IsDifferenceSet(g, ds);

   Print(dstest,"\n");

  if dstest then

    Add(difsetlist, [cn, [[vk, idk],[vh, idh]], ds]);
```

```
     Print(cn,". ",[[vk, idk],[vh, idh]]," ", dstest,"\n");
    else Print("*** Error! ", x," ***\n");
    fi;
   od; od;
  od;
 fi;
 if RemInt(cn, 1000) = 0 then  # Let’s record results from time to time
  Print(cn,". ", Size(difsetlist),"\n");
  PrintTo(outputfile, "difsetlist := ", difsetlist,";\n");
 fi;
od; # for cn in ...

PrintTo(outputfile, "difsetlist := ", difsetlist,";\n");
```

**Program that Analyzes Difference Sets where v=256:** This file reads in difference sets created by semidirect products and then sifts them for those that create a design with rank equal to 10 (and thus SDP.) Thus, it essentially takes the results of the first program and goes through all of the difference sets to see which ones give SDP designs.

```
LoadPackage("design");
Read(Concatenation(path,"DifSetProcedures_2021_08.txt"));
Read(Concatenation(path,"GeneralizedProductProcedures_2021_10.txt"));

inputfile := Concatenation(path,"1_Output_Difsets_16by16.txt");
Read(inputfile);
```

```
outputfile :=  Concatenation(path,"2_Output_sdplist.txt");


 Print("We read in difference sets constructed by 16 by 16 semidirect products\n
 ");
 Print("and sift them for SDP difference sets, that is, difference sets\n");
 Print("with binary rank equal to 10.\n\n");


 Print("Reading in storage file ", inputfile, " with ", Size(difsetlist),
 " difference sets.\n");


ranklist := [ ]; sdplist := [ ]; sdpcounter := 0;
counter := 0;


for x in difsetlist do
  counter := counter + 1;
  cn := x[1];
  ds := x[3];
  mat := IncidenceMatrixOfDifferenceSet( [256,cn], ds ); #This comes from
  Difference Set Procedure code
  rank := CodeRank( mat, 2 );
  Add(ranklist, rank);
  if RemInt(counter,500) = 0 then # from time to time we record &
                                  # store the current results.
```

```
    Print("Counter = ", counter, ", Catalogue number = ",cn,".
    ranks = ", Collected(ranklist),"\n");
    PrintTo(outputfile, "sdplist := ", sdplist,";\n");
  fi;
  if rank = 10 then
    Add(sdplist, x); sdpcounter:=sdpcounter+1;
  fi;
od;


Print(Collected(ranklist),"\n");
PrintTo(outputfile, "sdplist := ", sdplist,";\n");
```

**Program that gives the Non-Isomorphic SDP designs where v=256:** This file runs through all of the SDP designs that were given in previous program and gives all of the designs are non-isomorphic from each other.

```
LoadPackage("design");
Read(Concatenation(path,"/DifSetProcedures_2021_08.txt"));
Read(Concatenation(path,"/GeneralizedProductProcedures_2021_10.txt"));


Print("We examine difference sets with 2-rank equal to 10 and record\n");
Print("one difference set per isomorphism class of design.\n\n");


v := 256;
```

```
outputfile := Concatenation(path,"/3_Output_NonisomorphicSDP256.txt");

inputfile := Concatenation(path,"/2_Output_sdplist.txt");

Read(inputfile);


Print("Input file is ", inputfile, " with ", Size(sdplist),"

difference sets.\n");


NonIsomorphicSDP := [ ]; DifSetRepresentatives := [ ];

dscounter := 0;

for ds in sdplist do

  dscounter := dscounter+1;

  cn := ds[1]; difset := ds[3];

  g := SmallGroup( v, cn );

  design := PointsToInterval( LeftTranslates_Nums( g, difset ) );

  blocklist := [ ]; for list in design do Sort(list); Add(blocklist, list); od;

  SBIBD2 := BlockDesign( Size(blocklist), blocklist ); #


  newflag :=1;

  for SBIBD in NonIsomorphicSDP do

    iso := IsIsomorphicBlockDesign(SBIBD, SBIBD2);

    if iso then newflag := 0; fi; #

  od;
  if newflag = 1 then
    Add(NonIsomorphicSDP , SBIBD2);
    Add(DifSetRepresentatives, ds);
    autsize := Size(AutomorphismGroup(SBIBD2));
    Print("Adding new SBIBD with automorphism group of order ",
    Collected(Factors(autsize)),"\n");
  fi;
  if RemInt(dscounter, 100)=0 then # From time to time we store our work.
    Print("Completing ", dscounter, " difference sets, with
    ", Size(NonIsomorphicSDP)," different designs.\n");
  fi;
od; # for ds in ...
PrintTo(outputfile,"NonIsomorphicSDP := ",NonIsomorphicSDP,";\n");
AppendTo(outputfile,"DifSetRepresentatives :=",DifSetRepresentatives,";\n");
```

All of these codes were made using codes and programming found in GAP-systerm.org.[9]

# VITA

Matthew Williams

## EDUCATION

Master of Science student in Mathematics at Sam Houston State University, Fall 2020 – present. Thesis title: "Difference Sets and the Symmetric Difference Property."

Bachelor of Science (May 2020) in mathematics, Sam Houston State University, Huntsville, Texas.

## ACADEMIC EMPLOYMENT

Graduate Teaching Assistant, Department of Mathematics and Statistics, Sam Houston State University, June 2021 - present. Worked with two professors, Dr. Scott Chapman and Dr. Martin Malandro during the summer of 2021. Responsible for creating and hosting office/tutoring hours for students in need of assistance (Dr. Scott Chapman) or helped prepare a course that was taught in the Fall 2021 semester by looking over all of the materials, textbooks and assignments given for that course (Dr. Martin Malandro). Become an instructor (but not a formal professor) during the Fall 2021 semester. Responsible of delivering every lecture, and all other aspects of course management, such as, creating activities to monitor learning, grading and recording grades, suggesting solutions to assigned problems, addressing student performance issues, developing ways to improve learning and understanding, assigning final grades. Currently an embedded tutor that works with students who come to office/tutoring hours for students in need of assistance with assignments and/or topics that are taught by professors/other instructors.

Graduate Assistant, Department of Mathematics and Statistics, Sam Houston State University, September 2020 - May 2021. Assigned to three professors in the department to perform a variety of tasks for them throughout the year, which included grading papers and tutoring students on certain topics when needed.

**ACADEMIC AWARDS AND HONORS**

National Society of Collegiate Scholars, Sam Houston State University, February 2018-Lifetime

Alpha Lambda Delta, Sam Houston State University, January 2018-Present

Award for Academic Excellence (4.0 GPA), Fall 2017 and Spring 2018, Fall 2018 and Spring 2019

International Dean's List Society, February 2018-Lifetime

Alpha Chi National College Honor Society, Spring 2019-Present

National Society of Leadership and Success, Spring 2019-Present

Dean's List, Fall 2017 thru Spring 2020

President's List, Fall 2017 thru Spring 2020

2019 Outstanding Junior Mathematics Major, Spring 2019

2020 Outstanding Senior Mathematics Major, Spring 2020

Golden Key International Honor Society, Fall 2019

Avila Undergraduate Research Scholarship, Spring 2020 thru Summer 2020

Summa Cum Laude

2021 Outstanding Graduate Mathematics Major/Student, Spring 2021

Raven's Scholars, Spring 2021