On m-general sets in $\mathrm{AG}(k, q)$ and $\mathrm{PG}(k, q)$

Sidon sets, projective arcs and error-correcting codes

Tim Neumann

Advised By Prof. Robert Won

Department of Mathematics
George Washington University

May 2024

Senior Thesis in partial fulfillment of the requirements for obtaining
Special Honors in the Bachelor of Science in Mathematics

Acknowledgements

This thesis would not have been possible if it wasn't for the unwavering support of amazing people around me.

I am deeply grateful to the Department of Mathematics at GWU for providing a collaborative learning environment and the opportunity to conduct original research. I learned that doing research is mainly about asking the right questions, so I want to express my sincere gratitude to Professor Won for his time, dedication, patience and excellent supervision throughout the last year, steering me in the right directions and encouraging me to go further. I also owe special thanks to Professor Bonin, who taught me a new way of thinking, without which I would not feel prepared for the challenges that await me past my undergraduate studies.

I am thankful for the High Performance Computing team at GWU and the support they offered me throughout the last year. In particular, Joseph Creech's relentless assistance with the usage of the cluster was imperative to the success of the computational part of this thesis.

Furthermore, I would like to extend my thanks to researchers and experts in the field for the outstanding help I received. In particular, I want to thank Dr. Romera-Paredes for answering my questions around FunSearch, and the DeepMind team for showing that AI can indeed be used to make advances in (pure) mathematics. Many thanks also to Professor Pavese, who helped me understand a flaw in my approach to finding minimal m-general sets in projective space. A special thanks goes to Johannes Aalto, who embedded the publicly available DeepMind code into a workable environment without which I could have not run my experiment.

I could not have undertaken this journey without a magnificent support system. First and foremost, I want to thank my family for their love and support not only throughout this thesis, but my entire time in D.C. – Sharing the highs and lows of life with you encouraged me to work my plan and live my dream. (Also, thank you for playing the SET game every day and keeping a lively database of backup files in your e-mail account.) Many thanks also go to each of my friends at GWU and at home for making the last four years an unique and unforgettable time. I am especially grateful for Samiha Rao, Tom Cooklin, Emma Klinge, Fynn Schmitz and Aaron Grommes for sharing my passion for math, and for my friends on the soccer team and in the athletics department for being a second family to me. Lastly, I want to acknowledge my teachers and advisors both at GWU and at home, and in particular Ralf Gieße and Jakob Bohr for instilling in me a passion for mathematics and providing me with the solid foundation I needed to succeed at GWU.

Abstract

Sets of points in general position in geometric spaces are both inherently fascinating mathematical objects in their own right, and also bear useful connections to coding theory. In this thesis, we firstly examine 4-general sets in affine space, where they are related to well-studied sets in additive combinatorics called Sidon sets. In particular, we develop a new construction that proves an improved lower bound for the maximum size of 4-general sets in $\mathbb{F}_3^k$, present new computational results, and provide a proof-of-concept for a Large Language Model (LLM) approach to constructing large 4-general sets. In the second part of the thesis, we consider m-general sets in projective space and study their relationship to coding theory. We derive the well-known correspondence between certain subspace-evasive sets and linear error-correcting codes, illustrate how projective arcs give rise to maximal distance separable (MDS) codes, and obtain computational results on small and large (complete) m-general sets. Throughout, we hope to create an appreciation for how seemingly unrelated mathematical objects (such as Sidon sets and error-correcting codes) are united by the notion of m-general sets.

Introduction and Outline

It seems to be a recurring theme in mathematics that abstract objects relate different fields in unexpected ways, the Weil conjectures being one of the most famous examples of such happenstance. The structural depth of mathematical ideas is fascinating, and the fact that mathematicians work on similar problems in seemingly isolated fields intriguing.

This thesis is divided into two parts in order to treat the versatile notion of m-general sets in different geometries, and to shed light on the structures those sets give rise to and relate. In the first part, we introduce the affine space $\mathrm{AG}(k, q)$ as instance of a finite geometry and define m-general sets as well as Sidon sets, which bear a nice correspondence for $m = 4$ in $\mathrm{AG}(k, 3)$ and (up to a technical detail) in $\mathrm{AG}(k, 2)$. The central problem in this part is to find large (complete) Sidon sets in $\mathrm{AG}(k, 3)$, for which we employ three different approaches, each yielding a new perspective. Interestingly, the third approach involves the usage of a Large Language Model, contributing to the fairly recent development of utilizing Artificial Intelligence in (pure) mathematics. We also present computational results on small and large (complete) m-general sets in $\mathrm{AG}(k, 2)$ and $\mathrm{AG}(k, 3)$ for small dimensions k and $m = 3, 4, 5$.

In the second part of the thesis, we introduce projective space $\mathrm{PG}(k, q)$ as another instance of a finite geometry, and provide an overview over the field of coding theory. Building upon this, we develop and illustrate some well-known connections between generalizations of m-general sets called subspace-evasive set and certain error-correcting codes, and show how m-general sets are related to projective arcs and minimum-distance separable (MDS) error-correcting codes. This part is devised predominantly as a synopsis, relating concepts used in different fields of mathematics. We present computational results for m-general sets in $\mathrm{PG}(k, 2)$ and $\mathrm{PG}(k, 3)$ for small dimensions here as well.

Overall, we hope to shed light on the intricate connections that the notion of m-general sets in different geometries bear, and to motivate further work on this topic, especially in the context of Artificial Intelligence.

Table of Contents

PART 1

Sidon sets and m-general sets in affine space $\mathrm{AG}(k, q)$

1.1. Finite geometries: Affine Space

A *finite geometry* is a geometric space with finitely many points. Finite geometries enjoy wide interest in mathematics, partly due to their intrinsic fascination,[1] and partly due to their applications.[2] While most of our intuition is rooted in the Euclidean space $\mathbb{R}^n$, which is not a finite geometry, some of this intuition translates to finite geometries as well. In order to argue about these spaces rigorously, however, we shall first provide some definitions and framework for the spaces in question.

1.1.1. Fundamentals of $\mathrm{AG}(k, q)$.

Throughout this paper, let q be a prime power and let $\mathbb{F}_q$ denote the finite field of order q. We will also use the notation $\mathbb{F}$ to refer to a general field. Recall that when defining a vector space over $\mathbb{F}$, the origin (or the zero vector) plays a special role. For example, every vector subspace must pass through the origin. Any vector space can also be viewed as an *affine* space. However, when viewing a vector space as an affine space, the zero vector plays the same role as all of the other vectors in the space. For example, affine subspaces are not required to intersect the origin. The line $y = mx + b$ with $b \neq 0$ is an example of a one-dimensional affine subspace of $\mathbb{R}^2$.

Definition 1.1.1. (Affine space). Let $\mathbb{F}$ be a field. The *k-dimensional affine space over* $\mathbb{F}$ is the geometric structure whose points are elements of the vector space $\mathbb{F}^k$ and whose d-dimensional affine subspaces are the translates of the d-dimensional vector subspaces of $\mathbb{F}^k$.

When $\mathbb{F} = \mathbb{F}_q$, we use the notation $\mathrm{AG}(k, q)$ for the affine space $\mathbb{F}_q^k$.

Definition 1.1.2. (Flats). A d-dimensional affine subspace of an affine space is called a *d-dimensional flat*, a *d-flat*, or a *flat of codimension $k - d$*.

Whenever it is not clear from context, we will specify whether we are referring to a (linear) vector subspace or an affine subspace. We often call 0-dimensional flats *points*, 1-dimensional flats *lines*, 2-dimensional flats *planes* and $(k-1)$-dimensional flats *hyperplanes*.

[1] Terence Tao declared the capset problem in the finite geometry $\mathrm{AG}(n, 3)$, which we will have much more to say about later, as one of his “favourite open questions” in his blog back in 2007.

[2] Affine spaces are the mathematical underpinning of the SET game, for example.

Example 1.1.3. $\mathrm{AG}(2,3)$ is the 2-dimensional affine space over $\mathbb{F}_3$. There are $3^2 = 9$ points in $\mathrm{AG}(2,3)$, which we can arrange visually:

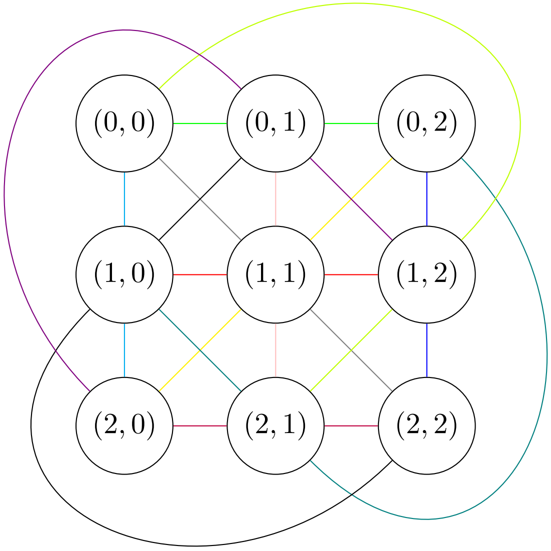


Figure 1. The affine plane $\mathrm{AG}(2,3)$

We have also drawn all twelve lines (that is, 1-dimensional flats) in $\mathrm{AG}(2,3)$. Observe that each point is on the same number of lines, and the vector $(0,0)$ plays no special role.

It is easy to count the number of affine subspaces of $\mathrm{AG}(k,q)$.

Theorem 1.1.4. For any $1 \leq d \leq k$, the number of d-dimensional subspaces of $\mathrm{AG}(k,q)$ is

$$q^{k-d}\frac{(q^k-1)(q^k-q)\dots(q^k-q^{d-1})}{(q^d-1)(q^d-q)\dots(q^d-q^{d-1})}.$$

Proof. We first count the number of d-dimensional vector subspaces of the vector space $\mathbb{F}_q^k$. Any such subspace has a basis consisting of d vectors, and so first we count the number of sets consisting of d linearly independent vectors. There are $q^k - 1$ choices for the first basis vector (excluding the zero vector), and each additional of the $d-1$ basis vectors must be linearly independent to the previous ones. Since the first basis vector blocks a 1-dimensional subspace containing q points, there are $q^k - q$ choices for the second vector. This 2-dimensional subspace blocks q^2 vectors, so for choosing a third linearly independent basis vector there are $q^k - q^2$ options. Proceeding inductively, there are $q^k - q^{i-1}$ choices for the i-th basis vector. Hence, there are $(q^k-1)(q^k-q)\dots(q^k-q^{d-1})$ choices that yield a basis for a d-dimensional vector subspace.

Observe that some of these bases determine the same subspaces. Hence we need to divide by the number of bases for any given d-dimensional vector subspace, which by the above reasoning is $(q^d-1)(q^d-q)\dots(q^d-q^{d-1})$. It follows that the quantity

$$\frac{(q^k-1)(q^k-q)\dots(q^k-q^{d-1})}{(q^d-1)(q^d-q)\dots(q^d-q^{d-1})}$$

counts the number of d-dimensional subspaces of $\mathbb{F}_q^k$. For the affine subspaces of $\mathrm{AG}(k,q)$, we need to account for the translates of such d-dimensional vector subspaces as well. There are q^k translates in total, but some translations yield the same affine subspace. In particular, since a d-dimensional subspace contains q^d points, exactly q^d translates yield the same affine subspace. Thus, we multiply by a coefficient of q^{k-d}, which proves the claim. $\square$

In $\mathrm{AG}(2,3)$ (see Example 1.1.3), there are $3^{(2-1)}\left(\frac{3^2-1}{3^1-1}\right) = 12$ one-dimensional subspaces, i.e., affine lines. It turns out that three points lying on a line in $\mathrm{AG}(2,3)$ can be stated in terms of an arithmetic condition, which we will exploit computationally later.

Theorem 1.1.5. In $\mathrm{AG}(k,3)$, three points $(x_1,x_2,\dots,x_k),(y_1,y_2,\dots,y_k),(z_1,z_2,\dots,z_k)$ form a line if and only if they sum to $\mathbf{0}$.

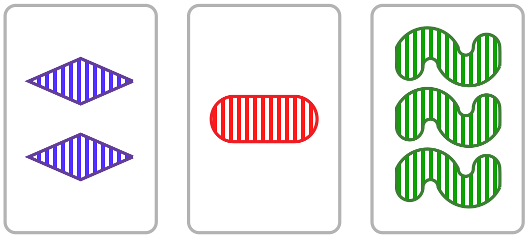
Proof. Since there are three field elements, a line is equivalent to the arithmetic progression $a, a+v, a+2v$ of length 3, for any point a and direction v. Now for the forward implication, note that we have $a+(a+v)+(a+2v) = 3a+3v = \mathbf{0}$ over $\mathbb{F}_3$. Conversely, assume that $\mathbf{x}+\mathbf{y}+\mathbf{z} = \mathbf{0}$ holds. Then we have $\mathbf{z} = \mathbf{0}-\mathbf{x}-\mathbf{y}$, and setting $\mathbf{a} = \mathbf{x}$ and $\mathbf{v} = \mathbf{y}-\mathbf{x}$ yields the arithmetic progression $\mathbf{a} = \mathbf{x}$, $\mathbf{a}+\mathbf{v} = \mathbf{y}$ and $\mathbf{a}+\mathbf{2v} = \mathbf{z}$. So $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ indeed form a line in $\mathrm{AG}(k,3)$. $\square$

We can identify another useful characteristic of lines in $\mathrm{AG}(k,3)$.

Theorem 1.1.6. In $\mathrm{AG}(k,3)$, three points $(x_1,x_2,\dots,x_k),(y_1,y_2,\dots,y_k),(z_1,z_2,\dots,z_k)$ form a line if and only if for each $1 \le i \le k$, the elements $x_i, y_i, z_i \in \mathbb{F}_3$ are either all the same or all distinct.

Proof. By Theorem 1.1.5, three points $(x_1,x_2,\dots,x_k),(y_1,y_2,\dots,y_k),(z_1,z_2,\dots,z_k)$ that form a line must satisfy $x_i+y_i+z_i = 0$ for each $1 \le i \le k$. Now without loss of generality, we can differentiate between three cases. Firstly, if all elements are the same, then they must be one of $0+0+0$, $1+1+1$, $2+2+2$, each of which evaluates to 0 over $\mathbb{F}_3$. Secondly, if two elements are the same, and the third differs, then the equal elements sum either to 0, 2 or 1, in neither of which cases a distinct third element can complete the sum to 0 over $\mathbb{F}_3$. Lastly, if all elements are distinct, then they must be of the form $1+2+3=0$. $\square$

This relation between subspaces and arithmetic progressions is a very useful idiosyncracy of $\mathbb{F}_3$. For example, by the above, both sets of 3-tuples $[(0,0,0),(0,0,1),(0,0,2)]$ and $[(0,0,2),(1,1,1),(2,2,0)]$ form a line in $\mathrm{AG}(3,3)$, which can be readily checked through component-wise addition. Now upon adding an additional dimension, we obtain $\mathrm{AG}(4,3)$, which consists of $3^4 = 81$ points that can be arranged in a $3 \times 3 \times 3 \times 3$ grid.[3] This space is well-known to everyone who has played the SET game before; in this popular[4] card game, every card displays one of three different characteristics for four different features (size, shape, color and shading). The goal of the game is to identify so-called SETs – collections of three cards, for which in each of the four features, the characteristics on the cards are either all the same or all distinct – as quickly as possible.



Example of a SET in the SET game.

This rule, compared to Theorem 1.1.6, shows that a SET in the game is simply a line in $\mathrm{AG}(4,3)$, which introduces many interesting problems in combinatorics, statistics and geometry.[5].

We now establish some terminology that allows us to describe how sets of points in affine space interact with one another.

Definition 1.1.7. (Affine combinations and affine independence). Let $\mathbb{F}$ be a field and let $\mathbb{F}^k$ be an affine space over $\mathbb{F}$. Given t points $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_t} \in \mathbb{F}^k$, an *affine combination* of these points is a linear combination $\sum_{i=0}^{t} c_i \mathbf{a_i}$ with $c_i \in \mathbb{F}$ where $\sum_{i=0}^{t} c_i = 1$.

A set of points is called *affinely dependent* if one of the points is an affine combination of the others. Otherwise, the set is called *affinely independent*.

Equivalently, a set $\{a_1, a_2, \ldots, a_t\}$ is affinely dependent if there exists a linear combination $\sum_{i=0}^{t} c_i a_i = \mathbf{0}$, where $\sum_{i=0}^{t} c_i = 0$ and the c_i's are not identically zero. This will play an important role in the computational approach to finding the maximal size of special sets in affine space developed in Section 1.3.

Example 1.1.8. The definition of affine combinations is familiar from parameterizing line segments in $\mathbb{R}^k$, where the line segment from a to b is given by $ta + (1-t)b$ with $t \in [0,1]$, such that the

[3] This space has three parallel hyperplanes isomorphic to $\mathrm{AG}(3,3)$.

[4] It is up to the discretion of the reader to decide whether SET is popular because or despite its deep connections to finite geometry.

[5] For an approachable and entertaining treatment, the reader is referred to "The Joy of SET: The Many Mathematical Dimensions of a Seemingly Simple Card Game" [MGGG17]

coefficients t and $1-t$ sum to 1. Letting t assume any real value gives an equation for the unique line through a and b in $\mathbb{R}^k$.

Continuing the resemblance of definitions from vector spaces, we define the *affine span* of a set to be the set of all affine combinations of points in a set. Then it is obvious that a set of points $\mathcal{S} \subseteq \mathbb{F}_q^k$ is affinely independent if no proper subset of $\mathcal{S}$ has the same affine span as $\mathcal{S}$.

Having introduced affine space, we now take a closer look at sets of points in $\mathrm{AG}(k,q)$ which have special properties.

1.1.2. m-general sets .

Given a set of points in a geometry, we are naturally interested in how those points are configured in space.

Definition 1.1.9. (General position). A set of points in $\mathbb{F}^k$ is said to be in *general position* if, for all d with $1 \leq d \leq k-1$, no $d+2$ points lie on a d-dimensional flat.

The following terminology was introduced by Bennett in [**Ben19**, Definition 1.1].

Definition 1.1.10. (m-general set). Let $3 \leq m \leq k+2$, and let $\mathcal{S}$ be a set of points in $\mathrm{AG}(k,q)$ with $|\mathcal{S}| \geq m$. Then $\mathcal{S}$ is called *m-general* if no m points of $\mathcal{S}$ lie on a common $(m-2)$-dimensional subspace of $\mathrm{AG}(k,q)$.

That is, a set is m-general if all of its size-m subsets are in general position. Observe that the restriction on the size of the set in the definition above guarantees that the set does not trivially fulfill the m-general property,[6] and the restrictions on m assure non-degeneracy.[7]

Equivalently, a m-general set is a set of points such that at most $m-1$ points lie on a single $(m-2)$-dimensional subspace [**Pav23**].

Definition 1.1.11. (Completeness and maximality). A m-general set is called *complete* if it is not contained in a larger m-general set, and *maximal* if it is complete and of largest size among all complete such sets.

These kinds of sets that have been studied extensively in the literature [**Pav23, TW21**]. In projective space, m-general sets bear deep connections to coding theory. In $\mathrm{AG}(k,3)$ the notion of a 3-general set is well-studied.

Definition 1.1.12. (Capset). A *cap* or *capset* is a set of points in $\mathrm{AG}(k,3)$ with no three points on a common line.

[6] However, we will still refer to a set of two points as being 3-general, if needed.

[7] Setting $m < 3$ or $m > k+2$ make talking about sets of points on $(m-2)$-dimensional subspaces degenerate, since those subspaces become either points or the entire space.

Note that a cap is simply a 3-general set over $\mathrm{AG}(k,3)$. Alternatively, in $\mathrm{AG}(k,3)$, by Theorem 1.1.5, we can define a cap to be a set of points that does not contain any three-term arithmetic progressions. The main result of [**HTW19**] is that a 4-general set in $\mathrm{AG}(k,3)$ corresponds to a *Sidon set* as described in Definition 1.1.18.

A very famous problem is the *capset problem*, which asks for the size of a maximal capset in $\mathrm{AG}(k,3)$. This problem has been solved explicitly only up to dimension $k = 6$, but in a groundbreaking paper from 2016, Ellenberg and Gijswijt improved the upper bound for the size of capsets to $o(2.756^k)$,[8] which is a very strong asymptotic result [**EG16**].

Interestingly, recent work by Google DeepMind has improved the lower bound of the capset problem for $k = 8$ to 512 using a computational approach involving an evolutionary algorithm on a Large Language Model (LLM) [**RPBN$^+$24**]. Hence, the capset problem remains an object of on-going research.

Theorem 1.1.13. [**TW21**] For $m > 3$ and $s \geq m$,[9] an m-general set of size s is also an m'-general set for $3 \leq m' \leq m-1$.

Proof. Let $\mathcal{S}$ be an m-general set in $\mathrm{AG}(k,q)$ where $|\mathcal{S}| > m$. To prove the claim, we need to show that for $3 \leq m' \leq m-1$, $\mathcal{S}$ is an m'-general set, i.e., that $\mathcal{S}$ intersects every m'-dimensional subspace in at most $m'+1$ points. For a contradiction, assume that there exists some m' in the for which at least $m'+2$ of the points in $\mathcal{S}$ lie on a m'-dimensional flat. Let $\mathcal{S}'$ be the subset of $m'+2$ such points. The affine span of $\mathcal{S}$ then has dimension at most m'. Then adding some $m-(m'+2)$ points from $\mathcal{S}$, the span of the resulting m points has dimension at most $m'+m-(m'+2) = m-2$. Hence, we have m from $\mathcal{S}$ lying on an $(m-2)$-dimensional flat, and so $\mathcal{S}$ is not m-general. □

Corollary 1.1.14. Let $\mathcal{S}$ be a set of points in $\mathrm{AG}(k,q)$. Then $\mathcal{S}$ is a $(k+1)$-general set if and only if the points of $\mathcal{S}$ are in general position.

Proof. If a set $\mathcal{S}$ is $(k+1)$-general, it is k'-general for $3 \leq k' \leq k+1$ by Theorem 1.1.13, so by Definition 1.1.9, the points in $\mathcal{S}$ are in general position. Conversely, if the points of $\mathcal{S}$ are in general position, then Definition 1.1.9 implies that no $k+1$ points of $\mathcal{S}$ lie on a $(k-1)$-dimensional flat, so the set is $(k+1)$-general. □

We see that the notion of m-general sets generalizes this of general position: By Corollary 1.1.14, the two coincide precisely for $m = k+1$, and we may have a set of points that are not in general position yet still form a m-general set for some $3 \leq m < k+1$.

[8] Denoting the size of a maximal capset in $\mathrm{AG}(k,3)$ with $r_3(k,3)$, this implies that $\lim_{k\to\infty} \frac{r_3(k,3)}{2.756^k} = 0$.

[9] We add the restriction $s \geq m$, since for every set of cardinality less than m, the set is automatically m-general by Definition 1.1.10.

1.1.3. Blocking sets and subspace-evasive sets.

In an m-general set in affine space, no m points may lie on the same $(m-2)$-dimensional flat. This notion has been further generalized by considering sets that intersect each d-dimensional flat a certain number of times [**DL11**, 1].

Definition 1.1.15. ((s,r)-subspace-evasive set). Let s and r be positive integers. A (s,r)-*subspace-evasive set* in a geometric space is a set of points that intersects every s-dimensional subspace at most r times.

Since the points of an m-general set intersect every $(m-2)$-dimensional subspace in at most $m-1$ points, it is an $(m-2, m-1)$-subspace-evasive set. Consequently, the terminology of subspace-evasive sets generalizes that of m-generality.

Example 1.1.16. Over $\mathrm{AG}(k,3)$, the capset problem is concerned with finding the largest $(1,2)$-subspace-evasive set, or equivalently, the smallest set of points that have at least one point on every line (since this blocking set will be the complement to the capset we seek).

A particular complementary object is discussed throughout literature [**BDGP23**, 1] as well.

Definition 1.1.17. ((s,m)-multiple blocking set). Let s and m be positive integers. An (s,m)-*multiple blocking set* in a geometric space is a set of points that intersects every s-dimensional subspace at least m times.

For the special case $m=1$, a s-blocking set is a set of points that contains at least one point on every s-dimensional subspace.[10]

It is clear from the above definitions that a (s,m)-blocking set is simply the complement of an $(s, D_s - m)$-evasive set, where D_s is the number of points on an s-dimensional subspace: In $\mathrm{AG}(k,q)$, we have $D_s = q^s$.[11]

General and subspace-evasive sets provide a broad framework, in which well-known objects like caps can be found. The connections described herein as well as in later sections are visualized in Graphic 2.3.4 for convenience.

One last important mathematical concept that bears connections to m-general sets, and which is of central importance to this thesis, is that of a Sidon set. Named after the Hungarian mathematician, Sidon sets originally arose in number theory, but have been applied to a variety of settings such as combinatorics, cryptography, and even harmonic analysis since. For intuition, Sidon sets are sets whose elements have pairwise distinct sums, but we will mostly utilize the standard definition given below.

[10] The authors of [**BDGP23**], whose work we borrow from for a framework of Section 2.3, use the term s-blocking set for a set of points that contains at least one point from every affine subspace of dimension $k-s$, i.e., codimension s.

[11] Also, the complement of an s-blocking set is a set of points that does not fully contain any s-dimensional subspace.

Definition 1.1.18. (Sidon set). Let $\mathcal{G}$ be an abelian group. A subset $\mathcal{A} \subseteq \mathcal{G}$ is called a *Sidon set* if the following condition holds: For group elements $a, b, c, d \in \mathcal{S}$, whenever $a + b = c + d$ holds, we have $\{a, b\} = \{c, d\}$.

Note that in the above definition, the group elements need not all be distinct. Interestingly, Sidon sets correspond to 4-general sets in $\mathrm{AG}(k, 3)$.

Theorem 1.1.19. [**HTW19**, Theorem 3.2] A subset $\mathcal{S}$ of $\mathrm{AG}(k, 3)$ is a 4-general set if and only if it is a Sidon set.

Proof. It is useful to consider the affine plane spanned by the three points $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$.

$\mathbf{a}$	$\mathbf{b}$	$-\mathbf{a} - \mathbf{b}$
$\mathbf{c}$	$-\mathbf{a} + \mathbf{b} + \mathbf{c}$	$\mathbf{a} - \mathbf{b} + \mathbf{c}$
$-\mathbf{a} - \mathbf{c}$	$\mathbf{a} + \mathbf{b} - \mathbf{c}$	$-\mathbf{b} - \mathbf{c}$

Table 1. Affine plane based on distinct and non-collinear points (a, b, c) [**HTW19**]

We prove the contrapositive of the forward direction first. Assume that $\mathcal{S}$ does not form a Sidon set in $\mathrm{AG}(k, 3)$, so there exist $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{F}_3^k$ with $\mathbf{a} + \mathbf{b} = \mathbf{c} + \mathbf{d}$ such that $\{\mathbf{a}, \mathbf{b}\} \neq \{\mathbf{c}, \mathbf{d}\}$. Then $\mathcal{S}$ contains either three distinct points $\mathbf{a}$,$\mathbf{b}$ and $\mathbf{c}$, where without loss of generality we have $\mathbf{a} + \mathbf{a} = \mathbf{b} + \mathbf{c}$, or four distinct points $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$, where without loss of generality we have $\mathbf{a} + \mathbf{b} = \mathbf{c} + \mathbf{d}$.[12] The former equality can be rewritten as

$$-\mathbf{a} = \mathbf{b} + \mathbf{c} \Leftrightarrow \mathbf{a} + \mathbf{b} + \mathbf{c} = \mathbf{0},$$

so the three points form a line by Theorem 1.1.5, and $\mathcal{S}$ is not 4-general. The latter equality implies that

$$\mathbf{d} = \mathbf{a} + \mathbf{b} - \mathbf{c},$$

so $\mathbf{d}$ lies on the affine plane spanned by the other three points, and $\mathcal{S}$ is again not 4-general.

Next we prove the contrapositive of the converse. So assume that $\mathcal{S}$ is not a 4-general set in $\mathrm{AG}(k, 3)$. Then either three points of $\mathcal{S}$ are collinear, or four points of $\mathcal{S}$ are coplanar. Now any line in $\mathrm{AG}(k, 3)$ is of the form $\mathbf{a}$, $\mathbf{b}$, $\mathbf{0} - \mathbf{a} - \mathbf{b} = -\mathbf{a} - \mathbf{b}$ for some $\mathbf{a}, \mathbf{b} \in \mathbb{F}_3^k$ by Theorem 1.1.5. So if three points in $\mathcal{S}$ are collinear, they assume the form $\mathbf{a}$, $\mathbf{b}$, $-\mathbf{a} - \mathbf{b}$, for which we have $\mathbf{b} + \mathbf{b} = -\mathbf{b} = \mathbf{a} + (-\mathbf{a} - \mathbf{b})$. Then equality of sums holds but $\{\mathbf{b}\} \neq \{\mathbf{a}, -\mathbf{a} - \mathbf{b}\}$, so $\mathcal{S}$ is not a Sidon set. Now if no three points in the set are collinear, then four points in $\mathcal{S}$ must be coplanar, and using Table 1.1.19 it can be verified that any set of 4 coplanar but non-collinear points in $\mathcal{S}$ gives rise to a labeling of points such that $\mathbf{a} + \mathbf{b} = \mathbf{c} + \mathbf{d}$, violating the Sidon property. □

[12] If all elements are equal, the Sidon property cannot be violated, and if there are precisely two distinct elements, the Sidon property cannot be violated either over $\mathbb{F}_3$, since $2\mathbf{x} = 2\mathbf{y}$ implies that $\mathbf{x} = \mathbf{y}$ in a field of characteristic larger than 2.

In $\mathrm{AG}(k,2)$, we are facing a similar situation, but since the characteristic of the field is 2, we need to be careful about the case in which $\mathbf{a}+\mathbf{a}=\mathbf{0}=\mathbf{b}+\mathbf{b}$ holds for $\mathbf{a},\mathbf{b}\in\mathbb{F}_2^k$. In this case, pairwise sums are not distinct, despite **a** and **b** possibly being points of a 4-general set. Hence, satisfying the restrictions of 4-general set is a weaker condition than those of a Sidon set in $\mathrm{AG}(k,2)$. However, we may adapt Definition 1.1.18 to read that $\mathcal{S}$ is a Sidon set if for group elements $a,b,c,d\in\mathcal{S}$, whenever $a+b=c+d$ *and* $a\neq b$, $c\neq d$ hold, we have $\{a,b\}=\{c,d\}$.[13] Then an arithmetic condition for m-general sets described by Won and Tait in [**TW21**, Theorem 2.5] can be adapted to show that, disregarding the trival solution over $\mathbb{F}_2$ described above, we maintain the equivalence between 4-general sets and Sidon sets.

These correspondences will allow us to prove new results on the size of maximal complete Sidon sets over affine space in Section 1.2.

[13] That is, we exclude the degenerate case in $\mathbb{F}_2$ from the definition.

1.2. Improving the lower bound of maximal Sidon sets in $\mathrm{AG}(k,3)$

Sidon sets have received attention in a wide array of fields, so it is natural to ask for the size of maximal Sidon sets. Since a set is 4-general in $\mathrm{AG}(k,3)$ if and only if it is a Sidon set, [**HTW19**, 3.2], we will focus on the affine space $\mathrm{AG}(k,3)$ in particular, noting that we have already encountered it as the ambient space for the capset problem and the SET game. Similar to capsets, the problem of finding large sets of points such that no four of them are coplanar has not been solved comprehensively.

1.2.1. Best-known bounds.

Denote the size of a maximal 4-general set (Sidon set) in $\mathrm{AG}(k,3)$ by $r_4(k,3)$. To find large maximal 4-general sets, consider first the case where k is even. It is a fact that $\mathbb{F}_3^k$ is additively isomorphic to $\mathbb{F}_3^{k/2} \times \mathbb{F}_3^{k/2}$, where furthermore $\mathbb{F}_3^{k/2}$ is isomorphic to $\mathbb{F}_{3^{k/2}}$ as vector spaces over $\mathbb{F}_3$. Hence, to find a maximal Sidon set in even dimension k, we simply have to find a Sidon set in the affine space with points in $\mathbb{F}_{3^{k/2}} \times \mathbb{F}_{3^{k/2}}$. In [**Cil12**, Example 1], it is shown that the set $\{(\mathbf{x}, \mathbf{x^2}) : \mathbf{x} \in \mathbb{F}_{3^{k/2}}\}$ is a Sidon set.

Theorem 1.2.1. [**Cil12**, Example 1] Let q be an odd prime power and let $p(X), r(X) \in \mathbb{F}_q[X]$ be non-constant polynomials of degree ≤ 2 such that $p(X) - \mu r(X)$ is not a constant for any $\mu \in \mathbb{F}_q$. Then the set $\mathcal{S} = \{(p(x), r(x)) : x \in \mathbb{F}_q\}$ is a Sidon set in $\mathbb{F}_q \times \mathbb{F}_q$.

In order to prove the above theorem, we first prove a lemma.

Lemma 1.2.2. Let $\mathcal{S} = \{(p(x), r(x)) : x \in \mathbb{F}_q\}$, where $p(X), r(X) \in \mathbb{F}_q[X]$. If for any $(e_1, e_2) \in \mathbb{F}_q \times \mathbb{F}_q$ with $(e_1, e_2) \neq (0,0)$, there exists at most one solution $(x_1, x_2) \in \mathbb{F}_q \times \mathbb{F}_q$ satisfying $(p(x_1), r(x_1)) - (p(x_2), r(x_2)) = (e_1, e_2)$, then $\mathcal{S}$ is a Sidon set.

Proof. We prove the contrapositive. So assume that $\mathcal{S}$ is not a Sidon set. So there exist $x_1, x_2, x_3, x_4 \in \mathbb{F}_q$ such that

$$(p(x_1), r(x_1)) + (p(x_2), r(x_2)) = (p(x_3), r(x_3)) + (p(x_4), r(x_4)),$$

where we have $\{(p(x_1), r(x_1)), (p(x_2), r(x_2)\} \neq \{(p(x_3), r(x_3)), (p(x_4), r(x_4)\}$. Without loss of generality, assume that we have

$$(p(x_1), r(x_1)) \neq (p(x_3), r(x_3)) \text{ and } (p(x_1), r(x_1)) \neq (p(x_4), r(x_4)),$$

which directly implies $x_1 \neq x_3$ and $x_1 \neq x_4$. Now

$$(p(x_1), r(x_1)) - (p(x_3), r(x_3)) = (p(x_4), r(x_4)) - (p(x_2), r(x_2),$$

so the difference $(p(x_1), r(x_1)) - (p(x_3), r(x_3)) = (e_1, e_2) \neq (0, 0)$[14] is not uniquely determined by (x_1, x_3), since $(x_4, x_2) \neq (x_1, x_3)$[15] satisfies $(p(x_4), r(x_4)) - (p(x_2), r(x_2)) = (e_1, e_2)$ as well. Hence, we find that, upon relabeling, there exists $(e_1, e_2) \neq (0, 0)$ for which the equation $(p(x_1), r(x_1)) - (p(x_2), r(x_2)) = (e_1, e_2)$ has more than one solution $(x_1, x_2) \in \mathbb{F}_q \times \mathbb{F}_q$, which completes the proof of the contrapositive of the lemma. □

Using this lemma, we can prove Theorem 1.2.1.

Proof of Theorem 1.2.1. Suppose that $p(X), r(X) \in \mathbb{F}_q[X]$ are two non-constant polynomials of degree at most 2 and that $p(X) - \mu r(X)$ is not a constant for any $\mu \in \mathbb{F}_q$. From the above lemma, if we can show that for every $(e_1, e_2) \in \mathbb{F}_q \times \mathbb{F}_q$ with $(e_1, e_2) \neq (0, 0)$, there exists at most one solution $(x_1, x_2) \in \mathbb{F}_q \times \mathbb{F}_q$ that satisfies $(p(x_1), r(x_1)) - (p(x_2), r(x_2)) = (e_1, e_2)$, it will follow that $\mathcal{S} = \{(p(x), r(x)) : x \in \mathbb{F}_q\}$ is a Sidon set.

So suppose we have $(e_1, e_2) \in \mathbb{F}_q$ with $(e_1, e_2) \neq (0, 0)$, and a pair $(x_1, x_2) \in \mathbb{F}_q \times \mathbb{F}_q$ that satisfies $(p(x_1), r(x_1)) - (p(x_2), r(x_2)) = (e_1, e_2)$. It is clear that $p(X)$ and $r(X)$ cannot both be linear, since otherwise, we may choose μ to be the quotient of the leading coefficients of $p(X)$ and $r(X)$ whence $p(X) - \mu r(X)$ would be constant. Hence, we may assume, without loss of generality, that $r(X)$ is quadratic, and that $p(X)$ is either linear or quadratic.

If $p(X)$ is linear, then we can write $p(X) = \alpha X + \beta$ for some $\alpha, \beta \in \mathbb{F}_q$ with $\alpha \neq 0$. It follows that

$$e_1 = p(x_1) - p(x_2) = \alpha x_1 + \beta - (\alpha x_2 + \beta) = \alpha(x_1 - x_2),$$

and since α is nonzero, we can relate x_1 and x_2 by $x_1 = \frac{e_1}{\alpha} + x_2$. Now since $r(X)$ is quadratic, it is of the form $r(X) = \gamma X^2 + \delta X + \epsilon$ for $\gamma, \delta, \epsilon \in \mathbb{F}_q$ with $\gamma \neq 0$. Then

$$\begin{aligned} e_2 = r(x_1) - r(x_2) =& \gamma x_1^2 + \delta x_1 + \epsilon - (\gamma x_2^2 + \delta x_2 + \epsilon) = \gamma(x_1^2 - x_2^2) + \delta(x_1 - x_2) = \\ & \gamma\left(\left(\frac{e_1}{\alpha} + x_2\right)^2 - x_2^2\right) + \delta\left(\left(\frac{e_1}{\alpha} + x_2\right) - x_2\right) = \gamma\left(\frac{e_1^2}{\alpha^2} + 2\frac{e_1}{\alpha}x_2\right) + \frac{\delta e_1}{\alpha}. \end{aligned}$$

Now $\gamma \neq 0$, so we may divide by γ when solving the above for x_2, which yields the unique solution $x_2 \in \mathbb{F}_q$.[16] Using $x_1 = \frac{e_1}{\alpha} + x_2$, we then find the unique solution x_1. Hence, if there exists a pair $(x_1, x_2) \in \mathbb{F}_q \times \mathbb{F}_q$ that satisfies $(p(x_1), r(x_1)) - (p(x_2), r(x_2)) = (e_1, e_2) \neq (0, 0)$ in the case where $r(X)$ is quadratic and $p(X)$ is linear, then this solution is unique, so there exists at most this one solution.

If $p(X)$ is quadratic, then we can write $p(X) = \alpha X^2 + \beta X + \gamma$ for some $\alpha, \beta, \gamma \in \mathbb{F}_q$ with $\alpha \neq 0$. Now in order to reduce this case to the first, choose μ so that $(p - \mu r)(X) \in \mathbb{F}_q[X]$ is linear. Then $(p - \mu r)(X) = \delta X + \epsilon$, for some $\delta, \epsilon \in \mathbb{F}_q$. Since we are assuming that $(p - \mu r)(X)$ is non-constant,

[14] The inequality follows directly from our observation that $(p(x_1), r(x_1)) \neq (p(x_3), r(x_3))$.

[15] The inequality follows directly from our observation that $x_1 \neq x_4$.

[16] Note that if $e_1 = 0$, this forces $e_2 = 0$, which we exclude by assumption.

we have $\delta \neq 0$. But recall that we have $(p(x_1), r(x_1)) - (p(x_2), r(x_2)) = (e_1, e_2) \neq (0, 0)$, so in particular $(p(x_1), \mu r(x_1)) - (p(x_2), \mu r(x_2)) = (e_1, \mu e_2)$, and hence

$$(p - \mu r)(x_1) - (p - \mu r)(x_2) = e_1 - \mu e_2,$$

where $(p - \mu r)(X)$ is a linear polynomial and $e_1 - \mu e_2 \neq 0$.[17] In complete analogy to the case that $p(X)$ is linear, we can then define a relation between x_1 and x_2 from the equation

$$e_1 - \mu e_2 = (p - \mu r)(x_1) - (p - \mu r)(x_2) = \delta x_1 + \epsilon - (\delta x_2 + \epsilon) = \delta(x_1 - x_2),$$

where $\delta \neq 0$ such that $x_1 = \frac{e_1 - \mu e_2}{\delta} + x_2$. With this, we can (exactly as in the first case) uniquely extract x_2 from the quadratic function $r(X)$ and the relation

$$e_2 = r(x_1) - r(x_2) = r\left(\frac{e_1 - \mu e_2}{\delta} + x_2\right) - r(x_2).^{18}$$

Then $x_1 = \frac{e_1 - \mu e_2}{\delta} + x_2$ uniquely determines x_1 as well. Hence, if there exists a pair $(x_1, x_2) \in \mathbb{F}_q \times \mathbb{F}_q$ that satisfies $(p(x_1), r(x_1)) - (p(x_2), r(x_2)) = (e_1, e_2) \neq (0, 0)$ in the case where $r(X)$ and $p(X)$ are both quadratic, then this solution is unique, so there exists at most this one solution.

Hence for any given functions $p(X)$ and $r(X)$ that satisfy the assumptions of the theorem, we see that for every $(e_1, e_2) \in \mathbb{F}_q \times \mathbb{F}_q$ with $(e_1, e_2) \neq (0, 0)$, there exists at most one solution $(x_1, x_2) \in \mathbb{F}_q \times \mathbb{F}_q$ that satisfies $(p(x_1), r(x_1)) - (p(x_2), r(x_2)) = (e_1, e_2)$. So by Lemma 1.2.2, it follows that $\mathcal{S} = \{(p(x), r(x)) : x \in \mathbb{F}_q\}$ is a Sidon set for such functions $p(X)$ and $r(X)$. □

COROLLARY 1.2.3. The set $\mathcal{S} = \{(\mathbf{x}, \mathbf{x^2}) : \mathbf{x} \in \mathbb{F}_q\}$ is a Sidon set.

PROOF. This follows directly from Theorem 1.2.1. □

The authors of [**HTW19**] show that for any Sidon set $\mathcal{S}$ in AG$(k, 3)$, we have the upper bound $|\mathcal{S}|(|\mathcal{S}| - 1) \leq 3^k - 1$ on the size of $\mathcal{S}$, which implies that the $\{(\mathbf{x}, \mathbf{x^2})\}$ construction is maximal in even dimension k. Consequently, as concluded in [**HTW19**, Theorem 3.4], we have $r_4(k, 3) = 3^{k/2}$ for even k.

In odd dimension however, the authors of [**HTW19**] did not attempt to construct large Sidon sets, instead providing the trivial lower bound $r_4(k, 3) \geq 3^{(k-1)/2} + 1$ by taking a maximal Sidon set in dimension $k - 1$, embedding it in dimension k, and adding an affinely independent point. Also, the exact size of a maximal Sidon set in odd dimension is known only when $k = 3$ and $k = 5$, with a non-trivial lower bound computed when $k = 7$ (through computational search). Table 2 recreates the bounds provided in [**HTW19**] with additional explicit data in dimensions greater than 8.

[17] If to the contrary $e_1 - \mu e_2 = 0$, then we had $(p - \mu r)(x_1 - x_2) = 0$, but $x_1 - x_2 \neq 0$ by the initial assumption that $(e_1, e_2) \neq (0, 0)$, so $(p - \mu r)(X)$ had at least the two roots 0 and $x_1 - x_2$ in $\mathbb{F}_q[X]$, which contradicts $(p - \mu r)(X)$ being linear, since a linear polynomial cannot have two distinct roots.

[18] Recall that we have $e_1 - \mu e_2 \neq 0$.

Dimension k	1	2	3	4	5	6	7	8	9	10	11	12	13	k even	k odd
$r_4(k,3) \geq$	2	3	5	9	13	27	33	81	82	243	244	729	730	$3^{k/2}$	$3^{(k-1)/2}+1$
$r_4(k,3) \leq$	2	3	5	9	13	27	47	81	140	243	421	729	1263	$3^{k/2}$	$\lceil 3^{k/2} \rceil$

TABLE 2. Best bounds on 4-general sets in AG$(k,3)$ as derived from [**HTW19**].[19]

We are therefore interested in improving the lower bound for a maximal Sidon set in odd dimension, and present a new construction for this in the next section.

1.2.2. A new construction.

We first observe that AG$(k,3)$ may be partitioned into three hyperplanes, each of which is isomorphic to AG$(k-1,3)$. Although this can be done in many ways, we consider those points in AG$(k,3)$ whose last coordinate is equal to $i \in \{0,1,2\}$, which we call the *i-hyperplane*. Then AG$(k,3)$ is the disjoint union of the 0-hyerplane, 1-hyperplane, and 2-hyperplane.

If k is odd, then in the 0-hyerplane, using the construction described above, we can construct a Sidon set of the form $\{(\mathbf{x},\mathbf{x^2},0) : \mathbf{x} \in \mathbb{F}_{3^{(k-1)/2}}\}$ in AG$(k,3)$. Following directly from Definition 1.1.18, this is a Sidon set in AG$(k,3)$. It is clear that we may add any point in the 1-hyperplane while retaining the Sidon property, by looking at the last coordinate in the sums.[20] Our aim is to add more than one point to the additional two hyperplanes of AG$(k,3)$ in a systematic way while maintaining the Sidon property.

THEOREM 1.2.4. For odd k with $k > 1$, we have the inequality

$$r_4(k,3) \geq 3^{(k-1)/2} + |\mathcal{S}|,$$

where $\mathcal{S} \subseteq \mathbb{F}_3^{(k-1)/2}$ is a Sidon set in the affine space AG$(\frac{k-1}{2},3)$.

The above Theorem claims that we can add points from some lower-dimensional Sidon set to the other two hyperplanes while maintaining Sidon property. In order to prove this improved lower bound, we give an explicit construction and prove its Sidon property.

PROOF OF THEOREM 1.2.4. We claim that the union

$$\mathcal{X} = \{(\mathbf{x},\mathbf{x^2},0) : \mathbf{x} \in \mathbb{F}_{3^{(k-1)/2}}\} \cup \{(\mathbf{0},\mathbf{y},1) : \mathbf{0} \in \mathbb{F}_3^{(k-1)/2}, \mathbf{y} \in \mathcal{S}\}$$

forms a Sidon set, i.e., that for any four elements $\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}$ in $\mathcal{X}$, the equality $\mathbf{a}+\mathbf{b} = \mathbf{c}+\mathbf{d}$ implies $\{\mathbf{a},\mathbf{b}\} = \{\mathbf{c},\mathbf{d}\}$. There are five cases to consider when choosing four elements from $\mathcal{X}$:

Case 1: $\mathbf{s_1},\mathbf{s_2},\mathbf{s_3},\mathbf{s_4} \in \{(\mathbf{x},\mathbf{x^2},0) : x \in \mathbb{F}_{3^{(k-1)/2}}\}$.

Since the points in $\{(\mathbf{x},\mathbf{x^2},0) : \mathbf{x} \in \mathbb{F}_{3^{(k-1)/2}}\}$ form a Sidon set, it is trivial that if $\mathbf{s_1}+\mathbf{s_2} = \mathbf{s_3}+\mathbf{s_4}$, then $\{\mathbf{s_1},\mathbf{s_2}\} = \{\mathbf{s_3},\mathbf{s_4}\}$.

[19] To the best of our knowledge, these bounds are the best-known results on 4-general sets in AG$(k,3)$.

[20] As mentioned above, this is in fact how the trivial bound in [**HTW19**] is obtained.

Case 2: $\mathbf{s_1}, \mathbf{s_2}, \mathbf{s_3} \in \{(\mathbf{x}, \mathbf{x^2}, 0) : \mathbf{x} \in \mathbb{F}_{3^{(k-1)/2}}\}$ and $\mathbf{s_4} \in \{(\mathbf{0}, \mathbf{y}, 1) : \mathbf{y} \in \mathcal{S}\}$.

Since the last coordinate of any pairwise sum differs for the above set, the sums are forced to be distinct.

Case 3: $\mathbf{s_1} \in \{(\mathbf{x}, \mathbf{x^2}, 0) : \mathbf{x} \in \mathbb{F}_{3^{(k-1)/2}}\}$ and $\mathbf{s_2}, \mathbf{s_3}, \mathbf{s_4} \in \{(\mathbf{0}, \mathbf{y}, 1) : \mathbf{y} \in \mathcal{S}\}$.

As in the second case, since the last coordinate of any pairwise sum differs for the above set, the sums are again forced to be distinct.

Case 4: $\mathbf{s_1}, \mathbf{s_2}, \mathbf{s_3}, \mathbf{s_4} \in \{(\mathbf{0}, \mathbf{y}, 1) : \mathbf{y} \in \mathcal{S}\}$.

Similar to the first case, this case is again trivial: The points satisfy the Sidon property since $\mathcal{S}$ is a Sidon set.

Case 5: $\mathbf{s_1}, \mathbf{s_2} \in \{(\mathbf{x}, \mathbf{x^2}, 0) : \mathbf{x} \in \mathbb{F}_{3^{(k-1)/2}}\}$ and $\mathbf{s_3}, \mathbf{s_4} \in \{(\mathbf{0}, \mathbf{y}, 1) : \mathbf{y} \in \mathcal{S}\}$.

This is the only non-trivial case. In order for the two sums to be equal, the last coordinate needs to match. Hence, we must have that $\mathbf{s_1} + \mathbf{s_3} = \mathbf{s_2} + \mathbf{s_4}$, i.e.,

$$(\mathbf{x_1}, \mathbf{x_1^2}, 0) + (\mathbf{0}, \mathbf{y_1}, 1) = (\mathbf{x_2}, \mathbf{x_2^2}, 0) + (\mathbf{0}, \mathbf{y_2}, 1).$$

This directly forces $\mathbf{x_1} = \mathbf{x_2}$ and $\mathbf{x_1^2} + \mathbf{y_1} = \mathbf{x_2^2} + \mathbf{y_2}$. Using the former equality, the latter reduces to $\mathbf{x_1^2} + \mathbf{y_1} = \mathbf{x_1^2} + \mathbf{y_2}$, which happens if and only if $\mathbf{y_1} = \mathbf{y_2}$. Thus, we have $\mathbf{x_1} = \mathbf{x_2}$ and $\mathbf{y_1} = \mathbf{y_2}$, which implies $\mathbf{s_1} = \mathbf{s_2}$ and $\mathbf{s_3} = \mathbf{s_4}$. Hence, $\mathbf{s_1} + \mathbf{s_3} = \mathbf{s_2} + \mathbf{s_4}$ forces $\{\mathbf{s_1}, \mathbf{s_3}\} = \{\mathbf{s_2}, \mathbf{s_4}\}$, which shows that this case satisfies the Sidon property as well. □

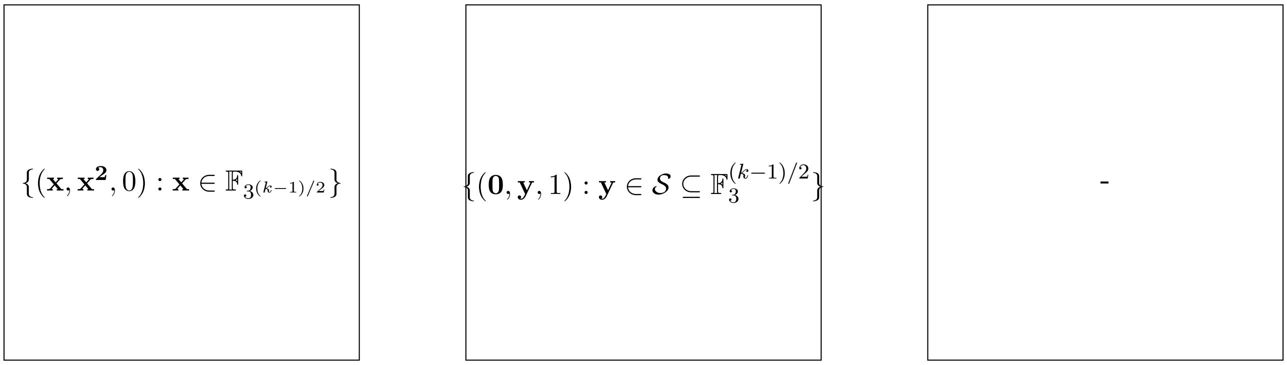

$\{(\mathbf{x}, \mathbf{x^2}, 0) : \mathbf{x} \in \mathbb{F}_{3^{(k-1)/2}}\}$
$\{(\mathbf{0}, \mathbf{y}, 1) : \mathbf{y} \in \mathcal{S} \subseteq \mathbb{F}_3^{(k-1)/2}\}$
-

FIGURE 2. Sidon set construction in AG$(k, 3)$ for odd k yielding the Sidon set

$$\mathcal{X} = \{(\mathbf{x}, \mathbf{x^2}, 0) : \mathbf{x} \in \mathbb{F}_{3^{(k-1)/2}}\} \cup \{(\mathbf{0}, \mathbf{y}, 1) : \mathbf{0} \in \mathbb{F}_3^{(k-1)/2}, \mathbf{y} \in \mathcal{S}\}.$$

1.2.2.1. *Analysis of the new lower bound.* Note that this construction, paired with our knowledge on the exact size of maximal Sidon sets in even dimensions, gives us a recursive way to determine new lower bounds for maximal Sidon sets in odd dimension. Before developing an explicit formula for this bound, we provide pseudocode to construct a large Sidon set in $\mathrm{AG}(k,q)$ in Algorithm 1).

Algorithm 1 Finding large Sidon set sizes in $\mathrm{AG}(k,3)$

```
Require: dimension $k$
  maxSize $\leftarrow 0$
  while $1 \geq 0$ do
      if $k$ is even then
          return maxSize $+ 3^{k/2}$
      else if $k = 3$ then
          return maxSize $+ 5$
      else if $k = 5$ then
          return maxSize $+ 13$
      else if $k = 7$ then
          return maxSize $+ 33$
      else
          $k \leftarrow (k-1)/2$
          maxSize $\leftarrow$ maxSize $+ 3^k$
      end if
  end while
  return maxSize
```

The above captures how we find lower bounds in odd dimensions. If k is even, then the algorithm outputs $3^{k/2}$, which is the size of a maximal Sidon set. If k is odd, we add $3^{k-1/2}$ to the running sum and start a new iteration of the loop on dimension $(k-1)/2$. Eventually, this procedure will terminate when k is even (for which we know the exact size of a maximal Sidon set), or k is either 3 or 5, which are the odd dimensions in which the exact size of $r_4(k,3)$ is known.[21]

For example, if $k = 10 \equiv 0 \pmod 2$, Algorithm 1 outputs a lower bound of 3^5 (which is tight), and if $k = 39 \equiv 7 \pmod{16}$, it outputs a lower bound of $3^{19} + 3^9 + 3^4 + 3^2$.

We are interested in a closed-form lower bound from Theorem 1.2.4 for odd k as well. Therefore, we first simplify the above algorithm by reducing it to two base cases: k being even and $k = 1$.

[21] We know the exact size for $k = 1$ as well, but note that the case $k = 1$ would never materialize, since it results from the stopping condition $k = 3$.

This gives slightly worse lower bounds than Theorem 1.2.4,[22] but will turn out to be helpful for the following argument.

Algorithm 2 Finding large Sidon set sizes in $\mathrm{AG}(k, q)$ (simplified)

```
Require: dimension $k$
  maxSize $\leftarrow 0$
  while $1 \geq 0$ do
      if $k$ is even then
          return maxSize $+ 3^{k/2}$
      else if $k = 1$ then
          return maxSize $+ 2$
      else
          $k \leftarrow (k-1)/2$
          maxSize $\leftarrow$ maxSize $+ 3^k$
      end if
  end while
  return maxSize
```

We give some special cases of the algorithm's output below.

- $k = 2s \rightarrow 3^{k/2}$
- $k = 4s + 1 \rightarrow 3^{(k-1)/2} + 3^{(k-1)/4}$
- $k = 8s + 3 \rightarrow 3^{(k-1)/2} + 3^{(k-3)/4} + 3^{(k-3)/8}$
- $k = 16s + 7 \rightarrow 3^{(k-1)/2} + 3^{(k-3)/4} + 3^{(k-7)/8} + 3^{(k-7)/16}$
- $\ldots$
- $k = 2^t - 1 \rightarrow 3^{(k-1)/2} + 3^{(k-3)/4} + 3^{(k-7)/8} + 3^{(k-15)/16} + \ldots + 3^{(k-(2^t-1))/2^t} + 1$

Observe that the difficulty of transforming the recursive procedure above lies in determining at which step $i \geq 0$, for any given odd initial dimension $k = k_0$, the integer $(k_i - 1)/2$ in the recursive step of the algorithms will be even, such that we can apply our knowledge of the size of Sidon sets in even dimension. A convenient way to determine this is to express k in binary representation and define t as the number of trailing 1s in this binary representation. Some cases of interest are given below.

- $k = 2s \rightarrow t = 0 \;\; (10 = 1010_b)$
- $k = 4s + 1 \rightarrow t = 1 \;\; (5 = 101_b)$
- $k = 8s + 3 \rightarrow t = 2 \;\; (11 = 1011_b)$
- $k = 16s + 7 \rightarrow t = 3 \;\; (23 = 10111_b)$
- $\ldots$
- $k = 2^t - 1 \rightarrow t = \log_2(k+1) \;\; (15 = 1111_b)$

[22] In particular, the lower bound results of Algorithm 2 match those of Algorithm 1 in every case except from when the recursion encounters the case $k = 5$. Here, Algorithm 1 adds $r_4(5, 3) = 13$ to the running sum, while Algorithm 2 adds $3^2 + 3^1 = 12$ to running sum.

Then inspection shows that the output of the recursive Algorithm 2 is equivalent to the closed-form lower bound

$$r_4(k,3) \geq \left(\sum_{i=1}^{t} 3^{\frac{k-(2^i-1)}{2^i}}\right) + 3^{\frac{k-(2^t-1)}{2^{t+1}}} = \frac{1}{3}\left(\sum_{i=1}^{t} \sqrt[2^i]{3^{k+1}}\right) + \frac{1}{\sqrt{3}} \sqrt[2^{t+1}]{3^{k+1}},$$

where the recursive nature of the bound is masked by prior determination of t. The formula precisely replicates Algorithm 2 and thus gives our best bounds for all dimensions except for the ones in which the recursion encounters the case $k = 5$ or $k = 7$, i.e., the cases where the binary representation of k starts with the pattern 101 or 111, respectively, followed by trailing 1s.[23] In these cases, we may add one point to the output of the formula. As an example, consider the case $k = 7 = (111)_2$, for which $t = 3$, and compare to the best known bounds given in Table 2.

Example 1.2.5.

$$r_4(7,3) \geq \frac{1}{3}\left(\sum_{i=1}^{3} \sqrt[2^i]{3^{7+1}}\right) + \frac{1}{\sqrt{3}} \sqrt[2^{3+1}]{3^{7+1}} = \frac{1}{3}\left(\sum_{i=1}^{3} \sqrt[2^i]{3^8}\right) + \frac{1}{\sqrt{3}} \sqrt[16]{3^8} =$$
$$\frac{1}{3}\left(\sqrt[2]{3^8} + \sqrt[4]{3^8} + \sqrt[8]{3^8}\right) + \frac{1}{\sqrt{3}} \sqrt[16]{3^8} = \frac{1}{3}(81 + 9 + 3) + \frac{\sqrt{3}}{\sqrt{3}} = 32.$$

1.2.2.2. *Asymptotics.* As is the case for many extremal problems, we are also interested in the asymptotic behaviour of $r_4(k,3)$ as k tends to infinity. Recall some notation from complexity theory: For real-valued functions f and g, we write

$$f(n) = \mathcal{O}(g(n)) \text{ if } \limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty \text{ (Big O) and}$$
$$f(n) = \Omega(g(n)) \text{ if } \liminf_{n\to\infty} \frac{f(n)}{g(n)} > 0 \text{ (Big Omega).}$$

In [**HTW19**], the authors show that $r_4(k,3) = \mathcal{O}(3^{k/2})$, and their results further imply that $r_4(k,3) = \Omega(3^{(k-1)/2})$ holds. Since the lower bound guarantees for odd dimensions are always worse than for even, consider odd k. Our weakest lower bound holds for $k = 2^n - 1$, where $n \geq 1$. Here, we have

$$r_4(k,3) \geq \frac{1}{3}\Big(\sum_{i=1}^{\log_2(k+1)} \sqrt[2^i]{3^{k+1}}\Big) + 1 = g(k)$$

and

$$\liminf_{k\to\infty} \frac{r_4(k,3)}{g(k)} > 0,$$

so we obtain

$$r_4(k,3) = \Omega(\frac{1}{3}\Big(\sum_{i=1}^{\log_2(k+1)} \sqrt[2^i]{3^{k+1}}\Big) + 1).$$

[23] The latter of which corresponds to a dimension of the form $k = 2^s - 1$ for some positive integer $s \geq 3$.

But note that

$$\Omega(\frac{1}{3}\Big(\sum_{i=1}^{\log_2(k+1)} \sqrt[2^i]{3^{k+1}}\Big) + 1) = \Omega(3^{(k-1)/2}),$$

so even though $g(k) \geq 3^{(k-1)/2} + 1$ holds, our construction does not improve the asymptotic lower bound according to Ω complexity considerations. The asymptotic behavior of $r_4(k, 3)$ thus remains an open question.

Note that our construction only uses one additional hyperplane (we referred to it as the 1-hyperplane), as visualized in Graphic 1.2.4. Further thought should be devoted to the question whether or not we can add points to the third hyperplane as well, or whether we can apply some sort of splitting amongst the two hyperplanes.[24]

1.2.3. Further improvements through computational search and new lower bounds.

Generally, computational approaches to finding new large m-general sets are only feasible in low dimensions, since the search space grows exponentially. For a rough estimate of the extent of those spaces, assume that we employ brute force computation to check each possible subsets of the search space. There are $\approx 2^{3^k}$ such subsets of points for $\mathbb{F}_3^k$, so in dimension $k = 7$, this amounts to $\approx 2^{3^7} = 2^{2187}$ subsets. Rewriting this in base 10,[25] we find that we have $\approx 2 \cdot 10^{658}$ subsets, each of which needs to be tested for m-generality in the worst-case. Comparing this to the number of baryons in the observable universe, which is estimated to be on the order of $\approx 10^{80}$, shall serve as example that traversing the entire search space, even for moderately small dimensions, is an intractable task.

Instead, we can choose to initialize a search with a set of points that is guaranteed to be m-general, which reduces the search space drastically. This way, we can hope to obtain new lower bounds in small dimensions, which however might not be guaranteed to be optimal (depending on the type of initialization). We provide pseudocode for this computational approach below.

[24] The computational results in small dimension below suggest that the sets resulting from our construction are complete but not maximal in sufficiently small dimensions, so it might be worthwhile to explore an entirely different construction.

[25] $\log_{10}(2^{2187}) = 2187 \log_{10}(2) \approx 658.34$

Algorithm 3 Finding large complete 4-general sets in $\mathrm{AG}(k, q)$

```
Require: dimension $k$, field order $q$
  maxSize $\leftarrow$ 0
  maxSet $\leftarrow$ [ ]
  Determine coefficient vectors for affine combinations
  set $\leftarrow$ {Initialize starting set}
  possible $\leftarrow$ {Determine points that may be added}
  call buildCap(set, possible)

  function BUILDCAP(set, possible)
      if len(possible) = 0 & len(set) > maxSize then
          maxSize $\leftarrow$ len(set)
          maxSet $\leftarrow$ set
      else
          for i in range(0, len(possible)) do
              set.append(possible[i])
              possible $\leftarrow$ {Determine new possible points}
              call buildCap(set, possible)
          end for
      end if
  end function
```

In general, there is a multitude of options to choose from for the initialization of a search.

- Empty initialization: Here, we traverse the entire search space, which is very inefficient.
- Initializing with $k+1$ affinely independent points: This gives better runtime and yields a comprehensive search. We employ this method in Section 1.3.2 to find complete m-general sets for different values of m and over different fields.
- Leverage the Sidon set construction from Theorem 1.2.4: This gives a good runtime, but restricts the search space considerably.
- Leverage the $(\mathbf{x}, \mathbf{x^2})$ construction: This initialization is less restrictive, but results in worse runtime than the above.
- Combinations of the above approaches

In Code Excerpt 0.0.3, we provide an explicit Python implementation of the initialization according to our construction from Theorem 1.2.4.

For our search of Sidon sets in $\mathrm{AG}(k, 3)$, we choose two different types of such initializations, starting from (i) a maximal Sidon set in lower dimension as implemented in `maximal-sidon-set-odd_dimension.py` and (ii) starting from a maximal Sidon set according to Theorem 1.2.4 as used in `maximal-sidon-set-new_construction.py`. For the special case $k \equiv 1 \pmod 4$, we additionally run a hybrid version of the two, as implemented in `maximal-sidon-set-odd_dimension_hybrid.py`. In all cases, the initialization scripts pass to the Python script `m-general-sets-AG-max_search.py`, which determines and traverses the remaining search space, trying to extend the given set as much as possible while maintaining the Sidon property. All computations were conducted on the Pegasus High Performance Computing Cluster at The George Washington University, and all code is accessible on the author's GitHub page at `https://github.com/timneumann1/Senior-Thesis-m-general-sets`.

Similar to the authors of [**HTW19**], who obtained a large Sidon set for $k = 7$ through computational search, using initialization (i) we find that a maximal Sidon set in $\mathrm{AG}(9, 3)$ must have size at least 91, compared to 82 (previous best lower bound), thereby surpassing the lower bound of 90 from Theorem 1.2.4. Similarly, we provide a maximal Sidon set in $\mathrm{AG}(11, 3)$ of size at least 256, compared to 244 (previous best lower bound), which matches the lower bound given by Theorem 1.2.4. Lastly, we construct a maximal Sidon set in $\mathrm{AG}(13, 3)$ of size 756 using initialization (ii) as well as the hybrid approach, which exceeds the trivial lower bound of 744. Since initialization (i) provides less points initially, the search space here is simply too large to surpass those results.

Hence, our computational approach helped us to increase the lower bound on $r_4(9, 3)$ to 91. The lexicographically first such Sidon set is given explicitly in the code output sample 0.0.4.

Interestingly, initializing the search with (ii) yields complete Sidon set for all odd $k \leq 13$. However, this may not hold true for larger k, so it remains an open question whether or not the new construction from Theorem 1.2.4 gives complete Sidon sets or not.

Using Theorem 1.2.4 and our computational results, we provide a comprehensive overview of lower bounds on $r_4(k, 3)$, comparing results from [**HTW19**], our improved lower bound, and our computational approach.

k	1	2	3	4	5	6	7	8	9	10	11	12	13
Lower Bound [**HTW19**]	2	3	5	9	13	27	33	81	82	243	244	729	730
Lower Bound 1.2.4	2	3	5	9	13	27	33	81	**90**	243	**256**	729	**756**
Lower Bound (Computational)	2	3	5	9	13	27	33	81	**91**	243	256	729	756

TABLE 3. Comparison of lower bounds on $r_4(k, 3)$

We summarize the best bounds in Table 4, where lower bounds follow from the results of this section and upper bounds follow from [HTW19, Proposition 3.3]

k	1	2	3	4	5	6	7	8	9	10	11	12	13	k even	k odd
$r_4(k,3) \geq$	2	3	5	9	13	27	33	81	91	243	256	729	756	$3^{k/2}$	Theorem 1.2.4
$r_4(k,3) \leq$	2	3	5	9	13	27	47	81	140	243	421	729	1263	$3^{k/2}$	$\lceil 3^{k/2} \rceil$

TABLE 4. Computational results on 4-general sets in $\mathrm{AG}(k,3)$

1.2.4. Considerations for 4-general set constructions in $\mathrm{AG}(k,2)$.

Having seen a construction of 4-general sets (Sidon sets) in $\mathrm{AG}(k,q)$, it is natural to ask whether or not the above arguments apply to $\mathrm{AG}(k,2)$ as well. Sidon sets in $\mathrm{AG}(k,2)$, in particular, have recently received attention in [CFG+22, CP23, RRW22] and throughout literature. Table 5 summarizes the dimensions in which the maximal size of a Sidon set in $\mathrm{AG}(k,2)$ are known.[Nag22, 5], and a general lower bound for the even and odd case.

Dimension k	1	2	3	4	5	6	7	8	9	10	k even	k odd
Size of maximal complete set	2	3	4	6	7	9	12	18	24	34	$2^{k/2}$	$2^{(k-1)/2}+1$

TABLE 5. Bounds on 4-general sets in $\mathrm{AG}(k,2)$

Interestingly, in even dimension it can be shown that a construction similar to the $(\mathbf{x}, \mathbf{x^2})$ method for $\mathrm{AG}(k,3)$ works in $\mathrm{AG}(k,2)$. We refer to [TW21, Section 5] for a proof that the set $\{(\mathbf{x}, \mathbf{x^3}) : \mathbf{x} \in \mathbb{F}_{3^{k/2}}\}$ forms a Sidon set in this case, where the additive isomorphism between $\mathbb{F}_2^n$ and $\mathbb{F}_{2^{n/2}} \times \mathbb{F}_{2^{n/2}}$ is used. Note however that the resulting Sidon set is *not* necessarily maximal, as was the case previously. Hence, for even dimensions k, we have the lower bound $2^{k/2}$, which is not necessarily tight, and for odd k, we have the lower bound $2^{(k-1)/2}+1$ for maximal Sidon sets in $\mathrm{AG}(k,2)$. Now a parallel argument to the one above shows that our construction of $\mathcal{X}$ above satisfies the Sidon property in the cases $(i), (ii), (iii)$ and (iv) of Proof 1.2.4. Note however that case (vi) splits into two cases again over $\mathbb{F}_2$:

For odd k, consider four points in $\mathcal{X} \subseteq \mathbb{F}_2^k$ such that $s_1 = (x_1, x_1^3, 0), s_2 = (x_2, x_2^3, 0)$ and $s_3 = (0, y_1, 1), s_4 = (0, y_2, 1)$, with $x_i, y_i \in \mathbb{F}_2^{(n-1)/2}$. In order for two pairwise sums to equate, the last coordinate needs to match. Over $\mathbb{F}_2$, this yields two possible combinations. Firstly, without loss of generality, assume that $s_1 + s_3 = s_2 + s_4$, i.e., $(x_1, x_1^3, 0) + (0, y_1, 1) = (x_2, x_2^3, 0) + (0, y_2, 1)$. This forces $x_1 = x_2$ and $x_1^3 + y_1 = x_2^3 + y_2$. Using the former equality, the latter reduces to $x_1^3 + y_1 = x_1^3 + y_2 \Leftrightarrow y_1 = y_2$. Thus, we have $x_1 = x_2$ and $y_1 = y_2$, which implies $s_1 = s_2$ and $s_3 = s_4$. Hence, $s_1 + s_3 = s_2 + s_4$ forces $\{s_1, s_3\} = \{s_2, s_4\}$.

Secondly, without loss of generality, assume that $s_1 + s_2 = s_3 + s_4$, i.e., $(x_1, x_1^3, 0) + (x_2, x_2^3, 0) = (0, y_1, 1) + (0, y_2, 1)$. This forces $x_1 = -x_2 = x_2$ (since we are working over $\mathbb{F}_2$), and $x_1^3 + x_2^3 = y_1 + y_2$. Using the former equality, the LHS of the latter equality reduces to $x_1^3 + x_1^3 = 2x_1^3 = 0$, so $y_1 = -y_2 = y_2$, so again we have $x_1 = x_2$ and $y_1 = y_2$, which implies $s_1 = s_2$ and $s_3 = s_4$. Hence, $s_1 + s_2 = s_3 + s_4$ does not force $\{s_1, s_3\} = \{s_2, s_4\}$, so the Sidon property is *not* maintained.

However, the above equalities precisely describe the trivial solution described in Section 1.1,[26] so our construction yields a Sidon set that has an additional constraint on the summands, and therefore a 4-general set.

The lower bound that results from this construction is analogous to the one provided for $\mathrm{AG}(k, 3)$, where we replace the 3 in the base of the exponents with a 2. This improves the trivial lower bound in odd dimensions. Our computational results using the $(\mathbf{x}, \mathbf{x^3}, 0)$ construction also exceed the trivial bounds, but do not improve upon the best-known bounds for maximal Sidon sets in low dimensions. Here, in low dimensions the authors of [**CP23**] notably improved some bounds based on an interesting correspondence with linear codes, and comparison with the best-known results of such codes [**CP23**, Table 2]. It would be interesting to see if similar considerations can yield improvements to the bounds on $r_4(k, 3)$. In Section 2.3, we will extensively treat the correspondence between m-general sets and coding theory over projective spaces.

1.2.5. Using Large Language Models to solve problems in mathematics.

To conclude this section on Sidon sets in affine space, we come back to $\mathrm{AG}(k, 3)$ and extend our theoretical and computational approaches to a third one, which offers an unexpected but fascinating angle at the problem in question. With the rise of Artificial Intelligence (AI), many promising lines of research have opened and new applications come within reach. For example, Neural networks demonstrated their ability to beat the best human Chess and Go players as described in [**SHS$^+$17, SHM$^+$16**], and Generative Pre-trained Transformers (GPT) [**BMR$^+$20**] created an anthropomorphic conversation setting. There has also been great usage of new methods of AI in the sciences, such as in molecular research in biology, where a DeepMind Neural Network demonstrated the ability to predict protein structures [**JEP$^+$21**]. Mathematics, however, has long been considered to be some sort of stronghold against the advances in Artificial Intelligence. While computers have been used for large-scale searches to assist in proving conjectures before,[27] and mathematical proofs can, upon transformation to an appropriate format, be checked for logical coherence by computers, the creative aspect of original discovery in mathematics has seemed to elude the digital world so far. Recently, however, a team from Google DeepMind made a contribution

[26] $\mathcal{S}$ is a Sidon set if for group elements $a, b, c, d \in \mathcal{S}$, whenever $a + b = c + d$ *and* $a \neq b$, $c \neq d$ hold, we have $\{a, b\} = \{c, d\}$.

[27] such as the Four color Theorem

to the capset problem that we encountered earlier, using a Large Language Model whose output provides creative insight into the problem.

1.2.5.1. *A breakthrough paper on the capset problem.* Recall that the capset problem asks for the maximal size of a set of points in AG$(k, 3)$ such that no three are collinear. Denote this quantity by $r_3(k, 3)$. As mentioned in Section 1.1, the exact value of $r_3(k, 3)$ is only known for $k \leq 6$, but a strong asymptotic result exists.[**EG16**] Still, interest in finding new sizes for large capsets has not faded. The latest advancement on the capset problem, however, was quite surprising: a DeepMind research team around lead researcher Bernardino Romera-Paredes improved the best-known lower bound of a capset in dimension $k = 8$ from 496 to 512 using Google Codey – a Large Language Model developed by Google – paired with an evolutionary algorithm. As detailed in their paper "Mathematical discoveries from program search with large language models" [**RPBN$^+$24**], LLMs can indeed be used to discover creative new solutions in mathematics, here in particular in extremal combinatorics. The main challenge hereby is described as follows:

> Large language models (LLMs) have demonstrated tremendous capabilities in solving complex tasks, [. . . but] sometimes suffer from confabulations [. . .]. This hinders the use of current large models in scientific discovery.

DeepMind's solution to this problem, a new approach named *FunSearch*,[28] is an evolutionary approach "pairing a pretrained LLM with a systematic evaluator"[**RPBN$^+$24**].

At the core of FunSearch is a priority function. While the team tackled a number of open problems in mathematics, in the specific case of the capset problem, this priority function $\mathbb{Z}_3^k \to \mathbb{R}$ assigns every point in affine space a priority to be added to a set. After priorities are assigned, the points are added to an initially empty set in descending order of priorities. After every addition, points that can no longer be added without violating the capset property (no three on a line) are discarded, and out of the remaining possible choices, the point with the next-highest priority is added.[29] Now the role of the LLM is to come up with priority functions that yield large capsets. This way, a capset is constructed from bottom up.

On the architecture level, the approach consists of a sampler, evaluator and programs database. The sampler samples priority functions from the database and sends them to the evaluator, which attaches the size of the capset that the function gives rise to and stores the pair of priority function and metric in the programs database.[30]

Initially, the priority function is trivial, assigning every point a priority of 0.0. Throughout, the LLM is queried repeatedly with constructing a new priority function, for which a new set is built, its size evaluated, attached to the program and stored in the database each time. A key element

[28] referring to the search of an appropriate priority function for a specific problem

[29] Ties are broken in favour of lexicographical order.

[30] In their actual experiments, this metric is an average of capset sizes over multiple dimensions.

here is that the LLM is prompted to not only find good priority functions, but improve upon the functions that are sampled from the database. This way, the evolutionary process is initialized and maintained.[31] Another aspect of this *evolutionary* algorithm is the structure of the database in islands, on each of which the aforementioned process is run in parallel. After a certain time, the weaker half of the island is discarded and the best programs (according to the largest size of capsets the priority functions gave rise to) are used to re-seed those islands. This process is repeated ad infinitum. By construction, the maximum size of a capset derived from a program per experiment is non-decreasing.

For the capset problem, FunSearch was able to create with a priority function that yielded a capset of size 512 in $\mathrm{AG}(8,3)$ and of size 1082 in $\mathrm{AG}(9,3)$. The explicit priority functions that gave rise to those sets can be found in [**RPBN$^+$24**, Figure 4b] and [**RPBN$^+$24**, Figure C.5, Supplementary Material]. On top of this result itself, the researchers were able to examine symmetries in the given capset that resembled constructions of a maximal cap (the Hill cap) of size 112 in $\mathrm{AG}(6,3)$:

> This cap set can be constructed as the union of four 128-element sets, each with a different defining property. For example, the first set contains a specific half of all vectors without a 0 entry, whereas the second set contains all vectors whose nonzero entries appear in 4 specific coordinates. These ideas are strikingly similar to the construction of the Hill cap, which results in the optimal 112-cap in $\mathbb{F}_3^6$.

So in some sense, the algorithm partook in a creative discovery process:[32] "FunSearch searches for programs that describe how to solve a problem, rather than what the solution is."[**RPBN$^+$24**]

1.2.5.2. *A LLM approach to* 4*-general sets in* $\mathrm{AG}(7,3)$.

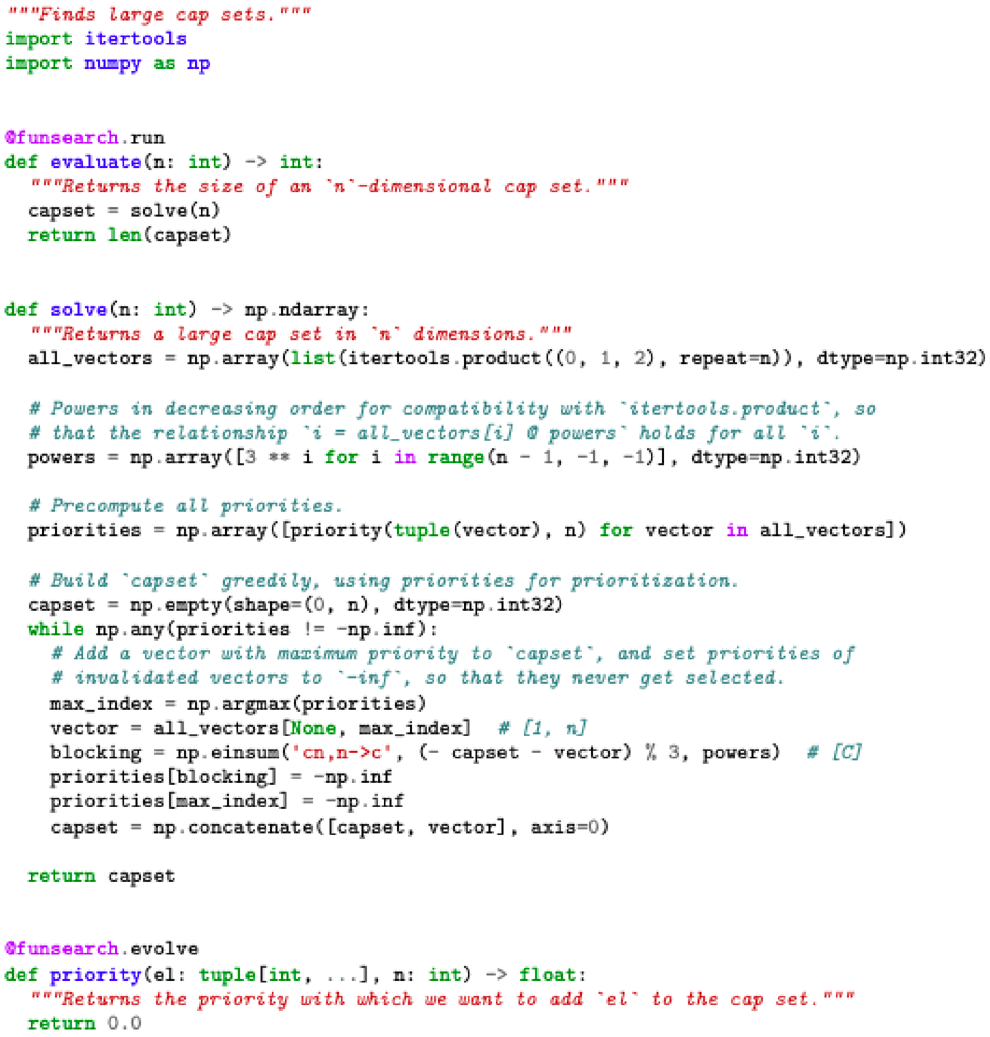
For our purposes of examining Sidon sets, we are interested in this approach primarily for its extension from 3-general to 4-general sets. It turns out that in essence, FunSearch can easily be adapted to tackle different problems, especially when the problem statement is similar. In order to see this, we provide the actual query prompt that the researchers in Figure 3.

While the `priority()` function handles the definition of the priority function, the construction of capsets based on the function is managed by the `solve()` function. In particular, the adding of points according to their priorities, and the blocking of infeasible points is handled in the while-loop, where the `einsum` function from the `numpy` library blocks points: For every newly added point and every point in the existing capset, the third point that would complete the affine line is discarded.[33] Now in order to obtain 4-general instead of 3-general sets, it is instructive to recall the structure of an affine plane in $\mathrm{AG}(k,3)$, as visualized in Table 1.1.19.

[31] The number of input functions can be chosen as hyperparameter.

[32] Of course, it was rather randomness that gave rise to the results, but one might argue that the evolutionary process as a whole, as instance of biomimicry, is creative.

[33] Discarding a point means setting its priority to $-\infty$.

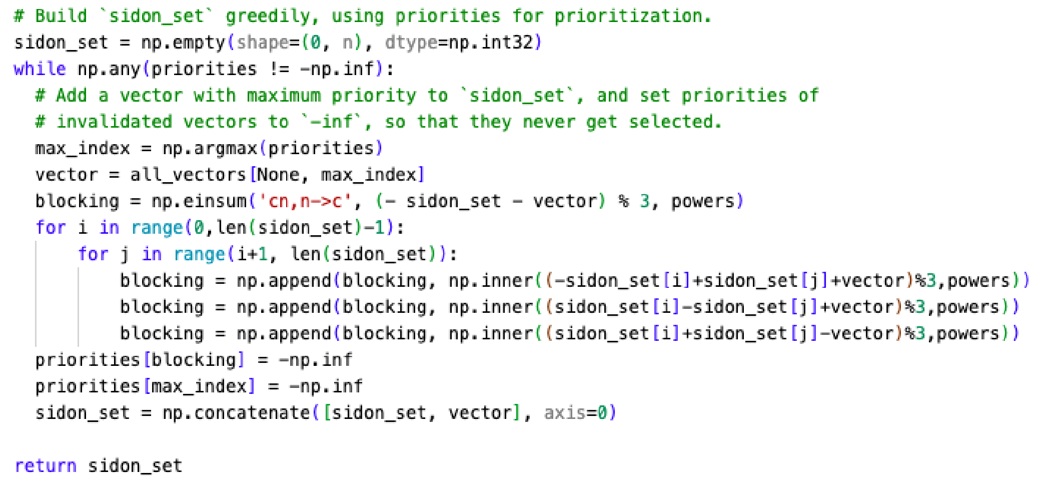


```
"""Finds large cap sets."""
import itertools
import numpy as np


@funsearch.run
def evaluate(n: int) -> int:
  """Returns the size of an `n`-dimensional cap set."""
  capset = solve(n)
  return len(capset)


def solve(n: int) -> np.ndarray:
  """Returns a large cap set in `n` dimensions."""
  all_vectors = np.array(list(itertools.product((0, 1, 2), repeat=n)), dtype=np.int32)

  # Powers in decreasing order for compatibility with `itertools.product`, so
  # that the relationship `i = all_vectors[i] @ powers` holds for all `i`.
  powers = np.array([3 ** i for i in range(n - 1, -1, -1)], dtype=np.int32)

  # Precompute all priorities.
  priorities = np.array([priority(tuple(vector), n) for vector in all_vectors])

  # Build `capset` greedily, using priorities for prioritization.
  capset = np.empty(shape=(0, n), dtype=np.int32)
  while np.any(priorities != -np.inf):
    # Add a vector with maximum priority to `capset`, and set priorities of
    # invalidated vectors to `-inf`, so that they never get selected.
    max_index = np.argmax(priorities)
    vector = all_vectors[None, max_index]  # [1, n]
    blocking = np.einsum('cn,n->c', (- capset - vector) % 3, powers)  # [C]
    priorities[blocking] = -np.inf
    priorities[max_index] = -np.inf
    capset = np.concatenate([capset, vector], axis=0)

  return capset


@funsearch.evolve
def priority(el: tuple[int, ...], n: int) -> float:
  """Returns the priority with which we want to add `el` to the cap set."""
  return 0.0
```

Figure 3. Program skeleton for capset query of Large Language Model [RPBN+24, Figure C.6, Supplementary Material]

Observe that for every triple of distinct and non-collinear points (a, b, c), such a plane consists of nine points. Two examples of how the six additional points are formed follow. The point $-a-b$ results from adding the direction $(b-a)$ to a twice: $a+2(b-a) = -a+2b = -a-b$, and (ii) the point $a-b+c$ results from adding the direction $(b-a)$ to a twice, and the direction $(c-a)$ once: $a+2(b-a)+(c-a) = a-b+c$.

So assume that there is a set of points stored in the `sidon_set` list variable. Then upon adding an additional point, the `einsum` command blocks all points that would complete a line with the points in the set. So in order to assure that we maintain the Sidon property, we must additionally block the remaining three points $-a+b+c$, $a-b+c$ and $a+b-c$ on any plane, where without loss of generality we may assume that the points a and b are part of the set, and c is the added



```
# Build `sidon_set` greedily, using priorities for prioritization.
sidon_set = np.empty(shape=(0, n), dtype=np.int32)
while np.any(priorities != -np.inf):
  # Add a vector with maximum priority to `sidon_set`, and set priorities of
  # invalidated vectors to `-inf`, so that they never get selected.
  max_index = np.argmax(priorities)
  vector = all_vectors[None, max_index]
  blocking = np.einsum('cn,n->c', (- sidon_set - vector) % 3, powers)
  for i in range(0,len(sidon_set)-1):
      for j in range(i+1, len(sidon_set)):
          blocking = np.append(blocking, np.inner((-sidon_set[i]+sidon_set[j]+vector)%3,powers))
          blocking = np.append(blocking, np.inner((sidon_set[i]-sidon_set[j]+vector)%3,powers))
          blocking = np.append(blocking, np.inner((sidon_set[i]+sidon_set[j]-vector)%3,powers))
  priorities[blocking] = -np.inf
  priorities[max_index] = -np.inf
  sidon_set = np.concatenate([sidon_set, vector], axis=0)

return sidon_set
```

Figure 4. Excerpt from program skeleton for Sidon set query of Large Language Model

point.[34] This is achieved by the nested for-loop in Figure 4. This way, all points are discarded that would yield a configuration of four points on a 2-flat in $\mathrm{AG}(k, 3)$. In other words, we guarantee to maintain Sidon property when adding any one of the remaining point.

Recreating the experiments conducted by DeepMind with this changed input specification, we were able to obtain Sidon sets of size 31 in $\mathrm{AG}(7, 3)$ after approximately $500,000$ queries of the Google Codey `code-bison` model,[35] using an API. One of the priority functions that yielded this result can be found in Code Excerpt 0.0.5. It is important to note that priority functions giving rise to Sidon sets of size 31 were found fairly early in the experiment. The author's GitHub page https://github.com/timneumann1/FunSearch-Sidon contains comprehensive code and instructions to run FunSearch experiments using Google's `code-bison` model for finding large capsets and Sidon sets.

Recalling that $r_4(k, 3) \geq 33$, we did not attain or improve the best-known maximal size of a Sidon set in this dimension, for which there are two main reasons.

Firstly, we used the Pegasus High Performance Cluster for the FunSearch experiment, but were not able to implement parallelism into our experiments. As stated in their paper, DeepMind's researchers employed a "distributed system that has three types of workers – a programs database, samplers and evaluators – which communicate asynchronously" [RPBN+24], tremendously increasing performance. The necessary infrastructure and code is partly proprietary, so that a replication in the scope of this thesis was not feasible.

[34] We must also block the added point itself, of course.

[35] For our experiment, we used a small temperature and reset time as hyperparameters for the first 250,000 queries, and then increased both parameters for the remaining 500,000 queries.

Secondly, the DeepMind team conducted 140 experiments, out of which only 4 yielded an improvement of the capset [**RPBN$^+$24**, Supplementary Material A.3], and a comparable FunSearch experiment on so-called Admissible Sets required generation of "approximately two million programs" [**RPBN$^+$24**, Supplementary Material, A.5]. Now our access to the LLM through the API introduces a considerable monetary cost, making large-scale implementation (on the order of millions of queries) infeasible in the scope of this thesis.

Hence, it is not of great surprise that our experiment did not result in an improvement of the best-known Sidon set size in dimension 7. But even though we did not improve the bound, we provided a proof-of-concept that the FunSearch approach works for Sidon sets as well, indication that improvements in the maximal Sidon set size might be achieved with this approach. In particular, we strongly expect more computational power and fast model access to improve the result, especially given the proximity to the capset problem and the improvement demonstrated therein by DeepMind.

Some suggestions for future include:

- The most straightforward area of improvement is adding parallelism, i.e., increasing the number of parallel workers. This will speed up sampling and evaluation significantly and thus make experiments less time-consuming.
- Experimenting with the choice of model itself might be of immense benefit in terms of monetary resources. There exist open source Large Language Models such as StarCoder [**LAZ$^+$23**] that might be capable of adequate performance and do not come with the API query costs that is required for using Google Codey.
- Since Sidon sets are generally not as well-known as capsets, it might be instructive to change the prompt to something more descriptive of the particular objects we try to construct. This is generally referred to as *prompt engineering*. Note however that the strength of FunSearch really lies in leveraging random evolutionary successes, so it is not clear if a more precise depiction of what we are looking for would help the LLM to respond with better priority function.
- The complexity of LLM responses varies greatly dependent on what temperature we set as hyperparameter. Usually, the temperature as measure of randomness can range from 0 (deterministic) to 1 (random). It is obvious that this setting has great impact on the evolution of the priority functions, so it might be worthwhile to experiment with different temperatures, or even a time-dependent temperature for model queries.
- Changing the number of dimensions in which a given priority function is evaluated on might yield some more useful ideas in the LLM's responses at the expense of additional computational cost (since the algorithm has to construct Sidon sets in multiple dimensions).

- The choice of which priority functions are used for a new prompt is based on a probability vector that assigns a higher probability of being selected to priority functions with better signatures (larger sizes of Sidon sets). The softmax function that is used to achieve this depends on the two hyperparameters `cluster_sampling_temperature_init` and `cluster_sampling_temperature_period`, which can be chosen differently in order to examine changes in performance.

Since many branches of AI research depend heavily on experiments, it is difficult to predict which of these suggestions are most likely to cause an improvement in results.

1.2.5.3. *LLMs in mathematics.* The FunSearch approach – repeated query of a language model towards the problem statement refined on previous best answers in an evolutionary approach – is an example of the rising influence of Artificial Intelligence on many areas of scientific research, and proof that this development does not halt before mathematics. It is likely that AI will advance to be a strong tool for mathematicians. The number of problems similar to the capset problem or the Sidon set problem discussed above that can be tackled using LLM approaches are numerous, and it will be interesting to see how this and other approaches leverage AI to extend human capabilities.

Throughout this section, we have approached the problem of finding $r_4(k, 3)$ from different angles, yielding insight from a theoretical, computational and AI perspective, which exemplifies that oftentimes there is more than one way to attack a question. In the next section, we examine m-general sets for $m = 3, 4, 5$ over $\mathbb{F}_2$ and $\mathbb{F}_3$ in affine space computationally.

1.3. A computational approach to m-general sets over affine spaces

In this next section, we describe a computational approach and determine a variety of small and large complete m-general sets with $m = 3, 4, 5$ over affine space in low dimensions and with respect to both $\mathbb{F}_2$ and $\mathbb{F}_3$. In Section 2.4 and in the context of projective space, similar considerations will provide a way to construct error-correcting codes. For affine space, this section's goal is to provide a tangible way of working with the objects discussed in previous sections, and to give some instances of (to the best of our knowledge) new results of small and large complete m-general sets.

1.3.1. Finding minimal and maximal complete m-general sets in $\mathrm{AG}(k, q)$.

The overall structure of the computational work in this more general setting still mimics the pseudocode provided in Algorithm 3, where the m-generality must be treated carefully according to specific m. So in order to find m-general sets in affine space, we again rely heavily on the notion of affine combinations. We begin with a set of points that is in general position in the respective space and hence guaranteed to be m-general. For $\mathrm{AG}(k, q)$, we choose a set of $k + 1$ points such that every new point introduces a new dimension. This set is affinely independent, and since "[a]ny two affinely independent sets of the same size are affinely equivalent" [**CFG⁺22**], this choice of starting set in the affine setting can be made without loss of generality. From here, we recursively traverse the search space in lexicographical order, blocking all points that cannot be added to the m-general set while maintaining its property at every step. This builds directly upon Definition 1.1.7 on affine combinations, where in the file `m-general-sets-AG.py`, we use the function `coeff_vectors` to determine all coefficient vectors of the required dimension for which the coordinate entries sum up to 1.[36] The parameter `opt` can be set to "min" or "max" in order to determine small or large complete m-general sets. The output of the file contains the size and points of any extremal set. As an example, the search for a small complete 4-general set in $\mathrm{AG}(3, 3)$ returns

"A complete 4-general set of minimal size 5 in $\mathrm{AG}(3, 3)$ is $[0, 1, 3, 9, 13]$."

The script `m-general-sets-AG-max_search.py` allows for an arbitrary initialization and traverses the remaining search space, which was helpful in Section 1.2 when searching for maximal Sidon sets. In order to verify results, the script `m-general-sets-AG-test.py` uses the arithmetic conditions described in Theorem 2.5 in [**TW21**] to determine whether or not a given input set of points in affine space is m-general or not.

1.3.2. Computational results in affine space.

In the following tables, we present results for a variety of complete m-general sets in $\mathrm{AG}(k, q)$ for $q = 2, 3$ and small k. Following the definition closely, we require that a set be of size m at least in

[36] Note that this code significantly expands, but fundamentally builds upon the code used for the computational part of [**HTW19**].

order to be considered m-general. Furthermore, we require $q^k < m \leq k + 2$, which excludes some trivial cases.

$\mathbb{F}_2$	m = 3			m = 4			m = 5		
k	small	large	reference	small	large	reference	small	large	reference
2	4	4	2^k	-	-	-	-	-	-
3	8	8	”	4	4	1.3 / [CFG+22]	-	-	-
4	16	16	”	6	6	1.3 / [CFG+22]	6	6	1.3
5	32	32	”	7	7	1.3 / [CFG+22]	7	7	1.3
6	64	64	”	8	9	1.3 / [CFG+22]	8	9	1.3
7	128	128	”	12	12	1.3 / [CP23]	12	12	1.3
8	256	256	”	≤ 16	18	1.3 / [CP23]	≤ 16	≥ 18	1.3
9	512	512	”	≤ 22	24	1.3 / [CP23]	≤ 22	≥ 23	1.3
10	1024	1024	”	≤ 28	34	1.3 / [CP23]	≤ 29	≥ 31	1.3

Table 6. m-general sets in $\mathrm{AG}(k, 2)$

$\mathbb{F}_3$	m = 3			m = 4			m = 5		
k	small	large	reference	small	large	reference	small	large	reference
2	4	4	[Bos47]	-	-	-	-	-	-
3	8	9	1.3 / [DM03]	5	5	.	-	-	-
4	16	20	[MGGG17] / [Pel70]	6	9	1.3 / [HTW19]	6	6	1.3
5	≤ 32	45	1.3 / [EFLS02]	≤ 11	13	1.3 / [HTW19]	7	12	1.3
6	≤ 64	112	1.3 / [Pot08]	≤ 17	27	1.3 / [HTW19]	≤ 13	≥ 14	1.3
7	≤ 128	≥ 236	1.3 / [RPBN+24]	≤ 28	≥ 33	1.3 / [HTW19]	≤ 17	≥ 19	1.3
8	≤ 256	≥ 512	1.3 / [RPBN+24]	≤ 43	81	1.3 / [HTW19]	≤ 22	≥ 25	1.3
9	≤ 512	≥ 1082	1.3 / [RPBN+24]	≤ 69	≥ 91	1.3 / 1.2	≤ 31	≥ 33	1.3

Table 7. m-general sets in $\mathrm{AG}(k, 3)$

PART 2

Projective arcs, error-correcting codes and m-general sets in projective space $\mathrm{PG}(k, q)$

2.1. Finite Geometries: Projective Space

In the previous sections, we have worked over affine spaces $\mathrm{AG}(k, q)$, which arise naturally from vector spaces $\mathbb{F}_q^k$. Another finite geometry that is closely related to vector and affine spaces is projective geometry, which has received wide attention in mathematics from quantum physics to coding theory.[1] This section will introduce the projective space $\mathrm{PG}(k, q)$, which will be the ambient space of interest for the remainder of this thesis.

2.1.1. Fundamentals of $\mathrm{PG}(k, q)$.

Definition 2.1.1. (Projective space) The k-dimensional (finite) *projective space* over $\mathbb{F}_q$, commonly denoted as $\mathrm{PG}(k, q)$, is the space whose s-dimensional subspaces are the $(s+1)$-dimensional vector subspaces of the vector space $\mathbb{F}_q^{k+1}$ for $0 \leq s \leq k$.

For example, the 0-dimensional subspaces (points) of a k-dimensional projective space are the 1-dimensional subspaces of $\mathbb{F}_q^{k+1}$, i.e., the spaces spanned by the lines passing through the origin in $\mathbb{F}_q^{k+1}$. Furthermore, the lines of $\mathrm{PG}(k, q)$ are the 2-flats (planes) of $\mathbb{F}_q^{k+1}$. This definition hints at a way of constructing $\mathrm{PG}(k, q)$.

Construction 2.1.2. The points of $\mathrm{PG}(k, q)$ correspond to the nonzero vectors in $\mathbb{F}_q^{k+1}$ under the equivalence relation $\sim$, where for two vectors v and w in $\mathbb{F}_q^{k+1}$, we have $v \sim w$ if there exists a nonzero scalar $\alpha \in \mathbb{F}_q$ such that $v = \alpha w$ [**BDGP23**, 4f.]. So v and w give rise to the same point in $\mathrm{PG}(k, q)$ if and only if $v \sim w$. Consequently, $\mathrm{PG}(k, q)$ is the set of equivalence classes $V/\sim$, where $V = \mathbb{F}_q^{k+1} \setminus \{\mathbf{0}\}$.

From this construction, we see how projective space arises from collapsing subspaces of a vector space in higher dimension.

Example 2.1.3. Let u, v and w be vectors in $\mathbb{F}_3^3$. Then if the three vectors are linearly dependent, i.e., if the matrix $\begin{bmatrix} u \ v \ w \end{bmatrix}$ does not have full rank, then they lie on a hyperplane through the origin, which is a 2-dimensional flat. Hence, the corresponding points in $\mathrm{PG}(2, 3)$ lie on a 1-dimensional flat, i.e., a projective line.

[1] There also exists a projective version of the SET game, as explained in [**DM03**, pp. 16f.].

Since the relation to affine spaces remains somewhat opaque here,[2] we want to mention a second way of thinking about projective space, where one starts with an affine space of the *same* dimension as our target space, and *completes* the affine space to obtain projective space. The most tangible instance is the projective plane: In order to obtain $\mathrm{PG}(2, q)$, take $\mathrm{AG}(2, q)$ and complete every set of parallel lines, i.e., the translates of a line, with a *point at infinity.* Then all points at infinity lie on a line at infinity, and no two lines are parallel in the resulting projective plane. As an example, consider the Fano plane $\mathrm{PG}(2, 2)$ below, which is the smallest projective space. To illustrate the connection to affine space in this example, we have marked a set of four points that forms an isomorphic copy of $\mathrm{AG}(2, 2)$ in the Fano Plane, with the other three points forming the line at infinity:

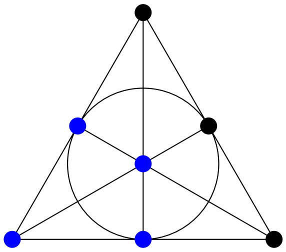


FIGURE 1. The Fano plane $\mathrm{PG}(2, 2)$, the smallest projective space

For the purpose of this thesis, we will work mainly with Construction 2.1.2, paired with a way of labeling points in projective space referred to as *homogeneous coordinates.* Note that in contrast to the usual k-tuples we worked with in affine space, points in projective space are equivalence classes of a higher-dimensional vector space. Hence, a vector subspace uniquely determines its corresponding subspace in projective space through the equivalence relation, but *not* vice versa, i.e., we lose information through the projection. Now homogeneous coordinates represent the points of $\mathrm{PG}(k, q)$ in the following manner: If $\mathbf{x} = (x_0, x_1, \dots, x_k)$ is a vector in $\mathbb{F}_q^{k+1} \setminus \{\mathbf{0}\}$, suppose that x_i is the first nonzero entry of $\mathbf{x}$[3], and express the corresponding point in projective space as

$$\left(0 : 0 : \dots : \frac{x_i}{x_i} : \frac{x_{i+1}}{x_i} : \frac{x_{i+2}}{x_i} : \dots : \frac{x_k}{x_i}\right) = \left(0 : 0 : \dots : 1 : \frac{x_{i+1}}{x_i} : \frac{x_{i+2}}{x_i} : \dots : \frac{x_k}{x_i}\right).$$

This way, we account for the fact that a point in projective space can originate from a set of collinear vectors.

EXAMPLE 2.1.4. The points $(1, 0, 0)$ and $(2, 0, 0) = 2(1, 0, 0)$ are collinear in $\mathbb{F}_3^3$ and give rise to the same point in $\mathrm{PG}(2, 3)$, namely $(1 : 0 : 0)$. Likewise, the points $(0, 2, 1)$ and $(0, 1, 2)$ in $\mathbb{F}_3^3$ are projected to $(0 : 1 : 2)$ in $\mathrm{PG}(2, 3)$, since $(0, 1, 2) = 2 \cdot (0, 2, 1)$.

[2] In some sense, Construction 2.1.2 could also start with an affine space, for which we designate an origin and then apply the equivalence relation, which sheds some light on the connection.

[3] Such an entry must be exist, since the set that we apply the equivalence relation to excludes the zero vector.

Remark 2.1.5. For the connection to affine space, note that if we fix the first coordinate of the homogeneous coordinates of $\mathrm{PG}(k,q)$ to be 1, then the remaining k of the $k+1$ components are the coordinates of $\mathbb{F}_q^k$, from which we obtain a representation of affine space of dimension k: Indeed, there exist isomorphic copies of $\mathrm{AG}(k,q)$ in $\mathrm{PG}(k,q)$. Additionally, homogeneous coordinates capture the phenomenon of *points at infinity*: for those points, up to choice of labeling, the first (homogeneous) coordinate is 0.

To provide a tangible example of Construction 2.1.2 for projective space and see homogeneous coordinates in action, consider how $\mathrm{PG}(2,3)$, the projective plane of order 3, arises from the vector space $\mathbb{F}_3^3$. To illustrate the connection to $\mathbb{F}_3^3$, we have colored pairs of points in the vector space that give rise to the same point in $\mathrm{PG}(2,3)$ accordingly.

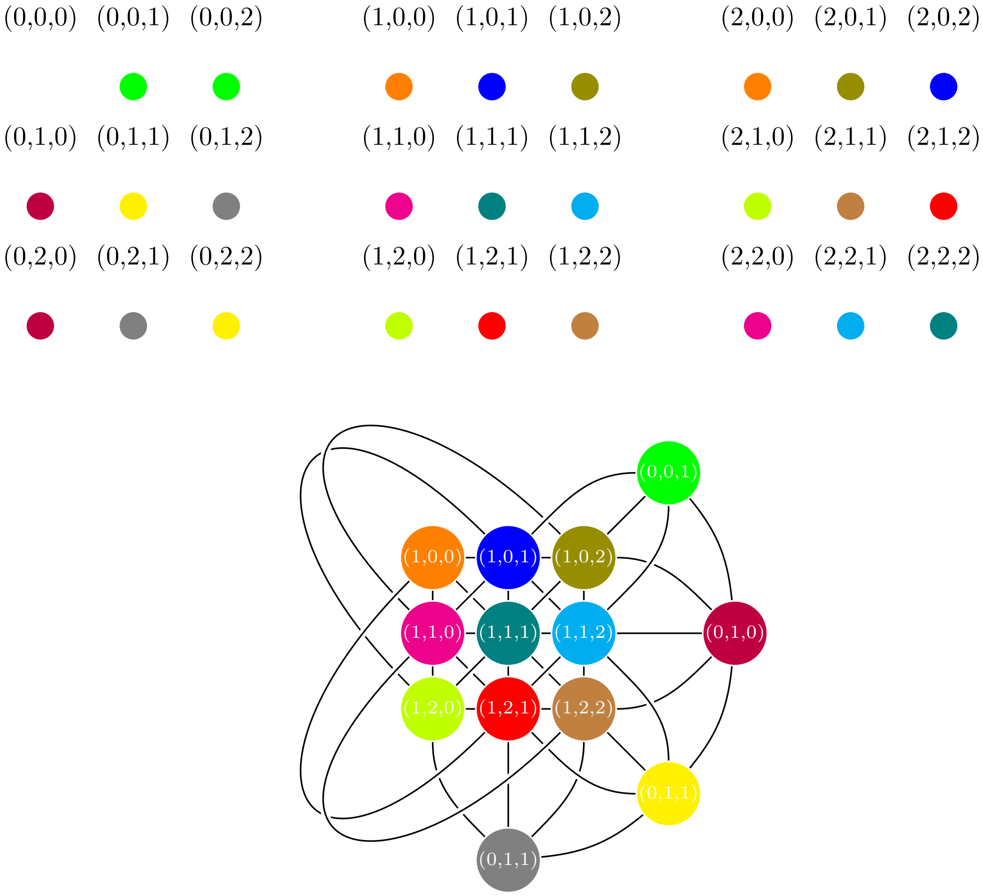


Figure 2. Projective space $\mathrm{PG}(2,3)$ arises from equivalence classes in $\mathbb{F}_3^3$

We see that the 13 equivalence classes of vectors are represented accordingly in $\mathrm{PG}(2,3)$ by 13 points.

By collapsing subspaces of a vector space of higher dimension, we can determine the size of projective subspaces by a counting argument. Since the 0-dimensional subspaces (points) in $\mathrm{PG}(k, q)$ arise from the vectors in $\mathbb{F}_q^{k+1}$, where we have $q-1$ nonzero collinear vectors for each vector, exactly these $q-1$ vectors will be collapsed to the same point in $\mathrm{PG}(k, q)$. It follows that there are $(q^{k+1}-1)/(q-1) = 1 + q + q^2 + \ldots + q^k$ distinct points in $\mathrm{PG}(k, q)$.[4] To see how homogeneous coordinates give us another perspective on counting those points, let i with $1 \leq i \leq k+1$ be the position of the first 1, counting from the left. For each index i, there are $(k+1) - i$ remaining positions with q choices each, so each i contributes q^{k+1-i} combinations. With the bounds on i given above, we see that this counts exactly $q^k + q^{k-1} + \ldots + q + 1$ points.

Example 2.1.6. By the counting arguments above, the Fano Plane $\mathrm{PG}(2, 2)$ shown in Graphic 2.1.3, which arises from applying the equivalence relation on $\mathbb{F}_2^3$,[5] contains

- $q^2 = 4$ triples of the form $(1 : x_1 : x_2)$ with $x_1, x_2 \in \{0, 1\}$,
- $q = 2$ triples of the form $(0 : 1 : x_2)$ with $x_2 \in\in \{0, 1\}$,
- one element of the form $(0 : 0 : 1)$.

We can also treat the general case.

Theorem 2.1.7. For $0 \leq s \leq k$, the number of s-dimensional subspaces in $\mathrm{PG}(k, q)$ is

$$\frac{(q^{k+1}-1)(q^{k+1}-q)\ldots(q^{k+1}-q^s)}{(q^{s+1}-1)(q^{s+1}-q)\ldots(q^{s+1}-q^s)}.$$

Proof. Recall from Theorem 1.1.4 that the number of $(s+1)$-dimensional subspaces of the vector space $\mathbb{F}_q^{k+1}$ is

$$\frac{(q^{k+1}-1)(q^{k+1}-q)\ldots(q^{k+1}-q^s)}{(q^{s+1}-1)(q^{s+1}-q)\ldots(q^{s+1}-q^s)},$$

and that the s-dimensional subspaces of $\mathrm{PG}(k, q)$ correspond precisely to the $(s+1)$-dimensional subspaces of $\mathbb{F}_q^{k+1}$. □

As an example, Theorem 2.1.7 dictates that in $\mathrm{PG}(2, 3)$, the number of 1-dimensional subspaces, i.e., projective lines, is $(3^3-1)(3^3-3)/(3^2-1)(3^2-3) = 13$, as can be seen in Remark 2.1.5. Furthermore, we already saw that there are $(3^3-1)/(3^1-1) = 13$ projective points (0-dimensional subspaces).

The above equality in the number of points and lines holds in general, and is a result of the so-called *plane duality*, which entails that points and lines play a symmetric role in projective planes. We

[4] For example, the projective plane $\mathrm{PG}(2, 3)$ consists of $3^2 + 3 + 1 = 13$ points.

[5] or equivalently from completing $\mathrm{AG}(2, 2)$

can verify this numerically: If $k = 2$, then by Theorem 2.1.7 the number of projective lines over $\mathrm{PG}(2, q)$ is

$$\frac{(q^3-1)(q^3-q)}{(q^2-1)(q^2-q)} = \frac{(q^3-1)(q^2-1)q}{(q^2-1)(q-1)q} = \frac{q^3-1}{q-1} = 1+q+q^2,$$

which is the number of points in a projective plane.

2.1.2. Connections to group theory.

The previous deliberations can also be generalized to the terminology of group theory. Here, the general linear group $\mathrm{GL}(k)$ is the group of $k \times k$ invertible matrices under matrix multiplication, which has the identity matrix as identity element and is closed under its operation since we have $\det AB = \det A \cdot \det B$ for any two $k \times k$ matrices A and B, and a matrix being invertible is equivalent to it having nonzero determinant. Then Construction 2.1.2 hints at the nature of the projective linear group of a vector space being the quotient group of the general linear group modulo the subgroup of the nonzero scalar transformations of the vector space.

Furthermore, the affine group is the group of all invertible affine transformations, and in accordance with the idea of projective completion, the affine group is a subgroup of the projective group [**Ewa71**, 241].

2.1.3. Sets over $\mathrm{PG}(k, q)$.

Similar to our treatment of affine space, we want to consider sets of points and their properties in projective space as well. Fortunately, all notions from Section 1.1.2 transfer to this new setting. In particular, we can define subspace-evasive, m-general and blocking sets in complete analogy to the affine setting, even though the geometric objects differ, of course. Since our main focus is on m-general sets, we restate the definition here:

Definition 2.1.8. (m-general set). Let $3 \leq m \leq k+2$, and let $\mathcal{P}$ be a set of points in $\mathrm{PG}(k, q)$ with $|\mathcal{P}| \geq m$. Then $\mathcal{P}$ is called *m-general set* if no m points in $\mathcal{P}$ lie on a single $(m-2)$-dimensional subspace of $\mathrm{PG}(k, q)$.

Also recall our statement in Section 1.1.2 that a (s, m)-blocking set is simply the complement of an $(s, D_s - m)$-evasive set, where D_s is the number of points on an s-dimensional subspace. Now in $\mathrm{PG}(k, q)$, via the equivalence relation we see that every point on a s-dimensional subspace of $\mathrm{PG}(k, q)$ originates from a 1-dimensional subspace of a $(s+1)$-dimensional subspace of $\mathbb{F}_q^{k+1}$. Now the q^{s+1} vectors in such a $(s+1)$-dimensional subspace of $\mathbb{F}_q^{k+1}$ define exactly $q^{s+1}/(q-1) = 1 + q + \ldots + q^s$ distinct 1-dimensional subspaces, which give rise to the points in projective space. Hence, for $\mathrm{PG}(k, q)$, we have $D_s = 1 + q + \ldots + q^s$.

Since there is a an isomorphic copy of affine space in projective space, we can state one more result about m-general sets.

Theorem 2.1.9. An m-general set in $\mathrm{AG}(k, q)$ is also a m-general set in $\mathrm{PG}(k, q)$. [Pav23]

Proof. Let $\mathcal{S}$ be an m-general set in $\mathrm{AG}(k, q)$, so $\mathcal{S}$ intersects every $(m-2)$-dimensional subspace in at most $m-1$ points. Now $\mathrm{PG}(k, q)$ contains an isomorphic copy of $\mathrm{AG}(k, q)$,[6] and every affine subspace of this isomorphic copy is a subset of a projective subspace of the same dimension. Let $\mathcal{S}'$ denote the corresponding set of points in projective space resulting from the natural isomorphism given by homogeneous coordinates. Since $\mathcal{S}'$ intersects solely the points of the isomorphic copy, and not the remaining points of projective space (the points at infinity), we have not added any points to the intersection with any subspace. Hence, $\mathcal{S}'$ intersects every $(m-2)$-dimensional subspace of $\mathrm{PG}(k, q)$ in at most $m-1$ points as well, so $\mathcal{S}'$ is a m-general set in projective space. □

Note that we have slightly abused notation in the theorem statement, which can be justified since the identification of points between affine and projective space is clear in this case. This theorem directly implies that bounds on the size of m-general sets in affine space will result into bounds for such sets in projective space.

To conclude this introduction on projective space, we mention one more important concept that arises frequently in this context [Pav23], and which adds useful terminology.

Definition 2.1.10. (s-arc). Let $\mathcal{P}$ be a set of projective points with $|\mathcal{P}| = s$ in $\mathrm{PG}(k, q)$ such that at most k lie on a projective hyperplane. Then $\mathcal{P}$ is called an *s-arc*.

Throughout, projective arcs[7] are $(k-1, k)$-subspace-evasive sets, and those objects take in an important role in the connection between projective sets and coding theory [BL20, Lan00]. In order to fully appreciate the connections between projective space and coding theory that m-general sets bridge, we next give a brief introduction to the important field of coding theory.

[6] Projective space indeed contains many copies of affine space, but the natural identification provided by considering the k-tuple that follows a 1 in the first coordinate of homogeneous coordinates is a straightforward one.

[7] We can also define arcs in the affine setting, where they are sets of points over $\mathbb{F}_q^k$ for which no $k+1$ of them lie on an affine hyperplane.

2.2. Coding Theory

With the rise of digital communication, much research in the 20th century was devoted to making exchange of communication not only secure, but also robust to errors. Since communication channels can introduce noise to transmitted messages, the emerging field of coding theory became concerned with creating error-correcting codes. Such codes add redundancy to a message in order to mitigate the risk of losing information when submitting it over a noisy channel. A fascinating field of application is space travel, where error-correcting codes are utilized to ensure reliable communication with satellites, for example. In the following, we introduce some fundamental definitions, general notions as well as special examples from the wide field coding theory. While only scratching the surface, this will enable us to see the deep connections between finite geometries and error-correcting codes.

2.2.1. Preliminaries.

From here on and for the remainder of the thesis, we suppress the boldface notation for vectors, since it will be clear from context in which space the given variables live.

Definition 2.2.1. (Alphabet, letter and words). Let $\mathcal{S}$ be a finite set, called the *alphabet*. The elements of $\mathcal{S}$ are called the *letters* of the alphabet. Then a *word* in $\mathcal{S}$ is a finite sequence of letters of the alphabet.

We write $\mathcal{S}^n = \{\mathbf{w} = a_1a_2\dots a_n \mid a_1,\dots,a_n \in \mathcal{S}\}$ for the set of all possible words of length n. In most cases, we have $\mathcal{S} = \mathbb{Z}_q$, with $q = 2$ (binary) or $q = 3$ (ternary), such that the length-n words are given by $\mathbb{Z}_q^n$, the n-dimensional discrete vector space over $\mathbb{Z}_q$.[8]

Definition 2.2.2. (Code). A *code* of length n over an alphabet S is a subset $\mathcal{C} \subseteq \mathcal{S}^n$.

A very well-known family of codes is formed by the Hamming codes, named after one of the most influential contributors to coding theory, Richard Hamming.

Example 2.2.3. For the so-called Hamming $[7,4]$ code, we have $\mathcal{S} = \mathbb{F}_2$ (it is a *binary* code), and $\mathcal{C}$ consists of all 7-bit words $abcdefg$ satisfying

$$a+b+c-e=0, \quad a+b+d-f=0, \quad a+c+d-g=0.$$

We can list all $2^4 = 16$ codewords explicitly:

$$\begin{aligned}\mathcal{C} = \{&0000000, 0001011, 0010101, 0011110, 0100110, 0101101, 0110011, 0111000,\\ &1000111, 1001100, 1010010, 1011001, 1100001, 1101010, 1110100, 1111111\}.\end{aligned}$$

[8] We refer to dimensionality with n instead of k for reasons that will become clear soon: In gist, the codewords originate from a smaller space, for which we want to use k.

We will frequently return to this example. One important observation is that the above code $\mathcal{C}$ contains the zero vector. Indeed, it is a so-called *linear code*.

2.2.2. Linear Codes.

Algebraic coding theory as a subfield of coding theory is concerned with such codes that are not only subsets, but (linear vector) subspaces of some word space $\mathcal{S}^n$.

Definition 2.2.4. (Linear code). [**Hil86**] Let $\mathcal{S}$ be a finite field of prime-power order q, denoted by $\mathbb{F}_q$. Then a (linear) subspace of the vector space $\mathbb{F}_q^n$ is called a *linear code*.

Linear codes allow us to apply the means of linear algebra. For example, we can find a set of linearly independent codewords that generate a code.

Definition 2.2.5. (Generator matrix). Let $\mathcal{C}$ be a linear code of dimension k over some ambient vector space $\mathbb{F}_q^n$. Then a *generator matrix* for $\mathcal{C}$ is a matrix $G \in M_{k\times n}(\mathbb{F}_q)$ whose rows form a basis for $\mathcal{C}$.

If G is a generator matrix for $\mathcal{C}$, then every element of $\mathcal{C}$ is a linear combination of the rows of G. Hence, we can write

$$C = \{y^t G \mid y \in \mathbb{F}_q^k\}.$$

Example 2.2.6. A generator matrix for the Hamming $[7, 4]$ code in Example 2.2.3 is given by

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

Here, the rows of G are simply the images of the messages 1000, 0100, 0010, and 0001 (the standard basis vectors of $\mathbb{F}_2^4$) inside of C. We see that G has rank 4, i.e., that those images are linearly independent and span a space of dimension 4 over $\mathbb{Z}_2$ with precisely $2^4 = 16$ vectors in it. From this, it also becomes clear why this code is referred to as the Hamming $[7, 4]$-code: We premultiply its generator matrix with elements in $\mathbb{F}_2^4$ to obtain elements in an isomorphic copy of $\mathbb{F}_2^4$ in $\mathbb{F}_2^7$. Note that with this code, we can send 4-bit messages by transmitting 7 bits of data.

The above holds in general: If $\mathcal{C}$ is a k-dimensional subspace of some vector space $\mathbb{F}_q^n$, then we call $\mathcal{C}$ a $[n, k]$-code [**Hil86**]. The quantity $n - k$ is oftentimes referred to as the redundancy of the code, since it captures the additional dimensions that are added to the message in order to make it robust to errors.

Now for a given linear code $\mathcal{C}$, there is a closely related code $\mathcal{C}^\perp$ that is connected to $\mathcal{C}$ via a geometric notion.

DEFINITION 2.2.7. (Dual code). The *dual code* of a linear code $\mathcal{C}$ is the code

$$\mathcal{C}^{\perp} = \{y \in \mathbb{F}_q^n : \langle x, y\rangle = 0 \, \forall x \in \mathcal{C}\}.$$

So the dual code consists of all codewords that are orthogonal to the codewords in $\mathcal{C}$. Since $\mathcal{C}^{\perp}$ is linear as well, it has a generator matrix H.

DEFINITION 2.2.8. (Parity-check matrix). Let $\mathcal{C}$ be a linear code of dimension k over $\mathbb{F}_q^n$. Then the *parity-check matrix* of $\mathcal{C}$ is a matrix $H \in M_{(n-k)\times n}(\mathbb{F}_q)$ whose rows form a basis of $\mathcal{C}^{\perp}$.

By construction, the rank of H is $n-k$, since the k-dimensional code $\mathcal{C}$ is the null space of H, i.e., we have $Hc = \mathbf{0}$ for every $c \in \mathcal{C}$. Hence, H has a dual character: It is the generator matrix for $C^{\perp}$ and the parity check matrix for $\mathcal{C}$.

EXAMPLE 2.2.9. The parity-check matrix for the Hamming $[7,4]$-code is given by

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

2.2.3. Distance of a code.

In order to define properties of a code, we introduce a measure on *how different* two codewords are, which suggests a distance metric. Owing to Richard Hamming again, the most commonly such metric is the so-called Hamming distance.

DEFINITION 2.2.10. (Hamming distance). Let $\mathcal{S}$ be an alphabet and let $x, y \in \mathcal{S}^n$ be two codewords of length n. Write $x = x_1x_2\dots x_n$ and $y = y_1y_2\dots y_n$. Then the *Hamming distance* $d(x,y)$ between x and y is defined as

$$d(x,y) = \big|\{i : x_i \neq y_i, i = 1, 2, \dots, n\}\big|.$$

Hence, the Hamming distance measures in how many letters two codewords differ.[9]

THEOREM 2.2.11. The Hamming distance d is a metric.

PROOF. There are three properties we need to verify: (i) non-negativity, (ii) symmetry and (iii) triangle inequality. So let $x, y, z \in \mathcal{S}^n$. For (i), since the Hamming distance is defined as the size of a set, we have $d(x,y) \geq 0$, where equality holds precisely when the set is empty.[10] Property (ii) follows

[9] Note that this definition does not depend on the code to be linear. Indeed, the concept of Hamming distance does not even depend on the notion of a code, and it exceeds coding theory in its applicability. A common area of application is biology, where DNA strings can be compared by the number of bases in which they differ.

[10] The set $\{i : x_i \neq y_i, i = 1, 2, \dots, n\}$ is empty if and only if none of the x_i's and y_i's differ, i.e., if and only if $x = y$.

directly from Definition 2.2.10. Lastly, d satisfies the triangle inequality $d(x,z) \leq d(x,y) + d(y,z)$, since if we can obtain y from x by changing l_1 letters and z from y by changing l_2 letters, we can certainly obtain z from x by changing at most $l_1 + l_2$ letters. Thus, the Hamming distance indeed is a metric on $\mathcal{S}^n$. □

For a codeword $x \in \mathbb{F}_q^n$, we can use the notion of the Hamming distance to define its weight.

Definition 2.2.12. (Weight). A codeword $x = [x_1\ x_2 \ldots x_n] \in \mathbb{F}_q^n$ has *weight* $w = w(x)$ if it has w nonzero components, i.e., if $w(x) = |\{w_i : w_i \neq 0, 1 \leq i \leq n\}| = w$.

Furthermore, it is instructive to apply the Hamming distance to a code $\mathcal{S}$ as a whole.

Definition 2.2.13. (Minimum distance of a code). The *minimum distance of a code* $\mathcal{C}$, usually denoted by $d = d(\mathcal{C})$,[11] is defined as $d(\mathcal{C}) := \min\{d(x,y) \mid x, y \in \mathcal{C}, x \neq y\}$.

Using Definitions 2.2.12 and 2.2.13, and observing that $w(x) = w(x - \mathbf{0})$, we can prove the following useful theorem that connects the weight to the minimum distance of a code.

Theorem 2.2.14. [**Hil86**] For every linear code $\mathcal{C}$, we have $d(\mathcal{C}) = \min\{w(x) : x \neq \mathbf{0} \in \mathcal{C}\}$.

Proof. Since $\mathcal{C}$ is a subspace, we know that $\mathbf{0} \in \mathcal{C}$. Hence, we have

$$d(\mathcal{C}) = \min\{d(x,y) \mid x, y \in \mathcal{C}, x \neq y\} \leq \min\{d(x,\mathbf{0}) \mid x \neq \mathbf{0} \in \mathcal{C}\} = \min\{w(x) : x \neq \mathbf{0} \in \mathcal{C}\}.$$

On the other hand, if $x, y \in \mathcal{C}$, then again since $\mathcal{C}$ is a subspace, we have $x - y \in \mathcal{C}$. Furthermore, $d(x,y)$ is exactly the number of nonzero entries in $x - y$, so $d(x,y) = d(x - y, \mathbf{0})$, which implies

$$d(x,y) = d(x - y, \mathbf{0}) \geq \min\{w(x) : x \neq \mathbf{0} \in \mathcal{C}\}.$$

Since this is true for all $x, y \in \mathcal{C}$, choose $x \neq y$ such that $d(x,y)$ is minimal. Then

$$d(\mathcal{C}) = \min_{x,y} d(x,y) \geq \min\{d(x,\mathbf{0}) \mid x \neq \mathbf{0} \in \mathcal{C}\}.$$

Combining both inequalities proves the claim. □

We have seen that the minimum size of the support of a nonzero vector in $\mathcal{C}$ is the Hamming distance d. With the next theorem, we relate the weight of a codeword to the generator matrix G that gives rise to $\mathcal{C}$.

Theorem 2.2.15. A codeword $x = y^t G \in \mathcal{S}^n$ has weight w if and only if y has dot product equal to zero with $n - w$ columns of G.

[11] It will be clear from context whether the letter d is referring to the function $d(x,y)$ defining a metric, or the minimum distance.

Proof. By Definition 2.2.12, a codeword $x \in \mathcal{C}$ has weight w if x has w nonzero entries. Recall that $x = y^t G$ is obtained by left-multiplying some $y \in \mathbb{F}_q^k$ into the $k \times n$ generator matrix $G = (G)_{ij}$ of $\mathcal{C}$. So we have

$$[x_1\ x_2 \dots x_n] = [y_1\ y_2 \dots y_k]G = [\sum_{i=1}^{k} y_i g_{1i}\ \ \sum_{i=1}^{k} y_i g_{2i} \dots \sum_{i=1}^{k} y_i g_{ni}] = [\langle y, g_1\rangle\ \ \langle y, g_2\rangle \dots \langle y, g_n\rangle],$$

where $\langle \cdot, \cdot \rangle$ is the ordinary dot product. It is easy to see that $x = [\langle y, g_1\rangle\ \ \langle y, g_2\rangle \dots \langle y, g_n\rangle]$ has w nonzero entries if and only if $n - w$ of the dot products of y with the g_i's are zero. Since the g_i's are precisely the columns of G, this proves the claim. □

Indeed, we can relate d to the parity-check matrix H of $\mathcal{C}$ as well.

Theorem 2.2.16. [Sin64] The Hamming distance of a code $\mathcal{C}$ is the minimum number d such that $d-1$ columns of a parity-check matrix H of $\mathcal{C}$ are linearly independent, while there exists a set of d columns of H that are linearly dependent.

Proof. Let d be the Hamming distance of $\mathcal{C}$. Then there exists some codeword $c \in \mathcal{C}$ such that $w(c) = d$. Recall that $Hc = \mathbf{0}$ by design. Since c has weight d, there are d nonzero entries in c. Interpret these as coefficients of the respective columns in H. Then with those non-trivial coefficients, the corresponding d columns are linearly combined to yield the zero vector, so they are linearly dependent. Now if there existed a set of $d-1$ linearly dependent columns, this would imply that some nonzero $c' \in \mathcal{C}$ with weight less than d was in the null space of H, which is a contradiction to Theorem 2.2.14 which states that the minimum weight over all nonzero codewords is precisely the Hamming distance of $\mathcal{C}$. For the converse, assume that there is a set of d columns of H that are linearly dependent, but the columns in any set of smaller cardinality are linearly independent. Then the nonzero codewords, which form the non-trivial part of the null space of H, must have minimum weight d, since otherwise we could linearly combine less than d columns non-trivially to obtain the zero vector, violating the assumed linear independence of sets of less than d columns. Hence the Hamming distance of $\mathcal{C}$ is d. □

For ease of notation, it is beneficial to refer to an $[n, k]$ code $\mathcal{C}$ over $\mathbb{F}_q$ with minimum distance $d = d(\mathcal{C})$ as $[n, k, d]_q$-code. Using this notation, our recurring example of a Hamming code can be expressed as linear $[7, 4, 3]_2$ code, or binary linear $[7, 4, 3]$-code. Indeed, there is a family of binary Hamming codes, all of which abide by the form $[2^m - 1, 2^m - m - 1, 3]_2$ for integers $m \geq 2$.

2.2.3.1. *Equivalence of codes.* Having introduced some main parameters for codes, it is helpful to see how different codes relate with respect to them.

Definition 2.2.17. (Equivalent linear codes). [Hil86, 50] Two $k \times n$ matrices generate *equivalent linear* $[n, k]_q$ *codes* if one matrix can be obtained from the other by a sequence of operations of the following types: (i) permutation of the rows, (ii) multiplication of a row by a nonzero scalar,

(iii) addition of a scalar multiple of one row to another, (iv) permutation of the columns, (v) multiplication of any column by a nonzero scalar.

As noted by Hill in [**Hil86**, 12], the distance of two equivalent codes is identical. This will be of importance when considering the connections between coding theory and projective space in Section 2.3.

2.2.4. Error-correcting capabilities.

With this general setup, we can finally make precise what it means for a code to be *error-correcting.*

Property 2.2.18. A code $\mathcal{C}$ corrects t errors if, for every $x \in \mathcal{S}^n$, there is at most one $y \in \mathcal{C}$ with $d(x, y) \leq t$.

This property follows from the necessity of any two codewords of $\mathcal{C}$ to be different enough to uniquely identify a transmitted message as a codeword of $\mathcal{C}$ – even under the presence of t errors. As a remark on linear algebra, recall that for a linear code $\mathcal{C}$, the codewords are of the form $y^t G$ for some $y \in \mathbb{F}_q^k$, or equivalently of the form $G^t y$. Thus, we can associate to G^t a linear transformation $\phi : \mathbb{Z}_q^k \to \mathbb{Z}_q^n$. Now ϕ clearly is not surjective, but for $\mathcal{C}$ to have error-correcting capabilities, ϕ must satisfy more than mere injectivity: It does not suffice to map pre-images to distinct images, but those images must abide by the constraint imposed by the minimum distance requirements.

The next theorem is connects the concepts of error-correction and minimum distance.

Theorem 2.2.19. [**Hil86**] A code $\mathcal{C}$ corrects t errors if and only if $d(\mathcal{C}) \geq 2t + 1$.

Proof. We prove both directions by their contrapositive. For the forward direction, suppose that $d(\mathcal{C}) \leq 2t$. Then there exist two words $x, y \in \mathcal{C}$ such that $d(x, y) \leq 2t$, i.e., x and y differ in at most $2t$ letters. If we have $d(x, y) \leq t$ as well, then there are two codewords within distance t from $x \in \mathcal{S}^n$, namely x and y, so $\mathcal{C}$ cannot possibly correct t errors. If otherwise $t < d(x, y) \leq 2t$ holds, then we can construct a word $c \in \mathcal{S}^n$ by copying x and changing t of the letters that differ between x and y to be the letters in y. Now for this c, we have $d(x, c) = t$ and $d(y, c) \leq t$, so there are two codewords within distance t from c, which implies that $\mathcal{C}$ does not correct t errors here either.

For the converse, assume that $\mathcal{C}$ does not correct t errors. Then there exists an element $c \in S^n$ and two codewords $x, y \in \mathcal{C}$ such that $d(x, c) \leq t$ and $d(y, c) \leq t$. But then by the triangle inequality and symmetry, it follows that $d(x, y) \leq d(x, c) + d(c, y) = d(x, c) + d(y, c) \leq 2t$. □

Example 2.2.20. By the above theorem, to show that the Hamming $[7, 4, 3]_2$ code corrects one error amounts to showing that $d(\mathcal{C}) \geq 3$, which by Theorem 2.2.14 is tantamount to showing that every nonzero $c \in \mathcal{C}$ has at least three nonzero entries. And by what we know about the parity-check matrix, it further suffices to show that if $x \in \mathbb{Z}_2^n$ has only one or two 1's, then it will not satisfy $Hx = \mathbf{0}$. Now x having a single 1 implies that H has a zero column, which it does not.[12] If x has exactly two 1's, this implies that one column of H is equal to another column of H scaled by -1. But since we are working over $\mathbb{Z}_2$, this means that H has two identical columns, which it does not. Also, from the listing of codewords we see that there exists a codeword in $\mathcal{C}$ with three nonzero entries, so we have $d(\mathcal{C}) = 3$, so the $[7, 4, 3]_2$ code indeed corrects one error.

2.2.4.1. *Encoding and Decoding.* To give an example of how the error-correction abilities of a linear code outlined above can be put into practice, we describe the standard procedure of decoding an encoded message using the $[7, 4, 3]_2$ code. Recall that a message is of the form $w = y^t G \in \mathbb{Z}_2^7$, where G is a generator matrix as seen in Example 2.2.6 and $y \in \mathbb{Z}_2^4$ is some word that we want to transmit. In order to decode the message w, we make use of the parity-check matrix H from Example 2.2.9, for which by definition $Hw = 0$ holds. Now assume that at most one error occurred during transmission, so that the receiver obtains the message w', which either contains no errors, i.e., $w' = w$, or differs in one symbol, i.e., $w' = w + e_i$, where the i-th symbol has the error and e_i is the corresponding standard basis vector. The receiver of the message can then compute Pw': If it is equal to 0, they can infer that $w = w'$, if not, we have $Pw' = P(w + e_i) = Pw + Pe_i = Pe_i$, which is the i-th column of P. Hence, this scheme can detect and correct one error in the transmission.

Now if the underlying field was not binary but of order q, the error might have been of the form $w' = w + se_i$, where $1 \leq s \leq q - 1$. Then the calculation above yields $Pw' = P(w + se_i) = Pw + P(se_i) = sPe_i$. But recall from Theorem 2.2.16 that any $d - 1$ columns of H are linearly independent, so in this example any 2 columns are linearly independent. Since sPe_i is a multiple of a column of P, and all of its columns are pairwise linearly independent, i.e., not multiples of one another, we can still uniquely identify the location of the error and correct it.

2.2.5. Bounds on codes.

Having set up some core concepts and definitions in the study of error-correcting codes, we briefly address the question of what constitutes a *good* code. We have already seen that the Hamming distance measures the error-correcting capabilities of a code, and maximizing the distance d indeed sounds like a reasonable criterion for a good code. However, large distance comes at a cost: Either the ambient space $\mathcal{S}^n$ is very large, or the dimensionality k of the code is small, i.e., we can only transmit a small number of messages. So unless we try to maximize one parameter with the other two being fixed, we encounter a trade-off situation. Hill summarizes this succinctly in [**Hil86**, 12], claiming that a good $[n, k, d]_q$ code has

> "small n (for fast transmission of messages), large k[13] (to enable transmission of a wide variety of messages) and large d (to correct many errors)."

[12] Note that we could have chosen other vectors for the basis of the dual code as row vectors of H, but none such choice would result in a zero column.

[13] Hill uses M instead of k.

Being aware of these considerations, we may also ask for the maximal size of an $[n, k, d]_q$-code, i.e., the maximal number of codewords that can be transmitted having fixed all three parameters n, k and d. Next, we give two examples of bounds on the size of a code.

Theorem 2.2.21. If $\mathcal{C}$ is a $[n, k, d]_q$-code of length n, then we have the following upper bound – the Hamming bound – on the size of $\mathcal{C}$:

$$|C| \leq \frac{q^n}{\sum_{k=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{k} (q-1)^k}.$$

Note that the upper limit of the sum in the denominator stems from the fact that a code of distance d corrects $\lfloor \frac{d-1}{2} \rfloor$ errors.[14] We refer to [**Hil86**, 2.16] for the proof of this theorem. Codes that achieve the Hamming bound are referred to as *perfect codes*. Not surprisingly, the Hamming codes are perfect codes, which is part of their special relevance in coding theory.

Another useful bound on the size of a code can be derived from Theorem 2.2.16. Here, we proved that for a $[n, k, d]_q$ code $\mathcal{C}$ and its parity-check matrix H, any $d-1$ columns of H are linearly independent, while there exist d columns that are linearly dependent. Now recall that H is a $(n-k) \times n$ matrix, so every column is an element of $\mathbb{F}_q^{n-k}$, which implies that we can never have more than $n-k$ linearly independent columns of H. This implies that $d \leq n-k+1 \Leftrightarrow k \leq n-d+1$, which gives rise to the so-called Singleton bound [**Sin64**, Theorem 1], named after Richard Singleton.

Theorem 2.2.22. If $\mathcal{C}$ is a $[n, k, d]_q$-code of length n, the size of $\mathcal{C}$ is bounded by the Singleton bound

$$|\mathcal{C}| < q^{n-d+1}.$$

Codes that achieve the Singleton bound are referred to as *maximum distance separable* (MDS) codes. These codes will be of particular interest to us in Section 2.3. It is noteworthy that both bounds approach the same question somewhat differently,[15] but that neither actually guarantees the existence of a code that is tight in the bound.[16]

Definition 2.2.23. (Maximum distance separable code). [**Rom92**, 235] A *maximum distance separable code* (MDS) code is a linear $[n, k, n-k+1]_q$ code.

From the above considerations, we derive the special role that MDS codes play in coding theory: For a MDS code, equality holds in $d \leq n-k+1$, so for given parameters n and k, an MDS code

[14] This theorem is also known as the sphere-packing bound [**Hil86**], since maximizing the number of codewords with minimum distance d in some ambient space can be rephrased as the problem of maximizing the number of non-intersecting spheres with radius d centered at the points in ambient space.

[15] None of them is a better bound in general, i.e., the quality of the bound can only be evaluated with respect to the given parameters.

[16] Both the Hamming and the Singleton bound simply pose theoretical thresholds on the size of an error-correcting code.

maximizes the distance d and thus its error-correcting capabilities. Given that a linear code $\mathcal{C}$ is a $[n, k, n-k+1]_q$ MDS code, we can list a variety of useful characterizations and properties of $\mathcal{C}$.

Properties 2.2.24.

(1) In the parity-check matrix H of $\mathcal{C}$, every set of $n-k$ columns of H is linearly independent, while there exists a set of $n-k+1$ columns that are linearly dependent.

Proof. This follows directly from Theorem 2.2.16. □

(2) Every set of k columns of the generator matrix G of $\mathcal{C}$ are linearly independent [**HHL+91**, 72f.].

Proof. Since $d = n-k+1$ and $\mathcal{C}$ is linear, every codeword must have support at least $n-k+1$, which is equivalent to stating that no nonzero codeword can have more than $n-(n-k+1) = k-1$ zeros. Now if k columns of the generator matrix were linearly dependent, then there would be a codeword with k zeros in the respective columns. Since this is not possible, any k columns are indeed independent. □

(3) The dual code $\mathcal{C}^\perp$ is a $[n, n-k, k+1]_q$ MDS code [**Rom92**, 235].

Proof. The dual code $\mathcal{C}^\perp$ has the generator matrix of $\mathcal{C}$ as its parity-check matrix. Refer to this matrix as H. Now from the second characteristic and the fact that $\mathcal{C}$ is a MDS code, we know that any k columns of H are linearly independent. Now $\mathcal{C}^\perp$ is a $[n, n-k, d^*]_q$ code, which is MDS if and only if $d^* = n-(n-k)+1 = k+1$. By Theorem 2.2.16, d^* is the minimum number such that d^*-1 columns of H are linearly independent, while there exists a set of d^* columns that are linearly dependent. Since we have seen that any k columns of H are linearly independent, we know that $d^*-1 \geq k \Leftrightarrow d^* \geq k+1$ must hold. But by the Singleton bound, we have $d^* \leq k+1$. Thus, $d^* = k+1$, and $\mathcal{C}^\perp$ is indeed a MDS code. □

There are various more characteristics of MDS codes [**Rom92**, 237], and those codes have been studied extensively throughout literatrue [**BL20, Lan00, HHL+91**], partly motivated by their rich connections to pure mathematics, but mainly because of their significant real-world applicability. Interestingly, a somewhat recent area of research is devoted to quantum MDS codes [**Bal21, JLLX10**], which are used in quantum-error correction for quantum computing and quantum communication. This shall serve as example that even though much of the groundbreaking work in this area has been done in the second half of the 20th century, coding theory remains an active field of research and interest.

2.2.6. Reed–Solomon codes.

We conclude this section by examining one particularly interesting example of MDS codes, the Reed–Solomon (RS) codes, which are used by various space agencies in satellite communication and constitute one of the most useful classes of error-correcting codes. Named after Irving S. Reed and Gustave Solomon, RS codes belong to the BCH codes, which is a family of cyclic codes.

Definition 2.2.25. (Cyclic codes). [HHL+91, 102] A linear code $\mathcal{C}$ over $\mathbb{F}_q$ is called a *cyclic code* if the cyclic shift of each codeword is also a codeword.

A cyclic code is an ideal in the polynomial ring $R_n = \mathbb{F}_q[x]/\langle x^n - 1\rangle$ [Rom92, 148], and as such an innately algebraic object. Furthermore, the codewords of a cyclic code can be identified with polynomials in x, where the components of the message words constitute the coefficients. This allows for the construction of a generator polynomial, which is the unique nonzero polynomial of minimum degree in $\mathcal{C}$ [HHL+91, 105]. Now RS codes as special type of cyclic codes oftentimes exhibit $q \geq n$, i.e., the order of the field is at least as large as the code length. So consider the case $q = n$ and let $g(x) = m_0 + m_1 x + m_2 x^2 + \ldots + m_{k-1}x^{k-1}$ be a generator polynomial for the RS code, where the m_i's are coefficients in $\mathbb{F}_q$. Using this, we can define the generator matrix

$$G = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ a_0 & a_1 & a_2 & \ldots & a_{q-1} \\ a_0^2 & a_1^2 & a_2^2 & \ldots & a_{q-1}^2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_0^{k-1} & a_1^{k-1} & a_2^{k-1} & \ldots & a_{q-1}^{k-1} \end{bmatrix},$$

where the a_i's are field elements.[17] Now a codeword is obtained by premultiplying a message word $(m_0, m_1, m_2, \ldots, m_{k-1})$ such that the codeword has the form $(g(a_0), g(a_1), g(a_2), \ldots g(a_{q-1}))$. For convenience, we identify the field elements naturally with $a_i = i$ for $0 \leq i \leq q-1$ such that we can express the generator matrix as

$$G = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 0 & 1 & 2 & \ldots & q-1 \\ 0 & 1 & 2^2 & \ldots & (q-1)^2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 1 & 2^{k-1} & \ldots & (q-1)^{k-1} \end{bmatrix}.$$

Now instead of storing the generator matrix of $\mathcal{C}$, which could be quite expensive for large field order or dimensionality, we can leverage to our advantage that one only needs to maintain the generator polynomial [HHL+91, 109]. Consequently, RS codes have gained great interest from industry, and some famous use cases that give testament to their wide applicability are error-correction in CDs and DVDs as well as encoding of transmissions to the Voyager spacecraft [Moo08].

Since Reed–Solomon codes have distance $d = n - k + 1$, a $[n, k, n-k+1]_q$ Reed–Solomon code can correct $\lfloor \frac{n-k}{2} \rfloor$ errors. Usually, the field order is typically chosen to be $q = 2^8$, but we will see an example of how to construct a Reed–Solomon code over $\mathbb{F}_7$ from our computational work in Section 2.4.

[17] This matrix is a so-called Vandermonde matrix.

2.3. Subspace-evasive sets over projective space and error-correcting codes

Having developed a foundation in finite projective geometry as well as coding theory, it is time to explore the connections between those abstract mathematical spaces over finite fields and error-correcting codes.

2.3.1. Error-Correcting codes and projective space.

Our first theorem extends Theorem 2.2.15 from the previous section by introducing a connection between linear codes and projective space. We assume that the linear codes in the remainder of this section are non-degenerate, i.e., that there is no coordinate in which all codewords are zero [**BDGP23**, 8].[18]

Theorem 2.3.1. [**Bis23**] A codeword $x = y^t G$ with $y \in \mathbb{F}_q^k$ has weight w if and only if the projective hyperplane induced by $y \cdot u = 0$ for $u \in \mathbb{F}_q^k$ has $n - w$ points in common, counting multiplicities, with the columns of G, when interpreted as points in $\mathrm{PG}(k-1, q)$.

Proof. In Theorem 2.2.15, we showed that a codeword $x = y^t G \in \mathcal{S}^n$ has weight w if and only if y has dot product equal to 0 with $n - w$ columns of the generator matrix of the code. Now for this equivalent statement about projective spaces, first note that the set $\{u \in \mathbb{F}_q^k : y \cdot u = 0\}$ indeed defines a hyperplane in $\mathbb{F}_q^k$, since the space is spanned by all vectors $u \in \mathbb{F}_q^k$ that are orthogonal to y, so it has normal vector y and codimension 1. Now upon interpreting the columns of the generator matrix as vectors in $\mathbb{F}_q^k$, since $y \cdot g_i = 0$ holds for $n-w$ of the g_i's, it follows that $n-w$ of the columns intersect the hyperplane. Recall Construction 2.1.2 of projective space from Section 2.1, where we collapse subspaces of $\mathbb{F}_q^k$ to subspaces of lower dimension in projective space. Then the hyperplane defined by $y \cdot u = 0$ in $\mathbb{F}_q^k$ is projected onto a projective hyperplane, i.e., a $(k-2)$-dimensional flat in $\mathrm{PG}(k-1, q)$, whereby each individual vector $u \in \mathbb{F}_q^k$ is mapped to a point in projective space. By construction, those projective points will lie on the projective hyperplane in $\mathrm{PG}(k-1, q)$, counting multiplicities. And since $n - w$ of the columns of the generator matrix intersect the hyperplane in $\mathbb{F}_q^k$, precisely $n - w$ of the projective points intersect this projective hyperplane, and vice versa. □

Caveat 2.3.2. In the above proof, we pointed out that we need to count multiplicities. This is necessary since some columns of the generator matrix might yield the same point – recall that points in $\mathrm{PG}(k, q)$ are 1-dimensional subspaces of $\mathbb{F}_q^{k+1}$ – in projective space, which happens precisely if two columns are linearly dependent.

Building upon this, we can paint the larger picture, relating $[n, k, d]_q$ codes and projective space.

Theorem 2.3.3. [**Bis23**] A linear $[n, k, \geq d]_q$ code gives rise to a multiset of n points in $\mathrm{PG}(k-1, q)$, counting multiplicities, that meets any projective hyperplane in at most $n-d$ points, and vice versa.

[18] This is equivalent to none of the columns of the generator matrix being all zero.

Proof. An $[n,k,d]_q$-code consists of codewords $x = y^t G$ with $y \in \mathbb{F}_q^k$, so upon projection on projective space, the entirety of the code will be described by a set of points in $\mathrm{PG}(k-1,q)$, where multiplicities greater than 1 are possible (i.e., the points form a multiset in projective space). Now let y_i be any vector in $\mathbb{F}_q^k$ and let w_i denote the weight of the codeword $x_i = {y_i}^t G$ for $1 \leq i \leq q^k$. We know that y_i induces some hyperplane $y_i \cdot u = 0$, consisting of vectors $u \in \mathbb{F}_q^k$ whose span has normal vector $y_i \in \mathbb{F}_q^k$. By the above theorem, the projective hyperplane resulting from this (vector space) hyperplane has $n - w_i$ points in common with the n columns of G when interpreted as projective points, counting multiplicities. Here, the expression $n - w_i$ is maximized when w_i is minimized. And by Theorem 2.2.14, the Hamming distance of the code is $d = \min_i\{w_i\}$, which implies that $n - w_i$ is maximized by d. Thus, every projective hyperplane has at most $n - d$ points in common with the columns of G.[19] (Note that if we allow for the Hamming distance of the code to be greater than d, the inequality remains valid). Lastly, every projective hyperplane in $\mathrm{PG}(k-1,q)$ originates from some hyperplane of $\mathbb{F}_q^k$, which is a $(k-1)$-dimensional flat and thus of the form $y_i \cdot u = 0$. Hence, a $[n,k,\geq d]_q$ code indeed gives rise to a multiset of points in $\mathrm{PG}(k-1,q)$ that intersects any projective hyperplane in at most $n - d$ points.

Conversely, consider a multiset of n projective points in $\mathrm{PG}(k-1,q)$ that meets any projective hyperplane in at most $n - d$ points. Interpret the homogeneous coordinates of the points as coordinates of the columns g_i of the generator matrix for a $[n,k]_q$ code.[20]. Since any hyperplane in projective space results from a (vector space) hyperplane of the form $y \cdot u = 0$, it follows from Theorem 2.3.1 that any codeword $x = y^t G$, with $y \in \mathbb{F}_q^k$ and G being the matrix containing the g_i's as columns, has weight d. Thus, the multiset indeed gives rise to a $[n,k,d]_q$ code. □

The above theorem yields a central equivalence between linear $[n,k,d]_q$ codes and projective space: The columns of a generator matrix for $\mathcal{C}$, which encapsulate information about the linear code, can be identified with points in projective space, and the maximal intersection size of those with the projective hyperplanes carries information about the distance of the code.[21] In the context of the dual code, we can associate the columns of the parity-check matrix H of $\mathcal{C}$ with points in the projective space $\mathrm{PG}(n-k-1,q)$ analogously.[22]

Remarks 2.3.4. Firstly, note that we do not have a clear bijection between linear $[n,k,d]$ codes and sets of points in projective space, since a projective point does not uniquely determine a corresponding column vectors for the generator matrix. Instead, when referring to *equivalence* of these objects, we understand that a set of projective points may give rise to different generator matrices, which however form *equivalent codes* in the sense of Definition 2.2.17 [**Hil86**, 180]. This

[19] When interpreted as projected points, and counting multiplicities.

[20] The columns of the generator matrix are recovered up to scalar multiples by transforming homogeneous coordinates to vector space coordinates.

[21] The distance of $\mathcal{C}$ is the minimal size of projective points *not* intersecting a hyperplane, over all hyperplanes.

[22] Also, since the generator matrix of $\mathcal{C}$ is the parity-check matrix of $\mathcal{C}^\perp$, the points in projective space originating from the above construction are associated to $\mathcal{C}^\perp$ as well.

is directly implied by Definition 2.2.17 (v), since the difference between the generator matrices will simply be the scale of the columns. In other words, a set of projective points with the assumed properties can be paired with a set of linear $[n, k, d]_q$ codes that exhibit the same parameters n, k and d.

Secondly, note how the theme of multisets brought up in Caveat 2.3.2 runs through the above proof as well. This motivates studying a special kind of code amongst the linear $[n, k, d]_q$ codes.

Definition 2.3.5. (Projective code). A linear $[n, k, d]_q$-code $\mathcal{C}$ is called *projective code* if the columns of the generator matrix G of $\mathcal{C}$ give rise to distinct points in $\mathrm{PG}(k-1, q)$.

Equivalently, any two columns of a generator matrix G of a projective code $\mathcal{C}$ are linearly independent.[23]

In the following, instead of allowing multiplicities to be counted,[24] we restrict our attention to projective codes. Next, we extend the connection between linear $[n, k, d]_q$ codes and points in $\mathrm{PG}(k-1, q)$ by using the concepts introduced in Section 1.1.2.

2.3.2. Error-Correcting codes and subspace-evasive sets.

Building upon Theorem 2.3.3, the goal of the next theorem is to establish an equivalence between linear $[n, k, d]_q$ codes and subspace-evasive sets in projective space.

Theorem 2.3.6. [Bis23] A (projective) linear $[n, k, d]_q$-code gives rise to a $(k-2, n-d)$-subspace-evasive set of size n in $\mathrm{PG}(k-1, q)$, and vice versa.

Proof. Recall from Definition 1.1.15 that a (s, m)-subspace-evasive set is a set of points that meets every s-dimensional subspace in at most m points. Now by Theorem 2.3.3, the set of projective points that originates from the columns of the generator matrix of a projective linear $[n, k, d]_q$-code meets every hyperplane of $\mathrm{PG}(k-1, q)$ in at most $n-d$ points. Since a hyperplane in $\mathrm{PG}(k-1, q)$ is a $(k-2)$-dimensional flat, we can restate the above to find that the set of projective points meets every $(k-2)$-dimensional flat in at most $n-d$ points. Then by Definition 1.1.15, this set is precisely a $(k-2, n-d)$-subspace evasive set of size n in $\mathrm{PG}(k-1, q)$.
The converse argument also follows directly from Theorem 2.3.3, noting that a $(k-2, n-d)$-subspace-evasive set of size n in $\mathrm{PG}(k-1, q)$ is a set that meets every projective hyperplane in at most $n-d$ points. □

Example 2.3.7. Recall our running example, the Hamming $[7, 4, 3]_2$ code, which we denote as $\mathcal{H}$. By the above theorem, $\mathcal{H}$ can be interpreted as a $(2, 4)$-subspace evasive set in $\mathrm{PG}(3, 2)$. In other words, upon projecting the columns of the generator matrix of $\mathcal{H}$ to $\mathrm{PG}(3, 2)$, the resulting points

[23] By Theorem 2.2.16, this directly implies that the Hamming distance of the dual code $\mathcal{C}^{\perp}$ is at least 3.

[24] which forces us to count the intersection with hyperplanes in projective space with multiplicities as well

will have the property that at most four of them intersect any projective plane.[25] Likewise, we can argue about the connection of the parity-check matrix to projective space. Therefore, first note that the dual code $\mathcal{H}^\perp$ of the $[7,4,3]_2$ code $\mathcal{H}$ is a $[7,3,4]_2$ code. Hence, the columns of the generator matrix for $\mathcal{H}^\perp$[26] form a $(1,3)$-subspace-evasive set[27] of size 7 in $\mathrm{PG}(2,2)$ – the entire Fano plane. Indeed, since the columns of both G and H are pairwise linearly independent, both $\mathcal{H}$ and $\mathcal{H}^\perp$ are projective codes in the sense of Definition 2.3.5.

Recall that an $(k-2, n-d)$-subspace-evasive set of size n in $\mathrm{PG}(k-1,q)$ can give rise to many different generator matrices, which may span different, but equivalent linear codes. In Section 2.2, when discussing some special families of codes, we identified minimum-distance separable (MDS) codes as particular objects of interest. Just as MDS codes are special types of codes, the following theorem is a special case of Theorem 2.3.6.

Theorem 2.3.8. [**BL20**, Theorem 17] For $n \geq k$, a (projective) linear MDS $[n,k,n-k+1]_q$ code gives rise to an n-arc in $\mathrm{PG}(k-1,q)$, and vice versa.

Proof. Let C be a linear MDS $[n,k,n-k+1]_q$ code. Then by Theorem 2.3.6, we know that C gives rise to a $(k-2, n-(n-k+1))$-subspace-evasive set, i.e., a $(k-2,k-1)$-subspace-evasive, in $\mathrm{PG}(k-1,q)$ of size n, and vice versa. Now by Definition 2.1.10, a $(k-2,k-1)$-subspace-evasive set in $\mathrm{PG}(k-1,q)$ of size n is precisely an n-arc in this space. □

Another way of seeing why the above theorem holds comes from the property of an MDS code that any set of k columns of its generator matrix is linearly independent. That means, any k columns span a k-dimensional subspace of the vector space $\mathbb{F}_q^k$, which in turn means they span the entire $\mathbb{F}_q^k$. But then any k points span, upon projection to $\mathrm{PG}(k-1,q)$, the entire $(k-1)$-dimensional projective space, which implies that at most $k-1$ lie on a hyperplane in $\mathrm{PG}(k-1,q)$. This in turn shows that the n columns as projective points form an n-arc in $\mathrm{PG}(k,q)$ [**BL20**].

Theorem 2.3.8 finally paves the way for the connection between linear $[n,k,d]_q$ codes and m-general sets. Recall that a m-general set is a $(m-2,m-1)$-subspace-evasive set. Hence, if an m-general set of projective points lives in $\mathrm{PG}(k-1,q)$ of appropriate dimension, we can interpret the set of size n as an n-arc in projective space. From the above discussion, it is clear that we have to choose $k=m$, such that an m-general set defines a $(k-2,k-1)$-subspace-evasive set, and therefore an arc, over $\mathrm{PG}(k-1,q)$. We can summarize this in the following corollary.

Corollary 2.3.9. For $k \geq 3$ and $n \geq k$, a linear MDS $[n,k,n-k+1]_q$ code gives rise to a k-general set over $\mathrm{PG}(k-1,q)$, and vice versa.

[25] The projective planes in $\mathrm{PG}(3,2)$ are Fano planes.

[26] I.e., the parity-check matrix for $\mathcal{H}$.

[27] at most three of the projective points lie on a line

Proof. This follows directly from the above since a k-general set of size n in $\mathrm{PG}(k-1,q)$ is indeed an n-arc. □

As a sanity check, recall from Property 2.2.24 (2) that in the generator matrix G of an MDS code, every set of k columns is linearly independent. So consider the columns of a generator matrix of a linear $[n,k,n-k+1]_q$ MDS code, interpreted as projective points. By the above corollary, those points form a k-general set in $\mathrm{PG}(k-1,q)$, which implies that at most $k-1$ points intersect any projective hyperplane. So any such set of k columns in $\mathbb{F}_q^k$ must be linearly independent, since otherwise its pendant of projective points would intersect a hyperplane in projective space in k points. Hence, we indeed obtain the expected consequence on linear independence from Corollary 2.3.9.

Lastly, recall from Property 2.2.24 (3) that the dual of a MDS code is a MDS code. Hence, a parity check matrix H of a $[n,k,n-k+1]_q$ MDS code $\mathcal{C}$ is likewise a generator matrix for a $[n,n-k,k+1]_q$ MDS code $\mathcal{C}^\perp$. Now $\mathcal{C}$ corresponds to a k-general set over $\mathrm{PG}(k-1,q)$, and by Theorem 2.3.9 we know that $\mathcal{C}^\perp$ corresponds to a $(n-k)$-general set over $\mathrm{PG}(n-k-1,q)$.[28] Interestingly, this gives a non-trivial way of pairing m-general sets over projective spaces, induced by duality of codes.

2.3.3. Some practical examples involving m-general sets.

Notably, Corollary 2.3.9 allows us to devise some examples of the correspondence developed in this section. Focusing on $m = 3,4,5$, we find the following implication of m-general sets in the appropriate projective spaces on the existence of (projective) linear MDS codes:

- *capsets in* $\mathrm{PG}(2,q)$
  For capsets ($m=3$), we know by Theorem 2.3.9 that 3-general sets over $\mathrm{PG}(2,q)$ give rise to linear MDS $[n,3,n-2]_q$ codes. In Section 2.4, we will show how to construct a special such code, a linear $[7,3,5]_7$ Reed–Solomon code, which can correct two errors over $\mathbb{F}_7$.[29] Further note that by the discussion above, we know that the dual $[7,4,4]_7$ code of this RS code is also MDS, and by Corollary 2.3.9, this code gives rise to a 4-general set of size 7 over $\mathrm{PG}(3,7)$.

- *Sidon sets in* $\mathrm{AG}(3,3)$
  Recall from Section 1.2 that Sidon sets in $\mathrm{AG}(k,3)$ are 4-general sets. Furthermore, m-general sets in $\mathrm{AG}(k,q)$ are m-general sets over $\mathrm{PG}(k,q)$ by Theorem 2.1.9, so a Sidon set $\mathcal{S}$ over $\mathrm{AG}(k,3)$ gives rise to a 4-general set over $\mathrm{PG}(k,3)$. By Corollary 2.3.9, a 4-general set over $\mathrm{PG}(3,q)$ in turn corresponds to an MDS code, so in particular a Sidon set of size n over $\mathrm{AG}(3,3)$ gives rise to a ternary $[n,4,n-3]$ MDS code. In Section 1.3 we

[28] This agrees with Property 2.2.24 (1), which states that any set of $n-k$ columns of H – the generator matrix of $C^\perp$ – is linearly independent.

[29] This RS code is a projective arc as well as a projective cap of size 7.

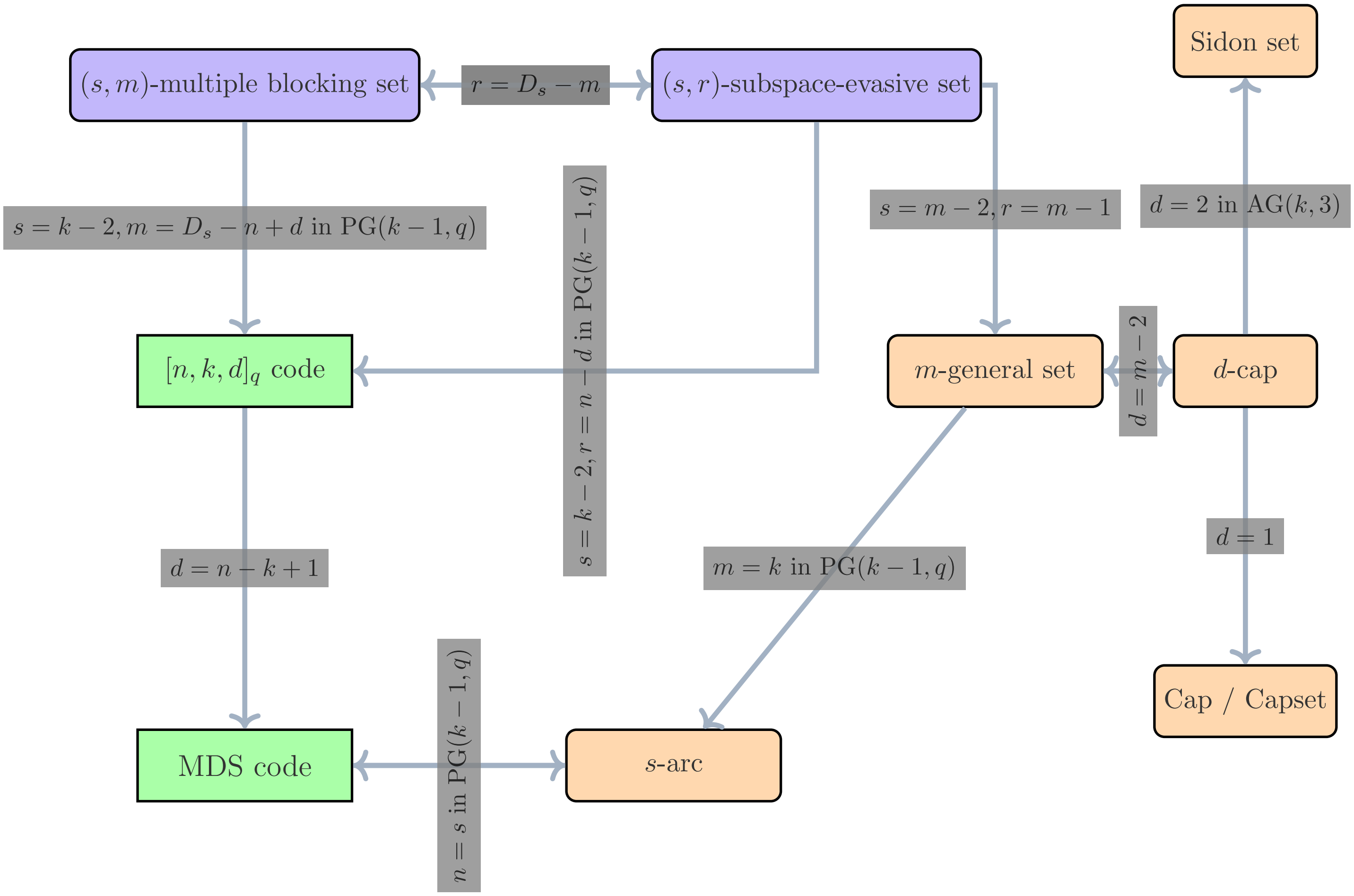
computationally determined maximal complete Sidon sets of maximal size 5, which give rise to a $[5,4,2]_3$ code. This code, however, is trivial [**Rom92**, 235] and does not correct any errors.

- 4-*general sets in* $\mathrm{PG}(3,q)$
  For the more general case, observe that a 4-general set over $\mathrm{PG}(3,5)$ gives rise to a MDS code. As will be shown in Section 2.4, a maximal complete such set has size 6, which implies equivalence of such 4-general sets in $\mathrm{PG}(3,5)$ and linear $[6,4,3]_5$ MDS codes, which can correct one error.
- 5-general sets in $\mathrm{PG}(4,q)$
  Likewise, we show that a maximal complete 5-general set in $\mathrm{PG}(4,7)$ has size at least 8, which gives rise to to a $[8,5,4]_7$ code.
- 224-general sets in $\mathrm{PG}(223,256)$
  The immensely useful Reed–Solomon $[255,223,33]_{256}$ code, which corrects 16 errors and has been used for NASA missions, resembles a 224-general set in $\mathrm{PG}(223,256)$.[30]

In this section, we have explored the deep connections that error-correcting codes and m-general sets in the projective geometry $\mathrm{PG}(k,q)$ bear. In particular, we have seen how linear codes can be viewed through the lense of geometry in projective spaces and vice versa. We conclude by summarizing some of the most important concepts and connections, after which we continue to the last section of the thesis, in which we present a computational approach to m-general sets in projective space.

[30] We choose to omit a computational search of projective space of dimension 223 over the field of order 256.

2.3.4. Summary Graphic.


Sidon set
(s,m)-multiple blocking set
$r = D_s - m$
(s,r)-subspace-evasive set
$s = m-2, r = m-1$
$d = 2$ in $\mathrm{AG}(k,3)$
$s = k-2, m = D_s - n + d$ in $\mathrm{PG}(k-1,q)$
$s = k-2, r = n-d$ in $\mathrm{PG}(k-1,q)$
$[n,k,d]_q$ code
m-general set
$d = m-2$
d-cap
$d = n-k+1$
$m = k$ in $\mathrm{PG}(k-1,q)$
$d = 1$
Cap / Capset
MDS code
$n = s$ in $\mathrm{PG}(k-1,q)$
s-arc

2.4. A computational approach to m-general sets over projective spaces

The appearance of general sets in coding theory, as treated in Section 2.3, and the possibility of constructive methods for codes pose a particular motivation for the computational study of m-general sets in projective space. Hence, we present a computational approach on finding m-general sets over projective space in in order to determine a variety of small and large complete m-general sets in $\mathrm{PG}(k,q)$ in low dimensions and with respect to $\mathbb{F}_2$, $\mathbb{F}_3$ and $\mathbb{F}_7$.[31] It is noteworthy that the coarse outline of the computational search is comparable in affine and projective space, but the conditions to be checked for points to be added rely on a different strategy, since the convenient formulations of affine combinations utilized in Section 2.4 do no transfer to projective space.

2.4.1. Finding minimal and maximal complete complete m-general sets in $\mathrm{PG}(k,q)$.

Our main insight for working with objects in the projective setting arises from leveraging the connection between homogeneous coordinates in $\mathrm{PG}(k,q)$ and vector coordinates in $\mathbb{F}_q^{k+1}$. The key ingredient for finding m-general sets in $\mathrm{PG}(k,q)$ here is the following chain of biconditional statements.

Theorem 2.4.1. A set of points $\mathcal{P}$ is m-general in $\mathrm{PG}(k,q)$ if and only if no m points in $\mathcal{P}$ lie on a $(m-2)$-dimensional flat of $\mathrm{PG}(k,q)$ if and only if no m vectors, obtained from transforming the homogeneous coordinates of the points in $\mathcal{P}$ to vector coordinates in $\mathbb{F}_q^{k+1}$, lie on a $(m-1)$-dimensional flat of $\mathbb{F}_q^{k+1}$ if and only if every m-subset of such vectors is linearly independent in $\mathbb{F}_q^{k+1}$.

Proof. We need to prove three biconditional statements, where the first statement follows directly from the definitions. For the second statement, we prove the forward direction by its contrapositive. So assume that some m vectors, obtained from transforming the homogeneous coordinates of points in $\mathcal{P}$ to vector coordinates in $\mathbb{F}_q^{k+1}$, lie on a $(m-1)$-dimensional flat of $\mathbb{F}_q^{k+1}$. Then the corresponding m projective points lie on a $(m-2)$-dimensional flat in $\mathrm{PG}(k,q)$ by Construction 2.1.2.

Conversely, assume that no m vectors obtained from transforming homogeneous coordinates to vector coordinates in $\mathbb{F}_q^{k+1}$ lie on a $(m-1)$-dimensional flat of $\mathbb{F}_q^{k+1}$. Then any corresponding m projective points will not lie on a $(m-2)$-dimensional flat in $\mathrm{PG}(k,q)$ by Construction 2.1.2 again. For the third biconditional statement, we prove both directions by their respective contrapositive. So assume that there exists a m-subset of vectors which is linearly dependent in $\mathbb{F}_q^{k+1}$. This directly implies that some m vectors lie on a $(m-1)$-dimensional flat. Conversely, assume that there exist m vectors in $\mathbb{F}_q^{k+1}$ that lie on a $(m-1)$-dimensional flat of $\mathbb{F}_q^{k+1}$. Since any basis for $(m-1)$-dimensional space has size exactly $m-1$, these vectors are linearly dependent. □

[31] We choose $\mathbb{F}_2$ and $\mathbb{F}_3$ since those fields are important in the context of binary and ternary codes, and $\mathbb{F}_7$ due to its relevance to the RS code alluded to in Section 2.3.

The above theorem provides the needed traction to handle projective m-general sets computationally: A set is m-general if and only if every m-subset of vectors in $\mathbb{F}_q^{k+1}$, obtained from transforming homogeneoues coordinates to vector coordinates, is linearly independent. Checking linear independence amounts to checking whether or not for every m-subset there exists a non-trivial linear combination of the vectors in the set that sums to the zero vector in $\mathbb{F}_q^{k+1}$.

Remark 2.4.2. Since projective points are 1-dimensional vector subspaces, there are multiple choices for the conversion of homogeneous to vector coordinates. We choose the simplest such conversion, not requiring any rescaling, as implemented in Code Excerpt 0.0.6.

Analogous to our approach in Section 1.3, we want to initialize our search with a set of projective points that is guaranteed to be m-general, and then traverse $\mathrm{PG}(k, q)$ recursively, adding as many points as possible by the condition described above. However, the fact that not all points are equivalent in projective space prevents the convenience of affinely equivalent starting sets employed in the affine search. In order to draw conclusion on minimality or maximality of set sizes, we use an empty initialization in small dimensions, which allows for a comprehensive search. However, in order to at least obtain bounds in higher dimensions ($k \geq 6$), we initialize with a non-empty set that is guaranteed to be m-general. Here, we replicate our approach in affine space and add vectors in $\mathbb{F}_q^{k+1}$ to a set such that each new point introduces a new dimension. This set of vectors, and in particular every m-subset of those vectors, is linearly independent. Then upon projecting to points in $\mathrm{PG}(k, q)$, we obtain the desired desired m-general starting set.[32] Next, we recursively traverse the search space, testing for every possible additional point if the resulting set satisfies m-generality, repeating the process until no more points can be added, followed by outputting minimal and maximal complete such sets. We provide pseudocode for the search of large m-general sets in $\mathrm{PG}(k, q)$ in Algorithm 4.

[32] This procedure amounts to choosing the $k+1$ standard basis vectors in this $\mathbb{F}_q^{k+1}$ followed by projecting to projective points. Now those projective points have the exact same homogeneous coordinates as the vectors they arise from have vector coordinates. When constructing projective points in our script by adding all possible homogeneous coordinates for a given field order and dimension, one can check that we find those “standard basis points” exactly in the 1st, 2nd, $(1+q+1)$st, $(1+q+q^2+1)$st etc. positions of the list of projective points created by the function `create_PG` in `m-general-sets-PG.py`. Besides from the first point, the indices of those points in the list admit the algebraic form $1 + q + \ldots + q^i = (q^{i+1} - 1)/(q - 1)$ for $0 \leq i \leq k - 1$, which we leverage when creating the starting set.

Algorithm 4 Finding large complete m-general sets in $\mathrm{PG}(k, q)$

```
Require: dimension $k$, field order $q$
  maxSize $\leftarrow 0$
  maxSet $\leftarrow$ [ ]
  set $\leftarrow$ {Initialize starting set}
  possible $\leftarrow$ {Determine points that may be added}
  call buildCap(set, possible)

  function BUILDCAP(set, possible)
    if len(possible) $= 0$ & len(set) $>$ maxSize then
      maxSize $\leftarrow$ len(set)
      maxSet $\leftarrow$ set
    else
      for i in range(0, len(possible)) do
        set.append(possible[i])
        possible $\leftarrow$ {Determine new possible points}
        call buildCap(set, possible)
      end for
    end if
  end function
```

Note that this algorithm works in close analogy to Algorithm 3, but that the details, e.g., of identifying possible points to be added to a given set, are hidden. The above considerations are implemented in the Python script `m-general-sets-PG.py`, supplemented by the script `m-general-sets-PG-test.py`, which directly uses the condition developed in Theorem 2.4.1 to check whether or not a given input set of projective points is m-general or not.

2.4.2. A computational illustration of the connection to error-correcting codes.

As described in Section 2.3.3, finding m-general sets over projective space enables us to construct MDS codes explicitly from the relation of projective arcs and generator matrices. Assume we have obtained some n-arc over $\mathrm{PG}(k-1, q)$, i.e., a set of n projective points that are k-general in projective space of dimension $k-1$ over the field $\mathbb{F}_q$. Then by Theorem 2.3, this set gives rise to a MDS $[n, k, n-k+1]$ code, for which a generator matrix can be found via the following procedure:

- Determine the homogeneous coordinates of each point, where each of the n points corresponds to a k-tuple.
- Populate a $k \times n$-matrix with the n tuples of length k. (For uniqueness of construction, we may arrange the columns by lexicographical ordering.)

Note that Definition 2.3.5 guarantees that every code constructed in this way is a projective code, since the columns of the generator or parity-check matrix will originate from distinct points in projective space. Also observe that the output of our script makes it easy to construct such matrices, since points are readily available in homogeneous coordinates.

2.4.2.1. *Constructing MDS codes from projective arcs.* In the following, we provide computational illustrations for the correspondences described in Section 2.3.3, explicitly constructing or checking the respective MDS codes with our scripts.

- capsets in $\mathrm{PG}(2, q)$

  By our discussion on Reed–Solomon codes in Section 2.2.6, we can construct a linear $[7, 3, 5]_7$ code, which can correct two errors over $\mathbb{F}_7$ using the Vandermonde generator matrix

  $$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 1 & 4 & 2 & 2 & 4 & 1 \end{bmatrix}$$

  Then this code gives rise to the following projective cap of 7 points over $\mathrm{PG}(2, 7)$, expressed in homogeneous coordinates:

  $$[1, 0, 0], [1, 1, 1], [1, 2, 4], [1, 3, 2], [1, 4, 2], [1, 5, 4], [1, 6, 1].$$

  For those points, we can use the script `m-general-sets-PG-test.py` to verify that the set is indeed a 3-general set.[33] Note that in general, all RS codes have a counterpart of projective arcs, but not every projective arc gives rise to an RS code, since those are special among the MDS codes.[34]

[33] The output being: “The set of points $[[1, 0, 0], [1, 1, 1], [1, 2, 4], [1, 3, 2], [1, 4, 2], [1, 5, 4], [1, 6, 1]]$ is a 3-general set in PG(2,7)? True.”

[34] All MDS codes give rise to projective arcs and vice versa.

- 4-general sets in $\mathrm{PG}(3, q)$

  Searching the output of capsets in $\mathrm{PG}(3, 5)$ that we generate with the script `m-general-sets-PG.py`, we find that a complete 4-general set of maximal size 6 in $\mathrm{PG}(3, 5)$ is given by

  $$[0, 0, 0, 1], [0, 0, 1, 0], [0, 1, 0, 0], [1, 0, 0, 0], [1, 1, 1, 1], [1, 2, 3, 4].$$

  Following the procedure outlined above (discarding the seventh point), we can create the generator matrix of a $[6, 4, 3]_5$ code from this set, choosing the order such that the identity matrix evolves:

  $$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 & 1 & 4 \end{bmatrix}$$

  Using the Matlab code `code_distance.m`, we can verify that the Hamming distance of the code defined by this generator matrix is indeed 3.

- 5-general sets in $\mathrm{PG}(4, q)$

  Considering the output of 5-general sets in $\mathrm{PG}(4, 7)$ that we generate with the script `m-general-sets-PG.py`, we find that a complete 5-general set of size 8 is given by

  $$[0, 0, 0, 0, 1], [0, 0, 0, 1, 0], [0, 0, 1, 0, 0], [0, 1, 0, 0, 0],$$
  $$[1, 0, 0, 0, 0], [1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 6, 5, 3, 4].$$

  Again following the procedure outlined above, we create the generator matrix of a $[8, 5, 4]_7$ code from this:

  $$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 2 & 6 \\ 0 & 0 & 1 & 0 & 0 & 1 & 3 & 5 \\ 0 & 0 & 0 & 1 & 0 & 1 & 4 & 3 \\ 0 & 0 & 0 & 0 & 1 & 1 & 5 & 4 \end{bmatrix}.$$

  Using the script `code_distance.m` again, we verify that the Hamming distance of the code defined by this generator matrix is indeed 4.

2.4.3. Computational results in projective space.

Having seen the tangible connection between m-general sets and error-correcting codes through projective arcs, we conclude this section by presenting results on the size of small and large complete m-general sets over $\mathrm{PG}(k,q)$ for $q = 2, 3$ and small k. For computational speed-up, we again employed the Pegasus High Performance Computing Cluster.

$\mathbb{F}_2$	m = 3			m = 4			m = 5		
k	small	large	reference	small	large	reference	small	large	reference
2	4	4	2.4 / 2^k	-	-	-	-	-	-
3	5	8	”	5	5	[Pav23, Table 1]	-	-	-
4	≤ 9	16	”	6	6	[Pav23, Table 1]	≤ 6	6	2.4
5	≤ 32	32	”	7	8	[Pav23, Table 1]	≤ 7	≥ 7	2.4
6	≤ 64	64	”	11	11	[Pav23, Table 1]	≤ 8	≥ 9	2.4
7	≤ 128	128	”	≤ 15	≥ 17	2.4	≤ 12	≥ 12	2.4
8	≤ 256	256	”	≤ 21	≥ 22	2.4	≤ 16	≥ 18	2.4

TABLE 1. m-general sets in $\mathrm{PG}(k,2)$

$\mathbb{F}_3$	m = 3			m = 4			m = 5		
k	small	large	reference	small	large	reference	small	large	reference
2	4	4	2.4 / [Bos47]	-	-	-	-	-	-
3	≤ 8	10	2.4 /[Bos47]	5	5	[Pav23, Table 1]	-	-	-
4	≤ 16	20	2.4 / [Pel70]	11	11	[Pav23, Table 2]	≤ 6	6	2.4
5	≤ 32	56	2.4 / [Hil83]	≤ 12	≥ 13	2.4	≤ 12	≥ 12	2.4
6	≤ 112	≥ 112	[BE11, Table 1]				≤ 13	≥ 13	2.4
7	≥ 248	≥ 248	[BE11, Table 1]	≤ 28	≥ 28	[Pav23, 5.3]			

TABLE 2. m-general sets in $\mathrm{PG}(k,3)$

The fact that a maximal 5-general set over $\mathrm{PG}(4,3)$ has size 6 follows from the fact that there exists a $[6,5,2]_3$ MDS code, but no $[7,5,3]_3$ MDS code, according to [Gra], whereas we saw in Section 2.3 that a 5-general set of size 7 would give rise to the latter code.

Summary and Outlook

Throughout this thesis, we have encountered the deep connections between m-general sets and Sidon sets as well as error-correcting codes, shedding light on how m-general sets in different geometries relate seemingly distinct mathematical objects.

The first part of our work was largely concerned with finding large Sidon sets in $\mathrm{AG}(k, 3)$ for odd dimensions k. While we worked on this question via three different approaches, giving a new construction and computational results to improve a lower bound on $r_4(k, 3)$, the overall problem remains unsolved and motivates further work. It could be rewarding to search for a potentially better construction of maximal such Sidon sets, and ultimately to give a tight bound on their size in odd dimension. The fact that a new line of research involving Large Language Models can be explored in this context is promising. Here, we ventured into an entirely new and promising area of research – the application of Large Language Models to (pure) mathematics. It would be fascinating to find out if the case $k = 7$ (and possible other cases) can be improved with the LLM approach, given appropriate computational capabilities and model access.

While giving a synopsis of the connections between m-general sets, projective arcs and error-correcting codes, and contributing tangible illustrations of the derived correspondences in the second part of the paper, we have only provided an introductory treatment of projective geometries and configurations of points herein. For example, there exists a multitude of striking connections to areas such as design theory that we did not cover. It might be of further interest to investigate whether the ideas and correspondences that we saw in the second part of the thesis, paired with known results on linear codes, can be leveraged to improve the given bounds in the first part further.

Overall, while some of the knowledge in the realm of this thesis has been established for some time already, it is encouraging to see that progress is still being made, be it with new theoretical insights or computational methods.

Code

The GitHub page of the author contains the Python and Matlab scripts that were used for the exploration of m-general sets in affine and projective spaces throughout this thesis. It can be accessed at https://github.com/timneumann1, where the repository accessible at https://github.com/timneumann1/Senior-Thesis-m-general-sets contains the code used for the computational and coding theory sections, while the repository accessible at https://github.com/timneumann1/FunSearch-Sidon contains the code for the FunSearch experiments, which are based on Google DeepMind's algorithm [RPBN+24] and the architecture provided by a GitHub user.

Below we collect code samples as referenced to throughout the thesis.

Code 0.0.3. Initialization of search according to the construction resulting from Theorem 1.2.4.

```
def sidon_recursion(n):
    if (n \% 2 == 0):
        return sidon(n)
    if (n == 1):
        return [[0],[1]]
    else:
        n = n-1
        list = []
        new_points_rec = sidon(n)
        for p in new_points_rec:
            list.append( np.concatenate(([0],p)) )

        new_points_rec = sidon_recursion(int(n/2))
        for p in new_points_rec:
            list.append( np.concatenate(([1],p,np.zeros(n//2, dtype = int))) )
        return list

def sidon(n):
    GF = galois.GF(3**int(n/2), repr="int")
    sidon_points = []
    for i in range(GF.order):
        element = GF(i)
        element_vec = element.vector()
        square_element = element ** 2
        square_element_vec = square_element.vector()
        sidon_element = np.concatenate((np.array(square_element_vec)[::-1],np.
                                                  array(element_vec)[::-1]))
        sidon_points.append(sidon_element)
    return sidon_points
```

```
def main():
    points = sidon_recursion(n)
    for p in points:
        sidon_set.append(p)
if __name__ == ’__main__’:
    global n
    n = <Input dimension>
    global sidon_set
    sidon_set = []
    main()
```

Code 0.0.4. Output of `maximal-sidon-set-odd_dimension.py` with for input dimension 9:

A complete 4-general set of size 91 in $\mathrm{AG}(9,3)$ is given by

[0, 82, 163, 252, 340, 418, 495, 580, 664, 757, 857, 929, 982, 1061, 1157, 1279, 1355, 1457, 1486, 1577, 1667, 1765, 1862, 1922, 1954, 2048, 2114, 2227, 2282, 2417, 2451, 2584, 2644, 2750, 2805, 2853, 2953, 2999, 3152, 3231, 3283, 3325, 3422, 3549, 3597, 3660, 3805, 3841, 3914, 4026, 4086, 4156, 4268, 4334, 4414, 4523, 4550, 4694, 4716, 4830, 4881, 4993, 5095, 5118, 5218, 5344, 5371, 5468, 5564, 5615, 5706, 5808, 5869, 5987, 5996, 6095, 6189, 6303, 6390, 6403, 6523, 6561, 6562, 6564, 6570, 6574, 6588, 6605, 6626, 6639, 13183].

Code 0.0.5. Sample priority function that gives rise to a Sidon set of size 31 in $\mathrm{AG}(7,3)$

```
def priority(el: tuple[int, ...], n: int) -> float:
  """Returns the priority with which we want to add ‘element‘ to the set with
        pairwise distinct sums."""
  """Improved version of ‘priority_v1‘."""

  mn, mx = np.min(el), np.max(el)

  return (
  np.min(
    np.abs(np.dot(el, list(range(n))) - np.dot(el, list(range(1, n + 1)))))+
  mn + mx +
  np.min(
      np.diff(el)) + (n - len(el)) *
        ((mn + mx - np.min(np.diff(el))) *
         (mn + mx - np.min(np.diff(el)) - 2) / 2 + mn + mx)
      )
```

Code 0.0.6. Computational conversion from homogeneous coordinates in $\mathrm{PG}(k,q)$ to vector coordinates in $\mathbb{F}_q^{k+1}$.

```python
def point_PG_to_VS(pointPG,q):
    '''
    This function transforms a point in projective space PG(n,q), expressed in its
    homogeneous coordinates, to a point in the vector space F_q^(n+1). This is
    achieved by rescaling the homogeneous coordinates based on a nonzero anchor.
    Note that by this way of expressing points in projective space, the
    transformation viewed as a function is injective but not surjective, i.e., no
    two points in projective space will yield the same vector in the higher-
    dimensional vector space, but not every point in the vector space will be
    covered by this construction.

    Input: point in projective space, order of the underlying field
    Output: point in vector space
    '''

    pointVS = [None]*len(pointPG)
    homogeneous_anchor = 0 # this anchor will be changed during execution since we exclude the zero vector in the input domain
    for i in range(len(pointPG)-1,-1,-1):
        pointVS[i] = pointPG[i]
        if pointVS[i] != 0:
            homogeneous_anchor = pointVS[i]
            loc_anchor = i

    for i in range(loc_anchor,len(pointPG)):
        pointVS[i] = (pointPG[i] * inverse_finite_field(homogeneous_anchor,q) ) % q

    return pointVS
```

Bibliography

[Bal21] Simeon Ball, *Some constructions of quantum MDS codes*, Designs, Codes and Cryptography **89** (2021), no. 5, 811–821.

[BDGP23] Anurag Bishnoi, Jozefien D'haeseleer, Dion Gijswijt, and Aditya Potukuchi, *Blocking sets, minimal codes and trifferent codes*.

[BE11] Jürgen Bierbrauer and Daniel Edel, *Large caps in projective Galois spaces*, 2011.

[Ben19] Michael Bennett, *Bounds on sizes of generalized caps in $AG(n, q)$ via the Croot-Lev-Pach polynomial method*, Journal of Combinatorial Theory, Series A **168** (2019), 255–271.

[Bis23] Anurag Bishnoi, *Sets of points meeting each subspace in a few points*, 2023.

[BL20] Simeon Ball and Michel Lavrauw, *Arcs in finite projective spaces*, 2020.

[BMR$^+$20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei, *Language Models are Few-Shot Learners*, 2020.

[Bos47] R. C. Bose, *Mathematical Theory of the Symmetrical Factorial Design*, Sankhyā: The Indian Journal of Statistics (1933-1960) **8** (1947), no. 2, 107–166.

[CFG$^+$22] Julia Crager, Felicia Flores, Timothy E. Goldberg, Lauren L. Rose, Daniel Rose-Levine, Darrion Thornburgh, and Raphael Walker, *How many cards should you lay out in a game of EvenQuads?: A detailed study of 2-caps in $AG(n, 2)$*, 2022.

[Cil12] Javier Cilleruelo, *Combinatorial problems in finite fields and sidon sets*, Combinatorica **32** (2012), no. 5, 497–511.

[CP23] Ingo Czerwinski and Alexander Pott, *Sidon sets, sum-free sets and linear codes*, 2023.

[DL11] Zeev Dvir and Shachar Lovett, *Subspace Evasive Sets*.

[DM03] Benjamin Lent Davis and Diane Maclagan, *The card game SET*, The Mathematical Intelligencer **25** (2003), no. 3, 33–40.

[EFLS02] Y. Edel, S. Ferret, I. Landjev, and L. Storme, *The Classification of the Largest Caps in $AG(5, 3)$*, Journal of Combinatorial Theory, Series A **99** (2002), no. 1, 95–110.

[EG16] Jordan S. Ellenberg and Dion Gijswijt, *On large subsets of $\mathbb{F}_q^n$ with no three-term arithmetic progression*, 2016.

[Ewa71] Günter Ewald, *Geometry: An Introduction*, Wadsworth Publishing Company, 1971.

[Gra] Markus Grassl, *Code Tables*, http://codetables.de/, Accessed: 2024-05-09.

[HHL$^+$91] D.G. Hoffman, D. Hoffman, D.A. Leonard, C.C. Lindner, K.T. Phelps, C.A. Rodger, and J.R. Wall, *Coding Theory: The Essentials*, Monographs and textbooks in pure and applied mathematics, M. Dekker, 1991.

[Hil83] Raymond Hill, *On Pellegrino's 20-caps in $S_{4,3}$*, North-holland Mathematics Studies **78** (1983), 433–447.

[Hil86] ______, *A First Course in Coding Theory*, Oxford Applied Linguistics, Clarendon Press, 1986.

[HTW19] Yixuan Huang, Michael Tait, and Robert Won, *Sidon sets and 2-caps in* $\mathbb{F}_3^n$, Involve, a Journal of Mathematics **12** (2019), no. 6, 995–1003.

[JEP$^+$21] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis, *Highly accurate protein structure prediction with AlphaFold*, Nature **596** (2021), no. 7873, 583–589.

[JLLX10] Lingfei Jin, San Ling, Jinquan Luo, and Chaoping Xing, *Application of classical Hermitian self-orthogonal MDS codes to quantum MDS codes*, IEEE Trans. Inform. Theory **56** (2010), no. 9, 4735–4740. MR 2807357

[Lan00] Ivan N. Landjev, *Linear codes over finite fields and finite projective geometries*, Discrete Mathematics (2000).

[LAZ$^+$23] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries, *Starcoder: may the source be with you!*, 2023.

[MGGG17] Liz McMahon, Gary Gordon, Hannah Gordon, and Rebecca Gordon, *The joy of SET*, Princeton University Press, Princeton, NJ; National Museum of Mathematics, New York, 2017, The many mathematical dimensions of a seemingly simple card game. MR 3559496

[Moo08] Eric Moorhouse, *Applied algebra: Reed–Solomon codes*, 2008.

[Nag22] Gábor Péter Nagy, *Thin Sidon sets and the nonlinearity of vectorial Boolean functions*, 2022.

[Pav23] Francesco Pavese, *On* 4*-general sets in finite projective spaces*, 2023.

[Pel70] G. Pellegrino, *Sul massimo ordine delle calotte in* $S_{4,3}$*.*, Matematiche (Catania) (1970), no. 25, 149–157.

[Pot08] Aaron Potechin, *Maximal caps in* $AG(6,3)$, Designs, Codes and Cryptography **46** (2008), no. 3, 243–259.

[Rom92] Steven. Roman, *Coding and information theory*, Graduate texts in mathematics; 134, Springer-Verlag, New York, 1992 (eng).

[RPBN$^+$24] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi, *Mathematical discoveries from program search with large language models*, Nature **625** (2024), no. 7995, 468–475.

[RRW22] Maximus Redman, Lauren Rose, and Raphael Walker, *A Small Maximal Sidon Set in* $\mathbb{Z}_2^n$, 2022.

[SHM$^+$16] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis, *Mastering the game of Go with deep neural networks and tree search*, Nature **529** (2016), no. 7587, 484–489.

[SHS$^+$17] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis, *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*, CoRR **abs/1712.01815** (2017).

[Sin64] Richard C. Singleton, *Maximum distance q-nary codes*, IEEE Trans. Inf. Theory **10** (1964), 116–118.

[TW21] Michael Tait and Robert Won, *Improved Bounds on Sizes of Generalized Caps in* $AG(n, q)$, SIAM Journal on Discrete Mathematics **35** (2021), no. 1, 521–531.