

**COMPUTING AUTOMORPHISM GROUPS OF PROJECTIVE
PLANES**

by

Jesse Victor Adamski

A Thesis Submitted to the Faculty of
The Charles E. Schmidt College of Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Science

Florida Atlantic University

Boca Raton, FL

December 2013

COMPUTING AUTOMORPHISM GROUPS OF PROJECTIVE
PLANES

by

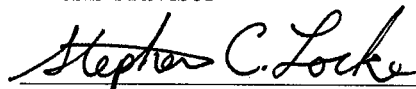
Jesse Victor Adamski

This thesis was prepared under the direction of the candidate's thesis advisor, Dr. Spyros Magliveras, Department of Mathematical Sciences, and has been approved by the members of his supervisory committee. It was submitted to the faculty of the Charles E. Schmidt College of Science and was accepted in partial fulfillment of the requirements for the degree of Master of Science.

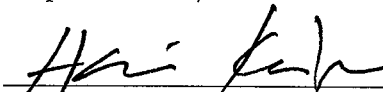
SUPERVISORY COMMITTEE:



Spyros Magliveras, Ph.D.
Thesis Advisor



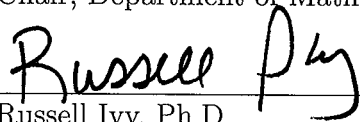
Stephen Locke, Ph.D.



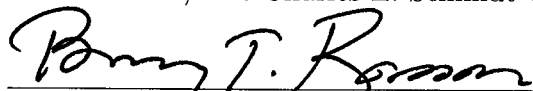
Hari Kalva, Ph.D.



Lee Klingler, Ph.D.
Chair, Department of Mathematical Sciences



Russell Ivy, Ph.D.
Interim Dean, The Charles E. Schmidt College of Science



Barry T. Rosson, Ph.D.
Dean, Graduate College



Date

ACKNOWLEDGEMENTS

I am indebted to several people, without whom I would not have been able to finish this thesis. My parents have been staunch supporters of me during my graduate work, and I would not have survived without them. I would also like to extend my thanks to Dr. Magliveras. His love of mathematics is surpassed only by his zeal to form deep bonds with mathematicians. I thank him for his encouragement, and for sharing his knowledge with me.

ABSTRACT

Author: Jesse Victor Adamski
Title: Computing Automorphism Groups of Projective Planes
Institution: Florida Atlantic University
Thesis Advisor: Dr. Spyros Magliveras
Degree: Master of Science
Year: 2013

The main objective of this thesis was to find the full automorphism groups of finite Desarguesian planes. A set of homologies were used to generate the automorphism group when the order of the plane was prime. When the order was a prime power p^a , $a \neq 1$, the Frobenius automorphism was added to the set of homologies, and then the full automorphism group was generated. The Frobenius automorphism was found by using the planar ternary ring derived from a coordinatization of the plane.

DEDICATION

This thesis is dedicated to the Everlasting Trinity, God of the Old and New Testaments.

For our sake he made him to be sin who knew no sin, so that in him we might become the righteousness of God (2 Corinthians 5:21, ESV).

This manuscript is submitted in loving memory of James Buerkle.

COMPUTING AUTOMORPHISM GROUPS OF PROJECTIVE PLANES

1	Introduction	1
1.1	History	1
1.2	Fundamental Definitions	2
1.3	Desargues' Theorem in Euclidean Geometry	4
1.4	Known Planes of Certain Orders	7
1.5	A Crucial Theorem	7
1.6	The Field Plane of Order p^a	8
1.7	Latin Squares and Projective Planes	9
1.7.1	Latin Squares	9
1.7.2	Orthogonal Latin Squares	10
2	Approach	17
2.1	The <i>ksims</i> Algorithm	17
2.2	Computing	19
2.3	Finding a Homology for the Plane of Order 4	21
2.4	<i>oldautgens</i>	22
2.5	Programs	25
3	Coordinatization of the Plane and the Frobenius Map	27
3.1	Coordinatization of the Plane	27
3.1.1	Program	33

3.2	Implementing the Frobenius Automorphism into <i>autgens</i>	34
3.2.1	Program	37
4	Conclusion	38
A	Programs in <i>AUTPPS</i>	40
	Bibliography	43

Chapter 1

Introduction

1.1 HISTORY

The subject of geometry found its first rigorous text in Euclid's *Elements*. Euclid tied together the major geometric results of his day, and proved them from a handful of axioms. In particular, he assumed the following: Given a line ℓ and a point P not on that line, there exists one and only one line going through P that is parallel to ℓ . As mathematics developed, geometry began to split up into several different areas of study. The first results relating to *projective* geometry are credited to Pappus of Alexandria during the third century B.C.E. [3, p. 3]. Centuries later, Renaissance artists, including da Vinci and Dürer, used perspective to give depth to their artwork [2, p. 3]. For example, if an artist were to paint railroad tracks going off into the distance, then she would paint it so that the rails would gradually approach each other. Eventually the rails would appear to meet each other. The astronomer Johann Kepler, who lived from 1571 to 1630, built off of this idea when he suggested that the Euclidean plane could be modified so that parallel lines would meet at *points at infinity* [6, p. 29]. Next there was Victor Poncelet (1788 to 1867) who wrote the first text on projective geometry [6, p. 30]. It was K. G. C. von Staudt (1798-1867) who began to treat these *points at infinity* like ordinary points of the plane [3, p. 4]. "But

we are not really working in projective geometry until we are prepared to forget the inferior status of such points and admit them into the community as full members having the same privileges as ordinary points,” according to Coxeter. [3, p.3]. The foundations of the projective geometry were formalized during the nineteenth century, and the subject continued to grow during the twentieth century [6, p.30].

1.2 FUNDAMENTAL DEFINITIONS

The modern conception of a *projective plane* can be stated using incidence, without explicitly using the idea of points at infinity.

Definition 1.2.1. A projective plane $\Pi = (\mathcal{P}, \mathcal{L})$ is a set of points \mathcal{P} and lines \mathcal{L} , called the elements of Π , together with an incidence relation $\mathcal{I} \subset \mathcal{P} \times \mathcal{L}$ between the points and lines such that:

- (i) Any two distinct points are incident with a unique line.
 - (ii) Any two distinct lines are incident with a unique point.
 - (iii) There exist four points in \mathcal{P} no three of which are incident with one line.
- [8, p.77].

If $(P, \ell) \in \mathcal{I}$, i.e. if point P is incident with line ℓ , we will write $P\mathcal{I}\ell$, or even $P \in \ell$, motivated by the intuitive notion that line ℓ is the set of all points incident with it.

A projective plane $\Pi = (\mathcal{P}, \mathcal{L})$ is said to be *finite* if and only if \mathcal{P} is a finite set.

Definition 1.2.2. Any set of points incident with a common line are said to be collinear [8, p.77].

If A and B are two distinct points in the plane, then the line incident with these two points will be denoted AB .

Definition 1.2.3. A set of three distinct non-collinear points A, B, C , together with the lines AB, BC, CA is called a triangle (denoted $\triangle ABC$) [8, p. 77].

We now present some elementary propositions about finite projective planes. For proof see the text by Hughes [8] on page 79.

Lemma 1.2.1. If Π is a finite projective plane, then the number of points on any line is the same as the number of points on any other line.

If we denote the above number by $n + 1$, we say that Π is of order n .

Lemma 1.2.2. If Π is a plane of order n , then the number of lines on any point is $n + 1$. Dually, the number of points on any line is $n + 1$.

Lemma 1.2.3. If $\Pi = (\mathcal{P}, \mathcal{L})$ is a finite projective plane of order n , then $|\mathcal{P}| = |\mathcal{L}| = n^2 + n + 1$.

Definition 1.2.4. Let Π be a finite projective plane of order n . Let P_1, P_2, \dots, P_m be a labeling of the points and $\ell_1, \ell_2, \dots, \ell_m$ be a labeling of the lines, where $m = n^2 + n + 1$. The incidence matrix A of Π is an $m \times m$ matrix of zeros and ones such that $a_{ij} = 1$ if and only if P_j is on ℓ_i [8, p. 85].

It is easy to see that if A is the incidence matrix of a projective plane of order n , then

$$AA^T = A^T A = J + nI$$

where J is the $m \times m$ matrix of all 1's, and I the $m \times m$ identity matrix, with $m = n^2 + n + 1$.

Definition 1.2.5. Let $\Pi = (\mathcal{P}, \mathcal{L})$ be a projective plane. A subspace $\Pi' = (\mathcal{P}', \mathcal{L}')$ is a subset of Π (so $\mathcal{P}' \subset \mathcal{P}$ and $\mathcal{L}' \subset \mathcal{L}$) such that the elements of Π' form a projective plane under the incidence relation of Π [8, p. 81].

Definition 1.2.6. An isomorphism from projective plane $\Pi_1 = (\mathcal{P}_1, \mathcal{L}_1)$ to $\Pi_2 = (\mathcal{P}_2, \mathcal{L}_2)$ is a bijective mapping $\alpha : \mathcal{P}_1 \rightarrow \mathcal{P}_2$ which maps all subspaces of Π_1 to subspaces of Π_2 , that is :

$$E \subseteq F \text{ in } \Pi_1 \text{ if and only if } E^\alpha \subseteq F^\alpha \text{ in } \Pi_2 \text{ [8, p. 21].}$$

In particular, an isomorphism $\alpha : \mathcal{P}_1 \rightarrow \mathcal{P}_2$ will map the lines of \mathcal{L}_1 to those of \mathcal{L}_2 where a line ℓ is identified with the set of points incident with ℓ .

Definition 1.2.7. An automorphism (or collineation) of a projective plane Π is an isomorphism from Π to Π [8, p. 21].

Definition 1.2.8. Let $\Pi = (\mathcal{P}, \mathcal{L})$ be a projective plane, and suppose that $V \in \mathcal{P}$ and $\ell \in \mathcal{L}$. A perspectivity (or (V, ℓ) -perspectivity) is an automorphism α that fixes ℓ pointwise and V linewise [8, p. 95].

In the above definition, point V is called the center, and line ℓ the axis of α [8, p. 95].

Definition 1.2.9. Let α be a (V, ℓ) – perspectivity. If $V \in \ell$, then α is called an elation [8, p. 95].

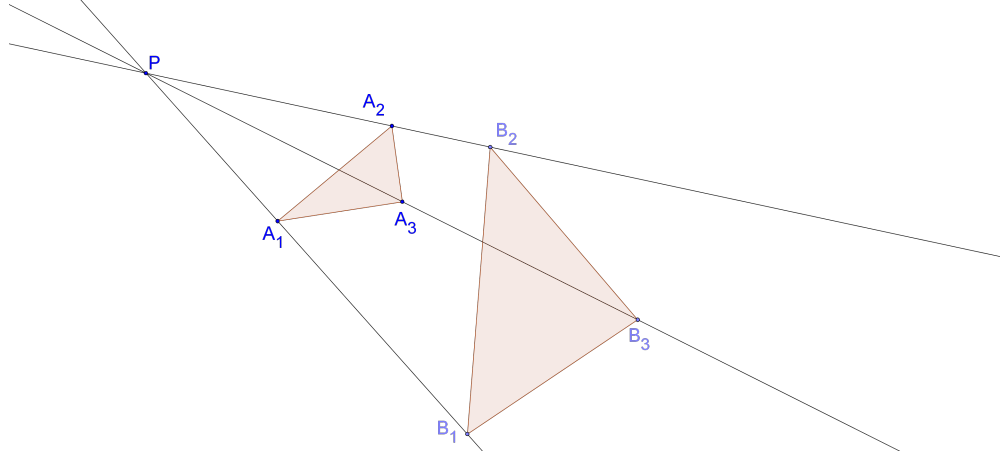
Definition 1.2.10. Let α be a (V, ℓ) – perspectivity. If $V \notin \ell$, then α is called a homology [8, p. 95].

1.3 DESARGUES' THEOREM IN EUCLIDEAN GEOMETRY

In this section, we temporarily consider the case of Euclidean geometry. However, we maintain the terms collinear and triangle analogously to the corresponding notions

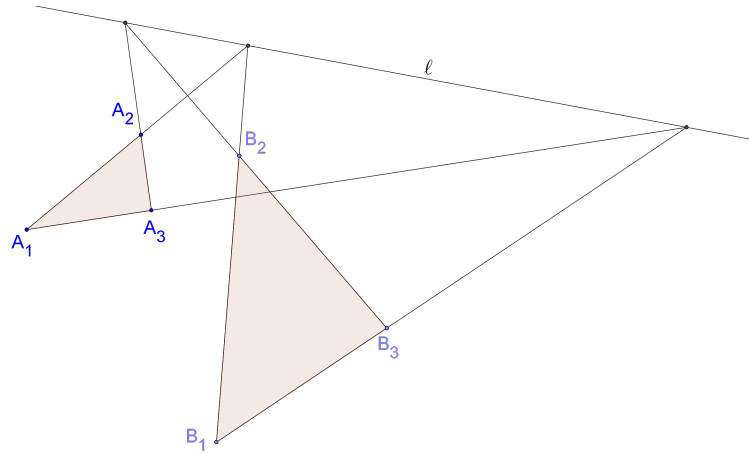
in Definitions 1.2.2 and 1.2.3. Also, given two lines ℓ_1 and ℓ_2 , we denote the point incident with these two lines as $\ell_1 \cap \ell_2$.

Definition 1.3.1. Two triangles $\triangle A_1A_2A_3$ and $\triangle B_1B_2B_3$ are said to be perspective with respect to a point P if P is incident with each line A_iB_i for $i \in \{1, 2, 3\}$ [1, p. 14].



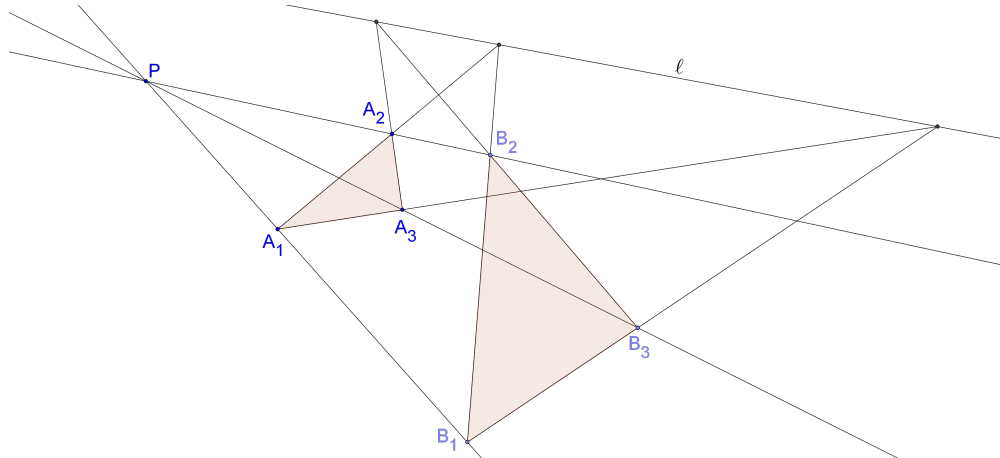
Similarly,

Definition 1.3.2. Two triangles $\triangle A_1A_2A_3$ and $\triangle B_1B_2B_3$ are said to be perspective with respect to a line ℓ if the 3 points $((A_iA_j) \cap (B_iB_j))$, for $i, j \in \{1, 2, 3\}, i \neq j$, are collinear [1, p. 14].



In Euclidean plane geometry, the concept of perspective triangles leads to the following theorem (which we present without proof):

Theorem 1.3.1. (Desargues' Theorem) Two triangles are perspective with respect to a point if and only if they are perspective with respect to a line [6, p. 46].



Now, let us consider the case of projective planes again. In general, Desargues' theorem does not always hold with respect to projective planes. [6, p. 176]. In fact, there are projective planes with order as small as 9 that do not satisfy Desargues' theorem [1, p. 88]. This directs us to the following definition:

Definition 1.3.3. A projective plane Π is said to be Desarguesian if, whenever two triangles in Π are perspective with respect to a point of Π , they are also perspective with respect to some line of Π (and conversely) [1, p. 14].

So *Desarguesian* planes are ones that satisfy Desargues' theorem. It is known that any finite plane of order $n \leq 8$ must be Desarguesian [1, p. 88]. However, there are non-Desarguesian planes with order as small as 9. In this thesis we will focus on planes that are Desarguesian.

1.4 KNOWN PLANES OF CERTAIN ORDERS

For $n \leq 15$ the following table provides us with what is known about the number of planes $\nu(n)$ of each order up to isomorphism :

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\nu(n)$	1	1	1	1	0	1	1	4	0	≥ 1	?	≥ 1	0	?

1.5 A CRUCIAL THEOREM

The following theorem attributed to Richard H. Bruck and Herbert John Ryser is a remarkable result connecting the existence of a projective plane of order n to number theory. Actually, the Bruck-Ryser theorem is often used to prove the non-existence of projective planes of a particular order n . Even though the conditions of the theorem are easy to state and check, the proof of the theorem is rather non-trivial. The theorem was proved by Bruck and Ryser in 1949 for projective planes, and a more general result, known as the Bruck-Ryser-Chowla theorem was proved for symmetric (v, b, r, k, λ) combinatorial designs in 1950.

Theorem 1.5.1. (The Bruck-Ryser) If $n \equiv 1$ or $2 \pmod{4}$ and there is a projective plane of order n , then n can be expressed as the sum of two squares (of integers) [8, p. 80].

Note that the Bruck Ryser theorem says nothing in the case where $n \equiv 0$ or $3 \pmod{4}$, for example, since $12 \equiv 0 \pmod{4}$, the Bruck-Ryser theorem cannot be of any use for $n = 12$. However, since $14 \equiv 2 \pmod{4}$ and 14 cannot be written as the sum of two squares, projective planes of order 14 cannot exist.

1.6 THE FIELD PLANE OF ORDER p^a

Here we assume the existence of the unique (up to isomorphism) Galois field of order p^a , $\mathbb{F} = \mathbb{F}_{p^a}$, where p is a prime and a is a positive integer.

Definition 1.6.1. Let $X = (\mathbb{F} \oplus \mathbb{F} \oplus \mathbb{F}) - \{(0, 0, 0)\}$. We define a relation \sim on X as follows: $(x_1, x_2, x_3) \sim (y_1, y_2, y_3) \iff \exists k \in \mathbb{F}^*$ such that $(x_1, x_2, x_3) = k(y_1, y_2, y_3)$. Let $V = X/\sim$ be the set of equivalence classes of X with respect to \sim . (Note that if $|\mathbb{F}| = q$, then $|V| = \frac{q^3-1}{q-1} = q^2 + q + 1$.) Let \mathcal{P} be a set of distinct representatives of V under \sim . So $|\mathcal{P}| = |V| = q^2 + q + 1$. Also, $\mathcal{P} = \{(x_i, y_i, z_i) \mid i \in \{1, 2, 3, \dots, q^2 + q + 1\}\}$. Let $\mathcal{L} = \{[x, y, z] \mid (x, y, z) \in \mathcal{P}\}$. The field plane $\Pi = (\mathcal{P}, \mathcal{L})$ is an incidence structure, with incidence relation \mathcal{I} defined by:

$$(x_1, x_2, x_3) \mathcal{I} [y_1, y_2, y_3] \iff \sum_{i=1}^3 x_i y_i = 0$$

for $(x_1, x_2, x_3) \in \mathcal{P}$ and $[y_1, y_2, y_3] \in \mathcal{L}$.

It is easy to show that the *field plane* is in fact a finite projective plane of order p^a . Since there is always a Galois field of order p^a we can construct a field plane of order p^a for any prime p and positive integer a . We can immediately conclude that there are an infinite number of projective planes.

There are still substantial open questions with regards to the orders of projective planes. As we saw above, for each prime power $n = p^a$ there is a plane of order n . However, does there exist a plane of order not a prime power? The only known result that gives insight into the possible orders of projective planes is the ***Bruck-Ryser Theorem 1.5.1***. It is possible to find a number n that satisfies this theorem which is not of the form p^a . For example, $n = 10$ is not a power of a prime, but it satisfies the theorem since $10 \equiv 2 \pmod{4}$ and $10 = 1^2 + 3^2$ [1, p.93]. However, it has been shown using an exhaustive computer search that there are no projective planes of

order 10 [10]. As of today, no one has found a projective plane of order other than p^a [12].

There is a special relationship between Desarguesian and field planes.

Theorem 1.6.1. Let Π be a finite projective plane. Then Π is Desarguesian if and only if it is a field plane [1, p. 77].

1.7 LATIN SQUARES AND PROJECTIVE PLANES

1.7.1 Latin Squares

Definition 1.7.1. Let R be a set of n distinct elements, which we take to be $\{1, 2, \dots, n\}$. A latin square of order n is an $n \times n$ matrix with entries from R such that each row and each column contains every element of R exactly once [8, p. 121].

For example, for $n = 1$ the following 1×1 matrix is a latin square: $\boxed{1}$. If we look at the case when $n = 3$, one example of a latin square is the following:

Example 1.7.1.

3	1	2
2	3	1
1	2	3

Note that without loss of generality, any latin square can be renamed so that the top row is in ascending order (so 1 2 3 ... n). Latin squares can be found by brute force, but we can use algebraic concepts to find them as well. Cayley showed that the multiplication tables, or Cayley tables, of groups give a special form of latin squares [4, p. 15]. To generalize this result, we need the following definition:

Definition 1.7.2. Let S be a set, and (\cdot) be a binary operation. (S, \cdot) is a quasigroup if $\forall a, b \in S, \exists x, y \in S$ such that each of the following equations have exactly one solution: $ax = b$ and $ya = b$ [4, p. 16].

This leads to the following useful theorem:

Theorem 1.7.1. The multiplication table of a quasigroup is a latin square [4, p. 16].

Let us take a look at a particular example of a quasigroup.

Example 1.7.2. Let S be the set of integers modulo 3, and define the binary operation (\cdot) on S , by $a \cdot b = 2a + b + 1$. We get the following multiplication table:

(\cdot)	0	1	2
0	1	2	0
1	0	1	2
2	2	0	1

 \longrightarrow

1	2	0
0	1	2
2	0	1

Where the multiplication table is indeed a latin square with its set being $R = \{0, 1, 2\}$ [4, p. 17].

In general, if we want to find one of the quasigroups (S, \cdot) of order n , then we can consider the set S of the integers modulo n with operation defined by $a \cdot b = ha + kb + l \pmod{n}$ with h, k , and l fixed integers where h and k are relatively prime to n [4, p. 17]

1.7.2 Orthogonal Latin Squares

Definition 1.7.3. Let $A = A(i, j)$ and $B = B(i, j)$ be two latin squares on the symbols $X = \{1, 2, \dots, n\}$. We say that A and B are orthogonal if and only if when A and B are superimposed (one is placed on top of the other) all possible pairs of $X \times X$ will appear, which means $\{(A(i, j), B(i, j)) \mid 1 \leq i \leq n, 1 \leq j \leq n\} = X \times X$.

If A and B are orthogonal latin squares, we denote this condition by $A \perp B$.

Example 1.7.3. For $X = \{1, 2, 3\}$ the following two latin squares are orthogonal:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{pmatrix}$$

It is possible to have a set of several latin squares, say k of them, of order n which are *mutually orthogonal* where for each pair of latin squares A, B from the set, $A \perp B$.

Example 1.7.4. For $X = \{1, 2, 3, 4, 5\}$, and $k = 3$ consider:

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \\ 3 & 4 & 5 & 1 & 2 \\ 4 & 5 & 1 & 2 & 3 \\ 5 & 1 & 2 & 3 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 1 & 2 \\ 5 & 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 & 1 \\ 4 & 5 & 1 & 2 & 3 \end{pmatrix} \quad C = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 1 & 2 & 3 \\ 2 & 3 & 4 & 5 & 1 \\ 5 & 1 & 2 & 3 & 4 \\ 3 & 4 & 5 & 1 & 2 \end{pmatrix}$$

Suppose that A is a latin square on the symbols $X = \{1, 2, \dots, n\}$, and suppose that π is a permutation in \mathcal{S}_n . We denote by $\pi(A)$ the result of renaming the symbols in A by their images under π . It is clear that $\pi(A)$ will also be a latin square. Moreover, if A and B are two orthogonal latin squares, then from $A \perp B$ follows that $A \perp \pi(B)$ and $\pi(A) \perp B$. An easy consequence of this fact is that if A_1, A_2, \dots, A_k is a set of mutually orthogonal latin squares (MOLS), we can select permutations $\pi_1, \pi_2, \dots, \pi_k$ so that $\pi_1(A_1), \pi_2(A_2), \dots, \pi_k(A_k)$ are mutually orthogonal, and all have their first rows precisely the sequence $(1, 2, 3, \dots, n)$. So, without loss of generality, we write MOLS in this form. The question now becomes, what is the maximum number of $n \times n$ MOLS that can be found?

Definition 1.7.4. Let G be a set of $n \times n$ mutually orthogonal latin squares. G is a complete set of mutually orthogonal latin squares if the size of G is $n - 1$ [4, p. 161].

A complete set gives an upper bound on the number of MOLS, which is seen in the next proposition.

Proposition 1.7.1. Any set of mutually orthogonal $n \times n$ latin squares has at most $n - 1$ matrices [8, p. 122].

So we see that *Example 1.7.3* is a complete set of 3×3 MOLS. However, *Example 1.7.4* is not a complete set, since a complete set would need one more matrix. A set of MOLS is a maximal set of mutually orthogonal latin squares if it is not a proper subset of a larger set of MOLS of that order. It is worth mentioning that there are integers k, n , with $k < n - 1$ for which there is a maximal set of k MOLS of order n .

We will see that projective planes can be derived from certain sets of MOLS. In the mean time, we look at the *tableau* corresponding to a set of mutually orthogonal latin squares. For a set of k MOLS where each matrix is $n \times n$, we will get a tableau (or matrix) of dimension $(n^2) \times (2+k)$. If $X = \{1, 2, \dots, n\}$ is the set of elements that form the MOLS, then in the first two columns we input all possible ordered pairs of $X \times X$ in ascending order. Next, for the remaining columns, if (i, j) was the ordered pair (coordinate) in the first two columns of row t , then for each of the k MOLS A^1, A^2, \dots, A^k we would input $A^m_{i,j}$ in row t , column $m + 2$.

1	1	$A^1_{1,1}$	$A^2_{1,1}$...	$A^k_{1,1}$
1	2	$A^1_{1,2}$	$A^2_{1,2}$...	$A^k_{1,2}$
\vdots	\vdots	\vdots	\vdots		\vdots
1	n	$A^1_{1,n}$	$A^2_{1,n}$...	$A^k_{1,n}$
2	1	$A^1_{2,1}$	$A^2_{2,1}$...	$A^k_{2,1}$
2	2	$A^1_{2,2}$	$A^2_{2,2}$...	$A^k_{2,2}$
\vdots	\vdots	\vdots	\vdots		\vdots
2	n	$A^1_{2,n}$	$A^2_{2,n}$...	$A^k_{2,n}$
\vdots	\vdots	\vdots	\vdots		\vdots
n	1	$A^1_{n,1}$	$A^2_{n,1}$...	$A^k_{n,1}$
\vdots	\vdots	\vdots	\vdots		\vdots
n	n	$A^1_{n,n}$	$A^2_{n,n}$...	$A^k_{n,n}$

Example 1.7.5. We find the tableau of the MOLS from Example 1.7.4 above.

1	1	1	1	1
1	2	2	2	2
1	3	3	3	3
1	4	4	4	4
1	5	5	5	5
2	1	2	3	4
2	2	3	4	5
2	3	4	5	1
2	4	5	1	2
2	5	1	2	3
3	1	3	5	2
3	2	4	1	3
3	3	5	2	4
3	4	1	3	5
3	5	2	4	1
4	1	4	2	5
4	2	5	3	1
4	3	1	4	2
4	4	2	5	3
4	5	3	1	4
5	1	5	4	3
5	2	1	5	4
5	3	2	1	5
5	4	3	2	1
5	5	4	3	2

As we will see, tableaux are useful in forming projective planes, but we can notice something else with regards to mutually orthogonal latin squares.

Proposition 1.7.2. If any two columns of the tableau T corresponding to a set of k MOLS are used as coordinates, then the remaining columns of T yield k MOLS [7, p. 190].

In *Example 1.7.5* we had 3 MOLS, where each matrix is 5×5 . We can now look at what happens if we have a complete set of mutually orthogonal latin squares.

Theorem 1.7.2. There exists a finite projective plane of order n if and only if there exists a complete set of $n - 1$ mutually orthogonal $n \times n$ latin squares. [8, p. 122].

Specifically, if we form a tableau from a complete set of mutually orthogonal latin squares, then we will be able to use the tableau to find a finite projective plane. Complete sets of MOLS are key in studying projective plane. If for a certain number n there is not a complete set of $n \times n$ MOLS, then there will not be a projective plane of order n . As we mentioned earlier, so far only projective planes of order p^a have been found, where p is a prime. In 1779, Leonhard Euler pondered whether there was a set of 2 mutually orthogonal latin squares of order $n = 6$, and he made the claim that there would not be such a pair of MOLS. [4, p.11]. This question went unanswered until G.R. Tarry proved Euler correct in 1900 [4, p.160]. In fact, Euler conjectured that if $n \equiv 2 \pmod{4}$, that there would not exist a pair of $n \times n$ orthogonal latin squares. However, this time Euler was disproved when R. C. Bose and S. S. Shrikhande found a pair of orthogonal latin squares of order 22, which is congruent to 2 (mod 4) [4, p.397]. Actually, it has been shown that Euler's conjecture is wrong for all $n > 6$ [4, p.416].

We will now use a set of mutually orthogonal latin squares to construct a finite projective plane. First, we will find the tableau from the orthogonal latin squares in

Example 1.7.3.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{pmatrix}$$

Example 1.7.6. The tableau we get is

1	1	1	1
1	2	2	2
1	3	3	3
2	1	2	3
2	2	3	1
2	3	1	2
3	1	3	2
3	2	1	3
3	3	2	1

Since we started with a complete set of 2 MOLS of order 3, we will get a plane of order 3. Our plane will therefore have $3^2 + 3 + 1 = 13$ points and 13 lines. To get the plane, we begin by choosing four numbers less than or equal to 13, which will be the points on a line l . In this example, we use $l = 1\ 2\ 5\ 7$. We adjoin l to the top of our tableau. Along the left side we fill in the rest of the numbers that we did not choose from $\{1, 2, \dots, n\}$. This modified tableau appears below to the left. Now that we have the modified tableau, we can get the plane. In the first row, we write the line l . In the modified tableau we look underneath the first point of l for any rows that contain the number 1. We see that the rows corresponding to the numbers 3, 4, and 6 contain ones, so the second line of the plane will be the first point of l with 3, 4, and 6, namely $1\ 3\ 4\ 6$. Similarly, we search for all of the rows that contain the number 2 in the first column, and we get the line $1\ 8\ 9\ 10$. We do the same for 3, and get $1\ 11\ 12\ 13$. Next, we look at the second column (the column underneath the second point of l) and repeat the process. We see that the rows corresponding to 3, 8, and 11 have ones, and so the next line will be the second point of l with 3, 8, and 11. We repeat this process for each column, and for each of the numbers 1, 2, and 3.

$$\begin{array}{c|cccc}
& 1 & 2 & 5 & 7 \\
\hline
3 & 1 & 1 & 1 & 1 \\
4 & 1 & 2 & 2 & 2 \\
6 & 1 & 3 & 3 & 3 \\
8 & 2 & 1 & 2 & 3 \\
9 & 2 & 2 & 3 & 1 \\
10 & 2 & 3 & 1 & 2 \\
11 & 3 & 1 & 3 & 2 \\
12 & 3 & 2 & 1 & 3 \\
13 & 3 & 3 & 2 & 1
\end{array}
\longrightarrow
P = \begin{array}{cccc}
1 & 2 & 5 & 7 \\
1 & 3 & 4 & 6 \\
1 & 8 & 9 & 10 \\
1 & 11 & 12 & 13 \\
2 & 3 & 8 & 11 \\
2 & 4 & 9 & 12 \\
2 & 6 & 10 & 13 \\
5 & 3 & 10 & 12 \\
5 & 4 & 8 & 13 \\
5 & 6 & 9 & 11 \\
7 & 3 & 9 & 13 \\
7 & 4 & 10 & 11 \\
7 & 6 & 8 & 12
\end{array}$$

We started with a complete set of order 3 orthogonal latin squares and got the

plane P . The indices in the matrix correspond to a numbering of the points in the plane. Each row of P corresponds to a line of the plane. For example, the fifth row down would correspond to the fifth line in the plane $l_5 = 2\ 3\ 8\ 11$ where 2, 3, 8, and 11 are the points on the line.

Chapter 2

Approach

2.1 THE *KSIMS* ALGORITHM

In 1968, Professor Charles Sims published a remarkable algorithm which facilitates computation with finite permutation groups. Various improved versions were developed since then [13, 9, 11]. These algorithms are of polynomial time complexity, and have the following properties in common: i) the input is a set of generators for a permutation group G , and the output consists of a structure of permutations called a *set of strong generators* for the group [13], ii) given any element $x \in \mathcal{S}_n$ there is an efficient algorithm to decide whether $x \in G$, iii) there is an efficiently computable bijective mapping from $\mathbb{Z}_{|G|}$ to G . Using this map, one can run through the elements of the group without duplication. The current version of the algorithm *ksims* used in this thesis was developed by Leo Chouinard, Jr. and Spyros Magliveras in the late 1970's at the University of Nebraska, Lincoln, and is based on a non-deterministic procedure. The algorithm is called *ksims* in honor of Donald Knuth and Charlie Sims. We use *ksims* often to recursively generate automorphism groups of projective planes. The input for *ksims* is a matrix (a_{ij}) , where each row i represents a single permutation with j mapping to a_{ij} . At its termination, the program returns the order of the group, and stores three objects in a predefined variable 'G': 1) the prescribed name of the group, 2) the set of generators of the group, and 3) the *logarithmic*

signature of the group.

Definition 2.1.1. Let G be a finite group and $A_1, A_2, \dots, A_s \subset G$. The ordered sequence $\alpha = [A_1, A_2, \dots, A_s]$ is a logarithmic signature of G if and only if every element of $g \in G$ can be represented uniquely in the form:

$$g = x_1 \cdot x_2 \cdots x_s \text{ where } x_i \in A_i.$$

In the following example, **ksims** is used to generate the alternating group A_5 .

Example 2.1.1. It is easy to show that $A_5 = \langle (1\ 2\ 3\ 4\ 5), (1)(2)(3\ 4\ 5) \rangle$ where the above generating permutations are in *cycle* form. The input for *ksims* is the matrix $\text{gens} = \begin{bmatrix} [2\ 3\ 4\ 5\ 1] \\ [1\ 2\ 4\ 5\ 3] \end{bmatrix}$, where each row corresponds to the bottom row of a permutation written in the standard Cartesian form. So, the first row of the input matrix corresponds to permutation $(1\ 2\ 3\ 4\ 5) = (\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \end{smallmatrix}) = [2\ 3\ 4\ 5\ 1]$ and the second row to $(1)(2)(3\ 4\ 5) = (\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 4 & 5 & 3 \end{smallmatrix}) = [1\ 2\ 4\ 5\ 3]$. The program *ksims* is run by typing $G \leftarrow \text{'A5' ksims gens}$, where G is the computed structure of the group. The program returns $|G| = 60$, while it stores the group name $(A5)$, *gens*, and the *logarithmic signature* of G . The computed output structure is:

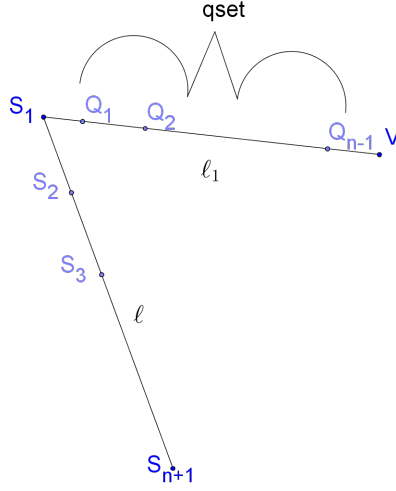
A5						3	8	12	15	15
						5	4	3	1	1
						1	2	3	4	5
						2	3	4	5	1
						3	4	5	1	2
						4	5	1	2	3
						5	1	2	3	4
	2	3	4	5	1	1	2	3	4	5
	1	2	4	5	3	1	3	4	2	5
						1	4	5	2	3
						1	5	3	2	4
						1	2	3	4	5
						1	2	4	5	3
						1	2	5	3	4
						1	2	3	4	5

The *logarithmic signature* of G is the 12×5 matrix found by taking the third object and removing the first, second, and last rows (the top two rows are present to facilitate computations using the logarithmic signature).

2.2 COMPUTING

In this project, we are interested in finding the full automorphism group of finite projective planes, and we use a specific type of perspectivity to generate these groups. We use perspectivities that fix a line l pointwise and fix a point V linewise where V is not on l , which as we saw in *Definition 1.2.10* is called a *homology*. We want to be able to find a homology α given any finite projective plane $\Pi = (\mathcal{P}, \mathcal{L})$. The program *fixpl* does this, and the following is a description of this program.

The incidence matrix A of Π needs to be obtained before *fixpl* is run. The program is run by inputting a point V and a line l . The program checks to see if V is on l , and if it is then an error code is returned. From here on, we assume that V is not on l . If n is the order of Π , then *fixpl* stores $n^2 + n + 1$ as r (which is the number of points and the number of lines in Π). Next, the program forms two $1 \times r$ vectors *pperm* and *lperm* which will represent the point and line permutations for α , respectively. These two vectors are initially filled with 0's, and as computation proceeds new entries are determined. *Fixpl* finds the points $\{S_1, S_2, \dots, S_{n+1}\}$ on line l using the program *ptset* and stores the points under the name of *set*. Next, program *lnon* determines the line l_1 between V and the first point S_1 in *set*. As previously mentioned, our perspectivity α will be fixing point V and the points S_i of l . However, there are still points on l_1 that are free to “move around”. We call these points *qset*, and they can be found by getting the set of points on l_1 and removing V and S .



We are now able to start filling elements into *pperm* and *lperm*. Since V and the points of *set* are fixed by α , the program fixes their corresponding points in *pperm*. For example, if our point V was point number 4, then the fourth element of *pperm* would be set to 4. Even so, the elements from *qset* are free to move around. The program chooses to map the first element Q_1 of *qset* to the second element Q_2 of *qset*, so this means that $pperm[Q_1]$ is set to Q_2 . Peculiarly enough, the information that we have specified in *pperm* is enough to determine the homology α completely. Since we have fixed certain points and mapped one element from *qset* to another, there will already be lines that get fixed, namely $\{l, VS_1, VS_2, \dots, VS_{n+1}\}$, while other lines are permuted, for example S_1Q_1 moves to S_2Q_2 . There will be other lines that can be determined from *pperm*, but we will not necessarily be able to determine all of *lperm* yet. We have a program *pimplies* that finds all line permutations that can be determined from current *pperm* information. Similarly, *limplies* is a program that inputs *lperm* and *pperm*, and finds all of the point permutations that are determined by *lperm*. The program *fixpl* uses *pimplies* to find more of *lperm*, and then uses *limplies* to fill in more of *pperm*. *Fixpl* now goes back and forth between *pimplies*

and *implies*, creating an avalanche of point and line implications that will not stop until both *pperm* and *lperm* are completely determined. Once there are no more entries in *pperm* to be filled, fixpl outputs *pperm*.

2.3 FINDING A HOMOLOGY FOR THE PLANE OF ORDER 4

In this section, we present an example where the fixpl process will be used to find a homology for the finite projective plane of order 4. The plane will be described using its incidence matrix A below. You should recall that each row represents a line of the plane, where 1's indicate which points are on the line. For example, the first row is the first line, and it contains the points 1 2 3 4 5 since there are 1's in the first, second, third, fourth, and fifth columns.

$$A = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ 16 \\ 17 \\ 18 \\ 19 \\ 20 \\ 21 \end{matrix} & \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ 1 & . & . & . & . & 1 & 1 & 1 & 1 & . & . & . & . & . & . & . & . & . & . & . & . \\ 1 & . & . & . & . & . & . & . & . & 1 & 1 & 1 & 1 & . & . & . & . & . & . & . & . \\ 1 & . & . & . & . & . & . & . & . & . & . & . & 1 & 1 & 1 & 1 & . & . & . & . & . \\ 1 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & 1 & 1 & 1 & . \\ . & 1 & . & . & . & 1 & . & . & . & 1 & . & . & . & 1 & . & . & . & 1 & . & . & . \\ . & 1 & . & . & . & 1 & . & . & . & 1 & . & . & . & 1 & . & . & . & 1 & . & . & . \\ . & 1 & . & . & . & . & 1 & . & . & . & 1 & . & . & . & 1 & . & . & . & 1 & . & . \\ . & 1 & . & . & . & . & . & 1 & . & . & . & 1 & . & . & . & 1 & . & . & . & 1 & . \\ . & . & 1 & . & . & 1 & . & . & . & . & 1 & . & . & . & . & 1 & . & . & 1 & . & . \\ . & . & 1 & . & . & . & 1 & . & . & . & . & 1 & . & . & . & . & 1 & . & . & . & 1 \\ . & . & . & 1 & . & 1 & . & . & . & 1 & . & . & . & . & . & 1 & . & . & 1 & . & . \\ . & . & . & 1 & . & . & 1 & . & . & 1 & . & . & . & . & . & 1 & . & . & . & 1 & . \\ . & . & . & 1 & . & . & . & 1 & . & . & . & 1 & . & 1 & . & . & 1 & . & . & . & . \\ . & . & . & 1 & . & . & . & . & 1 & . & . & 1 & . & 1 & . & . & . & 1 & . & . & . \\ . & . & . & . & 1 & . & . & 1 & . & 1 & . & . & . & . & . & 1 & . & 1 & . & . & . \\ . & . & . & . & 1 & 1 & . & . & . & . & 1 & . & . & 1 & . & . & . & . & . & . & 1 \\ . & . & . & . & 1 & . & . & . & 1 & . & 1 & . & . & . & . & 1 & . & 1 & . & . & . \end{bmatrix} \end{matrix}$$

If you look at A , you can notice that the sixth line of the plane does not contain the point 5. We will choose to use this line and this point to find a homology, so we set $l = 6$ and $V = 5$. Since the order of the plane is 4, $r = 21$, which is the number of

points and the number of lines in the plane. Next, we form two 1×21 permutation vectors, which start as $pperm = lperm = (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$. Line 6 contains the points (2 6 10 14 18), so *set* is defined to be these points. The first point of *l* is 2, so using the program *lnon* the line l_1 , the line through the points 2 and 5, is determined to be the line 1. The points on l_1 are (1 2 3 4 5), so the points 2 and 5 are removed to get $qset = (1\ 3\ 4)$. The points of *l* and the point *V* are fixed, so $pperm = (0\ 2\ 0\ 0\ 5\ 6\ 0\ 0\ 0\ 10\ 0\ 0\ 0\ 14\ 0\ 0\ 0\ 18\ 0\ 0\ 0)$. The first point from *qset* will be mapped to the second point from *qset*, so point 1 gets sent to point 3. This means that the first element of *pperm* is set to be 3, so *pperm* becomes (3 2 0 0 5 6 0 0 0 10 0 0 0 14 0 0 0 18 0 0 0). Now the programs *pimplies* and *limplies* will be used repeatedly until all of *pperm* is determined.

In the first iteration, *pimplies* sets *lperm* to be equal to (1 10 13 12 11 6 0 0 0 0 0 0 0 0 0 0 18 19 20 21). Also, it uses *limplies* to change *pperm* to (3 2 0 0 5 6 13 19 16 10 9 15 20 14 21 11 8 18 17 7 12).

In the second iteration, *pimplies* causes *lperm* to become (1 10 13 12 11 6 9 7 8 14 16 17 15 2 3 5 4 18 19 20 21). Finally, *limplies* modifies *pperm* to be (3 2 4 1 5 6 13 19 16 10 9 15 20 14 21 11 8 18 17 7 12).

After only two iterations, *fixpl* is able to determine a homology from the point $V = 5$ and the line $l = 6$. The program would return *pperm*. If this permutation was written in cycle notation, it would be

$$(1\ 3\ 4)(2)(5)(6)(7\ 13\ 20)(8\ 19\ 17)(9\ 16\ 11)(10)(12\ 15\ 21)(14)(18).$$

2.4 OLDAUTGENS

The program *fixpl* is able to find homologies for a Desarguesian plane. Another program, *oldautgens*, calls *fixpl* repeatedly in an attempt to generate the full auto-

morphism group of the plane. The input for *oldautgens* is an incidence matrix A of a finite projective plane. *Oldautgens* begins by finding a random point V and a random line ℓ in the plane, such that V is not on ℓ . Next, *fixpl* is used to find a homology from V and ℓ . This homology is added into a set of generators *gens*, which in turn is used to create a group G using the programs *ksims*. The process begins again with another homology α found from a new random point and line. If α is a new homology that is not in G , then it is added to *gens*, and the group G is regenerated. This cycle continues on. What if the homology α is already in G ? Perhaps, the process needs to be repeated a few times before a new homology $\alpha \notin G$ is found. We stipulate that if for 20 consecutive iterations each random homology α we find satisfies $\alpha \in G$, then we can safely assume that the group G is probably as large as it will get. The program then stops, and stores the group under the variable G .

We will check to see if *oldautgens* is accurately generating the full automorphism group of planes. We use *oldautgens* on finite projective planes of various orders, and compare the results to the actual known order of the full automorphism group of the planes.

Results from running <i>oldautgens</i>		
Order of Plane	Order of Automorphism Group Generated by <i>oldautgens</i>	Actual Order of Automorphism Group
3	5616	5616
4	60480	120960
5	372000	372000
8	16482816	49448448
9 (field plane)	42456960	84913920
11	212427600	212427600
16 (field plane)	4277145600	17108582400

Notice that *oldautgens* accurately finds the full automorphism group for planes of prime order. However, the program does not succeed for planes of order p^a , p a prime, $a > 1$. For example, the group generated for the plane of order 4 is half of what it should be, and similarly for the field plane of order 9. Further for the plane of order 8, the group is one-third of what it should be, and the field plane of order 16 yields a group one-fourth of the correct size. It would appear that if the order of the plane is p^a , then *autgens* is generating an automorphism group that is $1/a$ of the size of the actual group. It seems that in order to get the full automorphism group the collineation induced by the *Frobenius automorphism* of the field needs to be added to the set of generators.

Definition 2.4.1. Let \mathbb{F} be a field of characteristic p . The *Frobenius automorphism* is the function $\phi : \mathbb{F} \rightarrow \mathbb{F}$ defined such that $\forall a \in \mathbb{F}, \phi(a) = a^p$ [5, p. 549].

For $a, b, c, x, y, z \in \mathbb{F}$ if $ax + by + cz = 0$, then under the Frobenius automorphism $a^p x^p + b^p y^p + c^p z^p = 0$. This means that the Frobenius automorphism preserves incidences between points and lines. Therefore, the Frobenius automorphism induces a collineation of the plane. In the next chapter, a method for finding the Frobenius automorphism is discussed and utilized.

2.5 PROGRAMS

<i>fixpl</i>	
[0]	$pperm \leftarrow \mathbf{fixpl} \ u; \ pt; \ ln; \ r; \ lperm; \ qset; \ set; \ t; \ k1; \ k2$
[1]	$pt \leftarrow u[1] \ \diamond \ ln \leftarrow u[2]$
[2]	$v \leftarrow 1 \uparrow \rho A \ \diamond \ pperm \leftarrow lperm \leftarrow v\rho 0$
[3]	$set \leftarrow \mathbf{ptset} \ ln$
[4]	$\rightarrow er \times \iota \ 1 = pt \in \mathbf{ptset} \ ln$
[5]	$qset \leftarrow ptset \ \mathbf{lnon} \ pt, set[1]$
[6]	$t \leftarrow \sim qset \in pt, set \ \diamond \ qset \leftarrow t/qset$
[7]	$pperm[pt, set] \leftarrow pt, set$
[8]	$pperm[qset[1]] \leftarrow qset[2]$
[9]	$a1 : k1 \leftarrow +/pperm \neq 0$
[10]	$lperm \leftarrow lperm \ \mathbf{pimplies} \ pperm$
[11]	$pperm \leftarrow pperm \ \mathbf{limplies} \ lperm$
[12]	$k2 \leftarrow +/pperm \neq 0$
[13]	$\rightarrow a1 \times \iota k2 \neq k1$
[14]	$\rightarrow 0$
[15]	$er : 'pt \text{ on line error.'}$

<i>oldautgens</i>	
[0]	$G \leftarrow \mathbf{autgens} \ A; v; count; pt; ln; x; niterat; w; gens$
[1]	$v \leftarrow (\rho A)[2] \ \diamond \ w \leftarrow (1, v)\rho \ 0$
[2]	$gens \leftarrow (0, v)\rho \ \iota \ v \ \diamond \ count \leftarrow 0 \ \diamond \ niterat \leftarrow 20$
[3]	$G \leftarrow 'G' \ \mathbf{ksims} \ gens$
[4]	$a1 : pt \leftarrow ?v \ \diamond \ ln \leftarrow ?v$
[5]	$\rightarrow a1 \times \iota \ pt \in \mathbf{ptset} \ ln$
[6]	$x \leftarrow \mathbf{fixpl} \ pt, ln$
[7]	$\rightarrow a2 \times \iota \ ' - ' \in x \ \mathbf{In} \ G$
[8]	$count \leftarrow count + 1$
[9]	$\rightarrow end \times \iota \ count > niterat$
[10]	$\rightarrow a1$
[11]	$a2 : gens \leftarrow gens, [1]x$
[12]	$' _ _ _ _ '$
[13]	$G \leftarrow 'G' \ \mathbf{ksims} \ gens$
[14]	$\rightarrow a1$
[15]	$end : \mathbf{order} \ G$

Chapter 3

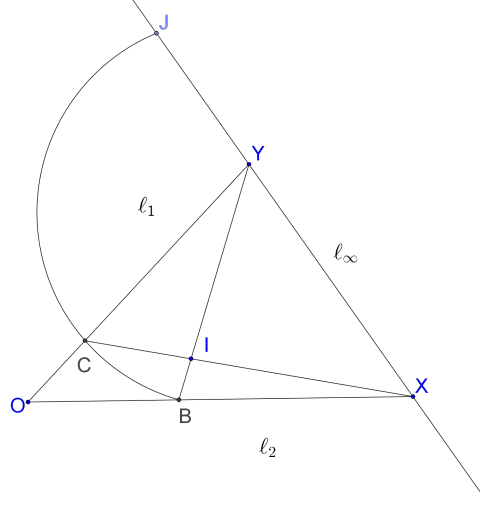
Coordination of the Plane and the Frobenius Map

3.1 COORDINATIZATION OF THE PLANE

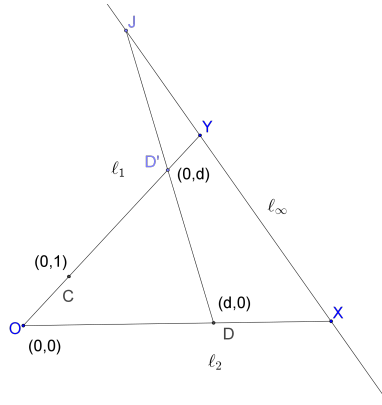
When discussing planes in elementary mathematics, the Cartesian coordinate system is often used. This coordinate system allows for each point in the plane to be uniquely named using an ordered pair of real numbers. Therefore, the set of all of the points in the plane can be written $\{(x, y) \mid x \in \mathbb{R} \text{ and } y \in \mathbb{R}\}$. This system depends on two axes named the x – *axis* and y – *axis*. The x – *axis* is a line consisting of the points $\{(x, 0) \mid x \in \mathbb{R}\}$, and similarly the y – *axis* is $\{(0, y) \mid y \in \mathbb{R}\}$. Hughes and Piper give a method for coordinatizing finite projective planes [8, p. 111-112]. In this section, this method is explained.

For the Cartesian coordinate system, the set of real numbers are used for the ordered pairs labeling points. The set of numbers that will be used for the finite projective plane system is $R = \{0, 1, 2, \dots, n - 1\}$ where n is the order of the plane. The first step in defining the coordinates is to pick picking three distinct lines that are not concurrent, and labeling them ℓ_1 , ℓ_2 , and ℓ_∞ . For the coordinatization, the line ℓ_1 will play a role similar to that of the y – *axis*, and ℓ_2 to that of the x – *axis*. The line ℓ_∞ is the line at infinity. Second, the points incident with each pair of these

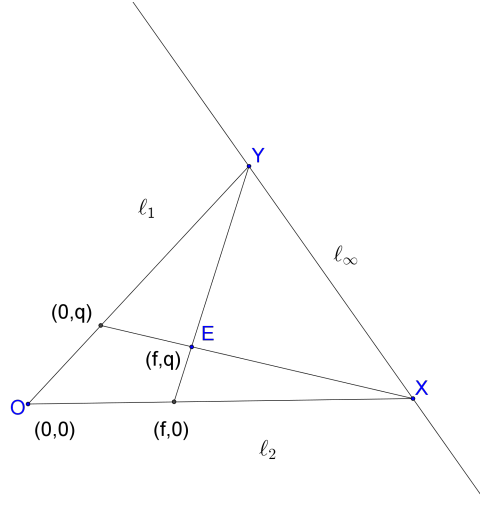
lines is labeled: so $X = \ell_2 \cap \ell_\infty$, $Y = \ell_\infty \cap \ell_1$, and $O = \ell_1 \cap \ell_2$. Third, an arbitrary point, not incident with ℓ_1 , ℓ_2 , or ℓ_∞ , is chosen and labeled I . Fourth, the following points are defined: $C = (XI) \cap \ell_1$, $B = (YI) \cap \ell_2$, and $J = (CB) \cap \ell_\infty$.



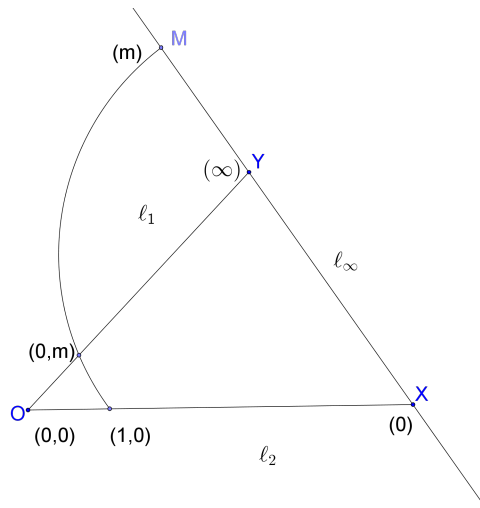
Now that these points and lines are defined, the process of assigning coordinates to the points can begin. The point O will act like the origin, and will be labeled $(0, 0)$. The point C is labeled $(0, 1)$. Recall that both O and C are on ℓ_1 . The remaining points on ℓ_1 are arbitrarily labeled $(0, 2), (0, 3), \dots, (0, n - 1)$ with the exception of the point Y , which will be given a special label later. The points on ℓ_2 will now be labeled. For all points D on the line ℓ_2 , such that $D \neq X$, if $D' = (JD) \cap \ell_1$ where $D' = (0, d)$, then label D as $(d, 0)$.



The axes ℓ_1 and ℓ_2 will now be used to label the remaining points in the plane, with the exception of the points on ℓ_∞ which will be christened last. For all points E that are not on ℓ_∞ , if $(XE) \cap \ell_1 = (0, q)$ and $(YE) \cap \ell_2 = (f, 0)$, then E is set to be (f, q) .



Finally, the points on ℓ_∞ will be labeled. If M is a point on ℓ_∞ and if the line incident with the points M and $(1, 0)$ meet the line ℓ_1 at $(0, m)$, then $M = (m)$. The point Y is set to be (∞) . All of the points of the finite projective plane have now been coordinatized.



Hughes and Piper also describe a method for coordinatizing the lines of the plane using the coordinatization of points [8, p. 112]. For any line ℓ that does not contain the point Y if $\ell \cap \ell_\infty = (m)$ and $\ell \cap \ell_1 = (0, k)$, then $\ell = [m, k]$. If Y is on ℓ and $\ell \cap \ell_2 = (b, 0)$, then $\ell = [b]$. The line ℓ_∞ is set equal to $[\infty]$.

Definition 3.1.1. A ternary ring is a nonempty set S together with a *ternary operation* T such that $\forall a, b, c \in S$ there exists a unique element $k \in S$ where $T(a, b, c) = k$ [8, p. 113].

In particular, there is a ternary ring that can be defined using the underlying incidence structure of a projective plane, along with a coordinatization of its points and lines.

Definition 3.1.2. Let Π be a projective plane coordinatized (using the method described above) by a set R . A planar ternary ring (R, T) with T defined such that $\forall a, b, c \in R$, $T(a, b, c) = k$ if and only if the point (b, c) is on $[a, k]$ [8, p. 113].

It is not difficult to show that when defined this way T is a ternary operation.

Theorem 3.1.1. Let Π be a projective plane coordinatized by a set R . If T is defined as above, then the following properties hold:

- (A) $T(a, 0, c) = T(0, b, c) = c$ for all $a, b, c \in R$
- (B) $T(a, 1, 0) = T(1, a, 0) = a$ for all $a \in R$
- (C) If $a, b, c, d \in R, a \neq c$, then there is a unique $x \in R$ such that $T(x, a, b) = T(x, c, d)$.
- (D) If $a, b, c \in R$ then there is a unique $x \in R$ such that $T(a, b, x) = c$
- (E) If $a, b, c, d \in R, a \neq c$, then there is a unique ordered pair $x, y \in R$ such that $T(a, x, y) = b$ and $T(c, x, y) = d$ [8, p. 113].

Planar Ternary Rings (PTRs) will provide a method of finding the underlying field of a finite Desarguesian plane. Given a PTR (R, T) , we define two binary operations as follows:

$$(i) \quad (R, +) \text{ where } \forall a, b \in R, a + b = T(1, a, b)$$

$$(ii) \quad (R^*, \cdot) \text{ where } \forall a, b \in R, a \cdot b = T(a, b, 0)$$

[8, p.117].

When defined this way, $(R, +)$ and (R^*, \cdot) are *loops* [8, p.117]. As seen before in Theorem 1.6.1, every finite Desarguesian plane is also a field plane. Therefore, each finite Desarguesian plane of order n has a corresponding field of order n . If R is a coordinatizing set of a finite Desarguesian plane Π , where $|\Pi| = |R| = n$, then the addition $(R, +)$ and multiplication (R^*, \cdot) defined above are the addition and multiplication of the field $\mathbb{F} = R$ of order n [8, p.120]. Consequently, if we coordinatize a finite Desarguesian plane, then we have a way of finding its underlying field. This gives us the ability to find the Frobenius automorphism.

This coordinatization method was implemented into our program *coord*. The input for *coord* is the incidence matrix A of a finite Desarguesian plane of order n . The program uses the above technique to coordinatize the plane. It also uses the planar ternary ring to find the addition and multiplication of the plane's field. Finally, the program stores three objects in a predefined variable z : 1) the addition table of the field pl , 2) the multiplication table of the field mu , and 3) the coordinatization of the points $crds$. The coordinatization $crds$ is a $(n^2 + n + 1) \times 2$ matrix where each row corresponds to a point written as an ordered pair. Also for the points on ℓ_∞ , if the point was coordinatized as (m) , then in $crds$ the point is stored as $(-1, m)$. The point (∞) is stored as $(-1, -1)$. We use *coord* on the plane of order 4 from Section 2.3.

Example 3.1.1. Given the incidence matrix A from Section 2.3, we type $z \leftarrow \text{coord } A$ in APL. The program stores the following under z

0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

0	0	0	0
0	1	2	3
0	2	3	1
0	3	1	2

0	0
0	1
0	2
0	3
-1	-1
-1	0
2	0
1	0
3	0
1	1
3	3
-1	1
2	2
2	1
-1	3
3	2
1	3
3	1
1	2
2	3
-1	2

where

+	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

.	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

The third item of z is the coordinatization of the points. In the third row we see $(0, 2)$, so the coordinatization of the third point is $(0, 2)$. We have a program *decoord* that when given coordinates will return the original point. For example, *decoord* $(0, 2)$ would return 3.

3.1.1 Program

<i>coord</i>	
[0]	$z \leftarrow \mathbf{coord} \ A; n; m; k; w; z; u; l1; l2; o; inf; li; x; y; i; c; b; j; l1r; l2r; v; D; D1; h; epts;$ $e; q; f; lir; M; m1$
[1]	$n \leftarrow (\rho A)[2] \diamond m \leftarrow (+/A[1;]) - 2 \diamond k \leftarrow h \leftarrow w \leftarrow z \leftarrow u \leftarrow 1 \diamond epts \leftarrow \iota \ n$
[2]	$crds \leftarrow n \ 2\rho \ 0 \diamond l1 \leftarrow \mathbf{ptset} \ 1 \diamond l2 \leftarrow \mathbf{ptset} \ 2 \diamond o \leftarrow \mathbf{pton} \ 1 \ 2$
[3]	$a1 : inf \leftarrow ?n \diamond li \leftarrow \mathbf{ptset} \ inf$
[4]	$\rightarrow a1 \times \iota \ o \in li$
[5]	$x \leftarrow \mathbf{pton} \ 2 \ inf \diamond y \leftarrow \mathbf{pton} \ inf \ 1$
[6]	$a2 : i \leftarrow ?n$
[7]	$\rightarrow a2 \times \iota \ i \in l1, l2, li$
[8]	$c \leftarrow \mathbf{pton} \ (\mathbf{lnon} \ x \ i) \ 1 \diamond b \leftarrow \mathbf{pton} \ (\mathbf{lnon} \ y \ i) \ 2$
[9]	$j \leftarrow \mathbf{pton} \ (\mathbf{lnon} \ c \ b) \ inf \diamond crds[o; 2] \leftarrow 0 \diamond crds[c; 2] \leftarrow 1$
[10]	$l1r \leftarrow (o \neq l1)/l1 \diamond l1r \leftarrow (c \neq l1r)/l1r \diamond l1r \leftarrow (y \neq l1r)/l1r$
[11]	$a3 : crds[l1r[k]; 2] \leftarrow (k + 1)$
[12]	$\rightarrow a3 \times \iota \ m > k \leftarrow k + 1$
[13]	$l2r \leftarrow (x \neq l2)/l2 \diamond v \leftarrow \rho \ l2r$
[14]	$a4 : D \leftarrow l2r[h] \diamond D1 \leftarrow \mathbf{pton} \ (\mathbf{lnon} \ j \ D) \ 1$
[15]	$crds[D; 1] \leftarrow crds[D1; 2]$
[16]	$\rightarrow a4 \times \iota \ v \geq h \leftarrow h + 1$
[17]	$a5 : epts \leftarrow (li[w] \neq epts)/epts$
[18]	$\rightarrow a5 \times \iota \ (m + 2) \geq w \leftarrow w + 1$
[19]	$a6 : e \leftarrow epts[z]$
[20]	$q \leftarrow crds[(\mathbf{pton} \ (\mathbf{lnon} \ x \ e) \ 1); 2] \diamond f \leftarrow crds[(\mathbf{pton} \ (\mathbf{lnon} \ y \ e) \ 2); 1]$
[21]	$crds[e;] \leftarrow f \ q$
[22]	$\rightarrow a6 \times \iota \ (\rho \ epts) \geq z \leftarrow z + 1$
[23]	$crds[y;] \leftarrow (-1 \ -1) \diamond lir \leftarrow (y \neq li)/li$
[24]	$a7 : M \leftarrow lir[u] \diamond m1 \leftarrow crds[(\mathbf{pton} \ (\mathbf{lnon} \ M \ b) \ 1); 2]$
[25]	$crds[M;] \leftarrow -1 \ m1$
[26]	$\rightarrow a7 \times \iota \ (\rho \ lir) \geq u \leftarrow u + 1$
[27]	$pl \leftarrow \mathbf{fplus} \ crds \diamond mu \leftarrow \mathbf{ftimes} \ crds$
[28]	$z \leftarrow (\subset pl), (\subset mu), (\subset crds)$

3.2 IMPLEMENTING THE FROBENIUS AUTOMORPHISM INTO *AUTGENS*

As mentioned previously, the underlying field operations, $(\mathbb{F}, +)$ and (\mathbb{F}^*, \cdot) , of a finite Desarguesian projective plane can be found when the plane is coordinatized. The program *coord* finds the addition table *pl*, the multiplication table *mu*, and the coordinatization of the points *crds*. Another program *frob* uses the multiplication table *mu* determined by *coord* to apply the Frobenius automorphism to elements of the coordinatizing set $R = \mathbb{F}$.

The program *oldautgens* will now be amended to become *autgens*. The new program *autgens* starts exactly like *oldautgens*. It uses homologies to find a set of generators *gens*, which in turn is used to find the group G . Now come the changes. *Autgens* finds the Frobenius automorphism α by applying *frob* to each of the points of the plane. The automorphism α is then added to *gens*. Finally, the group G is generated one last time, and *autgens* stores G as its result. The following table lists the size of the automorphism group generated by *autgens* from planes of various sizes.

Results from running <i>autgens</i>			
Order of Plane	Time to Run	Order of Automorphism Group Generated by <i>autgens</i>	Actual Order of Automorphism Group
3	1.951s	5616	5616
4	2.1s	120960	120960
5	20.917s	372000	372000
8	2m 1.046s	49448448	49448448
9 (field plane)	2m 40.496s	84913920	84913920
11	9m 33.818s	212427600	212427600
16 (field plane)	2h 29m 3.103s	17108582400	17108582400

The program *autgens* is able to successfully generate the full automorphism group

for the listed planes. Clearly, as the order of the plane gets larger, the time it takes *autgens* to finish increases. Is there a way to decrease the run time? Recall that in the program there was a constant named *niterat*. During the first phase of *autgens*, if the program finds a new homology α which is not in the group generated by the earlier (non-redundant) homologies in *gens*, then it adds α to the set of generators *gens*. However, if α is already in the group generated by *gens*, then *autgens* does not add α . If the program is unable to find a new homology *niterat* = 20 times, then it goes onto the “Frobenius” phase. We set *niterat* to 20 so that *autgens* would safely generate the full automorphism group. If we decrease the value of *niterat*, then the program will speed up. However, if we set *niterat* too low, there is a danger of generating only a proper subgroup of the automorphism group.

Probability of <i>autgens</i> finding the full automorphism group for different values of <i>niterat</i> (80 trials)			
<i>niterat</i>	Plane of Order 3	Plane of Order 4	Plane of Order 5
0	75 %	96.25 %	90 %
1	93.75 %	96.25%	97.5 %
2	98.75 %	100 %	100 %
3	100 %	100 %	100 %

Notice that when *niterat* was set to be 3, *autgens* was completely successfully in generating the full automorphism group for the three planes. In general, it appears that as the order of the plane increases, the probability of success for any individual value of *niterat* goes up (with the exception of *niterat* = 0 for the plane of order 5). We will now set *niterat* = 3 and check to see how quickly *autgens* finishes for planes of various orders.

Time for <i>autgens</i> to finish		
Order of Plane	Time to Run <i>niterat</i> = 3	Time to Run <i>niterat</i> = 20
3	0.723s	1.951s
4	2.028s	2.1s
5	6.618s	20.917s
8	1m 9.975s	2m 1.046s
9 (field plane)	1m 16.680s	2m 40.496s
11	2m 44.484s	9m 33.818s
16 (field plane)	23m 19.975s	2h 29m 3.103s

Note that the full automorphism group was successfully generated in each of the cases. When *niterat* was set to be equal to 3 the time went down significantly. For example, *autgens* was more than five times faster for the field plane of order 16 when *niterat* = 3. It would appear that when *niterat* = 3, *autgens* is able to safely, but quickly, calculate the full automorphism group.

3.2.1 Program

<i>autgens</i>	
[0]	$G \leftarrow \mathbf{autgens} \ A; v; count; pt; ln; x; niterat; w; z; q; h; gens$
[1]	$v \leftarrow (\rho A)[2] \diamond w \leftarrow (1, v)\rho \ 0 \diamond z \leftarrow \mathbf{coord} \ A \diamond q \leftarrow 1$
[2]	$gens \leftarrow (0, v)\rho \ \iota \ v \diamond count \leftarrow 0 \diamond niterat \leftarrow 20$
[3]	$G \leftarrow 'G' \ \mathbf{ksims} \ gens$
[4]	$a1 : pt \leftarrow ?v \diamond ln \leftarrow ?v$
[5]	$\rightarrow a1 \times \iota \ pt \in \mathbf{ptset} \ ln$
[6]	$x \leftarrow \mathbf{fixpl} \ pt, ln$
[7]	$\rightarrow a2 \times \iota \ ' - ' \in x \ \mathbf{In} \ G$
[8]	$count \leftarrow count + 1$
[9]	$\rightarrow end \times \iota \ count > niterat$
[10]	$\rightarrow a1$
[11]	$a2 : gens \leftarrow gens, [1]x$
[12]	$' - - - - '$
[13]	$G \leftarrow 'G' \ \mathbf{ksims} \ gens$
[14]	$\rightarrow a1$
[15]	$end : \mathbf{order} \ G$
[16]	$a4 : h \leftarrow crds[q;]$
[17]	$\rightarrow a5 \times \iota \ h[1] = -1$
[18]	$h[1] \leftarrow \mathbf{frob} \ h[1]$
[19]	$a5 : \rightarrow a6 \times \iota \ h[2] = -1$
[20]	$h[2] \leftarrow \mathbf{frob} \ h[2]$
[21]	$a6 : w[1; q] \leftarrow \mathbf{decoord} \ h$
[22]	$q \leftarrow q + 1$
[23]	$\rightarrow a4 \times \iota \ q \leq v$
[24]	$gens \leftarrow gens, [1]w$
[25]	$' - - - - - '$
[26]	$G \leftarrow 'G' \ \mathbf{ksims} \ gens$
[27]	$\mathbf{order} \ G$

Chapter 4

Conclusion

In this thesis, we wanted to generate the full automorphism group of finite projective Desarguesian planes. The main product of our work was the program *autgens* within the APL package named *AUTPPS*. The program *autgens* made extensive use of another program *fixpl*, which finds homologies in the plane, and the program *ksims* which generates a logarithmic signature of a permutation group from a set of generators. When we attempted to use *autgens* on non-Desarguesian planes, the program was unsuccessful. It would appear that our process for finding homologies does not work consistently on non-Desarguesian planes. As an example, for the Hall plane of order 9, *fixpl* was able to find only one $(V, 1)$ -homology (fixing the line 1 and any point V not on line 1). However, when *fixpl* was used on Desarguesian planes, it was successful in finding homologies for any point V and line ℓ where V is not on ℓ . This is why we restricted our focus to the case of Desarguesian planes. The program *autgens* successfully generates the full automorphism group of each Desarguesian plane we apply it too. We were able to speed up the program by tweaking certain things within the code, for example the constant *niterat*.

The question now becomes this: Would *autgens* be able to find the full automorphism group for *any* finite projective Desarguesian plane? More specifically, can we generate the full automorphism group using nothing but homologies and the Frobe-

nus automorphism? We suspect so, but we cannot say for certain. After a search of the literature, we were unable to find a theorem to this effect. It seems that if the order of the plane is p^1 , p a prime, then homologies can be used to generate the full automorphism group. Further, it would seem if the order of the plane is p^a , where a is not equal to 1, then the automorphism group can be generated by the Frobenius automorphism along with homologies. Additionally, we have seen evidence that the full automorphism group of a plane can be generated by as few as 4 generators: the Frobenius automorphism and three homologies. These could be topics for further study. Additionally, one could investigate how to modify *autgens* to work on non-Desarguesian planes.

Appendix A

Programs in *AUTPPS*

autgens

Input: An incidence matrix for a finite projective Desarguesian plane.

Output: The automorphism group.

coord

Input: An incidence matrix.

Output: Stores the following three things: pl - the addition table of the underlying skewfield, mu - the multiplication table of the underlying skewfield, $crds$ - the coordinatization of the points.

decoord

Preset: A coordinatization of points needs to be stored under the variable $crds$.

Input: A point written as an ordered pair from the coordinatization of the points.

Output: The point V . It is written according to the corresponding incidence matrix, where V is the column corresponding to the point.

fixpl

Preset: An incidence matrix needs to be stored under the variable A .

Input: A point and a line. The point V and line ℓ are entered as the integers corresponding to the desired point and line from the incidence matrix A .

Output: A permutation of points.

frob

Preset: A multiplication table needs to be stored under the variable mu . If the program *coord* is run beforehand, mu will already be set.

Input: A point V .

Output: A point V' , where V' is the result of applying the Frobenius automorphism to V .

getinc

Input: A design. If n is the order of the matrix, then the design is entered in as a $v \times (n + 1)$ matrix where $v = n^2 + n + 1$. Each row of the matrix represents a line, with the points as the entries of the row.

Output: An incidence matrix.

limplies

Preset: An incidence matrix needs to be stored under the variable A .

Input: A permutation of lines.

Output: The permutation of points that occur as a result of the permutation of lines.

lnon

Preset: An incidence matrix needs to be stored under the variable A .

Input: Two points. The points are entered in as integers p and q , where p and q corresponds to the p th and q th columns of an incidence matrix.

Output: The line that is incident with the two points.

lncset

Preset: An incidence matrix needs to be stored under the variable A .

Input: A point. The point is entered in as an integer n , where n corresponds to

the n th column of an incidence matrix (so n is the point).

Output: The set of lines incident with the point. It is returned as a vector of integers.

pimplies

Preset: An incidence matrix needs to be stored under the variable A .

Input: A permutation of points.

Output: The permutation of lines that occur as a result of the permutation of points.

pton

Preset: An incidence matrix needs to be stored under the variable A .

Input: Two lines. The lines are entered in as integers p and q , where p and q corresponds to the p th and q th rows of an incidence matrix.

Output: The point that is incident with the two lines.

ptset

Preset: An incidence matrix needs to be stored under the variable A .

Input: A line. The line is entered in as an integer m , where m corresponds to the m th row of an incidence matrix (which is a line).

Output: The set of points incident with the line. It is returned as a vector of integers.

BIBLIOGRAPHY

- [1] A. A. ALBERT AND R. SANDLER, *An introduction to finite projective planes*, Holt, Rinehart, and Winston, New York, NY, 1968.
- [2] H. S. M. COXETER, *Introduction to geometry*, John Wiley & Sons, New York, NY, 1963.
- [3] H. S. M. COXETER, *Projective geometry*, Blaisdell, New York, NY, 1964.
- [4] J. DÉNES AND A. D. KEEDWELL, *Latin squares and their applications*, Academic Press, New York, NY, 1974.
- [5] D. S. DUMMIT AND R. M. FOOTE, *Abstract algebra*, John Wiley & Sons, Hoboken, NJ, 3rd Edition, 2004.
- [6] W. T. FISHBACK, *Projective and euclidean geometry*, John Wiley & Sons, New York, NY, 2nd Edition, 1969.
- [7] M. HALL, JR., *Combinatorial theory*, John Wiley & Sons, New York, NY, 1967.
- [8] D. R. HUGHES AND F. C. PIPER, *Projective planes*, Springer-Verlag, New York, NY, 1973.
- [9] M. JERRUM, A compact representation for permutation groups, *J. Algorithms* **7** (1986), pp. 60–78.
- [10] C. W. H. LAM, The search for a finite projective plane of order 10, *Amer. Math. Monthly* **98** (1991), pp. 305–318.
- [11] S.S. MAGLIVERAS, A Cryptosystem from Logarithmic Signatures of Finite Groups, *Proceedings of the 29th Midwest Symposium on Circuits and Systems*, Elsevier Publ. Co. (1986), pp. 972-975.
- [12] M. SANIGA, M. PLANAT, AND H. ROSU, Mutually Unbiased Bases and Finite Projective Planes, *J. Opt. B: Quantum Semiclass* **6** (2004), pp. L19-L20.
- [13] C. C. SIMS, Computational methods in the study of permutation groups, *Computational Problems in Abstract Algebra*, Conf. Proceedings, Pergamon Press, (1970) pp. 169 – 183.