

ITERATED POINT-LINE CONFIGURATIONS IN PROJECTIVE PLANES

By

Mark Thomas Walters

Bachelor of Science
Miami University 2004

Master of Science
Miami University 2005

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy in

Mathematics

College of Arts and Sciences

University of South Carolina

2009

Accepted by:

Joshua Cooper, Major Professor

Linyuan Lu, Committee Member

Lili Ju, Committee Member

Peter Nyikos, Committee Member

Steven Mann, External Examiner

James Buggy, Interim Dean of the Graduate School

DEDICATION

This work is dedicated to my mother, Nancy Walters, who passed away February 29, 2008. There are no words that can express the sadness I have endured and the void that remains in losing her. She had always encouraged me to strive in academics and her relentless love and support helped to guide me through all of my schooling.

ACKNOWLEDGMENTS

First, I would like to thank my Ph.D. advisor, Josh Cooper, for everything he has done for me. Josh took me under his wing almost three years ago, shortly after we met, and has since taught me more about mathematics than any other person on the planet. It goes way beyond mathematics with this guy though... he has taught me how to pronounce a large part of the Hungarian dictionary (among other languages), how to use Google as a high-tech calculator, how to program in java and SAGE, and how to get a good deal on wedding flowers. He supplied almost all of the problems that we worked on in this dissertation, while consistently giving me valuable insight and ideas to try. He has been a great teacher, mentor, and friend to me and the countless hours that I have spent in his office have been worth every minute.

Next, I have to thank the other committee members: Professors László Székely, Linyuan Lu, Lili Ju, Peter Nyikos, and Steven Mann. Alongside Josh, these professors played an integral role in the progression of my graduate education. They have taught me a lot about discrete math, graph theory, numerical and applied math, set theory, and fixed income securities, respectively. In addition, they were brave enough to sit through either my oral comprehensive exams or my dissertation defense (or both), which must have been a little painful. I have to thank Professor George McNulty for providing his \LaTeX expertise, which helped make this dissertation look presentable. I would also like to thank a few professors from previous schools that have made major contributions to my education: Professors Mark Smith, Dan Pritikin, and Stephen Wright from Miami University and Professors Ralph Carlson and Dale McIntyre from Grove City College.

I also owe a lot to my family. My parents, Tom and Nancy Walters, have been the most supportive and loving parents that anyone could ever ask for. They, along with my sisters, Dana and Erin, have been to hundreds of baseball and basketball games, cross country and track meets, and various academic endeavors that I have pursued over the years from elementary school through college. My family has always encouraged me to excel in academics and sports. More importantly, they have taught me how to treat others, how to stand up for myself, and how to know the difference between right and wrong.

I certainly cannot leave out my fiancée, Ashley Kuhlmann. Over the last four years, she has learned way more about mathematics than she ever wanted to learn. Even though she was unable to contribute much to my actual dissertation, she was able to provide much-needed distractions at critical times during my research. Her love and support has helped me survive these four years of graduate school year at USC.

Finally, I need to thank the friends that I have made since I arrived here at USC four years ago. Aside from Ashley, I have spent more of my free time with Luke Owens than any other person. Just as I made my way into the acknowledgements section in Luke's dissertation, he is in mine for all the wrong reasons. Most of the time that I spent with Luke was horribly counterproductive to anything resembling academics. He could have easily been held responsible for me failing analysis during my first year, had I done so. He redeemed himself a bit by letting me borrow his study materials for the qualifying exams. His wife, Kate Owens, along with Matt Hielsberg, Chas Cavalier, Rob Tompkins, and Sandy and Luke Morrissey have also been great friends to me during these years. I know from experience that golf courses, bowling alleys, bars, concerts, pool tables, beach trips, poker games, and movie theaters are all atmospheres that are not conducive to academic progress... but I would not trade the memories for anything.

ABSTRACT

In this Ph.D. dissertation we analyze the behavior of iterated point-line configurations defined by the following process: begin with a set of four points in the real plane in general position. Add to this collection the intersections of all lines through pairs of these points. Iterate. Ismailescu and Radoičić (2004) showed that the limiting set is dense in the plane. Let IRP denote the Ismailescu/Radoičić plane found in this manner. We show that the number of points in IRP grows doubly exponentially. Next, we analyze the relationship between Pappus's Theorem and Desargues's Theorem in projective geometry and give a result that characterizes IRP in terms of Pappus. Through an analysis of cross products, we provide a simple convention to represent this iterative process. We also discuss the growth rate of this construction in the free projective plane. We prove that the lower bound is asymptotic to the trivial upper bound, which pins down this growth rate. Finally, we address some unanswered questions and conclude with a result in additive number theory that was proved along the way.

CONTENTS

DEDICATION	ii
ACKNOWLEDGMENTS	iii
ABSTRACT	v
LIST OF TABLES	viii
LIST OF FIGURES	ix
CHAPTER 1. INTRODUCTION	1
CHAPTER 2. IRP GROWTH RATE	4
2.1. Trivial Bounds	4
2.2. Nominal Improvement to the Upper Bound	6
2.3. Degree Bounds	7
2.4. Numerical Results	17
CHAPTER 3. CHARACTERIZATION OF IRP	21
3.1. Pappus and Desargues	21
3.2. Free \mathcal{F} -Plane	24
3.3. Representation by Coefficient Vectors	28
CHAPTER 4. FREE PROJECTIVE PLANE	34
CHAPTER 5. REMAINING QUESTIONS AND FURTHER WORK	39
5.1. Computational Results	39
5.2. Rank of IRP	40
5.3. IRP_n	44
5.4. Counting Pappus Configurations	44
5.5. The Free “Desarguian” Plane	45

CHAPTER 6. A RESULT IN ADDITIVE NUMBER THEORY	47
BIBLIOGRAPHY	49
APPENDICES:	
APPENDIX A. HOMOGENEOUS COORDINATES CODE	51
APPENDIX B. REDUCED COEFFICIENT VECTOR CODE	59
APPENDIX C. REDUCED COEFFICIENT VECTOR PROOF	63
APPENDIX D. FREE PROJECTIVE PLANE CODE	70

LIST OF TABLES

Table 4.1	Free Projective Plane Sequence	36
Table 4.2	Values for α_k and β_k	37
Table 5.1	Sequence for IRP	39
Table 5.2	IRP ₅ and IRP ₆	44

LIST OF FIGURES

Figure 1.1	The End of Stage 1	2
Figure 2.1	Illustrations for Case 2	5
Figure 2.2	5×5 Grid Example	9
Figure 2.3	Illustration for Case 2	10
Figure 2.4	Example with $N = 4$ and $k = 2$	12
Figure 3.1	Pappus's Theorem	21
Figure 3.2	Desargues's Theorem	22
Figure 3.3	Addition	23

CHAPTER 1

INTRODUCTION

Consider the iterative process of constructing points and lines in the real plane given by the following: begin with a set of points $P_1 = \{p_1, p_2, p_3, p_4\}$ in the real plane in general position. For each pair of points, construct the line passing through the pair. This will create a set of lines $L_1 = \{\ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ell_6\}$. Some of these constructed lines will intersect at points in the plane that do not belong to the set P_1 . Add any such point to the set P_1 to get a new set P_2 . Now, note that there exist some pairs of points in P_2 that do not lie on a line in L_1 , namely some elements of $P_2 \setminus P_1$. Add these missing lines to the set L_1 to get a new set L_2 . Iterate in this manner, adding points to P_k followed by adding lines to L_k .

Ismailescu and Radoičić showed in [11] – and Hillar and Rhea independently in [9] – that the limiting point set resulting from this process is dense in the plane. Grüne and Kamali in [6] show that iterated line *segment* intersections are generally dense in the interior of the convex hull of the starting set. A similar density result was proven by Bezdek and Pach in [2], where they take iterated circle intersections instead of lines. The authors of [10] consider taking incenters, centroids, and circumcenters of triangles formed by the configuration, and [1] considers orthocenters in addition to circumcenters. A natural follow-up question to these density results is: How many points are present at the k^{th} stage? We consider this problem here.

Now we introduce some notation for this iterative process. First, we use the term “general position” to require that for every $k \in \mathbb{N}$ no two lines in L_k are parallel. It will suffice to choose P_1 in a way so that its points have algebraically independent

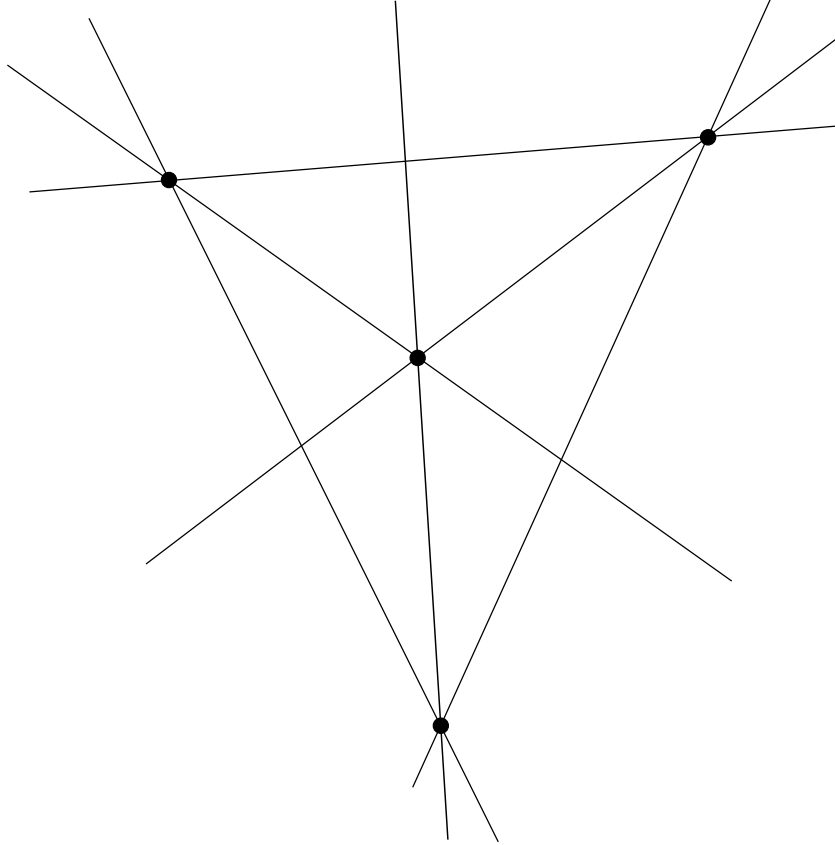


FIGURE 1.1. The End of Stage 1

coordinates. We show in Chapter 3 that the sequence $\{n_k\}_{k=1}^{\infty}$ does not depend on the choice of P_1 under our assumptions. The k^{th} stage is defined to consist of these two ordered steps:

- (1) Add each intersection of pairs of elements of L_k to P_{k+1} , and
- (2) Add a line through each of pair of elements of P_{k+1} to L_{k+1} .

Under this definition, we say that stage 1 begins with an empty plane and ends with the configuration of four points with six lines. In general, at the conclusion of stage k the plane has n_k points and m_k lines. We will denote the set of points at the end of stage k by P_k and likewise the set of lines at the end of stage k by L_k . Note that the convex hull of P_1 is a triangle in Figure 1.1, but it may also be a quadrilateral. The results presented here only require the general position assumption.

Note that a stage in this iterative process can be alternatively defined as follows:

- (1) Place a point at any intersection of a pair of lines for which a point does not already exist.
- (2) Take the dual of the configuration of points and lines (points become lines and lines become points).
- (3) Repeat steps 1 and 2.

Hence, points and lines play a very similar role in this process and we only need to consider bounds on one of the two quantities. Therefore, from here forward we will focus most of our attention on the values of n_k . Also, since the iterations of this process are performed under the settings defined by Ismailescu and Radoičić in [11], we will refer to this construction of points and lines in the real plane as the Ismailescu Radoičić Plane (IRP) .

CHAPTER 2

IRP GROWTH RATE

2.1. TRIVIAL BOUNDS

There are some trivial bounds on the number of points and lines at stage k that can be obtained with this notation. Since a point in P_k must lie at the intersection of at least two lines of L_{k-1} (with the exception of $k = 1$) we know that at the end of stage k , there are at most $\binom{m_{k-1}}{2}$ points. Similarly, since a line in L_k must contain at least two points from P_k we know that at the end of stage k there are at most $\binom{n_k}{2}$ lines. In other words,

$$n_k \leq \binom{m_{k-1}}{2} \quad \text{and} \quad m_k \leq \binom{n_k}{2}.$$

From this it follows that

$$n_{k+1} \leq \binom{m_k}{2} \leq \binom{\binom{n_k}{2}}{2} < \binom{\frac{n_k^2}{2}}{2} < \frac{\left(\frac{n_k^2}{2}\right)^2}{2} = \frac{n_k^4}{8} \quad (1)$$

and

$$m_{k+1} \leq \binom{n_{k+1}}{2} \leq \binom{\binom{m_k}{2}}{2} < \binom{\frac{m_k^2}{2}}{2} < \frac{\left(\frac{m_k^2}{2}\right)^2}{2} = \frac{m_k^4}{8}. \quad (2)$$

A trivial lower bound is given in the following:

PROPOSITION 1. *For all $k \in \mathbb{N}$, $n_{k+1} \geq n_k + 1$.*

PROOF. If this claim is false then we must have a stage at which the process stabilizes [11]. So, suppose that the process stabilizes after stage k and let $\text{conv}(P_k)$ denote the convex hull of P_k , where $|\text{conv}(P_k)|$ denotes the number of vertices of this

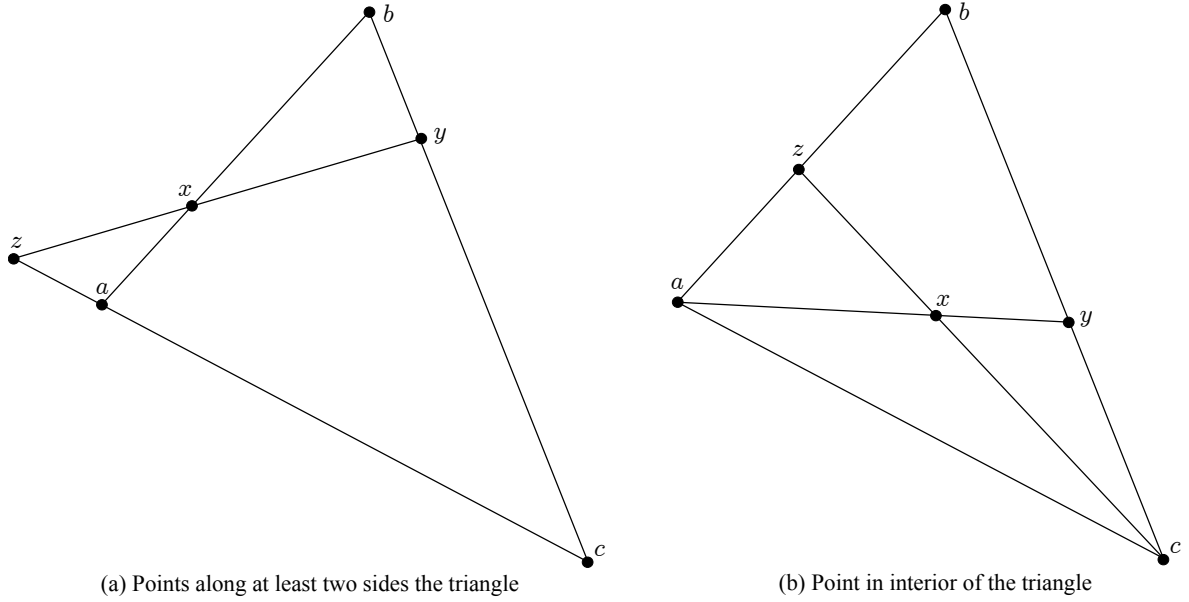


FIGURE 2.1. Illustrations for Case 2

convex hull. There are two cases. Suppose first that $|\text{conv}(P_k)| \geq 4$. In this case, we can find two nonadjacent, nonparallel sides of the convex hull, which lie on lines that intersect outside of the convex hull. This contradicts the stability supposition. In case 2 we have $|\text{conv}(P_k)| = 3$. Let $\{a, b, c\}$ be the set of vertices of the triangle forming the convex hull. Suppose that there exist points along at least two of the sides of the triangle defined by $\{a, b, c\}$, say $x \in ab$ and $y \in bc$. In this case, the point $z = ac \cap xy$ lies outside the convex hull (Figure 2.1(a)), again contradicting stability. So, there exist points along at most one of the sides of the triangle defined by $\{a, b, c\}$. Suppose that there exists some point x in the interior of $\{a, b, c\}$ and define $y = ax \cap bc$, $z = cx \cap ab$. In this case, we have $y \in bc$ and $z \in ab$, a contradiction to the assumption that at most one side of the triangle contains points (Figure 2.1(b)). The only remaining possibility is that P_k is comprised of $n_k - 1$ collinear points. But, the starting configuration of points and lines has the condition that for any line in L_1 , there are at least two points of P_1 not lying on it. Since we never remove any points during this process, then this must hold true for every stage, in particular stage k . This contradiction completes the proof. \square

2.2. NOMINAL IMPROVEMENT TO THE UPPER BOUND

We are able to slightly reduce the trivial upper bound by bounding the inherent overcount. First we need a few definitions. We define the *degree* of a point $p \in P_k$, denoted $d_k(p)$, to be the number of distinct lines incident upon p at the end of stage k . Similarly, the degree of a line $\ell \in L_k$, denoted $d_k(\ell)$, is the number of distinct points through which it passes at the end of stage k . Also, let

$$\delta_k = \min\{d_k(p) \mid p \in P_k\} \quad \text{and} \quad \bar{\delta}_k = \min\{d_k(\ell) \mid \ell \in L_k\}$$

and

$$\Delta_k = \max\{d_k(p) \mid p \in P_k\} \quad \text{and} \quad \bar{\Delta}_k = \max\{d_k(\ell) \mid \ell \in L_k\}.$$

A portion of the overcount using the trivial upper bound is due to the degree of a point.

LEMMA 2. *For all $k \in \mathbb{N}$,*

$$n_{k+1} \leq \frac{n_k^4}{8} - \left[\binom{\delta_k}{2} - 1 \right] n_k$$

PROOF. Note that the trivial upper bound assumes that each pair of distinct lines will generate a new point. The problem with this approach arises when a point has degree exceeding two. For instance, suppose $p \in P_k$ with $d_k(p) = 3$, i.e., there are three distinct lines $\ell_1, \ell_2, \ell_3 \in L_k$ passing through p . In stage $k+1$, the point p will be counted three times: once for $\ell_1 \cap \ell_2$, once for $\ell_1 \cap \ell_3$, and once for $\ell_2 \cap \ell_3$. Similarly, a point of degree 4 in stage k will be counted $\binom{4}{2} = 6$ times in stage $k+1$, and in general, a point of degree d in stage k will be counted $\binom{d}{2}$ times in stage $k+1$. Since p is only one point, it is overcounted $\binom{d}{2} - 1$ times. Note that each point in P_k has degree at least δ_k . It follows that each point in P_k will be overcounted at least $\binom{\delta_k}{2} - 1$ times. Since there are n_k points in P_k , the minimum overcount for n_{k+1} is

$\binom{\delta_k}{2} - 1) n_k$ and so we get

$$n_{k+1} \leq \frac{n_k^4}{8} - \left[\binom{\delta_k}{2} - 1 \right] n_k,$$

as desired. \square

2.3. DEGREE BOUNDS

Many of the arguments herein require some knowledge or control over the degrees of the points in the configuration. Hence, we must include some results that bound the degrees under various circumstances. We begin with the following observation:

PROPOSITION 3. *For all $k \in \mathbb{N}$, $\delta_k \geq 3$.*

PROOF. Suppose to the contrary that there exists some $k \in \mathbb{N}$ with $\delta_k < 3$. Since there are no points of degree 1, we must have $\delta_k = 2$. So there exists $p \in P_k$ with $d_k(p) = 2$, i.e., there exist two lines $\ell, \ell' \in L_k$ with $P_k \subseteq \ell \cup \ell'$. Note that $n_2 = 7$ and $\bar{\Delta}_2 = 3$ and so for all $\ell, \ell' \in L_2$, $P_2 \not\subseteq \ell \cup \ell'$. Since we never remove points in this iterative process, we know that if there exists $\ell, \ell' \in L_k$ with $P_k \subseteq \ell \cup \ell'$, then $k < 2$, i.e., $k = 1$. But we know that $\delta_1 = 3$, a contradiction. \square

Define an $n \times n$ *grid* to be any configuration of two collections of n parallel lines, where the one collection is not parallel to the other. We can obtain major improvements to the trivial lower bound using the following:

LEMMA 4. *The minimum size of a set S of parallel lines that passes through all of the intersections of an $n \times n$ grid, comprised of sets Q and R , where S is disjoint from Q and R , is $2n - 1$.*

PROOF. Suppose that Q and R are sets of parallel lines that comprise an $n \times n$ grid. Let S be a minimal witness set of s parallel lines passing through all intersections of the grid. We aim to show that $s \geq 2n - 1$. Without loss of generality, orient the grid so that the lines of S are vertical in the xy -plane and let $X = \{x_1, x_2, \dots, x_s\}$ be

the x -intercepts of the lines of S . So X is the collection of projected points, when we project the grid intersections onto the x -axis with this orientation. Let $\pi(p)$ denote the projection of a point p in the grid onto the x -axis. Arbitrarily choose lines ℓ_q, ℓ_r in the grid with $\ell_q \in Q$ and $\ell_r \in R$. Let q_1, q_2, \dots, q_n and r_1, r_2, \dots, r_n be the points of intersection of ℓ_q with R and ℓ_r with Q , respectively, where

$$\pi(q_1) \leq \pi(q_2) \leq \dots \leq \pi(q_n)$$

and

$$\pi(r_1) \leq \pi(r_2) \leq \dots \leq \pi(r_n).$$

Suppose also that $q_i = r_j$. Define A and B to be the sets of real numbers given by

$$A = \{\pi(q_1), \pi(q_2), \dots, \pi(q_n)\}$$

and

$$B = \{\pi(r_1) - \pi(r_j), \pi(r_2) - \pi(r_j), \dots, \pi(r_n) - \pi(r_j)\}.$$

Under this setting we have that $X = A + B$ and thus

$$s = |A + B|.$$

It is well known that

$$|A + B| \geq 2n - 1$$

for any pair A, B of sets of cardinality n and that equality is achieved when A and B are arithmetic progressions [12]. It follows that $s \geq 2n - 1$, completing the proof. \square

Using this lemma we can prove the following:

THEOREM 5. *For every $k \in \mathbb{N}$ we have $\delta_{k+1} \geq \min\{n_k - 1, 2\delta_k - 3\}$.*

PROOF. Let $p \in P_k$. It suffices to show that $d_{k+1}(p) \geq \min\{n_k - 1, 2\delta_k - 3\}$. There are two cases. In case 1, suppose each line in L_k that passes through p has degree 2. In this case, there are $d_k(p) + 1$ points at the beginning of stage k and so

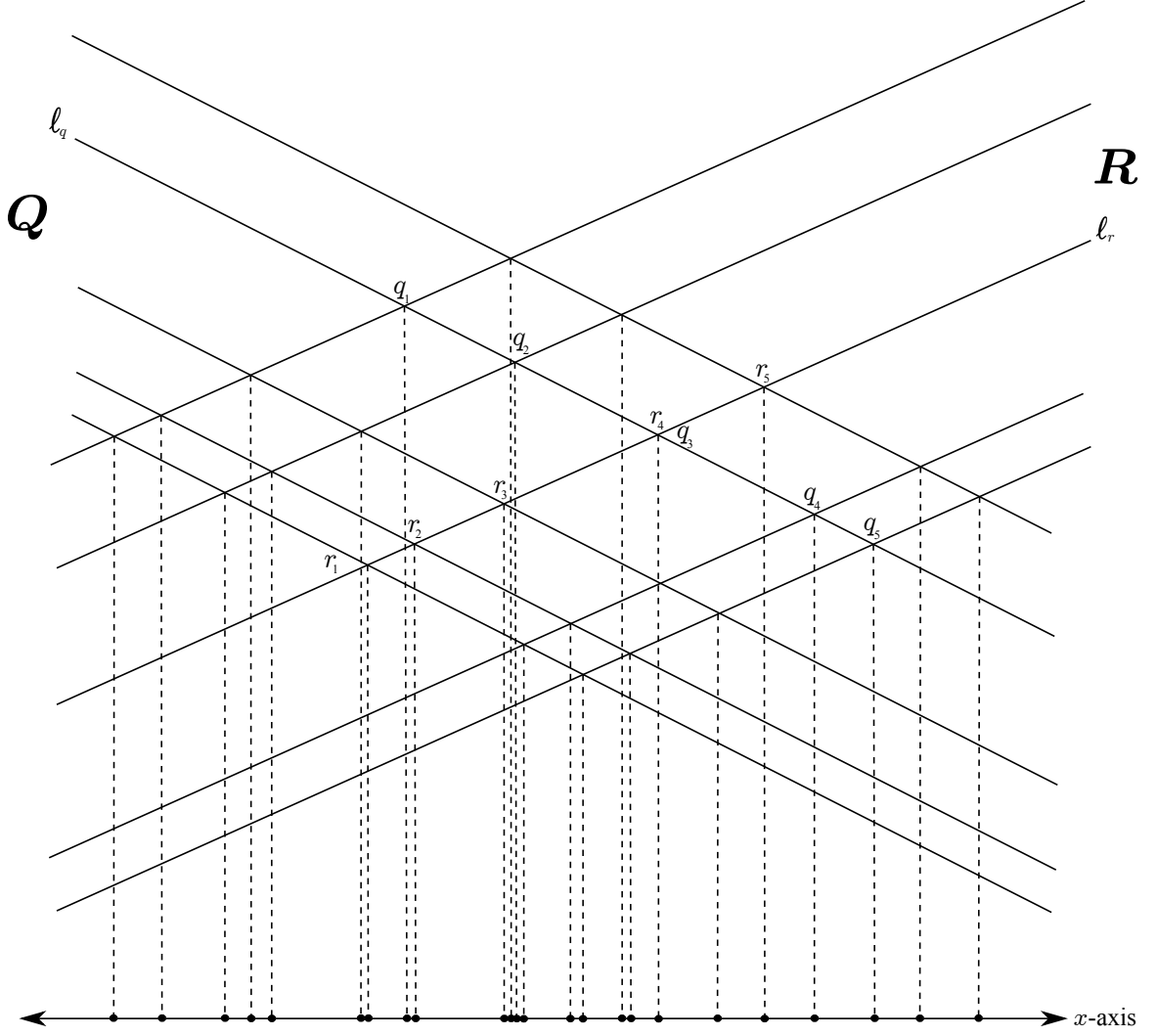


FIGURE 2.2. 5×5 Grid Example

$d_k(p) = n_k - 1$. Since we never remove lines, we get

$$d_{k+1}(p) \geq d_k(p) = n_k - 1 \geq \min\{n_k - 1, 2\delta_k - 3\}.$$

This completes the proof of case 1.

For case 2, suppose that there exists a line $\ell \in L_k$ that passes through p with $d_k(\ell) \geq 3$. Let $q, r \in P_k$ be the other two points on ℓ . Note that $d_k(q) \geq \delta_k$ and $d_k(r) \geq \delta_k$ and so there exist two sets of lines

$$L_q = \{\ell_{q_1}, \ell_{q_2}, \dots, \ell_{q_n}\} \subseteq L_k \setminus \ell \quad \text{and} \quad L_r = \{\ell_{r_1}, \ell_{r_2}, \dots, \ell_{r_m}\} \subseteq L_k \setminus \ell,$$

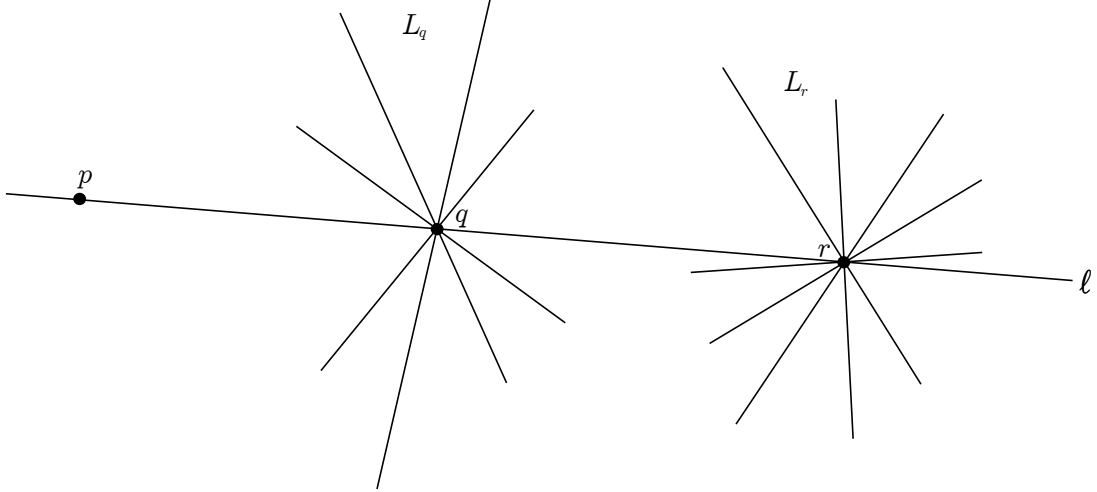


FIGURE 2.3. Illustration for Case 2

where $n, m \geq \delta_k - 1$ and the sets $L_q \cup \ell$ and $L_r \cup \ell$ consist of the lines incident upon q and r , respectively (See Figure 2.3). Now, consider the real plane as a subset of the real projective plane in the standard way and let ℓ be the line at infinity. We restrict our attention to arbitrarily chosen subsets $L_q' \subseteq L_q$ and $L_r' \subseteq L_r$, where $|L_q'| = |L_r'| = \delta_k - 1$. These lines form a $(\delta_k - 1) \times (\delta_k - 1)$ grid. Now in this grid we will place a point at each intersection for which one does not already exist during stage k . After doing so, we will construct a line through each pair of points for which one does not already exist. In particular, we will do so for pairs of points of the form (p, x) , where x lies at the intersection of lines from L_q' and L_r' . So, at the beginning of stage $k + 1$, there will be at least s lines incident upon p , where s denotes the number of lines necessary to adjoin p with all of the intersections of the grid. In other words, $d_{k+1}(p) \geq s$. Note that any lines passing through p would form a third collection of parallel lines to add to the grid. Therefore, s is at least the minimum number of parallel lines required to pass through all of the intersections of a $(\delta_k - 1) \times (\delta_k - 1)$ grid. Applying Lemma 4 yields

$$d_{k+1}(p) \geq s \geq 2(\delta_k - 1) - 1 = 2\delta_k - 3 \geq \min\{n_k - 1, 2\delta_k - 3\}.$$

□

Now by using techniques similar to the preceding proofs, we can obtain even faster growth of the minimum degree. We will then use the growth rate of δ_k to provide arguments for a better lower bound on n_k . First, let $cr(G)$ denote the crossing number of a graph, which is the minimum number of crossings in a planar drawing of the graph G . We will use the following lemma regarding crossing numbers (the proof can be found in [13]):

LEMMA 6. *If a graph G with n vertices and e edges has $e > 7.5n$, then we have*

$$cr(G) \geq \frac{e^3}{33.75n^2}.$$

We now use this crossing number inequality in the following theorem. The argument closely resembles Székely's proof [15] of the Szemerédi-Trotter Theorem (first appearing in [16]).

THEOREM 7. *Let $\mathcal{F} = \{F_1, F_2, \dots, F_N\}$ be a collection of $N \geq 4$ families, each of exactly $k \geq 2$ parallel lines, no two collections parallel to each other. Let V denote the set of intersection points of all pairs of lines in \mathcal{F} and let V_1 be those points that lie on F_1 . Then*

$$|V_1| \geq ck^2N^{1/2},$$

where $c \in \mathbb{R}$ is a positive constant that does not depend on N or k .

PROOF. Let A denote this configuration of $|V_1|$ points and Nk lines. Let i be the number of point-line incidences in A . Note that there are N different families of parallel lines in A , each containing exactly k lines. For all families except F_1 , each line contains exactly k points from V_1 and thus contains exactly $k - 1$ line segments which connect two points from A , call them edges. We know that $k \geq 2$ and so $k - 1 \geq k/2$. Hence, each line contains at least $k/2$ edges and if we add this up over all of the Nk lines, we see that the number of edges obtained in this manner is at least half of the

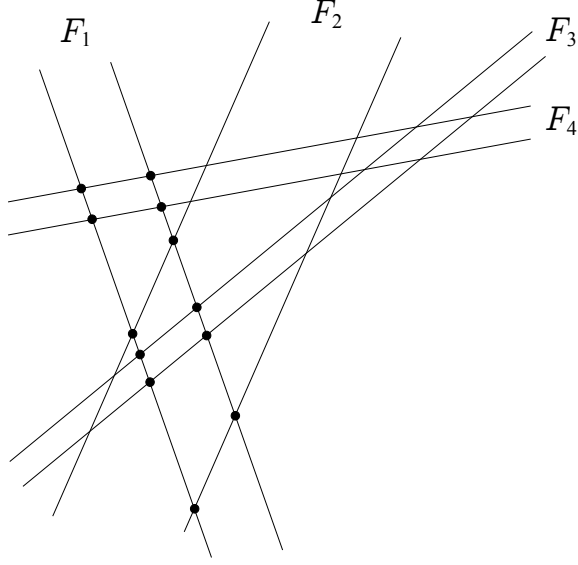


FIGURE 2.4. Example with $N = 4$ and $k = 2$

total number of incidences. In other words,

$$(\text{total number of edges}) \geq \frac{i}{2}.$$

Now, we can count the exact number of edges in A . For the k lines of F_1 , there are $|V_1| - k$ edges because all $|V_1|$ points lie on the lines of F_1 and for each line we must subtract one to count the number of edges. For each of the remaining $N - 1$ families there are exactly k lines, each containing exactly $k - 1$ edges, yielding a total of

$$(N - 1)k(k - 1)$$

edges. Adding these quantities together, we obtain a grand total of

$$|V_1| - k + (N - 1)k(k - 1)$$

edges, which simplifies to

$$|V_1| + Nk(k - 1) - k^2.$$

Now consider the graph G with $V(G) = V_1$ and $E(G)$ consisting of the aforementioned edges. Since all of the edges lie on one of Nk lines, and any two lines intersect in at

most one point, we have

$$\text{cr}(G) \leq (Nk)^2.$$

Applying the crossing number inequality, we obtain that either

$$|V_1| + Nk(k-1) - k^2 \leq 7.5|V_1| \tag{3}$$

or that

$$(Nk)^2 \geq \frac{(|V_1| + Nk(k-1) - k^2)^3}{33.75|V_1|^2} \tag{4}$$

In the case of (3) we get

$$Nk(k-1) - k^2 \leq 6.5|V_1|$$

which implies that

$$\frac{Nk(k-1) - k^2}{6.5} \leq |V_1|.$$

Now, since we know that $k \geq 2$ and $N \geq 4$, we have $k-1 \geq k/2$ and $N-2 \geq N^{1/2}$.

Combining this with the previous equation yields

$$|V_1| \geq \frac{Nk(k-1) - k^2}{6.5} \geq \frac{(N-2)k^2}{13} \geq c_1 k^2 N^{1/2}$$

for some positive constant $c_1 \in \mathbb{R}$.

In the case of (4) we have

$$33.75|V_1|^2(Nk)^2 \geq (|V_1| + Nk(k-1) - k^2)^3$$

and so

$$c_2|V_1|^{2/3}(Nk)^{2/3} \geq |V_1| + Nk(k-1) - k^2$$

for some positive constant $c_2 \in \mathbb{R}$. Recall that the right-hand-side of this inequality is $|E(G)|$, which is at least $i/2$. So we have

$$\frac{i}{2} \leq c_2|V_1|^{2/3}(Nk)^{2/3}.$$

Now, since each of the Nk lines in A must pass through at least k points, then there are at least Nk^2 incidences. From this it follows that

$$Nk^2 \leq i \leq c_3|V_1|^{2/3}(Nk)^{2/3}$$

for some positive constant $c_3 \in \mathbb{R}$. Hence,

$$N^3k^6 \leq c_4|V_1|^2(Nk)^2$$

and so

$$|V_1| \geq c_5k^2N^{1/2}$$

for some positive constants $c_4, c_5 \in \mathbb{R}$. So in both cases, we end up with our desired result. \square

Now, we can use the previous result to prove the following lemma regarding degree growth:

LEMMA 8. *There exists a positive constant $c \in \mathbb{R}$ such that for any point $p \in P_k$ with $d_k(p) = d$ we have*

$$d_{k+1}(p) \geq c\delta_k \left(\frac{n_k}{d}\right)^{1/2}.$$

PROOF. Let $p \in P_k$ with $d_k(p) = d$. By the pigeonhole principle, there exists some line ℓ through p with at least $s = \frac{n_k-1}{d}$ points on it (excluding p). Since each of these s points has at least the minimum degree, we know that there are at least $\delta_k - 1$ lines through each point (excluding ℓ). Consider the real plane as a subset of the real projective plane in the standard way and let ℓ be the line at infinity. If we restrict our attention to only the points on ℓ and the lines through them, then we obtain a grid of $s + 1$ families of parallel lines, one family for each of the points on ℓ . Each family of parallel lines contains at least $\delta_k - 1$ lines and no two families can be parallel (since they come from distinct points). We would like to restrict our attention to families of exactly $\delta_k - 1$ parallel lines. So for each family, except for the one generated by p , arbitrarily choose a subset of $\delta_k - 1$ lines and disregard all

other lines in that family. Let F be the family of lines through p and choose one family R to be a set of “reference” lines. Let P_0 denote the set of points that lie at the intersection of a reference line and one of the other $s - 1$ families (excluding F).

Now, during stage k , a point must be added to any intersection for which one does not already exist, in particular all points of P_0 . Also, a line must be added to connect any pair of points for which one does not already exist, in particular for the pairs in the set $T = \{(p, q) \mid q \in P_0\}$. Let t denote the number of distinct lines generated by pairs in the set T . Note that any such line can pass through at most $\delta_k - 1$ points of P_0 because all the points of P_0 lie in the family R , which contains exactly $\delta_k - 1$ lines. It follows that

$$d_{k+1}(p) \geq t \geq \frac{|P_0|}{\delta_k - 1} \geq \frac{|P_0|}{\delta_k}, \quad (5)$$

with the first inequality holding because any line generated by the set T must pass through p , and hence contributes to its degree in the stage.

Now, for the moment, exclude F from our collection of families and consider all other families of lines along with the points of P_0 . Suppose $s < 4$. Hence, $n_k - 1 < 4d$ and so $n_k < 4d + 1$, i.e., $n_k \leq 4d$. It follows that

$$\left(\frac{n_k}{d}\right)^{1/2} \leq 2.$$

Note that $d_{k+1}(p) \geq \delta_k$ for all $p \in P_{k+1}$ and so $d_{k+1}(p) \geq 2c\delta_k$ holds true for $c = \frac{1}{2}$. Hence,

$$d_{k+1}(p) \geq c\delta_k \left(\frac{n_k}{d}\right)^{1/2}$$

for some positive constant $c \in \mathbb{R}$, as desired. Now suppose that $s \geq 4$. Since we also know that $\delta_k \geq 3$, i.e., $\delta_k - 1 \geq 2$ for all $k \in \mathbb{N}$, we can apply Theorem 2 to this configuration with $F_1 = R$, $N = s$, and $k = \delta_k - 1$. It follows that

$$|P_0| \geq c_1(\delta_k - 1)^2 \left(\frac{n_k - 1}{d}\right)^{1/2}$$

for some positive constant $c_1 \in \mathbb{R}$. Now, note that $\delta_k \geq 2$ and $n_k \geq 4$, which implies that $\delta_k - 1 \geq \frac{1}{2}\delta_k$ and $n_k - 1 \geq \frac{3}{4}n_k$. Combining this with the previous equation, we obtain

$$|P_0| \geq c_1 \left(\frac{\delta_k}{2}\right)^2 \left(\frac{3n_k}{4d}\right)^{1/2} \geq c_2 \delta_k^2 \left(\frac{n_k}{d}\right)^{1/2} \quad (6)$$

for some positive constant $c_2 \in \mathbb{R}$. If we combine (5) and (6) we get

$$d_{k+1}(p) \geq c_2 \delta_k \left(\frac{n_k}{d}\right)^{1/2},$$

as desired. This completes the proof. \square

Note that $|P_0| \leq n_{k+1}$ and so there exists a positive constant $c \in \mathbb{R}$ such that

$$n_{k+1} \geq c \delta_k^2 \left(\frac{n_k}{d}\right)^{1/2}$$

holds for any point $p \in P_k$ with $d_k(p) = d$. In particular, it must hold for $p \in P_k$ chosen with $d_k(p) = \delta_k$. In this case

$$n_{k+1} \geq c \delta_k^2 \left(\frac{n_k}{\delta_k}\right)^{1/2} = c \delta_k^{3/2} n_k^{1/2}. \quad (7)$$

Now we are able to provide an improved lower bound on the minimum degree, which will be used to improve the lower bound on n_k .

LEMMA 9. *There exists some positive constant $c_2 \in \mathbb{R}$ such that given $k \in \mathbb{N}$, $\epsilon \geq 0$, and any positive constant $c_1 \in \mathbb{R}$ with $\delta_k \geq c_1 n_k^\epsilon$, we have*

$$\delta_{k+1} \geq c_2 n_k^{\left(\frac{1+2\epsilon}{3}\right)}.$$

PROOF. Suppose that $\delta_k \geq c_1 n_k^\epsilon$ for some $k \in \mathbb{N}$, $\epsilon \geq 0$, and positive constant $c_1 \in \mathbb{R}$. Define $\alpha \in \mathbb{R}$ by

$$\alpha = \frac{1 + 2\epsilon}{3}.$$

Let $p \in P_k$ with $d_k(p) = d$. There are two cases: either $d < n_k^\alpha$ or $d \geq n_k^\alpha$. If $d < n_k^\alpha$, then by Lemma 8 we have

$$d_{k+1}(p) \geq c_0 \delta_k \left(\frac{n_k}{n_k^\alpha} \right)^{1/2} = c_0 \delta_k n_k^{\frac{1-\alpha}{2}}$$

for some positive constant $c_0 \in \mathbb{R}$. Since $\delta_k \geq c_1 n_k^\epsilon$ and $\alpha = (1 + 2\epsilon)/3$, we must have

$$\begin{aligned} d_{k+1}(p) &\geq c_0 \delta_k n_k^{\frac{1-\alpha}{2}} \\ &\geq c_0 c_1 n_k^\epsilon n_k^{\frac{1-\epsilon}{3}} \\ &= c_2 n_k^{\left(\frac{1+2\epsilon}{3}\right)}, \end{aligned}$$

where $c_2 = c_0 c_1$. If instead $d \geq n_k^\alpha$, then obviously we have

$$d_{k+1}(p) \geq d \geq n_k^\alpha \geq c_2 n_k^{\left(\frac{1+2\epsilon}{3}\right)},$$

where $c_2 \leq 1$. So, in both cases, we have the conclusion that

$$d_{k+1}(p) \geq c_2 n_k^{\left(\frac{1+2\epsilon}{3}\right)}$$

and this will hold true for any $p \in P_k$. Since the choice of $p \in P_k$ was arbitrary, we have

$$\delta_{k+1} \geq c_2 n_k^{\left(\frac{1+2\epsilon}{3}\right)},$$

as desired. This completes the proof. \square

2.4. NUMERICAL RESULTS

Now, we can obtain some numerical results from Lemma 9. Note first that if $\epsilon = 0$ in Lemma 9, then it is easy to find a positive constant $c_1 \in \mathbb{R}$ that satisfies the hypotheses of the theorem, namely $c = 3$. If we apply the lemma with these values, then we get a positive constant $c_2 \in \mathbb{R}$ such that

$$\delta_k \geq c_2 n_{k-1}^{1/3} \tag{8}$$

for all $k \geq 2$. Further recall that the trivial upper bound yields

$$n_{k-1} \geq (8n_k)^{1/4} \quad (9)$$

for all $k \geq 2$. Combining (8) and (9), we get that there exists a positive constant $c_3 \in \mathbb{R}$ such that

$$\delta_k \geq c_2 n_{k-1}^{1/3} \geq c_2 [(8n_k)^{1/4}]^{1/3} \geq c_3 n_k^{1/12}$$

for $k \geq 2$. Now, we can apply Lemma 9 with $\epsilon = \frac{1}{12}$ for any $k \geq 2$. Since

$$\frac{1 + 2(\frac{1}{12})}{3} = \frac{7}{18},$$

we get that there exists a positive constant $c_4 \in \mathbb{R}$ such that

$$\delta_k \geq c_4 n_{k-1}^{7/18} \quad (10)$$

for all $k \geq 2$. Now, we combine (10) with (9), to obtain that there exists a positive constant $c_5 \in \mathbb{R}$ such that

$$\delta_k \geq c_4 n_{k-1}^{7/18} \geq c_4 [(8n_k)^{1/4}]^{7/18} \geq c_5 n_k^{7/72}$$

for all $k \geq 2$.

This process can be iterated and the limiting value of $\epsilon > 0$ is found by setting

$$\epsilon = \frac{1 + 2\epsilon}{12}$$

which implies that $\epsilon = 0.1 - o(1)$. Now, using Lemma 9 ($\epsilon = 0.1 - o(1)$) with (7), we obtain

$$\begin{aligned} n_{k+1} &\geq c \delta_k^{3/2} n_k^{1/2} \\ &\geq c \left(c' n_{k-1}^{\frac{1+2(0.1-o(1))}{3}} \right)^{3/2} n_k^{1/2} \\ &\geq c'' n_{k-1}^{1.1-o(1)} \end{aligned} \quad (11)$$

for some positive constants $c, c', c'' \in \mathbb{R}$. Using (11) along with the trivial upper bound, we obtain the following theorem:

THEOREM 10. *There exist positive constants $c_1, c_2 \in \mathbb{R}$ such that*

$$c_1 4^{1.0488^k} \leq n_k \leq c_2 4^{4^k}$$

for all $k \in \mathbb{N}$.

PROOF. Note first that $n_1 = 4$ and $n_2 = 7$. From repeated use of (11) we get that there exist positive constants $a_1, a_2, a_3, a_4 \in \mathbb{R}$ such that

$$a_1 4^{(1.1-o(1))^k} \leq n_{2k+1} \leq a_2 4^{4^{2k+1}}$$

and

$$a_3 7^{(1.1-o(1))^{k-1}} \leq n_{2k} \leq a_4 4^{4^{2k}}.$$

Taking square roots, it follows that there exist positive constants $c_1, c_2 \in \mathbb{R}$ such that

$$c_1 4^{1.0488^k} \leq n_k \leq c_2 4^{4^k},$$

(since $\sqrt{1.1} > 1.0488$) as desired. □

Theorem 10 shows that the growth of n_k is indeed doubly-exponential, as the trivial upper bound suggests. However, a considerable gap still remains between the exponents. This gap can be narrowed slightly by using Theorem 10 to improve itself:

THEOREM 11. *There exist positive constants $c_1, c_2 \in \mathbb{R}$ such that*

$$c_1 4^{1.0639^k} \leq n_k \leq c_2 4^{4^k}$$

for all $k \in \mathbb{N}$.

PROOF. We can repeat the arguments from Theorem 10 to get that there exists a positive constant $c \in \mathbb{R}$ such that

$$n_{k+1} \geq cn_{k-1}^{0.6-o(1)}n_k^{1/2}$$

for $k \geq 2$ from (11). However, now we can use the lower bound in Theorem 10 to obtain

$$\begin{aligned} n_{k+1} &\geq cn_{k-1}^{0.6-o(1)}n_k^{1/2} \\ &\geq cn_{k-1}^{0.6-o(1)}\left(c'n_{k-1}^{\sqrt{1.1}}\right)^{1/2} \\ &= c''n_{k-1}^{0.6+0.5\sqrt{1.1}-o(1)} \\ &> c''n_{k-1}^{1.1244-o(1)} \end{aligned}$$

for some positive constants $c, c', c'' \in \mathbb{R}$. So we get a new lower bound of $n_k \geq c4^{1.0603^k}$ for some positive constant $c \in \mathbb{R}$ since $\sqrt{1.1244} > 1.0603$. Now we can repeat this argument over and over, using the current lower bound to improve itself. The limiting value for the lower bound can be found by solving the equation $\sqrt{0.5x+0.6} = x$, which has a positive root given by

$$x = \frac{5 + \sqrt{265}}{20} > 1.0639.$$

□

Theorem 11 gives the best known bounds for n_k . While we have no rigorous argument providing improvements of the lower bound, computational results and heuristic reasoning suggest that the actual growth rate of n_k is closer to the stated upper bound.

CHAPTER 3

CHARACTERIZATION OF IRP

3.1. PAPPUS AND DESARGUES

We will refer heavily to the following two theorems of projective geometry:

THEOREM 12. (*Pappus*) *If alternate vertices of a hexagon are collinear, then the three points of intersection of opposite edges are collinear.*

THEOREM 13. (*Desargues*) *In a projective space, two triangles are in perspective axially if and only if they are in perspective centrally.*

Define a projective plane to be *Pappian* if its points and lines satisfy Pappus's Theorem and *Desarguian* if its points and lines satisfy Desargues's Theorem. Since points and lines in the real projective plane satisfy Pappus's Theorem and IRP is

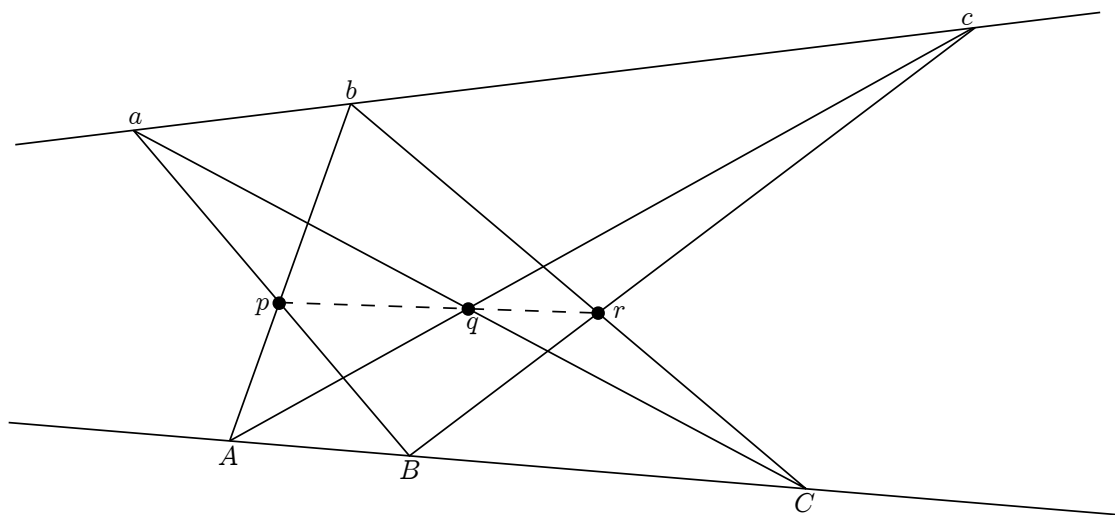


FIGURE 3.1. Pappus's Theorem

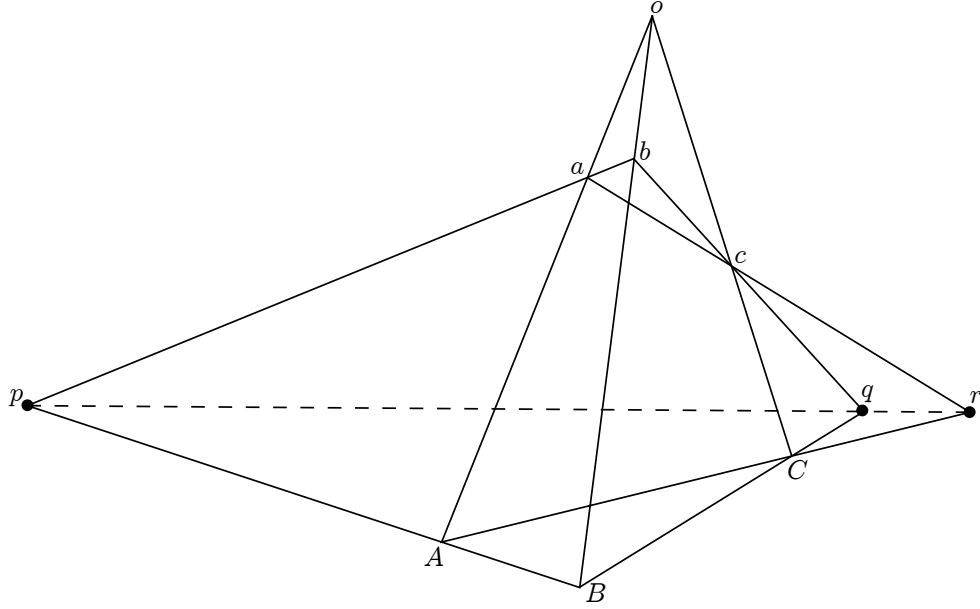


FIGURE 3.2. Desargues's Theorem

constructed in the real plane, then IRP is Pappian. It is well known that in any projective plane, Pappus's Theorem implies Desargues's Theorem. It follows that IRP is Desarguian and coordinatizable. It is also well known that the coordinate ring is a division ring and since Pappus's Theorem is equivalent to commutativity in the division ring, the coordinate ring must be a field, call it F .

PROPOSITION 14. F has characteristic 0.

PROOF. To see this, we must first define addition in F . Using homogeneous coordinates, let $X = (1 : 0 : 0)$, $Y = (0 : 1 : 0)$, $O = (0 : 0 : 1)$ and $U = (1 : 1 : 1)$. We define a collineation (an isomorphism from IRP to itself) fixing each point in the spans of X, Y, O , and U . Using this notation, the line XY represents the line at infinity. Finally, let $I = OU \cap XY$, which has homogeneous coordinates $I = (1 : 1 : 0)$. Given distinct elements $x, y \in F$ there exist two points $X_1 = (x : 0 : 1)$ and $Y_1 = (y : 0 : 1)$, both of which lie on OX . Note that the homogeneous coordinates of the point $W = X_1Y \cap OI$ are given by $(x : x : 1)$. Define $Z = Y_1I \cap WX$, which has homogeneous coordinates $(x + y : x : 1)$. Now, the point $YZ \cap OX$ has homogeneous coordinates given by $(x + y : 0 : 1)$.

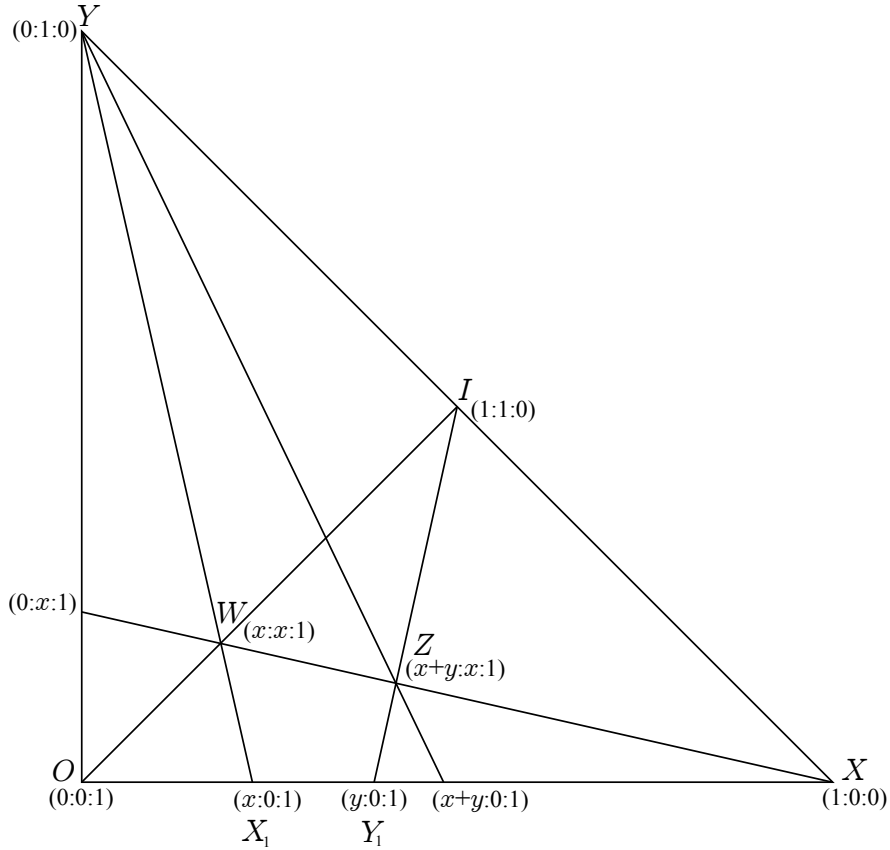


FIGURE 3.3. Addition

Now that addition has been defined, we aim to show that for any nonzero $x \in F$ and $n \in \mathbb{N}$, $nx = x + x + \cdots + x \neq 0$, i.e., the characteristic of F is zero. We begin with the point $X_1 = (x : 0 : 1)$ and we add it to itself to get $X_2 = (2x : 0 : 1)$. We add it again to get $X_3 = (3x : 0 : 1)$, and again to get $X_4 = (4x : 0 : 1)$, and so on. If we can show that each time we get a new nonzero first coordinate, then F must have characteristic zero. In the construction, X_1Y and WX will remain the same for each sum, where $W = (x : x : 1)$. If we let $Z_n = X_nI \cap WX$, then we have $((n+1)x : 0 : 1) = YZ_n \cap OX$. Suppose that $n \in \mathbb{N}$ is the smallest n such that $(n+1)x = 0$. We see that $YZ_n = OY$ and so Z_n lies on OY . By construction, $WX \cap OY = (0 : x : 1)$ and since $Z_n = X_nI \cap WX$ we have $X_nI \cap OY = (0 : x : 1)$. Note that the point I lies on the line X_nI and so X_nI must be the line joining the points I and $(0 : x : 1)$. But, $X_nI \cap OX = (nx : 0 : 1)$ by definition. The only way

that this can occur is in the event that $X_n I = OI$. It follows that $nx = 0$, contrary to the minimality assumption. \square

3.2. FREE \mathcal{F} -PLANE

Given any Pappian projective plane Π , based on the arguments above, we know that Π is coordinatizable. Let $(a_1 : a_2 : a_3)$, $(b_1 : b_2 : b_3)$, $(c_1 : c_2 : c_3)$, and $(d_1 : d_2 : d_3)$ be the homogeneous coordinates of four distinct points a, b, c, d in Π and define $\Gamma(a, b, c, d)$ to be the plane generated by a, b, c, d . The generation of $\Gamma(a, b, c, d)$ occurs one stage at a time, inducing an order to the creation of points and lines. The first stage consists of selecting the four points a, b, c, d and constructing a line between each pair of points. In a manner identical to that of IRP, each stage consists of the following steps:

- (1) Place a point at each intersection of a pair lines for which one does not already exist.
- (2) Construct a line through each of pair of points for which one does not already exist.

For both steps above, the order in which we add elements to $\Gamma(a, b, c, d)$ within the step is arbitrary and will not affect the resulting plane. Note that each point and line in $\Gamma(a, b, c, d)$ can be represented by homogeneous coordinates $(f : g : h)$, where each coordinate is a polynomial function of the 12 variables $\{a_i, b_i, c_i, d_i\}_{i=1}^4$. It is clear that the four initial points have coordinates that are already functions (monomials) of those twelve variables. To see how we will obtain future points and lines, consider two lines $(f_1 : g_1 : h_1)$ and $(f_2 : g_2 : h_2)$ in $\Gamma(a, b, c, d)$, where $f_j, g_j, h_j \in \mathbb{Z}[\{a_i, b_i, c_i, d_i\}_{i=1}^4]$ for $j = 1, 2$. These lines can be thought of as intersecting planes in \mathbb{R}^3 :

$$f_1x + g_1y + h_1z = 0 \quad \text{and} \quad f_2x + g_2y + h_2z = 0.$$

The intersection of these planes is a line parallel to the vector given by the cross product

$$\begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ f_1 & g_1 & h_1 \\ f_2 & g_2 & h_2 \end{vmatrix} = (g_1h_2 - g_2h_1 : f_2h_1 - f_1h_2 : f_1g_2 - f_2g_1),$$

which yields the homogeneous coordinates of the intersection point of the two lines in $\Gamma(a, b, c, d)$. This process can also be used to find the line that passes through two points in $\Gamma(a, b, c, d)$, because of duality.

Now, we define some notation. Given a point x and a line ℓ in a projective plane, let $x \sim \ell$ denote that x lies on ℓ .

DEFINITION 15. Let Π_1 and Π_2 be projective planes with point sets $P(\Pi_1)$ and $P(\Pi_2)$ and line sets $L(\Pi_1)$ and $L(\Pi_2)$, respectively. Suppose that φ is a map from Π_1 to Π_2 such that

$$\varphi : P(\Pi_1) \rightarrow P(\Pi_2) \quad \text{and} \quad \varphi : L(\Pi_1) \rightarrow L(\Pi_2).$$

We say that φ is an epimorphism if it is surjective and

$$x \sim \ell \implies \varphi(x) \sim \varphi(\ell).$$

In other words, an epimorphism is a surjection from one plane onto another that preserves incidences. Given any collection of points P , we say that P is a collection of *general* points if no three points of P are collinear.

DEFINITION 16. Let \mathcal{F} be a family of projective planes. We say that a projective plane Π is a *free \mathcal{F} -plane* if for every $\Pi_0 \in \mathcal{F}$, for any four general points $a, b, c, d \in \Pi_0$ there exists an epimorphism φ from Π to $\Gamma(a, b, c, d)$.

LEMMA 17. If \mathcal{F} consists of the set of Pappian planes, then IRP is a free \mathcal{F} -plane.

Note that if \mathcal{G} consists of the set of Pappian *and* Desarguanian planes, then a free \mathcal{G} -plane is a free \mathcal{F} -plane and vice versa because Pappus's Theorem implies Desargues's Theorem [8].

PROOF. Let Π be a Pappian plane and let $(a_1 : a_2 : a_3), (b_1 : b_2 : b_3), (c_1 : c_2 : c_3),$ and $(d_1 : d_2 : d_3)$ be the homogeneous coordinates of four points a, b, c, d in Π . Given the algebraically independent homogeneous coordinates of our four initial points in IRP, $(x_1 : x_2 : x_3), (x_4 : x_5 : x_6), (x_7 : x_8 : x_9),$ and $(x_{10} : x_{11} : x_{12}),$ each point (and line) attained in the IRP can be represented with three homogeneous coordinates, each of which is a polynomial with integer coefficients over the variables $\{x_i\}_{i=1}^{12}$. Suppose we have a point in IRP represented by $(f, g, h) \in \mathbb{Z}[x_1, x_2, \dots, x_{12}]^3$. The map given by

$$\varphi(f : g : h) = (f(a, b, c, d) : g(a, b, c, d) : h(a, b, c, d))$$

where $(a, b, c, d) = (a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3, d_1, d_2, d_3)$ will give the necessary epimorphism.

Since we are using homogeneous coordinates, we can use an induction argument to see that φ is a well-defined function. First, enumerate the points and lines of IRP in the order in which they appear and do the same for $\Gamma(a, b, c, d)$. Let $\{s_i\}_{i=1}^{\infty}$ be the sequence of points and lines from IRP and $\{t_i\}_{i=1}^{\infty}$ be the sequence of points and lines from $\Gamma(a, b, c, d)$. By the definition of φ , we have

$$\varphi(x_1 : x_2 : x_3) = (a_1 : a_2 : a_3)$$

$$\varphi(x_4 : x_5 : x_6) = (b_1 : b_2 : b_3)$$

$$\varphi(x_7 : x_8 : x_9) = (c_1 : c_2 : c_3)$$

$$\varphi(x_{10} : x_{11} : x_{12}) = (d_1 : d_2 : d_3)$$

Now, suppose that φ is well-defined for all points and lines of IRP up to s_j and without loss of generality suppose s_{j+1} is a point. The existence of s_{j+1} in IRP is due to at least two intersecting lines s_i and s_ℓ with homogeneous coordinates $(f_i : g_i : h_i)$

and $(f_\ell : g_\ell : h_\ell)$, respectively, where $i \leq j$ and $\ell \leq j$. Using the cross product, we get that the homogeneous coordinates of the point s_{j+1} are given by

$$(f_{j+1} : g_{j+1} : h_{j+1}) = (g_i h_\ell - g_\ell h_i : f_\ell h_i - f_i h_\ell : f_i g_\ell - f_\ell g_i).$$

Let $\alpha_k = (k_1 a, k_2 b, k_3 c, k_4 d)$ be another representation of $\alpha = (a, b, c, d)$ in the plane $\Gamma(a, b, c, d)$, where $\bar{k} = (k_1, k_2, k_3, k_4) \in \mathbb{R}^4$. In other words, the four points of α_k are the initial four points of α , each scaled by a different real number. By the induction hypothesis, we know that φ is well-defined for s_i and s_ℓ and so

$$(f_i(\alpha_k) : g_i(\alpha_k) : h_i(\alpha_k)) = C_i(\bar{k})(f_i(\alpha) : g_i(\alpha) : h_i(\alpha))$$

and

$$(f_\ell(\alpha_k) : g_\ell(\alpha_k) : h_\ell(\alpha_k)) = C_\ell(\bar{k})(f_\ell(\alpha) : g_\ell(\alpha) : h_\ell(\alpha)),$$

for some polynomial functions C_i and C_ℓ of \bar{k} . It follows that

$$(f_{j+1}(\alpha_k) : g_{j+1}(\alpha_k) : h_{j+1}(\alpha_k)) = C_i(\bar{k})C_\ell(\bar{k})(f_{j+1}(\alpha) : g_{j+1}(\alpha) : h_{j+1}(\alpha)),$$

because the terms of each coordinate of s_{j+1} have one polynomial from $\{f_i, g_i, h_i\}$ and one from $\{f_\ell, g_\ell, h_\ell\}$. Hence, φ is well-defined on IRP by the principal of mathematical induction.

Next, we need to show that φ preserves incidences. Suppose that φ preserves incidences up to s_j and without loss of generality suppose s_{j+1} is a point in IRP. The existence of s_{j+1} in IRP is due to at least two intersecting lines s_i and s_ℓ with homogeneous coordinates $(f_i : g_i : h_i)$ and $(f_\ell : g_\ell : h_\ell)$, respectively, where $i \leq j$ and $\ell \leq j$. Using the cross product, we obtain that

$$s_{j+1} = (g_i h_\ell - g_\ell h_i : f_\ell h_i - f_i h_\ell : f_i g_\ell - f_\ell g_i).$$

By definition,

$$\varphi(s_i) = (f_i(a, b, c, d) : g_i(a, b, c, d) : h_i(a, b, c, d))$$

and

$$\varphi(s_\ell) = (f_\ell(a, b, c, d) : g_\ell(a, b, c, d) : h_\ell(a, b, c, d)).$$

Applying the inductive hypothesis to the lines s_i and s_ℓ in IRP, we obtain that $\varphi(s_i)$ and $\varphi(s_\ell)$ are intersecting lines in $\Gamma(a, b, c, d)$. Using the cross product again, we see that the point of intersection of $\varphi(s_i)$ and $\varphi(s_\ell)$ in $\Gamma(a, b, c, d)$ is given by

$$(g_i(\alpha)h_\ell(\alpha) - g_\ell(\alpha)h_i(\alpha) : f_\ell(\alpha)h_i(\alpha) - f_i(\alpha)h_\ell(\alpha) : f_i(\alpha)g_\ell(\alpha) - f_\ell(\alpha)g_i(\alpha)),$$

where $\alpha = (a, b, c, d)$. From the definition of φ , it is clear that this point of intersection in $\Gamma(a, b, c, d)$ is precisely $\varphi(s_{j+1})$, which finishes the inductive proof.

Finally, we need to prove that φ is surjective. Now, suppose to the contrary that φ is not surjective. Then, we can choose some point t_k from $\Gamma(a, b, c, d)$ such that there does not exist $i \in \mathbb{N}$ with $\varphi(s_i) = t_k$. Further, suppose that k is the smallest such k . Note that there exists a pair of lines t_i and t_j that intersect at t_k (causing its existence), where $i < k$ and $j < k$. By the minimality of k , we know that there exist some $\alpha, \beta \in \mathbb{N}$ with $\varphi(s_\alpha) = t_i$ and $\varphi(s_\beta) = t_j$. Note that s_α and s_β are lines in IRP and so they must intersect at some point s_γ . Using the preservation of incidences by φ , since $s_\gamma \sim s_\alpha$ we know $\varphi(s_\gamma) \sim \varphi(s_\alpha) = t_i$ and since $s_\gamma \sim s_\beta$ then we know $\varphi(s_\gamma) \sim \varphi(s_\beta) = t_j$. Now $\varphi(s_\gamma)$ is a point in $\Gamma(a, b, c, d)$ that lies at the intersection of the lines t_i and t_j . But, the intersection of t_i and t_j is t_k . So, $\varphi(s_\gamma) = t_k$, a contradiction. Hence, φ is surjective, completing the proof. \square

3.3. REPRESENTATION BY COEFFICIENT VECTORS

Let P and L denote the limiting sets of points and lines of IRP, respectively, i.e.,

$$P = \lim_{k \rightarrow \infty} P_k \qquad L = \lim_{k \rightarrow \infty} L_k.$$

Suppose that we represent each point $p \in P$ by a triple of homogeneous coordinates $(p_1 : p_2 : p_3)$, where $p_i \in \mathbb{R}$ for $1 \leq i \leq 3$. Let $a = (x_1 : x_2 : x_3)$, $b = (x_4 : x_5 : x_6)$,

$c = (x_7 : x_8 : x_9)$ and $d = (x_{10} : x_{11} : x_{12})$ denote the initial four points of IRP, where $x_i \in \mathbb{R}$ for $1 \leq i \leq 12$. Now, if we treat each point as a vector in \mathbb{R}^3 , then the cross product of two points will yield a third vector in \mathbb{R}^3 whose coordinates are the homogeneous coordinates of the line passing through the two points. By duality, the cross product of two lines will yield their intersection point. This idea can be represented by considering the points to be vectors in \mathbb{R}^3 passing through the origin and the lines to be planes containing the origin. Two lines (planes) intersect at a unique point (vector) and two points (vectors) lie on a unique line (plane). Proceeding in this manner, we can generate IRP one stage at a time, with the first stage illustrated in the following tables:

Point	Coordinates
a	$(x_1 : x_2 : x_3)$
b	$(x_4 : x_5 : x_6)$
c	$(x_7 : x_8 : x_9)$
d	$(x_{10} : x_{11} : x_{12})$

Line	Coordinates
ab	$(x_2x_6 - x_3x_5 : x_3x_4 - x_1x_6 : x_1x_5 - x_2x_4)$
ac	$(x_2x_9 - x_3x_8 : x_3x_7 - x_1x_9 : x_1x_8 - x_2x_7)$
ad	$(x_2x_{12} - x_3x_{11} : x_3x_{10} - x_1x_{12} : x_1x_{11} - x_2x_{10})$
bc	$(x_5x_9 - x_6x_8 : x_6x_7 - x_4x_9 : x_4x_8 - x_5x_7)$
bd	$(x_5x_{12} - x_6x_{11} : x_6x_{10} - x_4x_{12} : x_4x_{11} - x_5x_{10})$
cd	$(x_8x_{12} - x_9x_{11} : x_9x_{10} - x_7x_{12} : x_7x_{11} - x_8x_{10})$

By virtue of the cross product, it is clear from this initialization that each point in IRP is representable by homogeneous coordinates, each of which is a polynomial in the 12 variables $\{x_i\}_{i=1}^{12}$. In other words, $P \subset \mathbb{Z}[x_1, x_2, \dots, x_{12}]^3$, and from this it can be seen that n_k is well-defined under our hypotheses. It is easy to show that if $u, v \in \Gamma^3$, where Γ denotes the set of all polynomials in the twelve variables x_1, x_2, \dots, x_{12} over \mathbb{R} , $w = u \times v$, and

$$d_u = \max_{1 \leq i \leq 3} \deg(u_i) \qquad d_v = \max_{1 \leq i \leq 3} \deg(v_i),$$

then

$$d_w = \max_{1 \leq i \leq 3} \deg(w_i) \leq d_u d_v.$$

Further, it seems that this maximum would be realized in most cases, because otherwise one of two events must occur: The first involves a cancellation of polynomial terms to reduce the degree and the second occurs when each coordinate possesses a common factor, which can be factored out due to the nature of homogeneous coordinates. It turns out that the latter occurs, and it occurs with regularity under this setting. Before we discuss this phenomenon, we require some notation to simplify these polynomials:

DEFINITION 18. *We will refer to the following degree 4 monomials as terms of type (i, j) for $1 \leq i \leq 12$, $1 \leq j \leq 3$:*

<i>Type</i>	<i>Monomial</i>
(1, 1)	$x_1 x_4 x_8 x_{12}$
(2, 1)	$x_1 x_4 x_9 x_{11}$
(3, 1)	$x_1 x_5 x_7 x_{12}$
(4, 1)	$x_1 x_6 x_7 x_{11}$
(5, 1)	$x_1 x_5 x_9 x_{10}$
(6, 1)	$x_1 x_6 x_8 x_{10}$
(7, 1)	$x_2 x_4 x_7 x_{12}$
(8, 1)	$x_3 x_4 x_7 x_{11}$
(9, 1)	$x_2 x_4 x_9 x_{10}$
(10, 1)	$x_3 x_4 x_8 x_{10}$
(11, 1)	$x_2 x_6 x_7 x_{10}$
(12, 1)	$x_3 x_5 x_7 x_{10}$

<i>Type</i>	<i>Monomial</i>
(1, 2)	$x_2 x_5 x_7 x_{12}$
(2, 2)	$x_2 x_5 x_9 x_{10}$
(3, 2)	$x_2 x_4 x_8 x_{12}$
(4, 2)	$x_2 x_6 x_8 x_{10}$
(5, 2)	$x_2 x_4 x_9 x_{11}$
(6, 2)	$x_2 x_6 x_7 x_{11}$
(7, 2)	$x_1 x_5 x_8 x_{12}$
(8, 2)	$x_3 x_5 x_8 x_{10}$
(9, 2)	$x_1 x_5 x_9 x_{11}$
(10, 2)	$x_3 x_5 x_7 x_{11}$
(11, 2)	$x_1 x_6 x_8 x_{11}$
(12, 2)	$x_3 x_4 x_8 x_{11}$

<i>Type</i>	<i>Monomial</i>
(1, 3)	$x_3 x_6 x_7 x_{11}$
(2, 3)	$x_3 x_6 x_8 x_{10}$
(3, 3)	$x_3 x_4 x_9 x_{11}$
(4, 3)	$x_3 x_5 x_9 x_{10}$
(5, 3)	$x_3 x_4 x_8 x_{12}$
(6, 3)	$x_3 x_5 x_7 x_{12}$
(7, 3)	$x_1 x_6 x_9 x_{11}$
(8, 3)	$x_2 x_6 x_9 x_{10}$
(9, 3)	$x_1 x_6 x_8 x_{12}$
(10, 3)	$x_2 x_6 x_7 x_{12}$
(11, 3)	$x_1 x_5 x_9 x_{12}$
(12, 3)	$x_2 x_4 x_9 x_{12}$

THEOREM 19. Suppose $p = (p_1 : p_2 : p_3) \in P$, where $p_1, p_2, p_3 \in \mathbb{Z}[x_1, x_2, \dots, x_{12}]$.

(1) For $1 \leq j \leq 3$, each term from p_j is of type (i, j) for some $1 \leq i \leq 12$.

Moreover, for $1 \leq i \leq 12$ and $c \in \mathbb{Z}$, the following are equivalent:

(a) the monomial of type $(i, 1)$ appears as a term in the polynomial p_1 with the coefficient c .

(b) the monomial of type $(i, 2)$ appears as a term in the polynomial p_2 with the coefficient $-c$.

(c) the monomial of type $(i, 3)$ appears as a term in the polynomial p_3 with the coefficient c .

It follows that p is determined by its first coordinate. Hence, we can uniquely represent p with a “coefficient vector” $v \in \mathbb{Z}^{12}$, where v_i contains the coefficient of the term of type $(i, 1)$ present in the first coordinate of p .

(2) Suppose that $v \in \mathbb{Z}^{12}$ is the coefficient vector for a point $p \in P$. For each $1 \leq i \leq 6$, we have $v_{2i} = -v_{2i-1}$ and the following relations hold:

$$v_7 = -v_1 - v_3 \qquad v_9 = v_1 - v_5 \qquad v_{11} = v_3 + v_5.$$

Consequently, p can be uniquely represented by the reduced coefficient vector $(v_1, v_3, v_5) \in \mathbb{Z}^3$.

PROOF. We proceed by induction on the stage k . Recall that for any given point $(x : y : z)$ in homogeneous coordinates and for any $\rho \neq 0$, the multiple $\rho(x : y : z)$ is an equivalent representation. Note that for $k = 1$, each of the points $a, b, c, d \in P_1$ can be written in the desired form by performing the following multiplications:

$$a = (x_1 : x_2 : x_3)(x_4x_8x_{12} - x_4x_9x_{11} - x_5x_7x_{12} + x_6x_7x_{11} + x_5x_9x_{10} - x_6x_8x_{10})$$

$$b = (x_4 : x_5 : x_6)(x_1x_8x_{12} - x_1x_9x_{11} - x_2x_7x_{12} + x_3x_7x_{11} + x_2x_9x_{10} - x_3x_8x_{10})$$

$$c = (x_7 : x_8 : x_9)(x_1x_5x_{12} - x_1x_6x_{11} - x_2x_4x_{12} + x_3x_4x_{11} + x_2x_6x_{10} - x_3x_5x_{10})$$

$$d = (x_{10} : x_{11} : x_{12})(x_1x_5x_9 - x_1x_6x_8 - x_2x_4x_9 + x_3x_4x_8 + x_2x_6x_7 - x_3x_5x_7)$$

Let $m_{(i,j)}$ be the monomial of type (i, j) for $1 \leq i \leq 12$ and $1 \leq j \leq 3$. We can see by inspection that

$$a_1 = m_{(1,1)} - m_{(2,1)} - m_{(3,1)} + m_{(4,1)} + m_{(5,1)} - m_{(6,1)}$$

$$a_2 = -m_{(1,2)} + m_{(2,2)} + m_{(3,2)} - m_{(4,2)} - m_{(5,2)} + m_{(6,2)}$$

$$a_3 = m_{(1,3)} - m_{(2,3)} - m_{(3,3)} + m_{(4,3)} + m_{(5,3)} - m_{(6,3)}$$

$$b_1 = m_{(1,1)} - m_{(2,1)} - m_{(7,1)} + m_{(8,1)} + m_{(9,1)} - m_{(10,1)}$$

$$b_2 = -m_{(1,2)} + m_{(2,2)} + m_{(7,2)} - m_{(8,2)} - m_{(9,2)} + m_{(10,2)}$$

$$b_3 = m_{(1,3)} - m_{(2,3)} - m_{(7,3)} + m_{(8,3)} + m_{(9,3)} - m_{(10,3)}$$

$$c_1 = m_{(3,1)} - m_{(4,1)} - m_{(7,1)} + m_{(8,1)} + m_{(11,1)} - m_{(12,1)}$$

$$c_2 = -m_{(3,2)} + m_{(4,2)} + m_{(7,2)} - m_{(8,2)} - m_{(11,2)} + m_{(12,2)}$$

$$c_3 = m_{(3,3)} - m_{(4,3)} - m_{(7,3)} + m_{(8,3)} + m_{(11,3)} - m_{(12,3)}$$

$$d_1 = m_{(5,1)} - m_{(6,1)} - m_{(9,1)} + m_{(10,1)} + m_{(11,1)} - m_{(12,1)}$$

$$d_2 = -m_{(5,2)} + m_{(6,2)} + m_{(9,2)} - m_{(10,2)} - m_{(11,2)} + m_{(12,2)}$$

$$d_3 = m_{(5,3)} - m_{(6,3)} - m_{(9,3)} + m_{(10,3)} + m_{(11,3)} - m_{(12,3)}.$$

Also, it is straightforward to show that the three equivalent conditions stated in the first part of the theorem hold for each of the four initial points. From this we get the coefficient vectors for $a, b, c,$ and d :

$$a = (1, -1, -1, 1, 1, -1, 0, 0, 0, 0, 0, 0),$$

$$b = (1, -1, 0, 0, 0, 0, -1, 1, 1, -1, 0, 0),$$

$$c = (0, 0, 1, -1, 0, 0, -1, 1, 0, 0, 1, -1),$$

and

$$d = (0, 0, 0, 0, 1, -1, 0, 0, -1, 1, 1, -1).$$

It is clear that each of these four vectors from \mathbb{Z}^{12} satisfy the linear relations in the second part of the theorem. Thus, we can write the reduced coefficient vectors from \mathbb{Z}^3 :

$$a = (1, -1, 1) \quad b = (1, 0, 0) \quad c = (0, 1, 0) \quad d = (0, 0, 1)$$

which concludes the proof of the base case $k = 1$.

Note that the existence of a point $p \in P_{k+1}$ is due to the existence of four points $\alpha, \beta, \gamma, \delta \in P_k$. In particular, given a point $p \in P_{k+1}$ there exist points $\alpha, \beta, \gamma, \delta \in P_k$ such that $p = (\alpha \times \beta) \times (\gamma \times \delta)$. Suppose that $\alpha, \beta, \gamma, \delta \in P_k$ each satisfy the conditions of the theorem and with $p = (\alpha \times \beta) \times (\gamma \times \delta)$. Let

$$\alpha = (\alpha_1, \alpha_2, \alpha_3) \quad \beta = (\beta_1, \beta_2, \beta_3) \quad \gamma = (\gamma_1, \gamma_2, \gamma_3) \quad \delta = (\delta_1, \delta_2, \delta_3)$$

be the reduced coefficient vectors for these points. Using SAGE to perform the computations (See Appendix C), we see that p satisfies the conditions of the theorem with the reduced coefficient vector $p = (p_1, p_2, p_3)$, where

$$\begin{aligned} p_1 = & -\alpha_3\beta_1\gamma_2\delta_1 + \alpha_1\beta_3\gamma_2\delta_1 + \alpha_2\beta_1\gamma_3\delta_1 - \alpha_1\beta_2\gamma_3\delta_1 \\ & + \alpha_3\beta_1\gamma_1\delta_2 - \alpha_1\beta_3\gamma_1\delta_2 - \alpha_2\beta_1\gamma_1\delta_3 + \alpha_1\beta_2\gamma_1\delta_3, \end{aligned}$$

$$\begin{aligned} p_2 = & -\alpha_3\beta_2\gamma_2\delta_1 + \alpha_2\beta_3\gamma_2\delta_1 + \alpha_3\beta_2\gamma_1\delta_2 - \alpha_2\beta_3\gamma_1\delta_2 \\ & + \alpha_2\beta_1\gamma_3\delta_2 - \alpha_1\beta_2\gamma_3\delta_2 - \alpha_2\beta_1\gamma_2\delta_3 + \alpha_1\beta_2\gamma_2\delta_3, \end{aligned}$$

$$\begin{aligned} p_3 = & -\alpha_3\beta_2\gamma_3\delta_1 + \alpha_2\beta_3\gamma_3\delta_1 + \alpha_3\beta_1\gamma_3\delta_2 - \alpha_1\beta_3\gamma_3\delta_2 \\ & + \alpha_3\beta_2\gamma_1\delta_3 - \alpha_2\beta_3\gamma_1\delta_3 - \alpha_3\beta_1\gamma_2\delta_3 + \alpha_1\beta_3\gamma_2\delta_3. \end{aligned}$$

This concludes the induction proof. □

CHAPTER 4

FREE PROJECTIVE PLANE

In the free projective plane, we can obtain trivial upper bounds for the number of points and lines at the end of stage k using the same arguments that we used for IRP. Let η_k and μ_k denote the number of points and lines, respectively, at the end of stage k in the free projective plane. It follows that:

$$\eta_{k+1} \leq \binom{\mu_k}{2} < \frac{\mu_k^2}{2}$$

and

$$\mu_k \leq \binom{\eta_k}{2} < \frac{\eta_k^2}{2}.$$

We conjecture that the lower bound is actually quite close to this upper bound. Suppose that

$$\mu_{k-1} \leq \sqrt{2\eta_k}(1 + \alpha_k)$$

and

$$\eta_k \leq \sqrt{2\mu_k}(1 + \beta_k)$$

for some $k \in \mathbb{N}$, where $\alpha_k > 0$ and $\beta_k > 0$. Combining the first bound with the second, we get

$$\begin{aligned} \mu_{k-1} &\leq \sqrt{2\eta_k}(1 + \alpha_k) \\ &\leq \sqrt{2\sqrt{2\mu_k}(1 + \beta_k)}(1 + \alpha_k) \\ &= 2^{3/4}\mu_k^{1/4}(1 + \beta_k)^{1/2}(1 + \alpha_k). \end{aligned}$$

Now, in the free projective plane, the only time we do not create a line between a pair of distinct points is in the event that the line already exists. Hence, when we estimate η_{k+1} by $\binom{\mu_k}{2}$, the overcount is solely due to the degree of points at the end of stage k (this is identical to the overcount discussed in Section 2.2). For instance, suppose that $p \in P_k$ is such that $d_k(p) = 3$, i.e., there are three distinct lines passing through p , call them ℓ_1, ℓ_2 , and ℓ_3 . Using $\binom{\mu_k}{2}$ to count the points in stage $k+1$, we will count p three times: once for $\ell_1 \cap \ell_2$, once for $\ell_1 \cap \ell_3$, and once for $\ell_2 \cap \ell_3$. Obviously, p is just a single point, and so it has been overcounted twice. In general, if p is a point in stage k , it will be counted $\binom{d_{k+1}(p)}{2}$ times, where $d_{k+1}(p)$ denotes the number of lines that pass through it at the end of stage k . It follows that

$$\begin{aligned}\eta_{k+1} &= \binom{\mu_k}{2} - \sum_{j=1}^{\eta_k} \left[\binom{d_{k+1}(v_j)}{2} - 1 \right] \\ &= \binom{\mu_k}{2} + \eta_k - \sum_{j=1}^{\eta_k} \left[\binom{d_{k+1}(v_j)}{2} \right]\end{aligned}$$

Since a point can have at most η_k lines passing through it at the end of stage k , then

$$\max_{p \in P_k} d_{k+1}(p) = \eta_k.$$

Hence,

$$\eta_{k+1} \geq \binom{\mu_k}{2} + \eta_k - \eta_k \binom{\eta_k}{2}.$$

Note that

$$\mu_k \leq \binom{\eta_k}{2} = \frac{\eta_k(\eta_k - 1)}{2} < \eta_k(\eta_k + 2)$$

follows from the trivial upper bound on μ_k . From this it is straightforward to verify that

$$\eta_{k+1} \geq \frac{\mu_k^2}{2} - \frac{\eta_k^3}{2}$$

and since $\eta_k < \mu_{k-1}^2/2$ we get

$$\eta_{k+1} \geq \frac{\mu_k^2}{2} - \frac{\mu_{k-1}^6}{16}.$$

At this point, we can apply the inductive hypothesis to obtain that

$$\begin{aligned}\eta_{k+1} &\geq \frac{\mu_k^2}{2} - \frac{(2^{3/4}\mu_k^{1/4}(1+\beta_k)^{1/2}(1+\alpha_k))^6}{16} \\ &= \frac{\mu_k^2}{2} - \sqrt{2}\mu_k^{3/2}(1+\beta_k)^3(1+\alpha_k)^6 \\ &= \frac{\mu_k^2}{2} \left(1 - \frac{2^{3/2}(1+\beta_k)^3(1+\alpha_k)^6}{\mu_k^{1/2}}\right).\end{aligned}$$

Now, define

$$a_{k+1} = \frac{2^{3/2}(1+\beta_k)^3(1+\alpha_k)^6}{\mu_k^{1/2}}$$

and note that

$$\mu_k \leq \frac{\sqrt{2\eta_{k+1}}}{\sqrt{1-a_{k+1}}}. \quad (12)$$

We use an analogous argument for μ_{k+1} to obtain that

$$\eta_{k+1} \leq \frac{\sqrt{2\mu_{k+1}}}{\sqrt{1-b_{k+1}}},$$

where

$$b_{k+1} = \frac{2^{3/2}(1+\alpha_{k+1})^3(1+\beta_k)^6}{\eta_k^{1/2}}.$$

Using a combinatorial algorithm written in java (see Appendix D), we are able to determine the number of points and lines through the first four stages. The results are listed in Table 4.1:

TABLE 4.1. Free Projective Plane Sequence

k	η_k	μ_k
1	4	6
2	7	9
3	13	33
4	295	37,266

We can use these results to determine α_i and β_i for the first few values of i . First note that α_1 does not make sense because μ_0 is undefined. By the definitions above,

$$\eta_1 \leq \sqrt{2\mu_1}(1+\beta_1) \iff 4 \leq \sqrt{2 \cdot 6}(1+\beta_1) \implies \beta_1 \geq \frac{2}{\sqrt{3}} - 1. \quad (13)$$

Since we would like β_1 to be as small as possible while still satisfying (13), we choose $\beta_1 = 0.1548$. Using analogous arguments, we obtain the results listed in Table 4.2 below:

TABLE 4.2. Values for α_k and β_k

k	α_k	β_k
2	0.6036	0.6500
3	0.7651	0.6002
4	0.3586	0.0806

We use the lower bound on η_5 from above to get

$$\begin{aligned}
\eta_5 &\geq \frac{\mu_4^2}{2} \left(1 - \frac{2^{3/2}(1 + \beta_4)^3(1 + \alpha_4)^6}{\mu_4^{1/2}} \right) \\
&= \frac{37,266^2}{2} \left(1 - \frac{2^{3/2}(1.0806)^3(1.3586)^6}{37,266^{1/2}} \right) \\
&> 613,648,536 \\
&> 0.4418(37,266)^2 \\
&= 0.4418\mu_4^2
\end{aligned}$$

It follows that

$$\mu_4 < \frac{\sqrt{2\eta_5}}{\sqrt{0.8836}} < \sqrt{2\eta_5}(1.0639) \implies \alpha_5 = 0.0639.$$

Using a similar argument with μ_5 and the trivial upper bound $\eta_5 \leq (\mu_4^2)/2$ we get

$$\begin{aligned}
\mu_5 &\geq \frac{\eta_5^2}{2} \left(1 - \frac{2^{3/2}(1 + \alpha_5)^3(1 + \beta_4)^6}{\eta_5^{1/2}} \right) \\
&> \frac{613,648,536^2}{2} \left(1 - \frac{2^{3/2}(1.0639)^3(1.0806)^6}{613,648,536^{1/2}} \right) \\
&> 0.4998\eta_5^2
\end{aligned}$$

It follows that

$$\mu_5 < \frac{\sqrt{2\mu_5}}{\sqrt{0.9996}} < \sqrt{2\mu_5}(1.0003) \implies \beta_5 = 0.0003.$$

Suppose that we know α_k and β_k with $k \geq 5$. From (12), we know that

$$\mu_k \leq \sqrt{2\eta_{k+1}} \cdot \frac{1}{\sqrt{1 - \alpha_{k+1}}}.$$

We would like to have

$$\mu_k \leq \sqrt{2\eta_{k+1}}(1 + \alpha_{k+1}),$$

therefore we choose α_{k+1} so that

$$1 + \alpha_{k+1} = \frac{1}{\sqrt{1 - \alpha_{k+1}}}.$$

Assuming that $\alpha_k < 0.1$ and $\beta_k < 0.1$, we get that

$$\alpha_{k+1} = \frac{1}{\sqrt{1 - \alpha_{k+1}}} - 1 < \frac{1}{1 - \frac{2^{3/2}(1.1)^9}{\sqrt{\mu_k}}} - 1 = f(\mu_k).$$

A similar argument will show that

$$\beta_{k+1} = \frac{1}{\sqrt{1 - \beta_{k+1}}} - 1 < \frac{1}{1 - \frac{2^{3/2}(1.1)^9}{\sqrt{\eta_{k+1}}}} - 1 = g(\eta_{k+1}).$$

Since $\eta_5 > 613,648,536$, $\mu_5 > 1.88 \cdot 10^{17}$, and both $\{\eta_k\}_{k=1}^\infty$ and $\{\mu_k\}_{k=1}^\infty$ are nondecreasing, we know that $\beta_{k+1} < \alpha_{k+1} < 1.6 \cdot 10^{-8} < 0.1$. Further, since $\{\eta_k\}_{k=1}^\infty$ and $\{\mu_k\}_{k=1}^\infty$ both grow doubly-exponentially, then $f(\mu_k)$ and $g(\eta_{k+1})$ both tend to zero at a doubly-exponential rate. It follows that α_k and β_k tend to zero at a doubly-exponential rate. From this it follows that

$$\eta_{k+1} \rightarrow \frac{\mu_k^2}{2} \quad \text{and} \quad \mu_k \rightarrow \frac{\eta_k^2}{2}$$

as $k \rightarrow \infty$.

CHAPTER 5

REMAINING QUESTIONS AND FURTHER WORK

5.1. COMPUTATIONAL RESULTS

An obvious modern-day approach to this problem is to write computer software that simulates this iterative process. We tried this with several different programs written in SAGE, java, and Mathematica, each of which outputs the sequence $n_1, m_1, n_2, m_2, \dots$ as far as it can go. The SAGE code is provided (Appendices A and B) for two such programs. Thankfully, all of the results agreed and the most efficient program (Appendix B) was able to calculate the sequence up to $n_5 = 719,725$ in about two hours on a 2.0 GHz processor with 4.0 GB of RAM. The following table illustrates the results:

TABLE 5.1. Sequence for IRP

k	n_k	m_k
1	4	6
2	7	9
3	13	25
4	97	1741
5	719,725	?

Despite being a short sequence, we can analyze it quite a bit. One interesting feature of this sequence is that for $2 \leq k \leq 4$, the quantity $n_k - n_{k-1}$ divides the quantity $n_{k+1} - n_k$. In other words, the number of points added in stage k divides the number of points added in stage $k + 1$. The natural question is whether this feature continues throughout the infinite sequence. Also, note that the sequence of ratios $\frac{\log n_{k+1}}{\log n_k}$ (which

tracks the exponent in the growth rate of n_k) is given by:

$$\begin{aligned}\frac{\log n_2}{\log n_1} &\approx 1.4037 \\ \frac{\log n_3}{\log n_2} &\approx 1.3181 \\ \frac{\log n_4}{\log n_3} &\approx 1.7835 \\ \frac{\log n_5}{\log n_4} &\approx 2.9481\end{aligned}$$

An important question is whether this sequence of logarithm ratios is monotone (after the first element). If so, then we could immediately increase the lower bound from that which is given in Theorem 11 to $n_k \geq c4^{2.9481^k}$. Based on these results (as well as all of the results in this paper), we pose the following conjecture:

CONJECTURE 20. *There exist positive constants $c_1, c_2 \in \mathbb{R}$ such that*

$$c_1 4^{4-o(1)^k} \leq n_k \leq c_2 4^{4^k}$$

for all $k \in \mathbb{N}$

Although there is no proof of any bounds better than those stated in this paper, it seems that this sequence grows much closer to the least upper bound than to the greatest lower bound. Also, it seems unlikely that a computer will be able to complete an exhaustive search for n_6 (or even m_5) any time soon. The reason is that in order to find n_6 the computer needs to examine $\binom{719,725}{4} > 10^{22}$ different 4-sets of points for membership. So, it seems that any further numbers of the sequence will require a theoretical argument or a much faster machine.

5.2. RANK OF IRP

Note that Lemma 17 states that IRP is a free \mathcal{F} -plane (for \mathcal{F} consisting of all Pappian planes), but says nothing about it being unique. In order to further characterize IRP, we need some terms:

DEFINITION 21. Let Π be any plane with points $P(\Pi)$ and lines $L(\Pi)$. We say that $S \subset P(\Pi)$ is independent if for each $T \subsetneq S$ and each $x \in S \setminus T$ we have $x \notin \Gamma(T)$.

DEFINITION 22. We say that a set of points $S \subset P(\Pi)$ is a generating set for the plane Π if $\Gamma(S) = \Pi$. If S is an independent generating set for Π , then we say that S forms a basis for Π .

DEFINITION 23. Define the rank of Π , denoted $\text{rank}(\Pi)$, to be the cardinality of the largest basis for Π .

CONJECTURE 24. $\text{rank}(\text{IRP}) = 4$

If $S \subset P(\Pi)$ is a set of points in a plane Π with $|S| \leq 3$, then $P(\Gamma(S)) = S$, i.e., the iterative process is stable. Hence, (assuming Π has more than three points) S cannot be a generating set and thus cannot form a basis for Π . For this reason, it is clear that $\text{rank}(\text{IRP}) \geq 4$ because any set of four general points is independent. So, what remains is to show that $\text{rank}(\text{IRP}) \leq 4$, i.e., that there does not exist a set of five points that forms a basis for IRP . It is seen by inspection that in IRP any set of four general points from stage 2 forms a basis for IRP . Using SAGE, we proved that each set of four general points from stage 3 forms a basis for IRP . Note that in order to prove that a given set of four or more independent points $S \subset P(\Pi)$ forms a basis for $\Pi = \Gamma(\{a, b, c, d\})$ it suffices to show that $\{a, b, c, d\} \subset \Gamma(S)$. In fact, it suffices to show even less:

PROPOSITION 25. Each set of four general points from $\Pi = \Gamma(\{a, b, c, d\})$ forms a basis for Π if and only if for each set $P = \{p_1, p_2, p_3, p_4\}$ of four general points from Π (with the exception of $\{a, b, c, d\}$), there exists a point $x \in \Gamma(P) \cap \Pi$ such that

$$\sum_{p \in P'} \text{stage}(p) < \sum_{p \in P} \text{stage}(p),$$

where $P' = (P \setminus \{p_i\}) \cup \{x\}$ for some $1 \leq i \leq 4$ and P' is a general set.

In other words, P forms a basis for Π if and only if P always generates some point x that is older than its youngest point and the new set is a general set. Before we prove this proposition, a lemma is needed:

LEMMA 26. *Given disjoint sets of points $P = \{p_1, p_2, p_3, p_4\}$ and $Q = \{q_1, q_2, q_3, q_4\}$, where P is a general set, there exist $i, j \in \{1, 2, 3, 4\}$ such that $(P \setminus \{p_i\}) \cup \{q_j\}$ is a general set.*

PROOF. Let $P = \{p_1, p_2, p_3, p_4\}$ and $Q = \{q_1, q_2, q_3, q_4\}$ be disjoint sets of points such that P is a general set. Note that, for $1 \leq i \leq 4$, $P_i = P \setminus \{p_i\}$ is a set of three non-collinear points. Define T_i to be the extended triangle formed by the points of P_i , where the line segments that form the edges of the triangle are extended infinitely in both directions. It follows that

$$\bigcap_{1 \leq i \leq 4} T_i = \{p_1 p_2 \cap p_3 p_4, p_1 p_3 \cap p_2 p_4, p_1 p_4 \cap p_2 p_3\} = \{p_5, p_6, p_7\},$$

where p_5, p_6 , and p_7 are the three points generated by P in the second stage of the iterative process. Suppose that $q_j \in Q \setminus \{p_5, p_6, p_7\}$, where $1 \leq j \leq 4$. Then there exists an extended triangle T_i such that q_j does not lie on T_i . It follows that $P_i \cup \{q_j\}$ is a general set. \square

Now, we can prove Proposition 25:

PROOF. To prove necessity, suppose that each set of four general points forms a basis for Π . Let $P = \{p_1, p_2, p_3, p_4\}$ be a set of four general points from Π , where $P \neq \{a, b, c, d\}$. Since P is assumed to be a basis for Π , we know that $\{a, b, c, d\} \subset \Gamma(P)$. Suppose first that $P \cap \{a, b, c, d\} = \emptyset$. Applying Lemma 26 with $P = P$ and $Q = \{a, b, c, d\}$, we obtain some $x \in \{a, b, c, d\}$ and some $1 \leq i \leq 4$ such that $P' = (P \setminus \{p_i\}) \cup \{x\}$ is a general set. Since a, b, c, d are the only points from stage

1, we know that $\text{stage}(x) < \text{stage}(p_i)$ and so

$$\sum_{p \in P'} \text{stage}(p) < \sum_{p \in P} \text{stage}(p).$$

In the event that $P \cap \{a, b, c, d\} \neq \emptyset$, we simply employ a variant of the argument in Lemma 26 to obtain the desired result. This completes the proof of necessity.

To prove sufficiency, suppose that $P_0 = \{p_1, p_2, p_3, p_4\}$ is a set of four general points from $\Pi = \Gamma(\{a, b, c, d\})$ and let $x \in \Gamma(P_0) \cap \Pi$ be such that

$$\sum_{p \in P_1} \text{stage}(p) < \sum_{p \in P_0} \text{stage}(p),$$

where $P_1 = (P \setminus \{p_i\}) \cup \{x\}$ for some $1 \leq i \leq 4$ and P_1 is a general set. It suffices to show that P_0 forms a basis for Π . Note that since $x \in \Gamma(P_0)$ we have $\Gamma(P_1) \subseteq \Gamma(P_0)$. But, since P_1 is a set of four general points from Π , we can apply the hypothesis to P_1 to obtain a new set P_2 of four general points from Π such that $\Gamma(P_2) \subseteq \Gamma(P_1)$ and

$$\sum_{p \in P_2} \text{stage}(p) < \sum_{p \in P_1} \text{stage}(p).$$

In general, we can create a sequence of sets of points $\{P_i\}_{i=0}^k$ such that if $i < j$ then $\Gamma(P_j) \subseteq \Gamma(P_i)$ and

$$\sum_{p \in P_j} \text{stage}(p) < \sum_{p \in P_i} \text{stage}(p).$$

This sequence must terminate with the set $P_k = \{a, b, c, d\}$ for some $k \in \mathbb{N}$ because

$$\sum_{p \in P_k} \text{stage}(p) = 4$$

is the minimum of such sums. Since $0 < k$ we get that $\Pi = \Gamma(\{a, b, c, d\}) \subseteq \Gamma(P_0)$ and since P_0 are four general points from Π we get that $\Gamma(P_0) \subseteq \Pi$. Hence, $\Gamma(P_0) = \Pi$, i.e., P_0 forms a basis for Π . \square

As mentioned above, it is still unclear whether each set of four general points forms a basis for IRP. Using SAGE, we found that there exist several sets of four

general points $P \subset P_4 \setminus P_3$ such that $\Gamma_3(P) \cap P_3 = \emptyset$. Of course this is not proof that these sets fail to be bases; however, it is certainly suggestive that it is possible.

5.3. IRP_n

It is a natural question to ask what happens when we begin this iterative process with more than four general points? Let IRP_4 denote IRP constructed with four initial points, IRP_5 denote IRP constructed with five initial points, and in general let IRP_n denote IRP constructed with n initial points. It is still unclear how such planes behave; however, using SAGE we were able to simulate the first two stages to get a short sequence for IRP_5 and IRP_6 . These results are illustrated in Table 5.2:

TABLE 5.2. IRP_5 and IRP_6

IRP_5			IRP_6		
k	n_k	m_k	k	n_k	m_k
1	5	10	1	6	15
2	20	100	2	51	870

It seems likely that there exists some reduction (similar to that of Theorem 19 in Section 3.3) that will allow us to represent each point in IRP_n uniquely by a vector in \mathbb{Z}^3 ; however, since we have not spent much time analyzing these cases, we have not found such a reduction.

5.4. COUNTING PAPPUS CONFIGURATIONS

In this section, we explore the number of occurrences of Pappus's Theorem in stage k of this iterative process. Let $PC_{k-\frac{1}{2}}$ denote the number of Pappus configurations present halfway through stage k . Since a Pappus configuration is comprised of two distinct lines and six distinct points (three from each line), we can provide an upper bound of

$$PC_{k-\frac{1}{2}} \leq \binom{m_{k-1}}{2} \left(\frac{\overline{\Delta}_{k-\frac{1}{2}} - 1}{3} \right)^2 < \frac{m_{k-1}^2 \overline{\Delta}_{k-\frac{1}{2}}^6}{2 \cdot 36} = \frac{m_{k-1}^2 \overline{\Delta}_{k-\frac{1}{2}}^6}{72}.$$

Similarly, we get a lower bound of

$$PC_{k-\frac{1}{2}} \geq \binom{m_{k-1}}{2} \binom{\bar{\delta}_{k-\frac{1}{2}} - 1}{3}^2 > \frac{(m_{k-1} - 1)^2 (\bar{\delta}_{k-\frac{1}{2}} - 3)^6}{2 \cdot 36} = \frac{(m_{k-1} - 1)^2 (\bar{\delta}_{k-\frac{1}{2}} - 3)^6}{72}.$$

It follows that

$$\frac{(m_{k-1} - 1)^2 (\bar{\delta}_{k-\frac{1}{2}} - 3)^6}{72} < PC_{k-\frac{1}{2}} < \frac{m_{k-1}^2 \bar{\Delta}_{k-\frac{1}{2}}^6}{72}$$

Since we know that $\bar{\Delta}_{k-\frac{1}{2}} \leq m_{k-1}$ we get

$$\frac{(m_{k-1} - 1)^2 (\bar{\delta}_{k-\frac{1}{2}} - 3)^6}{72} < PC_{k-\frac{1}{2}} < \frac{m_{k-1}^8}{72}.$$

From Section 3.2, we know that IRP is a free ‘‘Pappian’’ plane (and quite possibly *the* free ‘‘Pappian’’ plane). It follows that the overcount from using the trivial upper bound given in (1) is solely due to the degree issue discussed in Section 2.2 combined with the Pappus Configurations discussed here. However, we need to find better bounds on $PC_{k-\frac{1}{2}}$ before we can make any progress with this idea.

5.5. THE FREE ‘‘DESARGUIAN’’ PLANE

In Section 3.1 we mentioned the fact that Pappus’s Theorem implies Desargues’s Theorem, but not vice versa. The two are equivalent in any finite plane, because a finite skew field is a field (Wedderburn’s Theorem), but there are certain infinite planes that are Desarguian and not Pappian. This suggests that there exist free ‘‘Desarguian’’ planes, i.e., free \mathcal{G} -planes, where \mathcal{G} denotes the set of all Desarguian planes. If we let \mathcal{F} denote the set of all Pappian planes, then we have $\mathcal{G} \subsetneq \mathcal{F}$. A free \mathcal{G} -plane must contain some free \mathcal{F} -plane because a free \mathcal{G} -plane is permitted to violate Pappus’s Theorem, whereas a free \mathcal{F} -plane must satisfy Pappus’s Theorem. However, a free \mathcal{G} -plane must be contained in the free projective plane (analyzed in Chapter 4) because the free projective plane is permitted to violate Desargues’s Theorem, whereas a free \mathcal{G} -plane must satisfy Desargues’s Theorem. So, we have this

plane that “lies between” IRP and the free projective plane. We also know that this plane is constructed over a skew field that is not a field. It would be instructive to learn more about such a plane in an effort to better understand the planes discussed in this paper.

CHAPTER 6

A RESULT IN ADDITIVE NUMBER THEORY

LEMMA 27. *If A is a geometric progression $\{a, ar, ar^2, \dots, ar^{k-1}\}$ then*

$$\min_{|B|=k} |A + B| = \binom{k+1}{2},$$

which is achieved by $B = A$.

PROOF. First, we aim to show that

$$|A + B| \geq \binom{k+1}{2} = \frac{k^2 + k}{2}.$$

Let $B = \{b_0, b_1, \dots, b_{k-1}\}$. Note that $B_0 = \{a + b_0, ar + b_0, \dots, ar^{k-1} + b_0\}$ is a set of k distinct elements of $A + B$. Similarly $B_1 = \{a + b_1, ar + b_1, \dots, ar^{k-1} + b_1\}$ is a set of k distinct elements of $A + B$. The only way an element could lie in $B_0 \cap B_1$ is if

$$ar^i + b_0 = ar^j + b_1$$

for some $0 \leq i, j \leq k-1$. In this case, $i > j$ and

$$b_1 - b_0 = ar^i - ar^j = a(r^i - r^j)$$

which implies that there exists at most one such element (since $b_1 - b_0$, a , and r are all fixed). Let $B_i = \{a + b_i, ar + b_i, \dots, ar^{k-1} + b_i\}$ for $0 \leq i \leq k-1$. By a similar argument $|B_i \cap B_j| \leq 1$ for all $1 \leq i, j \leq k-1$. It's clear that

$$A + B = \bigcup_{0 \leq i \leq k-1} B_i.$$

It's also clear that

$$\left| \bigcup_{0 \leq i \leq k-1} B_i \right| \leq k^2$$

since each B_i has k elements. Now build $A + B$ by building $\cup_{0 \leq i \leq k-1} B_i$ one set at a time. Note that B_0 has all distinct elements. Now B_1 may have at most one element that lies in B_0 and so our overcount is at most 1. Next, $B_2 \cap (B_0 \cup B_1) \leq 2$ since $B_2 \cap B_0 \leq 1$ and $B_2 \cap B_1 \leq 1$. So our overcount is at most $1 + 2 = 3$. Similarly, $B_3 \cap (B_0 \cup B_1 \cup B_2) \leq 3$, so the overcount is at most $1 + 2 + 3 = 6$. Proceeding inductively in this way we get a maximum overcount of

$$1 + 2 + 3 + \dots + k - 1 = \frac{k(k-1)}{2}.$$

So, the minimum size of $\cup_{0 \leq i \leq k-1} B_i$ is given by

$$k^2 - \frac{k(k-1)}{2} = \frac{2k^2 - (k^2 - k)}{2} = \frac{k^2 + k}{2} = \frac{k(k+1)}{2} = \binom{k+1}{2}.$$

Thus,

$$\min_{|B|=k} |A + B| \geq \binom{k+1}{2}.$$

To see that this is achieved by $B = A$ is trivial:

$$B_0 = \{2a, a + ar, a + ar^2, \dots, a + ar^{k-1}\} \quad (\text{all distinct})$$

$$B_1 = \{ar + a, 2ar, ar + ar^2, \dots, ar + ar^{k-1}\} \quad (\text{one duplicate})$$

$$B_2 = \{ar^2 + a, ar^2 + ar, 2ar^2, \dots, ar^2 + ar^{k-1}\} \quad (\text{two duplicates})$$

\vdots

$$B_{k-1} = \{ar^{k-1} + a, ar^{k-1} + ar, \dots, ar^{k-1} + ar^{k-2}, 2ar^{k-1}\} \quad (\text{k-1 duplicates})$$

□

BIBLIOGRAPHY

1. G. Ambrus and A. Bezdek, *On iterative processes generating dense point sets*, Period. Math. Hungar. **53** (2006), no. 1-2, 27–44. MR2286458 (2008c:52007)
2. Károly Bezdek and János Pach, *A point set everywhere dense in the plane*, Elem. Math. **40** (1985), no. 4, 81–84. MR803080 (86m:51026)
3. Peter J. Cameron, *Projective and polar spaces*, QMW Maths Notes, vol. 13, Queen Mary and Westfield College School of Mathematical Sciences, London, 199?. MR1153019 (93c:51011)
4. Rey Casse, *Projective geometry: an introduction*, Oxford University Press, Oxford, 2006. MR2264641 (2007f:51001)
5. Giorgio Donati, *Pappus' configuration in non commutative projective geometry with application to a theorem of A. Schleiernmacher*, Rend. Circ. Mat. Palermo (2) **50** (2001), no. 2, 325–328. MR1847050 (2002g:51004)
6. Ansgar Grüne and Sanaz Kamali, *On the density of iterated line segment intersections*, Comput. Geom. **40** (2008), no. 1, 23–36. MR2392650 (2009b:52040)
7. Marshall Hall, *Projective planes*, Trans. Amer. Math. Soc. **54** (1943), 229–277. MR0008892 (5,72c)
8. Gerhard Hessenberg, *Beweis des Desarguesschen Satzes aus dem Pascalschen*, Math. Ann. **61** (1905), no. 2, 161–172. MR1511339
9. Christopher J. Hillar and Darren L. Rhea, *A result about the density of iterated line intersections in the plane*, Comput. Geom. **33** (2006), no. 3, 106–114. MR2199803 (2007b:52004)
10. Margherita Iorio, Dan Ismailescu, Radoš Radoičić, and Manuel Silva, *On point sets containing their triangle centers*, Rev. Roumaine Math. Pures Appl. **50** (2005), no. 5-6, 677–693. MR2204145 (2006k:52003)
11. Dan Ismailescu and Radoš Radoičić, *A dense planar point set from iterated line intersections*, Computational Geometry **27** (2004), 257–267.

12. Melvyn B. Nathanson, *Additive number theory: Inverse problems and the geometry of sumsets*, Springer, 1996.
13. János Pach and Géza Tóth, *Graphs drawn with few crossings per edge*, *Combinatorica* **17** (1997), no. 3, 427–439. MR1606052 (99b:05043)
14. J. T. Stafford, *Noncommutative projective geometry*, Proceedings of the International Congress of Mathematicians, Vol. II (Beijing, 2002) (Beijing), Higher Ed. Press, 2002, pp. 93–103. MR1957024 (2004k:16077)
15. László A. Székely, *Crossing numbers and hard Erdős problems in discrete geometry*, *Combin. Probab. Comput.* **6** (1997), no. 3, 353–358. MR1464571 (98h:52030)
16. Endre Szemerédi and William T. Trotter, Jr., *Extremal problems in discrete geometry*, *Combinatorica* **3** (1983), no. 3-4, 381–392. MR729791 (85j:52014)

APPENDIX A

HOMOGENEOUS COORDINATES CODE

```
#####
# Function cross_simple                                     #
# Input: 3-dimensional vectors v and w                   #
# Output: Cross product of v with w, reduced by common factor #
#                                                         #
# This function is used to find the homogeneous coordinates of the #
# line joining two points or the intersection point of two lines. #
#####

def cross_simple(v,w):
    z = v.cross_product(w)
    f = R(z[0])
    g = R(z[1])
    h = R(z[2])
    D = (f.gcd(g)).gcd(h)
    if D != 0:
        f /= D
        g /= D
        h /= D
    return vector([f,g,h])
    return vector([0,0,0])

#####
# function areSame                                       #
# Input: 3-dimensional vectors v and w                   #
# Output: 1 if v and w represent the same point/line, 0 otherwise #
#                                                         #
# This function is used to determine if two points are the same. #
#####

def areSame(v,w):
    if v==w or v==w:
        return True
    return False
```

```
#####
# function isInList                                                    #
# Input:  3-dimensional vector v and list of vectors L                #
# Output: 1 if v is an element of L, 0 otherwise                      #
#                                                                 #
# This function is a boolean test to see if a vector belongs to a   #
# set.                                                                 #
#####
```

```
def isInList(v,L):
    for w in L:
        if areSame(v,w):
            return 1
    return 0
```

```
#####
# function isSubset                                                    #
# Input:  Two lists containing 3-dimensional vectors                  #
# Output: 1 if list1 is a subset of list2, 0 otherwise              #
#                                                                 #
# This function is a boolean test to see if one set is a subset of  #
# another.                                                            #
#####
```

```
def isSubset(list1,list2):
    for v in list1:
        if not isInList(v,list2):
            return 0
    return 1
```

```
#####
# function nextStage                                                  #
# Input:  Sets of points P(k) and lines L(k)                        #
# Output: Sets of points P(k+1) and lines L(k+1)                    #
#                                                                 #
# This function completes a full stage of the iterative process.    #
#####
```

```
def nextStage(P,L):
    copyP = copy(P)
    copyL = copy(L)
    for i in range(len(copyL)-1):
        for j in range(i+1,len(copyL)):
```

```

        v = cross_simple(copyL[i],copyL[j])
        if not isInList(v,copyP):
            copyP.append(v)
    for i in range(len(copyP)-1):
        for j in range(i+1,len(copyP)):
            w = cross_simple(copyP[i],copyP[j])
            if not isInList(w,copyL):
                copyL.append(w)
    return [copyP,copyL]

```

```

#####
# function nextStep                                                    #
#Input:  Sets of points P(k) and lines L(k)                          #
# Output: Set of points P(k+1)                                       #
#                                               #
# This function completes a half stage of the iterative process.    #
#####

```

```

def nextStep(P,L):
    copyP = copy(P)
    for i in range(len(L)-1):
        for j in range(i+1,len(L)):
            v = cross_simple(L[i],L[j])
            if not isInList(v,copyP):
                copyP.append(v)
    return copyP

```

```

#####
# function nextStepRandom                                            #
# Input:  Sets of points P(k) and lines L(k), number of new points #
# N                                             #
# Output: P(k) unioned with a random set of N points from          #
# P(k+1)-P(k)                                                       #
#                                               #
# This function creates a random set of N points from the next    #
# stage.                                                             #
#####

```

```

def nextStepRandom(P,L,N):
    copyP = copy(P)
    for n in range(N):
        if n % 10 == 0:
            print n
        while 1:

```

```

        i = floor(len(L)*random())
        while 1:
            j = floor(len(L)*random())
            if i != j:
                break
        v = cross_simple(L[i],L[j])
        if not isInList(v,copyP):
            copyP.append(v)
            break
    return copyP

#####
# function makeLines                                     #
# Input:  set of points P                               #
# Output: set of lines L created by P                  #
#                                                #
# This function constructs the lines defined by P after one #
# iteration.                                           #
#####

def makeLines(P):
    L = []
    for i in range(len(P)-1):
        p1 = P[i]
        for j in range(i+1,len(P)):
            p2 = P[j]
            l = cross_simple(p1,p2)
            if not isInList(l,L):
                L.append(l)
    return L

#####
# function collinearTest                               #
# Input:  set of lines L                               #
# Output: 1 if any two lines from L are the same, 0 otherwise #
#                                                #
# This function is a boolean test to see if a set contains distinct #
# lines.                                              #
#####

def collinearTest(L):
    for i in range(len(L)-1):
        line1 = L[i]
        for j in range(i+1,len(L)):

```

```

        line2 = L[j]
        if areSame(line1,line2):
            return 0
    return 1

#####
# function oldLineTest                                     #
# Input:  sets of lines L and lines                       #
# Output: 1 if two sets are disjoint, 0 otherwise        #
#                                                #
# This function is a boolean test to see if two sets of lines have #
# nonempty intersection.                                #
#####

def oldLineTest(L,lines):
    for l in L:
        if isInList(l,lines):
            return 0
    return 1

#####
# function pointSearch                                     #
# Input:  sets of points p and lines l, number of iterations N #
# Output: vector containing indices of 4 points if found, [0,0,0,0] #
#         otherwise                                         #
#                                                #
# This function searches for 4 general points of IRP such that they #
# form a basis for IRP and that all six lines formed in the first #
# iteration are "new" lines. The existence of such points provides #
# an inherent distinction between IRP and the free projective plane.#
#####

def pointSearch(p,l,N):
    for n in range(N):
        print n
        i = floor(len(p)*random())
        while 1:
            j = floor(len(p)*random())
            if i != j:
                break
        while 1:
            k = floor(len(p)*random())
            if (i != k) and (j != k):
                break

```

```

while 1:
    l = floor(len(p)*random())
    if (i != l) and (j != l) and (k != l):
        break
w = p[i]
x = p[j]
y = p[k]
z = p[l]
P = [w,x,y,z]
L = makeLines(P)
if collinearTest(L):
    if oldLineTest(L,l):
        if basisCheck(P,p):
            return [i,j,k,l]
return [0,0,0,0]

```

```

#####
# function basisCheck                                     #
# Input:  potential basis in question bPoints, current point set #
# P(k)                                           #
# Output: 1 if basis is determined within 2 iterations, 0 otherwise #
#                                               #
# This function is a boolean test to see if a set of points forms a #
# basis for IRP.                                     #
#####

```

```

def basisCheck(bPoints,points):
    count = 0
    bLines = makeLines(bPoints)
    while count < 2:
        count += 1
        [bPoints,bLines] = nextStage(bPoints,bLines)
        print 'iteration', count, 'complete'
        if check13(bPoints,points):
            return 1
    return 0

```

```

#####
# function check13                                       #
# Input:  potential basis in question bPoints, current point set #
# P(k)                                           #
# Output: 1 if any 4 general points from P3 are found, 0 otherwise #
#                                               #
# This function is a subroutine of the basisCheck function. #
#####

```

```
#####
```

```
def check13(bPoints,points):  
    I = []  
    for i in range(13):  
        if isInList(points[i],bPoints):  
            I.append(i)  
    if len(I) < 4:  
        return 0  
    for i in range(len(I)-3):  
        w = points[i]  
        for j in range(i+1,len(I)-2):  
            x = points[j]  
            for k in range(j+1,len(I)-1):  
                y = points[k]  
                for l in range(k+1,len(I)):  
                    z = points[l]  
                    P = [w,x,y,z]  
                    L = makeLines(P)  
                    if collinearTest(L):  
                        return 1  
    return 0
```

```
# Establishes the variables x1-x12.  
R.<x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12>=ZZ []
```

```
a = vector([x1,x2,x3])  
b = vector([x4,x5,x6])  
c = vector([x7,x8,x9])  
d = vector([x10,x11,x12])
```

```
ab = cross_simple(a,b)  
ac = cross_simple(a,c)  
ad = cross_simple(a,d)  
bc = cross_simple(b,c)  
bd = cross_simple(b,d)  
cd = cross_simple(c,d)
```

```
P1 = [a,b,c,d]  
L1 = [ab,ac,ad,bc,bd,cd]
```

```
[P2,L2] = nextStage(P1,L1)  
[len(P2),len(L2)]
```

```
[P3,L3] = nextStage(P2,L2)
```

`[len(P3), len(L3)]`

`:`

APPENDIX B

REDUCED COEFFICIENT VECTOR CODE

```

# import bisect routine from library
from bisect import bisect

#####
# function adjust
# Input: vector v
# Output: vector -v
#
# This function negates a vector v, if necessary.
#####

def adjust(v):
    if ((v[0] < 0) or (v[0] == 0 and v[1] < 0)
        or (v[0] == 0 and v[1] == 0 and v[2] < 0)):
        for i in range(len(v)):
            v[i] = -v[i]

#####
# function isSubset
# Input: Sorted lists L1 and L2
# Output: 1 if L1 is a subset of L2, 0 otherwise
#
# This function is a boolean test to see if one set is a subset of
# another.
#####

def isSubset(L1,L2):
    for v in L1:
        inSpot = bisect(L2,v)
        if v != L2[inSpot-1]:
            return 0
    return 1

```

```
#####
# function crunch                                                    #
# Input: reduced coefficient vectors of four points                 #
# Output: reduced coefficient vector of resulting point,           #
#         or [0,0,0] if four points are collinear.                 #
#                                                                    #
# This function computes the reduced coefficient vector of the     #
# point obtained by intersecting the line through the first pair of #
# points with the line through the second pair of points. In the   #
# event that the four points are collinear, such a point does not  #
# exist and so [0,0,0] is returned.                                #
#####
```

```
def crunch(w1,w2,w3,x1,x2,x3,y1,y2,y3,z1,z2,z3):
    p = -w3*x1*y2*z1 + w1*x3*y2*z1 + w2*x1*y3*z1 - w1*x2*y3*z1
        + w3*x1*y1*z2 - w1*x3*y1*z2 - w2*x1*y1*z3 + w1*x2*y1*z3
    q = -w3*x2*y2*z1 + w2*x3*y2*z1 + w3*x2*y1*z2 - w2*x3*y1*z2
        + w2*x1*y3*z2 - w1*x2*y3*z2 - w2*x1*y2*z3 + w1*x2*y2*z3
    r = -w3*x2*y3*z1 + w2*x3*y3*z1 + w3*x1*y3*z2 - w1*x3*y3*z2
        + w3*x2*y1*z3 - w2*x3*y1*z3 - w3*x1*y2*z3 + w1*x3*y2*z3
    D = (p.gcd(q)).gcd(r)
    if D != 0:
        p = p/D
        q = q/D
        r = r/D
        return vector([p,q,r])
    else:
        return vector([1,0,0])
```

```
#####
# function nextStage                                                #
# Input: current set of points P(k) [reduced coefficient form]    #
# Output: set of points at the next stage P(k+1)                  #
#                                                                    #
# This function performs one stage of the iterative process.      #
#####
```

```
def nextStage(P):
    copyP = copy(P)
    for i in range(len(P)-3):
        w = P[i]
        for j in range(i+1,len(P)-2):
            x = P[j]
```

```

    for k in range(j+1,len(P)-1):
        y = P[k]
        for l in range(k+1,len(P)):
            z = P[l]
            p = crunch(w[0],w[1],w[2],x[0],x[1],x[2],
                       y[0],y[1],y[2],z[0],z[1],z[2])
            q = crunch(w[0],w[1],w[2],y[0],y[1],y[2],
                       x[0],x[1],x[2],z[0],z[1],z[2])
            r = crunch(w[0],w[1],w[2],z[0],z[1],z[2],
                       x[0],x[1],x[2],y[0],y[1],y[2])
            newPoints = [p,q,r]
            for v in newPoints:
                adjust(v)
                inSpot = bisect(copyP,v)
                if not v == copyP[inSpot-1]:
                    copyP.insert(inSpot,v)

return copyP

#####
# function basisCheck                                                    #
# Input:  potential basis in question bPoints, initial 4 points        #
# abcd                                         #
# Output: 1 if basis is determined within 2 iterations, 0 otherwise    #
#                                               #
# This function is a boolean test to see if a set of points forms a   #
# basis for IRP.                                                         #
#####

def basisCheck(bPoints,abcd):
    count = 0
    while count < 3:
        count += 1
        bPoints = nextStage(bPoints)
        if isSubset(abcd,bPoints):
            return 1
    return 0

# Initial four points a,b,c,d
a = vector([1,-1,1])
b = vector([1,0,0])
c = vector([0,1,0])
d = vector([0,0,1])

# We use the order d,c,a,b so that P1 is sorted.

```

```
P1 = [d,c,a,b]
```

```
P2 = nextStage(P1)  
len(P2)
```

```
P3 = nextStage(P2)  
len(P3)
```

```
P4 = nextStage(P3)  
len(P4)
```

```
P5 = nextStage(P4)  
len(P5)
```

APPENDIX C

REDUCED COEFFICIENT VECTOR PROOF

```

# Establishes the variables

R.<a1,a2,a3,b1,b2,b3,c1,c2,c3,d1,d2,d3,
  x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12>=ZZ[]

#####
# Function cross_simple                                     #
# Input: 3-dimensional vectors v and w                     #
# Output: Cross product of v with w, reduced by common factor #
#                                                     #
# This function is used to find the homogeneous coordinates of the #
# line joining two points or the intersection point of two lines. #
#####

def cross_simple(v,w):
    z = v.cross_product(w)
    f = R(z[0])
    g = R(z[1])
    h = R(z[2])
    D = (f.gcd(g)).gcd(h)
    if D != 0:
        f /= D
        g /= D
        h /= D
    return vector([f,g,h])
    return vector([0,0,0])

w = vector([a1*x1*x4*x8*x12 - a1*x1*x4*x9*x11 + a2*x1*x5*x7*x12
- a2*x1*x6*x7*x11 + a3*x1*x5*x9*x10 - a3*x1*x6*x8*x10
+ (-a1-a2)*x2*x4*x7*x12 - (-a1-a2)*x3*x4*x7*x11 + (a1-a3)*x2*x4*x9*x10
- (a1-a3)*x3*x4*x8*x10 + (a2+a3)*x2*x6*x7*x10 - (a2+a3)*x3*x5*x7*x10,
-a1*x2*x5*x7*x12 + a1*x2*x5*x9*x10 - a2*x2*x4*x8*x12 + a2*x2*x6*x8*x10
- a3*x2*x4*x9*x11 + a3*x2*x6*x7*x11 - (-a1-a2)*x1*x5*x8*x12
+ (-a1-a2)*x3*x5*x8*x10 - (a1-a3)*x1*x5*x9*x11 + (a1-a3)*x3*x5*x7*x11

```

$$\begin{aligned}
& - (a_2+a_3)*x_1*x_6*x_8*x_{11} + (a_2+a_3)*x_3*x_4*x_8*x_{11}, a_1*x_3*x_6*x_7*x_{11} \\
& - a_1*x_3*x_6*x_8*x_{10} + a_2*x_3*x_4*x_9*x_{11} - a_2*x_3*x_5*x_9*x_{10} \\
& + a_3*x_3*x_4*x_8*x_{12} - a_3*x_3*x_5*x_7*x_{12} + (-a_1-a_2)*x_1*x_6*x_9*x_{11} \\
& - (-a_1-a_2)*x_2*x_6*x_9*x_{10} + (a_1-a_3)*x_1*x_6*x_8*x_{12} - (a_1-a_3)*x_2*x_6*x_7*x_{12} \\
& + (a_2+a_3)*x_1*x_5*x_9*x_{12} - (a_2+a_3)*x_2*x_4*x_9*x_{12}]
\end{aligned}$$

$$\begin{aligned}
x = \text{vector}(& [b_1*x_1*x_4*x_8*x_{12} - b_1*x_1*x_4*x_9*x_{11} + b_2*x_1*x_5*x_7*x_{12} \\
& - b_2*x_1*x_6*x_7*x_{11} + b_3*x_1*x_5*x_9*x_{10} - b_3*x_1*x_6*x_8*x_{10} \\
& + (-b_1-b_2)*x_2*x_4*x_7*x_{12} - (-b_1-b_2)*x_3*x_4*x_7*x_{11} + (b_1-b_3)*x_2*x_4*x_9*x_{10} \\
& - (b_1-b_3)*x_3*x_4*x_8*x_{10} + (b_2+b_3)*x_2*x_6*x_7*x_{10} - (b_2+b_3)*x_3*x_5*x_7*x_{10}, \\
& -b_1*x_2*x_5*x_7*x_{12} + b_1*x_2*x_5*x_9*x_{10} - b_2*x_2*x_4*x_8*x_{12} + b_2*x_2*x_6*x_8*x_{10} \\
& - b_3*x_2*x_4*x_9*x_{11} + b_3*x_2*x_6*x_7*x_{11} - (-b_1-b_2)*x_1*x_5*x_8*x_{12} \\
& + (-b_1-b_2)*x_3*x_5*x_8*x_{10} - (b_1-b_3)*x_1*x_5*x_9*x_{11} + (b_1-b_3)*x_3*x_5*x_7*x_{11} \\
& - (b_2+b_3)*x_1*x_6*x_8*x_{11} + (b_2+b_3)*x_3*x_4*x_8*x_{11}, b_1*x_3*x_6*x_7*x_{11} \\
& - b_1*x_3*x_6*x_8*x_{10} + b_2*x_3*x_4*x_9*x_{11} - b_2*x_3*x_5*x_9*x_{10} \\
& + b_3*x_3*x_4*x_8*x_{12} - b_3*x_3*x_5*x_7*x_{12} + (-b_1-b_2)*x_1*x_6*x_9*x_{11} \\
& - (-b_1-b_2)*x_2*x_6*x_9*x_{10} + (b_1-b_3)*x_1*x_6*x_8*x_{12} - (b_1-b_3)*x_2*x_6*x_7*x_{12} \\
& + (b_2+b_3)*x_1*x_5*x_9*x_{12} - (b_2+b_3)*x_2*x_4*x_9*x_{12}])
\end{aligned}$$

$$\begin{aligned}
y = \text{vector}(& [c_1*x_1*x_4*x_8*x_{12} - c_1*x_1*x_4*x_9*x_{11} + c_2*x_1*x_5*x_7*x_{12} \\
& - c_2*x_1*x_6*x_7*x_{11} + c_3*x_1*x_5*x_9*x_{10} - c_3*x_1*x_6*x_8*x_{10} \\
& + (-c_1-c_2)*x_2*x_4*x_7*x_{12} - (-c_1-c_2)*x_3*x_4*x_7*x_{11} + (c_1-c_3)*x_2*x_4*x_9*x_{10} \\
& - (c_1-c_3)*x_3*x_4*x_8*x_{10} + (c_2+c_3)*x_2*x_6*x_7*x_{10} - (c_2+c_3)*x_3*x_5*x_7*x_{10}, \\
& -c_1*x_2*x_5*x_7*x_{12} + c_1*x_2*x_5*x_9*x_{10} - c_2*x_2*x_4*x_8*x_{12} + c_2*x_2*x_6*x_8*x_{10} \\
& - c_3*x_2*x_4*x_9*x_{11} + c_3*x_2*x_6*x_7*x_{11} - (-c_1-c_2)*x_1*x_5*x_8*x_{12} \\
& + (-c_1-c_2)*x_3*x_5*x_8*x_{10} - (c_1-c_3)*x_1*x_5*x_9*x_{11} + (c_1-c_3)*x_3*x_5*x_7*x_{11} \\
& - (c_2+c_3)*x_1*x_6*x_8*x_{11} + (c_2+c_3)*x_3*x_4*x_8*x_{11}, c_1*x_3*x_6*x_7*x_{11} \\
& - c_1*x_3*x_6*x_8*x_{10} + c_2*x_3*x_4*x_9*x_{11} - c_2*x_3*x_5*x_9*x_{10} \\
& + c_3*x_3*x_4*x_8*x_{12} - c_3*x_3*x_5*x_7*x_{12} + (-c_1-c_2)*x_1*x_6*x_9*x_{11} \\
& - (-c_1-c_2)*x_2*x_6*x_9*x_{10} + (c_1-c_3)*x_1*x_6*x_8*x_{12} - (c_1-c_3)*x_2*x_6*x_7*x_{12} \\
& + (c_2+c_3)*x_1*x_5*x_9*x_{12} - (c_2+c_3)*x_2*x_4*x_9*x_{12}])
\end{aligned}$$

$$\begin{aligned}
z = \text{vector}(& [d_1*x_1*x_4*x_8*x_{12} - d_1*x_1*x_4*x_9*x_{11} + d_2*x_1*x_5*x_7*x_{12} \\
& - d_2*x_1*x_6*x_7*x_{11} + d_3*x_1*x_5*x_9*x_{10} - d_3*x_1*x_6*x_8*x_{10} \\
& + (-d_1-d_2)*x_2*x_4*x_7*x_{12} - (-d_1-d_2)*x_3*x_4*x_7*x_{11} + (d_1-d_3)*x_2*x_4*x_9*x_{10} \\
& - (d_1-d_3)*x_3*x_4*x_8*x_{10} + (d_2+d_3)*x_2*x_6*x_7*x_{10} - (d_2+d_3)*x_3*x_5*x_7*x_{10}, \\
& -d_1*x_2*x_5*x_7*x_{12} + d_1*x_2*x_5*x_9*x_{10} - d_2*x_2*x_4*x_8*x_{12} + d_2*x_2*x_6*x_8*x_{10} \\
& - d_3*x_2*x_4*x_9*x_{11} + d_3*x_2*x_6*x_7*x_{11} - (-d_1-d_2)*x_1*x_5*x_8*x_{12} \\
& + (-d_1-d_2)*x_3*x_5*x_8*x_{10} - (d_1-d_3)*x_1*x_5*x_9*x_{11} + (d_1-d_3)*x_3*x_5*x_7*x_{11} \\
& - (d_2+d_3)*x_1*x_6*x_8*x_{11} + (d_2+d_3)*x_3*x_4*x_8*x_{11}, d_1*x_3*x_6*x_7*x_{11} \\
& - d_1*x_3*x_6*x_8*x_{10} + d_2*x_3*x_4*x_9*x_{11} - d_2*x_3*x_5*x_9*x_{10} \\
& + d_3*x_3*x_4*x_8*x_{12} - d_3*x_3*x_5*x_7*x_{12} + (-d_1-d_2)*x_1*x_6*x_9*x_{11} \\
& - (-d_1-d_2)*x_2*x_6*x_9*x_{10} + (d_1-d_3)*x_1*x_6*x_8*x_{12} - (d_1-d_3)*x_2*x_6*x_7*x_{12} \\
& + (d_2+d_3)*x_1*x_5*x_9*x_{12} - (d_2+d_3)*x_2*x_4*x_9*x_{12}])
\end{aligned}$$

```

wx = cross_simple(w,x)
yz = cross_simple(y,z)
p = cross_simple(wx,yz)
p

```

```

(a3*b2*c2*d1*x3*x5*x7*x10 - a2*b3*c2*d1*x3*x5*x7*x10 +
a3*b2*c3*d1*x3*x5*x7*x10 - a2*b3*c3*d1*x3*x5*x7*x10 -
a3*b2*c1*d2*x3*x5*x7*x10 + a2*b3*c1*d2*x3*x5*x7*x10 -
a2*b1*c3*d2*x3*x5*x7*x10 - a3*b1*c3*d2*x3*x5*x7*x10 +
a1*b2*c3*d2*x3*x5*x7*x10 + a1*b3*c3*d2*x3*x5*x7*x10 -
a3*b2*c1*d3*x3*x5*x7*x10 + a2*b3*c1*d3*x3*x5*x7*x10 +
a2*b1*c2*d3*x3*x5*x7*x10 + a3*b1*c2*d3*x3*x5*x7*x10 -
a1*b2*c2*d3*x3*x5*x7*x10 - a1*b3*c2*d3*x3*x5*x7*x10 -
a3*b2*c2*d1*x2*x6*x7*x10 + a2*b3*c2*d1*x2*x6*x7*x10 -
a3*b2*c3*d1*x2*x6*x7*x10 + a2*b3*c3*d1*x2*x6*x7*x10 +
a3*b2*c1*d2*x2*x6*x7*x10 - a2*b3*c1*d2*x2*x6*x7*x10 +
a2*b1*c3*d2*x2*x6*x7*x10 + a3*b1*c3*d2*x2*x6*x7*x10 -
a1*b2*c3*d2*x2*x6*x7*x10 - a1*b3*c3*d2*x2*x6*x7*x10 +
a3*b2*c1*d3*x2*x6*x7*x10 - a2*b3*c1*d3*x2*x6*x7*x10 -
a2*b1*c2*d3*x2*x6*x7*x10 - a3*b1*c2*d3*x2*x6*x7*x10 +
a1*b2*c2*d3*x2*x6*x7*x10 + a1*b3*c2*d3*x2*x6*x7*x10 +
a3*b1*c2*d1*x3*x4*x8*x10 - a1*b3*c2*d1*x3*x4*x8*x10 -
a2*b1*c3*d1*x3*x4*x8*x10 + a1*b2*c3*d1*x3*x4*x8*x10 -
a3*b2*c3*d1*x3*x4*x8*x10 + a2*b3*c3*d1*x3*x4*x8*x10 -
a3*b1*c1*d2*x3*x4*x8*x10 + a1*b3*c1*d2*x3*x4*x8*x10 +
a3*b1*c3*d2*x3*x4*x8*x10 - a1*b3*c3*d2*x3*x4*x8*x10 +
a2*b1*c1*d3*x3*x4*x8*x10 - a1*b2*c1*d3*x3*x4*x8*x10 +
a3*b2*c1*d3*x3*x4*x8*x10 - a2*b3*c1*d3*x3*x4*x8*x10 -
a3*b1*c2*d3*x3*x4*x8*x10 + a1*b3*c2*d3*x3*x4*x8*x10 +
a3*b2*c3*d1*x1*x6*x8*x10 - a2*b3*c3*d1*x1*x6*x8*x10 -
a3*b1*c3*d2*x1*x6*x8*x10 + a1*b3*c3*d2*x1*x6*x8*x10 -
a3*b2*c1*d3*x1*x6*x8*x10 + a2*b3*c1*d3*x1*x6*x8*x10 +
a3*b1*c2*d3*x1*x6*x8*x10 - a1*b3*c2*d3*x1*x6*x8*x10 -
a3*b1*c2*d1*x2*x4*x9*x10 + a1*b3*c2*d1*x2*x4*x9*x10 +
a2*b1*c3*d1*x2*x4*x9*x10 - a1*b2*c3*d1*x2*x4*x9*x10 +
a3*b2*c3*d1*x2*x4*x9*x10 - a2*b3*c3*d1*x2*x4*x9*x10 +
a3*b1*c1*d2*x2*x4*x9*x10 - a1*b3*c1*d2*x2*x4*x9*x10 -
a3*b1*c3*d2*x2*x4*x9*x10 + a1*b3*c3*d2*x2*x4*x9*x10 -
a2*b1*c1*d3*x2*x4*x9*x10 + a1*b2*c1*d3*x2*x4*x9*x10 -
a3*b2*c1*d3*x2*x4*x9*x10 + a2*b3*c1*d3*x2*x4*x9*x10 +
a3*b1*c2*d3*x2*x4*x9*x10 - a1*b3*c2*d3*x2*x4*x9*x10 -
a3*b2*c3*d1*x1*x5*x9*x10 + a2*b3*c3*d1*x1*x5*x9*x10 +
a3*b1*c3*d2*x1*x5*x9*x10 - a1*b3*c3*d2*x1*x5*x9*x10 +
a3*b2*c1*d3*x1*x5*x9*x10 - a2*b3*c1*d3*x1*x5*x9*x10 -
a3*b1*c2*d3*x1*x5*x9*x10 + a1*b3*c2*d3*x1*x5*x9*x10 -
a3*b1*c2*d1*x3*x4*x7*x11 - a3*b2*c2*d1*x3*x4*x7*x11 +

```

$a1*b3*c2*d1*x3*x4*x7*x11 + a2*b3*c2*d1*x3*x4*x7*x11 +$
 $a2*b1*c3*d1*x3*x4*x7*x11 - a1*b2*c3*d1*x3*x4*x7*x11 +$
 $a3*b1*c1*d2*x3*x4*x7*x11 + a3*b2*c1*d2*x3*x4*x7*x11 -$
 $a1*b3*c1*d2*x3*x4*x7*x11 - a2*b3*c1*d2*x3*x4*x7*x11 +$
 $a2*b1*c3*d2*x3*x4*x7*x11 - a1*b2*c3*d2*x3*x4*x7*x11 -$
 $a2*b1*c1*d3*x3*x4*x7*x11 + a1*b2*c1*d3*x3*x4*x7*x11 -$
 $a2*b1*c2*d3*x3*x4*x7*x11 + a1*b2*c2*d3*x3*x4*x7*x11 +$
 $a3*b2*c2*d1*x1*x6*x7*x11 - a2*b3*c2*d1*x1*x6*x7*x11 -$
 $a3*b2*c1*d2*x1*x6*x7*x11 + a2*b3*c1*d2*x1*x6*x7*x11 -$
 $a2*b1*c3*d2*x1*x6*x7*x11 + a1*b2*c3*d2*x1*x6*x7*x11 +$
 $a2*b1*c2*d3*x1*x6*x7*x11 - a1*b2*c2*d3*x1*x6*x7*x11 +$
 $a3*b1*c2*d1*x1*x4*x9*x11 - a1*b3*c2*d1*x1*x4*x9*x11 -$
 $a2*b1*c3*d1*x1*x4*x9*x11 + a1*b2*c3*d1*x1*x4*x9*x11 -$
 $a3*b1*c1*d2*x1*x4*x9*x11 + a1*b3*c1*d2*x1*x4*x9*x11 +$
 $a2*b1*c1*d3*x1*x4*x9*x11 - a1*b2*c1*d3*x1*x4*x9*x11 +$
 $a3*b1*c2*d1*x2*x4*x7*x12 + a3*b2*c2*d1*x2*x4*x7*x12 -$
 $a1*b3*c2*d1*x2*x4*x7*x12 - a2*b3*c2*d1*x2*x4*x7*x12 -$
 $a2*b1*c3*d1*x2*x4*x7*x12 + a1*b2*c3*d1*x2*x4*x7*x12 -$
 $a3*b1*c1*d2*x2*x4*x7*x12 - a3*b2*c1*d2*x2*x4*x7*x12 +$
 $a1*b3*c1*d2*x2*x4*x7*x12 + a2*b3*c1*d2*x2*x4*x7*x12 -$
 $a2*b1*c3*d2*x2*x4*x7*x12 + a1*b2*c3*d2*x2*x4*x7*x12 +$
 $a2*b1*c1*d3*x2*x4*x7*x12 - a1*b2*c1*d3*x2*x4*x7*x12 +$
 $a2*b1*c2*d3*x2*x4*x7*x12 - a1*b2*c2*d3*x2*x4*x7*x12 -$
 $a3*b2*c2*d1*x1*x5*x7*x12 + a2*b3*c2*d1*x1*x5*x7*x12 +$
 $a3*b2*c1*d2*x1*x5*x7*x12 - a2*b3*c1*d2*x1*x5*x7*x12 +$
 $a2*b1*c3*d2*x1*x5*x7*x12 - a1*b2*c3*d2*x1*x5*x7*x12 -$
 $a2*b1*c2*d3*x1*x5*x7*x12 + a1*b2*c2*d3*x1*x5*x7*x12 -$
 $a3*b1*c2*d1*x1*x4*x8*x12 + a1*b3*c2*d1*x1*x4*x8*x12 +$
 $a2*b1*c3*d1*x1*x4*x8*x12 - a1*b2*c3*d1*x1*x4*x8*x12 +$
 $a3*b1*c1*d2*x1*x4*x8*x12 - a1*b3*c1*d2*x1*x4*x8*x12 -$
 $a2*b1*c1*d3*x1*x4*x8*x12 + a1*b2*c1*d3*x1*x4*x8*x12,$
 $a3*b1*c2*d1*x3*x5*x8*x10 + a3*b2*c2*d1*x3*x5*x8*x10 -$
 $a1*b3*c2*d1*x3*x5*x8*x10 - a2*b3*c2*d1*x3*x5*x8*x10 -$
 $a2*b1*c3*d1*x3*x5*x8*x10 + a1*b2*c3*d1*x3*x5*x8*x10 -$
 $a3*b1*c1*d2*x3*x5*x8*x10 - a3*b2*c1*d2*x3*x5*x8*x10 +$
 $a1*b3*c1*d2*x3*x5*x8*x10 + a2*b3*c1*d2*x3*x5*x8*x10 -$
 $a2*b1*c3*d2*x3*x5*x8*x10 + a1*b2*c3*d2*x3*x5*x8*x10 +$
 $a2*b1*c1*d3*x3*x5*x8*x10 - a1*b2*c1*d3*x3*x5*x8*x10 +$
 $a2*b1*c2*d3*x3*x5*x8*x10 - a1*b2*c2*d3*x3*x5*x8*x10 -$
 $a3*b2*c2*d1*x2*x6*x8*x10 + a2*b3*c2*d1*x2*x6*x8*x10 +$
 $a3*b2*c1*d2*x2*x6*x8*x10 - a2*b3*c1*d2*x2*x6*x8*x10 +$
 $a2*b1*c3*d2*x2*x6*x8*x10 - a1*b2*c3*d2*x2*x6*x8*x10 -$
 $a2*b1*c2*d3*x2*x6*x8*x10 + a1*b2*c2*d3*x2*x6*x8*x10 -$
 $a3*b1*c2*d1*x2*x5*x9*x10 + a1*b3*c2*d1*x2*x5*x9*x10 +$
 $a2*b1*c3*d1*x2*x5*x9*x10 - a1*b2*c3*d1*x2*x5*x9*x10 +$
 $a3*b1*c1*d2*x2*x5*x9*x10 - a1*b3*c1*d2*x2*x5*x9*x10 -$

$a2*b1*c1*d3*x2*x5*x9*x10 + a1*b2*c1*d3*x2*x5*x9*x10 -$
 $a3*b1*c2*d1*x3*x5*x7*x11 + a1*b3*c2*d1*x3*x5*x7*x11 +$
 $a2*b1*c3*d1*x3*x5*x7*x11 - a1*b2*c3*d1*x3*x5*x7*x11 +$
 $a3*b2*c3*d1*x3*x5*x7*x11 - a2*b3*c3*d1*x3*x5*x7*x11 +$
 $a3*b1*c1*d2*x3*x5*x7*x11 - a1*b3*c1*d2*x3*x5*x7*x11 -$
 $a3*b1*c3*d2*x3*x5*x7*x11 + a1*b3*c3*d2*x3*x5*x7*x11 -$
 $a2*b1*c1*d3*x3*x5*x7*x11 + a1*b2*c1*d3*x3*x5*x7*x11 -$
 $a3*b2*c1*d3*x3*x5*x7*x11 + a2*b3*c1*d3*x3*x5*x7*x11 +$
 $a3*b1*c2*d3*x3*x5*x7*x11 - a1*b3*c2*d3*x3*x5*x7*x11 -$
 $a3*b2*c3*d1*x2*x6*x7*x11 + a2*b3*c3*d1*x2*x6*x7*x11 +$
 $a3*b1*c3*d2*x2*x6*x7*x11 - a1*b3*c3*d2*x2*x6*x7*x11 +$
 $a3*b2*c1*d3*x2*x6*x7*x11 - a2*b3*c1*d3*x2*x6*x7*x11 -$
 $a3*b1*c2*d3*x2*x6*x7*x11 + a1*b3*c2*d3*x2*x6*x7*x11 -$
 $a3*b2*c2*d1*x3*x4*x8*x11 + a2*b3*c2*d1*x3*x4*x8*x11 -$
 $a3*b2*c3*d1*x3*x4*x8*x11 + a2*b3*c3*d1*x3*x4*x8*x11 +$
 $a3*b2*c1*d2*x3*x4*x8*x11 - a2*b3*c1*d2*x3*x4*x8*x11 +$
 $a2*b1*c3*d2*x3*x4*x8*x11 + a3*b1*c3*d2*x3*x4*x8*x11 -$
 $a1*b2*c3*d2*x3*x4*x8*x11 - a1*b3*c3*d2*x3*x4*x8*x11 +$
 $a3*b2*c1*d3*x3*x4*x8*x11 - a2*b3*c1*d3*x3*x4*x8*x11 -$
 $a2*b1*c2*d3*x3*x4*x8*x11 - a3*b1*c2*d3*x3*x4*x8*x11 +$
 $a1*b2*c2*d3*x3*x4*x8*x11 + a1*b3*c2*d3*x3*x4*x8*x11 +$
 $a3*b2*c2*d1*x1*x6*x8*x11 - a2*b3*c2*d1*x1*x6*x8*x11 +$
 $a3*b2*c3*d1*x1*x6*x8*x11 - a2*b3*c3*d1*x1*x6*x8*x11 -$
 $a3*b2*c1*d2*x1*x6*x8*x11 + a2*b3*c1*d2*x1*x6*x8*x11 -$
 $a2*b1*c3*d2*x1*x6*x8*x11 - a3*b1*c3*d2*x1*x6*x8*x11 +$
 $a1*b2*c3*d2*x1*x6*x8*x11 + a1*b3*c3*d2*x1*x6*x8*x11 -$
 $a3*b2*c1*d3*x1*x6*x8*x11 + a2*b3*c1*d3*x1*x6*x8*x11 +$
 $a2*b1*c2*d3*x1*x6*x8*x11 + a3*b1*c2*d3*x1*x6*x8*x11 -$
 $a1*b2*c2*d3*x1*x6*x8*x11 - a1*b3*c2*d3*x1*x6*x8*x11 +$
 $a3*b2*c3*d1*x2*x4*x9*x11 - a2*b3*c3*d1*x2*x4*x9*x11 -$
 $a3*b1*c3*d2*x2*x4*x9*x11 + a1*b3*c3*d2*x2*x4*x9*x11 -$
 $a3*b2*c1*d3*x2*x4*x9*x11 + a2*b3*c1*d3*x2*x4*x9*x11 +$
 $a3*b1*c2*d3*x2*x4*x9*x11 - a1*b3*c2*d3*x2*x4*x9*x11 +$
 $a3*b1*c2*d1*x1*x5*x9*x11 - a1*b3*c2*d1*x1*x5*x9*x11 -$
 $a2*b1*c3*d1*x1*x5*x9*x11 + a1*b2*c3*d1*x1*x5*x9*x11 -$
 $a3*b2*c3*d1*x1*x5*x9*x11 + a2*b3*c3*d1*x1*x5*x9*x11 -$
 $a3*b1*c1*d2*x1*x5*x9*x11 + a1*b3*c1*d2*x1*x5*x9*x11 +$
 $a3*b1*c3*d2*x1*x5*x9*x11 - a1*b3*c3*d2*x1*x5*x9*x11 +$
 $a2*b1*c1*d3*x1*x5*x9*x11 - a1*b2*c1*d3*x1*x5*x9*x11 +$
 $a3*b2*c1*d3*x1*x5*x9*x11 - a2*b3*c1*d3*x1*x5*x9*x11 -$
 $a3*b1*c2*d3*x1*x5*x9*x11 + a1*b3*c2*d3*x1*x5*x9*x11 +$
 $a3*b1*c2*d1*x2*x5*x7*x12 - a1*b3*c2*d1*x2*x5*x7*x12 -$
 $a2*b1*c3*d1*x2*x5*x7*x12 + a1*b2*c3*d1*x2*x5*x7*x12 -$
 $a3*b1*c1*d2*x2*x5*x7*x12 + a1*b3*c1*d2*x2*x5*x7*x12 +$
 $a2*b1*c1*d3*x2*x5*x7*x12 - a1*b2*c1*d3*x2*x5*x7*x12 +$
 $a3*b2*c2*d1*x2*x4*x8*x12 - a2*b3*c2*d1*x2*x4*x8*x12 -$

$$\begin{aligned}
& a3*b2*c1*d2*x2*x4*x8*x12 + a2*b3*c1*d2*x2*x4*x8*x12 - \\
& a2*b1*c3*d2*x2*x4*x8*x12 + a1*b2*c3*d2*x2*x4*x8*x12 + \\
& a2*b1*c2*d3*x2*x4*x8*x12 - a1*b2*c2*d3*x2*x4*x8*x12 - \\
& a3*b1*c2*d1*x1*x5*x8*x12 - a3*b2*c2*d1*x1*x5*x8*x12 + \\
& a1*b3*c2*d1*x1*x5*x8*x12 + a2*b3*c2*d1*x1*x5*x8*x12 + \\
& a2*b1*c3*d1*x1*x5*x8*x12 - a1*b2*c3*d1*x1*x5*x8*x12 + \\
& a3*b1*c1*d2*x1*x5*x8*x12 + a3*b2*c1*d2*x1*x5*x8*x12 - \\
& a1*b3*c1*d2*x1*x5*x8*x12 - a2*b3*c1*d2*x1*x5*x8*x12 + \\
& a2*b1*c3*d2*x1*x5*x8*x12 - a1*b2*c3*d2*x1*x5*x8*x12 - \\
& a2*b1*c1*d3*x1*x5*x8*x12 + a1*b2*c1*d3*x1*x5*x8*x12 - \\
& a2*b1*c2*d3*x1*x5*x8*x12 + a1*b2*c2*d3*x1*x5*x8*x12, \\
& a3*b1*c2*d1*x3*x6*x8*x10 - a1*b3*c2*d1*x3*x6*x8*x10 - \\
& a2*b1*c3*d1*x3*x6*x8*x10 + a1*b2*c3*d1*x3*x6*x8*x10 - \\
& a3*b1*c1*d2*x3*x6*x8*x10 + a1*b3*c1*d2*x3*x6*x8*x10 + \\
& a2*b1*c1*d3*x3*x6*x8*x10 - a1*b2*c1*d3*x3*x6*x8*x10 + \\
& a3*b2*c2*d1*x3*x5*x9*x10 - a2*b3*c2*d1*x3*x5*x9*x10 - \\
& a3*b2*c1*d2*x3*x5*x9*x10 + a2*b3*c1*d2*x3*x5*x9*x10 - \\
& a2*b1*c3*d2*x3*x5*x9*x10 + a1*b2*c3*d2*x3*x5*x9*x10 + \\
& a2*b1*c2*d3*x3*x5*x9*x10 - a1*b2*c2*d3*x3*x5*x9*x10 - \\
& a3*b1*c2*d1*x2*x6*x9*x10 - a3*b2*c2*d1*x2*x6*x9*x10 + \\
& a1*b3*c2*d1*x2*x6*x9*x10 + a2*b3*c2*d1*x2*x6*x9*x10 + \\
& a2*b1*c3*d1*x2*x6*x9*x10 - a1*b2*c3*d1*x2*x6*x9*x10 + \\
& a3*b1*c1*d2*x2*x6*x9*x10 + a3*b2*c1*d2*x2*x6*x9*x10 - \\
& a1*b3*c1*d2*x2*x6*x9*x10 - a2*b3*c1*d2*x2*x6*x9*x10 + \\
& a2*b1*c3*d2*x2*x6*x9*x10 - a1*b2*c3*d2*x2*x6*x9*x10 - \\
& a2*b1*c1*d3*x2*x6*x9*x10 + a1*b2*c1*d3*x2*x6*x9*x10 - \\
& a2*b1*c2*d3*x2*x6*x9*x10 + a1*b2*c2*d3*x2*x6*x9*x10 - \\
& a3*b1*c2*d1*x3*x6*x7*x11 + a1*b3*c2*d1*x3*x6*x7*x11 + \\
& a2*b1*c3*d1*x3*x6*x7*x11 - a1*b2*c3*d1*x3*x6*x7*x11 + \\
& a3*b1*c1*d2*x3*x6*x7*x11 - a1*b3*c1*d2*x3*x6*x7*x11 - \\
& a2*b1*c1*d3*x3*x6*x7*x11 + a1*b2*c1*d3*x3*x6*x7*x11 - \\
& a3*b2*c2*d1*x3*x4*x9*x11 + a2*b3*c2*d1*x3*x4*x9*x11 + \\
& a3*b2*c1*d2*x3*x4*x9*x11 - a2*b3*c1*d2*x3*x4*x9*x11 + \\
& a2*b1*c3*d2*x3*x4*x9*x11 - a1*b2*c3*d2*x3*x4*x9*x11 - \\
& a2*b1*c2*d3*x3*x4*x9*x11 + a1*b2*c2*d3*x3*x4*x9*x11 + \\
& a3*b1*c2*d1*x1*x6*x9*x11 + a3*b2*c2*d1*x1*x6*x9*x11 - \\
& a1*b3*c2*d1*x1*x6*x9*x11 - a2*b3*c2*d1*x1*x6*x9*x11 - \\
& a2*b1*c3*d1*x1*x6*x9*x11 + a1*b2*c3*d1*x1*x6*x9*x11 - \\
& a3*b1*c1*d2*x1*x6*x9*x11 - a3*b2*c1*d2*x1*x6*x9*x11 + \\
& a1*b3*c1*d2*x1*x6*x9*x11 + a2*b3*c1*d2*x1*x6*x9*x11 - \\
& a2*b1*c3*d2*x1*x6*x9*x11 + a1*b2*c3*d2*x1*x6*x9*x11 + \\
& a2*b1*c1*d3*x1*x6*x9*x11 - a1*b2*c1*d3*x1*x6*x9*x11 + \\
& a2*b1*c2*d3*x1*x6*x9*x11 - a1*b2*c2*d3*x1*x6*x9*x11 + \\
& a3*b2*c3*d1*x3*x5*x7*x12 - a2*b3*c3*d1*x3*x5*x7*x12 - \\
& a3*b1*c3*d2*x3*x5*x7*x12 + a1*b3*c3*d2*x3*x5*x7*x12 - \\
& a3*b2*c1*d3*x3*x5*x7*x12 + a2*b3*c1*d3*x3*x5*x7*x12 +
\end{aligned}$$

$$\begin{aligned}
& a3*b1*c2*d3*x3*x5*x7*x12 - a1*b3*c2*d3*x3*x5*x7*x12 + \\
& a3*b1*c2*d1*x2*x6*x7*x12 - a1*b3*c2*d1*x2*x6*x7*x12 - \\
& a2*b1*c3*d1*x2*x6*x7*x12 + a1*b2*c3*d1*x2*x6*x7*x12 - \\
& a3*b2*c3*d1*x2*x6*x7*x12 + a2*b3*c3*d1*x2*x6*x7*x12 - \\
& a3*b1*c1*d2*x2*x6*x7*x12 + a1*b3*c1*d2*x2*x6*x7*x12 + \\
& a3*b1*c3*d2*x2*x6*x7*x12 - a1*b3*c3*d2*x2*x6*x7*x12 + \\
& a2*b1*c1*d3*x2*x6*x7*x12 - a1*b2*c1*d3*x2*x6*x7*x12 + \\
& a3*b2*c1*d3*x2*x6*x7*x12 - a2*b3*c1*d3*x2*x6*x7*x12 - \\
& a3*b1*c2*d3*x2*x6*x7*x12 + a1*b3*c2*d3*x2*x6*x7*x12 - \\
& a3*b2*c3*d1*x3*x4*x8*x12 + a2*b3*c3*d1*x3*x4*x8*x12 + \\
& a3*b1*c3*d2*x3*x4*x8*x12 - a1*b3*c3*d2*x3*x4*x8*x12 + \\
& a3*b2*c1*d3*x3*x4*x8*x12 - a2*b3*c1*d3*x3*x4*x8*x12 - \\
& a3*b1*c2*d3*x3*x4*x8*x12 + a1*b3*c2*d3*x3*x4*x8*x12 - \\
& a3*b1*c2*d1*x1*x6*x8*x12 + a1*b3*c2*d1*x1*x6*x8*x12 + \\
& a2*b1*c3*d1*x1*x6*x8*x12 - a1*b2*c3*d1*x1*x6*x8*x12 + \\
& a3*b2*c3*d1*x1*x6*x8*x12 - a2*b3*c3*d1*x1*x6*x8*x12 + \\
& a3*b1*c1*d2*x1*x6*x8*x12 - a1*b3*c1*d2*x1*x6*x8*x12 - \\
& a3*b1*c3*d2*x1*x6*x8*x12 + a1*b3*c3*d2*x1*x6*x8*x12 - \\
& a2*b1*c1*d3*x1*x6*x8*x12 + a1*b2*c1*d3*x1*x6*x8*x12 - \\
& a3*b2*c1*d3*x1*x6*x8*x12 + a2*b3*c1*d3*x1*x6*x8*x12 + \\
& a3*b1*c2*d3*x1*x6*x8*x12 - a1*b3*c2*d3*x1*x6*x8*x12 + \\
& a3*b2*c2*d1*x2*x4*x9*x12 - a2*b3*c2*d1*x2*x4*x9*x12 + \\
& a3*b2*c3*d1*x2*x4*x9*x12 - a2*b3*c3*d1*x2*x4*x9*x12 - \\
& a3*b2*c1*d2*x2*x4*x9*x12 + a2*b3*c1*d2*x2*x4*x9*x12 - \\
& a2*b1*c3*d2*x2*x4*x9*x12 - a3*b1*c3*d2*x2*x4*x9*x12 + \\
& a1*b2*c3*d2*x2*x4*x9*x12 + a1*b3*c3*d2*x2*x4*x9*x12 - \\
& a3*b2*c1*d3*x2*x4*x9*x12 + a2*b3*c1*d3*x2*x4*x9*x12 + \\
& a2*b1*c2*d3*x2*x4*x9*x12 + a3*b1*c2*d3*x2*x4*x9*x12 - \\
& a1*b2*c2*d3*x2*x4*x9*x12 - a1*b3*c2*d3*x2*x4*x9*x12 - \\
& a3*b2*c2*d1*x1*x5*x9*x12 + a2*b3*c2*d1*x1*x5*x9*x12 - \\
& a3*b2*c3*d1*x1*x5*x9*x12 + a2*b3*c3*d1*x1*x5*x9*x12 + \\
& a3*b2*c1*d2*x1*x5*x9*x12 - a2*b3*c1*d2*x1*x5*x9*x12 + \\
& a2*b1*c3*d2*x1*x5*x9*x12 + a3*b1*c3*d2*x1*x5*x9*x12 - \\
& a1*b2*c3*d2*x1*x5*x9*x12 - a1*b3*c3*d2*x1*x5*x9*x12 + \\
& a3*b2*c1*d3*x1*x5*x9*x12 - a2*b3*c1*d3*x1*x5*x9*x12 - \\
& a2*b1*c2*d3*x1*x5*x9*x12 - a3*b1*c2*d3*x1*x5*x9*x12 + \\
& a1*b2*c2*d3*x1*x5*x9*x12 + a1*b3*c2*d3*x1*x5*x9*x12)
\end{aligned}$$

APPENDIX D

FREE PROJECTIVE PLANE CODE

```
package pointslines;
import java.sql.Array;
import java.util.*;
import org.omg.PortableInterceptor.HOLDING;

public class Main {

/*****
* function intersect
* Input: two sets S1 and S2
* Output: set containing elements common to both S1 and S2
*
* This function returns the intersection of two sets.
*****/
    public static TreeSet intersect(TreeSet S1, TreeSet S2){
        TreeSet copy = (TreeSet) S1.clone();
        copy.retainAll(S2);
        return copy;
    }

/*****
* function getListElement
* Input: a list and an index
* Output: the element of the list at the index provided
*
* This function returns a requested element from a LinkedHashSet.
* If the list does not contain an element at the index, it returns
* the empty set.
*****/
    public static TreeSet getListElement(LinkedHashSet<Set> list,
        int index){
        Iterator iter = list.iterator();
        for(int j = 0; j < index; j++){
            if(iter.hasNext()){
                iter.next();
            }
        }
    }
}
```

```

        else{
            TreeSet empty_set = new TreeSet();
            return empty_set;
        }
    }
    if(iter.hasNext()){
        return((TreeSet)iter.next());
    } else {
        TreeSet empty_set = new TreeSet();
        return empty_set;
    }
}

/*****
* function getSetIndex                                     *
* Input: a list and an index                               *
* Output: the element of the list at the index provided  *
*                                                         *
* This function returns a requested element from a TreeSet. If the *
* list does not contain an element at the index, it returns -1. *
*****/
public static int getSetIndex(TreeSet list, int index){
    Iterator iter = list.iterator();
    for(int j = 0; j < index; j++){
        if(iter.hasNext()){
            iter.next();
        }
        else{
            return -1;
        }
    }
    if(iter.hasNext()){
        return((Integer)iter.next());
    } else {
        return -1;
    }
}

/*****
* function addPoint                                       *
* Input: points set, two indices, and a temporary set    *
* Output: void                                           *
*                                                         *
* This function adds the point determined by the two indices to the *
* points set.                                            *
*****/

```

```

    public static void addPoint(LinkedHashSet<Set> points, int index1,
        int index2, TreeSet<Integer> temp) {
        Set copy = (Set) temp.clone();
        copy.add(index1);
        copy.add(index2);
        points.add(copy);
    }

/*****
* function removePoint
* Input: points set, two indices, and a temporary set
* Output: void
*
* This function removes the point determined by the two indices
* from the points set
*****/
    public static void removePoint(LinkedHashSet<Set> points,
        int index1, int index2, TreeSet<Integer> temp) {
        Set copy = (Set) temp.clone();
        copy.add(index1);
        copy.add(index2);
        points.remove(copy);
    }

/*****
* function updateLines
* Input: lines set, lines indices, point index
* Output: void
*
* This function updates the lines set by adding the index of the
* new point to both of the lines that pass through it.
*****/
    public static void updateLines(LinkedHashSet<Set> lines,
        int index1, int index2, int ptIndex) {
        ((TreeSet)getListElement(lines, index1)).add(ptIndex);
        ((TreeSet)getListElement(lines, index2)).add(ptIndex);
    }

    public Main() {
    }

    public static void main(String[] args) {
        //initializations
        final int numLoops = 6;
        int lineAindex = 0, lineBindex = 0, numLines = 0,
            numPoints = 0;

```

```

LinkedHashSet<Set> points = new LinkedHashSet();
LinkedHashSet<Set> lines = new LinkedHashSet();
LinkedHashSet<Set> temp1 = new LinkedHashSet();
TreeSet<Integer> temp = new TreeSet();

// Initialize the first six lines
for(int i = 0; i < 5; i++) {
    for(int j = i + 1; j < 4; j++) {
        Set copy = (Set) temp.clone();
        copy.add(i);
        copy.add(j);
        lines.add(copy);
        numLines++;
    }
}

// Initialize the first four points
for(int i = 0; i < 4; i++){
    Set copy = (Set) temp.clone();
    Iterator iter = lines.iterator();
    while(iter.hasNext()) {
        Set line = (Set) iter.next();
        if(line.contains(i)){
            copy.add(lineAindex);
        }
        lineAindex++;
    }
    points.add(copy);
    numPoints++;
    lineAindex = 0;
}

// Output sizes of points and lines sets
System.out.println(points.size() + " elements in points: "
    + points);
System.out.println(lines.size() + " elements in lines: "
    + lines);

/*****
* Main Loop of the Program: *
* *
* This loop does all of the work. It compares all pairs of *
* lines to see if their intersection point is already in *
* the points set. If so, then it moves on to the next pair *
* of lines. If not, then it adds the point and updates the *
* lines. It concludes by dualizing the plane and printing *
*****/

```

```

* out updated sizes of the two sets. *
*****/
for(int k = 0; k < numLoops; k++) {
    for(int i = 0; i < numLines; i++){
        Iterator iter = lines.iterator();
        for(int j = 0; j < i; j++){
            if(iter.hasNext()) {
                iter.next();
            }
        }
        while(iter.hasNext()) {
            lineBindex = lineAindex;
            TreeSet lineA = (TreeSet) iter.next();
            while(iter.hasNext()){
                lineBindex++;
                TreeSet lineB = (TreeSet) iter.next();
                TreeSet intersection = intersect(lineA,lineB);
                if(intersection.isEmpty()){
                    addPoint(points, lineAindex, lineBindex,
                        temp);
                    numPoints++;
                    updateLines(lines, lineAindex, lineBindex,
                        numPoints - 1);
                }
            }
        }
        lineAindex++;
    }

    //swap points and lines
    temp1 = lines;
    lines = points;
    points = temp1;
    numPoints = points.size();
    numLines = lines.size();

    //Print results and reset indices
    System.out.println("numPoints = " + numPoints + " and
        numLines = " + numLines);
    System.out.println("points: " + points);
    System.out.println("lines: " + lines);
    lineAindex = 0;
    lineBindex = 0;
}
}
}
}

```