

# Deterministic construction of covering sets in projective Galois planes with small cover numbers

John Anker Corneliussen  
anker@diku.dk

December 3, 1999

## Abstract

The present work is closely related to the question of the existence of a universal constant covering number for finite projective planes. This question has been open since it was asked by the famous P. Erdős some 20 years ago. This is however a deep question, and as long as the existence problems of projective planes in general are unsolved, this question cannot be answered with a yes at all. If however the answer is no it would be sufficient to find lower bounds that enable us to construct a hierarchy of planes having strictly growing covering numbers. This is why this the projective planes within this work is limited to the planes that can be coordinatized over finite fields.

I will present the two major results of my experiments with Galois planes built over finite fields.

First there is construction of fast algorithms that select blocking sets of sizes less than  $q \log_2 q$  providing covering numbers less than the Abbot-Liu upper bound of  $4 \log_2 q$ .

Next there is a formula for blocking sets with the constant cover number 6 that holds for Galois planes built over Galois fields of characteristic 2, followed by a short proof.

Last there is the implementation issues of the algorithms and a substantial amount of data, obtained by running some of the algorithms on all projective Galois planes with a base less than 1400.

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Definitions and Theorems</b>	<b>6</b>
2.1	Basic definitions	6
2.1.1	Finite projective plane	6
2.1.2	Duality meta theorem	6
2.2	Line Cover, Blocking set and Covering Number	6
2.2.1	Point Cover of a line	7
2.2.2	Blocking Set	7
2.2.3	Covering Number	7
2.3	Finite fields	7
2.3.1	The characteristic	8
2.3.2	The multiplicative group	8
2.3.3	Generating finite fields	8
2.4	Projective Galois planes	8
2.4.1	Coordinates of points	9
2.4.2	Coordinates of lines	9
2.4.3	Dot product and Cross product	9
2.4.4	Point-line incidence	9
2.4.5	Line intersection	9
2.4.6	The affine plane and $L_\infty$	10
2.5	Main theorems	10

2.5.1	The theorem of Bezout . . . . .	10
2.5.2	Theorems of conic sections . . . . .	10
2.5.3	The theorem of Boros and Bruen-Fisher . . . . .	11
2.6	Height functions . . . . .	11
2.6.1	Height of lines . . . . .	11
2.6.2	Heights of sets of points . . . . .	11
2.6.3	The line profile of a set of points . . . . .	12
2.6.4	The height of points . . . . .	13
2.6.5	Heights of sets of points on a line . . . . .	13
2.6.6	Heights of zero sets of polynomials . . . . .	13
2.6.7	Heights of irreducible factors . . . . .	13
2.7	Elliptic curves . . . . .	13
<b>3</b>	<b>Analysis</b>	<b>14</b>
3.1	The map $N(x) = \langle x, x \rangle$ . . . . .	14
3.1.1	The fibre of $N(x) = 0$ , the radicals . . . . .	14
3.1.2	$N(x)$ when $q = 1 \pmod{4}$ . . . . .	16
3.1.3	$N(x)$ when $q = 3 \pmod{4}$ . . . . .	16
3.1.4	$N(x)$ in <i>char</i> 2 . . . . .	16
3.2	The linear group over $\mathbb{F}_q \times \mathbb{F}_q \times \mathbb{F}_q$ . . . . .	16
3.2.1	$PGL(2, \mathbb{F}_q)$ . . . . .	17
3.2.2	Targeting points or lines . . . . .	17
3.3	Conic sections . . . . .	17
3.3.1	Maximal sized sets of height 2 . . . . .	17
3.3.2	Linear transformations of conic sections . . . . .	18
3.3.3	Adding conic number two . . . . .	18
3.4	Systems of conic sections . . . . .	18
3.4.1	Disjoint conic orbits . . . . .	18
3.4.2	Conic orbits with 2 fixpoints . . . . .	19
3.5	Covering number better than $2 \log_2(q^2 + q + 1)$ . . . . .	19
3.6	Cover number 6 in characteristic 2 . . . . .	19
<b>4</b>	<b>Using the multiplicative group</b>	<b>20</b>
<b>5</b>	<b>Covering number solvers</b>	<b>21</b>
5.1	Introduction to the solvers . . . . .	21
5.2	Small covering number solvers picking 1 point at a time . . . . .	22
5.2.1	Simpler brute . . . . .	22
5.2.2	Smarter brute . . . . .	22
5.2.3	Elliptic cover . . . . .	23
5.2.4	Least height cover . . . . .	25
5.2.5	Pisinger and Illes' coverings . . . . .	26
5.3	Small covering number solvers using conic sections . . . . .	27
5.3.1	Norm fibre cover . . . . .	27
5.3.2	Hyperbola cover . . . . .	29
5.3.3	Hyperbola cover with reducer . . . . .	29
<b>6</b>	<b>Conclusions</b>	<b>30</b>
6.1	structure . . . . .	30
6.2	Does Erös' constant hold . . . . .	32

<b>7</b>	<b>Tables</b>	<b>33</b>
7.1	General information of solutions . . . . .	33
7.2	Lineprofiles of solutions . . . . .	33
7.3	Lineprofiles in char 2 . . . . .	34
7.4	Distributions of $i$ -secants . . . . .	34
<b>8</b>	<b>Implementation</b>	<b>34</b>
8.1	Language selection . . . . .	34
8.2	Primering class . . . . .	34
8.3	Finite field class . . . . .	36
8.3.1	Initialisation . . . . .	38
8.3.2	Addition and subtraction . . . . .	38
8.3.3	Multiplication and division . . . . .	38
8.3.4	Square roots and powers . . . . .	38
8.4	Plane class . . . . .	39
8.4.1	Point - line incidence . . . . .	40
8.4.2	Parametric line construction . . . . .	41
8.4.3	Miscellaneous polynomials . . . . .	41
8.4.4	Matrix class . . . . .	41

## 1 Introduction

When the good Paul Erdős left us in 1996, he had written loads of papers. These papers are still of great inspiration to many mathematicians and computer scientists all over the world. Among his publications a great number are of open problems. For a complete collection please see

<http://www.math.upenn.edu/~chung/ep.dvi>

The present work deal with one of these open problems, asking whether there exists blocking sets in finite projective planes, having covering numbers less than some universal constant. I will refer to this question through out the present text, as Erdős' problem, though we are familiar with this great mans urge to state hard problems and nasty question, so this is just another one of these that has been open for soon two decades.

The purpose of this project is to use computer simulations as a tool to gain more insight into the problem, and to reveal structure that can be used to construct deterministic algorithms, that produce blocking sets with small covering numbers. The kind of deterministic algrithms are not just simpleminded brute force traversals, as this approach could mean waiting for so long, because we simply do not know to much about the structure of the projective planes. If I in advance had considered even getting close to results very hard, I would probably have chickened out at an early stage of this project. I simply do not like the idea of struggling with neither NP-completeness nor Halting. I have learned to keep a an eye out for these kind of problems, and in this context there wasn't any bells ringing, not even once, despite the possibility of

I had a very strong belief in what we could call "the rigid structure" of the finite fields would be a more of helping hand than an enemy observing matters, somewhat like the force in Starwars... So I just needed a generic workbench talking the language of prime numbers. That is why I dug out Kernigan and Richie, polished the dragon book and slept with The C++ programming language, second edition under my pillow, and got lost from the world as we know it for quite some time.

The result of this is weird form of escapism is what I proudly can present as the implementation, that I have designed to be my friend when mining for information and beautiful structure, that could lead further on in the endless search for knowledge, not yet seen nor known. All right, it is what we could call an implementation of finite fields, with applications in projective Galois planes.

The natural language selection for the implementation was C++. There are mainly two reasons for this, the personal one for me to get to know the language, the other was the overloading of operators

and function calls in this piece of art. The latter make it possible for a dot product to look and feel like a dot product - and you might find this silly, but I think, I would have got bad dreams of brackets, if I had chosen Scheme as an example. Believe me ! I have been seriously relieved by this feature in the experimental phase of the project, so it really does matter.

The approach that i have used is that of algebraical geometry over finite fields, thus making results valid for some projective planes. I am very well aware that this limits the extendability of my findings, but it is on the other hand possible to bring in coordinates and algebra as a convenient tool, but then again stresses the need for the calculator.

It has been possible to keep concepts like lines as zero sets of degree-1 polynomials and conic sections as zero sets of irreducible degree-2 polynomials, this way giving a well-founded carpet under the intuition and flightly thought doing the actual survey.

In fact some of the phenomenons observed are so strange, that it is totally counter intuitive. Try for instance, to imagine what we refer to as a circle, can have the property, that all of it's tangents intersect in a single point. This is the case when we are dealing with the fields of characteristic 2.

I have deliberately not presented a picture of a line nor a conic section, because it is almost impossible, not to give a false impression, of what is going on. Keep in mind these planes are not metric spaces, so what we under euclidian circumstances would consider as zero length is a perfect conic section generator for these planes, and what we connect with the unit circle is completely singular for these planes.

We are very far away from the normal interpretation of even a line. So the most "normal" way to describe phenomenons like lines is that of zero sets ! The geometric intuition can only rely on this kind of abstraction of a line. That is why you will have to see finite fields as your cute little friends, equipped with their  $p = 0$  property, making intuition go haywire.

So illustrating conic sections intersecting lines is a joke. You would probably not even be able to distinguish the line from the conic section in a plane over a prime field. This could be represented in ordinary 2-d, but the associations drawn from such an illustration would rather be that of cubic art or group-tables, than lines, because of the omnipresent cyclicity of primefields, speaking a language of its own.

It is the fact that conic section and line intersection obey basically the same rules as that of projective geometry of reals things like a line is incident with a conic section in either 0,1 or 2 points. This property is described by the theorem of Bezout, when interpreting lines and conic sections as zero sets of polynomials.

It is maybe too ambitious of me to go ahead and solve the open problem, but still i hope that this computational study will bring new insight to the nature of the problem, and that it will uncover some structural properties, this way supporting the theoreticians with some empirical results and perhaps fragments of an idea that will close the problem for good.

I have selected to study Galois planes, that are planes over finite fields, because these planes can be equipped with coordinates. It has come to my belief that such a constant that Erdős suggested does not exist. After studying projective Galois planes for two years, I would rather propose the lower bound is logarithmic in the number of points in the plane for Galois planes. Of course I can't prove it, but it seems that all my experiments hit a logarithmic wall.

There are in fact great many mathematicians that have been puzzled by finite projective planes, and yet not much is known. Basic properties like when finite projective planes exist, is still not fully known. It is still an open question if there exist a projective plane having 111 points. A canadian group has done exhaustive search of  $S_111$ , and for those that don't what this is, it is the permutation group of 111 elements. For the computer scientists it is the same as finding all Hamiltonian paths in a graph with 111 vertices. This task is exponential in both time and space using dynamic programming, or else we have to visit every 111! computational leaves. This problem is known as the projective plane base 10 problem. By the way the guys found that base 10 did not exist, but that is not a proof. This is merely just an indication, because what if the program had a bug, even if it didn't, then what about flaws in the cache ram or main memory, leading to erroneous conclusions ? The message is clear we use computers to help us find the appropriate things, and then use math to prove it.

Let us not go deep down in existential problems and nihilistic not knowing philosophical discussions. Let us instead focus on the problem of Paul Erdős and the things that really are known and where to put my work.

There are scientists that have studied this problem in the past twenty years. This is people like E. Ughi, who has done a lot of work with conic sections in projective Galois planes. She has proved that the union of a fixed number of conic sections cannot cover all of the lines of the plane, when the underlying field get large enough, and is not of characteristic 2. This observation tell us that Erdős' proposed constant covering number cannot be obtained in terms of conic sections.

On the other hand she proved that a logarithmic number of suitably chosen conic sections do form blocking sets. Later Abbott and Liu tightened the upper bound giving a more explicit logarithmic upper bound for conic section coverings.

Recently David Pisinger and Tibor Illes made a project, that showed that some of the best known upper bounds on Galois planes could be brought down. Their results were however obtained by using sophisticated A.I., never the less it was their work that sent me in orbit, studying these matters. They were concerned about finding minimal blocking sets with minimal covering numbers, and their experiments were very fruitful. They made computations for all projective Galois planes up to base 607. Their algorithm were however based upon simulated annealing, and to produce these results they have used 48 days around the clock of computation time on powerful computers.

What is interesting is that my work show deterministic algorithms, that for some instances, produce better results, and that much faster ideed. One of my conic section solvers will do all Galois planes less than 600 in a matter of hours. Of course instances like the Galois plane base 3125 take a looong time. This particular instance took my machines 72 hours.

The algorithms I have constructed, are not just interesting because they are deterministic, but they come up with good solutions, that are close to Pisinger and Illes' results, and always are better for some planes. So one of my contribution might be something like, yes we can see that it is hard to go below some limit, and yes we can get close to that limit fast. But still considering Galois planes only is only worth something regarding Erdős' constant, if we can prove a the lower bound for covering numbers, using these kind as a ram to break the ice. So this computational study might help shed some light over fairly large instances of projective planes. The largest plane I have thrown my algorithms at is a plane having 4097 points incident with each line, finding the magic 12287 points out of 16781313 possible in linear time. This is math at work.

Though this project seem like a math project, it is a major project in computer science, that relate to math, and the computer scientific part of this project, is not only bringing the algebra of finite fields to work inside a computer, though this is supposed to be, and has been, a major part of this project, but also the process of taking some wild combinatorical problem, and use systematic and empirical effort to obtain deterministic algorithms and solvers.

We have to keep in mind that even if this problem seem to be a toy problem, that cannot save neither the whales nor the world as we know it, then toy problems can be very useful for beleivers in reductionism. We seem to understand larger problems in terms of reduced forms. This way we classify problems by the equivalent toy problems of our fancy.

Using math is not bad when attacking combinatorical problems. It seem to me that whensoever we have a hard problem, we can give it a mathematical formulation, and use the math to attack the problem further. Anyway there is a rich amount of information written by mathematicians, that could be useful for computer scientists for many purposes.

The present project is at least to me a great example of math and computer science walking hand in hand, - not just computer science using math without contributions to math.

Some of the math used in this work is reasonably harsh and hard to swallow. Every now and then i might expect too much from the reader, that is expected to be another graduate student, albeit I have made serious effort to reduce the math to what is necessary, and supply things with comments and remarks. In despite of the remarks and all, I might have gone too far at times, and let me make an attempt to justify it. I might have gone blind on what I think is trivial stuff, because I have been studying algebraic geometry, algebraic numbertheory and taken U.\* name it of algebra courses. Not that I claim to be some kind of expert in this field, but some times i seem to forget the trivial meaning of trivial.

---

\*Read: any course given by Professor Christian U. Jensen at Mathematical institute in Copenhagen

Before we go on into matters, I would like to say thank you to the creators of the egcs compiler, and the founders of the GNU public license concept for the opportunity to work with such a rich garden of tools, and thanks to all the guys out there, that has been bringing nourishing clear water to a guy walking in a wasted and almost empty desert of despair.

## 2 Definitions and Theorems

### 2.1 Basic definitions

#### 2.1.1 Finite projective plane

**Definition 1** Let  $\Pi(q)$  denote a collection of points and subsets of these points. Each of these subsets is referred to as a line, and we define the intersection of two lines as the intersection of the corresponding subsets.  $\Pi(q)$  is assumed to satisfy four axioms:

- (i) Each pair of distinct points lies on a unique line.
- (ii) Each pair of distinct lines intersects at a uniquely determined point.
- (iii) There are four points of which at most two lie on the same line.
- (iv)  $\Pi(q)$  has a line with  $q + 1$  points, where  $q$  is a given integer such that  $q \geq 2$ .

A collection  $\Pi(q)$  of points and lines satisfying all of the above axioms is called a finite projective plane of order  $q$ . It can be deduced (see for example, Kárteszi [Kárteszi]) from these axioms that

- Each line has  $q + 1$  points.
- There are  $q + 1$  lines through each point.
- $\Pi(q)$  has in total  $q^2 + q + 1$  points and  $q^2 + q + 1$  lines.

#### 2.1.2 Duality meta theorem

Sometimes it is nice to know that we can put a dual angle to a problem, and in projective planes in general we such a dual relation between points and lines. To understand the extend of duality take deep thought, but it is often very useful, and can serve as a major shortcut for some problems, and ensure that only one proof is needed instead of two for others.

**Theorem 1** Let  $S$  be any statement about projective planes, and let  $S^*$  be its dual statement defined by replacing each occurrence, in  $S$ , of the word “line” with the word “point”, and each “point” by “line”. Then if  $S$  is valid for all projective planes then  $S^*$  also holds for all projective planes. [Albert-Sandler, p.13]

## 2.2 Line Cover, Blocking set and Covering Number

Let us get a bit closer to something that look and feel like something, that is directly linked to Erdős problem. We have to understand what the above stated concepts cover, in order to understand Erdős question, now that we know what a projective plane is. The line cover are simply the lines incident with a union of points. It is a map from arbitrary sets of points into a collection of sets of lines in the powerset of lines. A set of points is called a blocking set, when it as a set, has all lines as line cover, and yet does not contain the all points of any line. We disinguish between the covering number of a blocking set and the covering number of a plane. If a blocking set in a finite projective plane has covering number  $\gamma$ , and the blocking set has minimal covering number of all covering sets in this particular plane, then the plane has covering number  $\gamma + 1$ . In general I use the term covering number with respect to blocking sets, and another term called property  $B(\gamma + 1)$  with respect to the general property of the plane, or a family of planes. It is as important to distinguish here, as it is to distinguish problem instances from the general problem.

**Definition 2** *The line cover of a single point  $p$  is the set of lines in  $\Pi$  incident in  $p$ . For a set  $S$  consisting of  $k$  points the line cover of the union of points is defined to be the union of the line covers of the points.*

$$LCOV(\{p\}) =_{def} \{l \in \Pi | p \text{ incident with } l\}$$

$$LCOV(S) =_{def} \bigcup_{p \in S} LCOV(\{p\})$$

### 2.2.1 Point Cover of a line

Applying the duality meta theorem thm. 1 to the LCOV functor we get a definition a dual functor mapping lines into points.

**Definition 3**

$$PCOV(\{l\}) =_{def} \{p \in \Pi | l \text{ incident with } p\}$$

$$PCOV(S) =_{def} \bigcup_{l \in S} PCOV(\{l\})$$

### 2.2.2 Blocking Set

**Definition 4** *A set  $B$  of points in  $\Pi$  is called a blocking set when  $LCOV(B)$  is the set of all lines of  $\Pi$  and  $B$  does not contain a line.*

### 2.2.3 Covering Number

**Definition 5** *The covering number  $\gamma(B)$  of a blocking set  $B$  is defined as*

$$\gamma(B) = \max_{l \in \Pi} (|B \cap PCOV(\{l\})|)$$

*I will refer in to this  $\gamma(B)$  in the section with the presented solvers, as the “ $\gamma$ -value” leaving the actual found blocking set to be implicitly mentioned.*

**Definition 6** *A finite projective plane is said to have property  $B(k)$  if there exist a blocking set  $S$  having covering number less than  $k$ . This represent a notation after Bernstein with origin in work done in two-colouring of hypergraphs.*

## 2.3 Finite fields

This is not supposed to be a lecture in abstract algebra, nevertheless we will have to face this important notion of finite fields. A field is a domain, where you can do computations like  $+$ ,  $-$ ,  $*$  and  $/$  with anything but zero. The distributive laws hold for fields. Fields have cousins called rings in which the distributive laws also must hold. Rings and fields are related through homomorphisms over maximal ideals. Fields are also rings, just a bit out of the ordinary. For the division to work, the ideals in a ring have to be trivial. It has to be an integral domain, which mean that if  $a * b = 0$  then either  $a$  or  $b$  is equal to zero. Common rings that everybody know is quotient rings of the natural numbers. The rings of remainders modulo a prime number are all finite fields, and make out the simplest forms of finite fields. So the next chunk of definitions are related to finite fields and properties, that is used actively within the project.

**Definition 7** *A finite field is a field with a finite number of elements. It is denoted  $\mathbb{F}_q$  where  $q$  is a prime power.*

### 2.3.1 The characteristic

**Definition 8** *The characteristic of a Galois field is defined as the additive order of 1. it is denoted char.*

### 2.3.2 The multiplicative group

**Theorem 2** *The multiplicative group of a finite field is cyclic. [Jensen, p.3.46]*

**Remark 1** *A cyclic group has a generator, this element can be used as an iterator, so that every element is equal to a unique number of compositions of the generator with itself. This can be useful for speeding up multiplication and division in finite fields.*

### 2.3.3 Generating finite fields

To get further than merely just using planes base prime numbers, that of course might be the ace up the sleeve hunting down Erdős constant, we might as well have a look at the other planes built over the rest of the finite fields, that is why concepts like irreducible polynomial, polynomial ring and maximal ideals are taken out of the algebra books and put into action, so that we may see more instances of projective planes, with a convenient representation.

**Definition 9** *A polynomial  $p(x)$  over a field  $K$  is said to be irreducible in  $K[X]$  if  $p(x)$  has no non-trivial divisors in  $K[X]$ , and  $p(x)$  is not a constant, where trivial divisors are constants and  $p$  itself.*

Finite fields are generated as splitting fields over  $\mathbb{Z}/p\mathbb{Z}$ . We generate a maximal ideal in  $\mathbb{Z}/p\mathbb{Z}[X]$  by selecting an irreducible polynomial  $f$  of degree  $n$  as generator. Hence we note that factorizing a ring over a maximal ideal yields a field, and we generate the field as  $\mathbb{Z}/p\mathbb{Z}[X]/(f)$ . We may picture the elements as polynomials over  $GF(p)$  with degree less than  $n$ . The arithmetic operations are inherited from the polynomial ring. The addition in the field is  $p$ -cyclic and the multiplication is  $p^n - 1$ -cyclic. By evaluating these polynomials in  $p$  we obtain a map from the finite field that is 1-1 onto the natural numbers  $0 \leq k < p^n$ , and this is a convenient and adequate representation inside a computer.

**Example 1**  *$GF(4)$  is equal to  $GF(2)[X]/(x^2 + x + 1)$*

*The four polynomials are  $0, 1, x$  and  $x + 1$ . In this case the multiplicative group has 3 elements, and can be generated by  $x$  or  $x + 1$ .*

## 2.4 Projective Galois planes

In this section I will introduce the customary of normalized coordinates, by introducing 3-d finite fields, and do projections along lines. And pick some points as canonical representatives for those of the 3-d lines, this way we obtain the projective 2-d points. As points are represented as 3-d lines, projective 2-d lines are projections of 3-d planes, so naturally we see point-line incidence as a 3-d zero space problem. The zero space of a plane in 3-d is one dimensional, and correspond to a point in the projective plane. Similarly the zero space of a 3-d line is two dimensional, corresponding to the lines incident with the point. These concepts are stressed forward so you don't have to get confused over the amount of abstract nonsense. As I might have said "Yes, linear algebra is useful again !".

**Definition 10**  *$PG(2, p^n)$  is the formal notation for a projective Galois plane. A Galois plane has projective coordinates in  $GF(p^n)$ .*

Galois planes are projective planes that can be constructed over a Galois field by using the lines from  $(0,0,0)$  in a three dimensional vectorspace as representatives for the points of the Galois plane.

So we might view the points of the projective plane as equivalence classes of points in  $\mathbb{F}_q \times \mathbb{F}_q \times \mathbb{F}_q$ . This means that for  $x_i, y_i$  and  $c$  in  $\mathbb{F}_q$  we have

**Definition 11**

$$(x_1 : x_2 : x_3) = (y_1 : y_2 : y_3) \iff c(x_1, x_2, x_3) = (y_1, y_2, y_3) \neq (0, 0, 0) \text{ for some } c \neq 0$$

### 2.4.1 Coordinates of points

Normalized representatives are in my work selected as follows :

**Definition 12** *Projective coordinates are obtained from coordinates in  $\mathbb{F}_q \times \mathbb{F}_q \times \mathbb{F}_q$  as equivalence classes using a normalisation scheme. Given  $(x, y, z) \in \mathbb{F}_q \times \mathbb{F}_q \times \mathbb{F}_q$  then*

- if  $z \neq 0$ , the normalized coordinates are  $(\frac{x}{z}, \frac{y}{z}, 1)$
- if  $z = 0 \wedge y \neq 0$ , the normalized coordinates are  $(\frac{x}{y}, 1, 0)$
- if  $z = 0 \wedge y = 0$ , the normalized coordinates are  $(1, 0, 0)$

### 2.4.2 Coordinates of lines

Lines in a projective Galois plane are projections of the span of two different lines through origo in  $\mathbb{F}_q \times \mathbb{F}_q \times \mathbb{F}_q$ . We represent coordinates for lines with the projection of the normal to this plane in origo.

**Definition 13** *The line  $l$  with coordinates is given by the equation*

$$PCOV((a : b : c)_{line}) = \{(x : y : z)_{point} \in \Pi \mid ax + by + cz = 0\}$$

### 2.4.3 Dot product and Cross product

**Definition 14** *The dot product  $\langle x, y \rangle$  for sets of coordinates is defined as*

$$\langle (x_1, x_2, x_3), (y_1, y_2, y_3) \rangle = x_1y_1 + x_2y_2 + x_3y_3$$

*note that it is only well defined modulo def.12.*

**Definition 15** *The cross product  $x \times y$  where  $x \neq y$  is defined as*

$$(x_1, x_2, x_3) \times (y_1, y_2, y_3) = (x_2y_3 - y_2x_3, y_1x_3 - x_1y_3, x_1y_2 - x_2y_1)$$

### 2.4.4 Point-line incidence

Definition 14 is useful to establish the point-line incidence so that

**Remark 2** *It is obvious from the definition of the dot product, that the point  $P : (p_1 : p_2 : p_3)$  is incident with the line  $L : (l_1 : l_2 : l_3) \iff \langle P, L \rangle = 0$*

### 2.4.5 Line intersection

We can use the cross product to determine line intersection, since the crossproduct gives a representative of the normal of the plane generated by two different representatives in  $\mathbb{F}_q \times \mathbb{F}_q \times \mathbb{F}_q$ , so

**Definition 16** *Given  $L_1 \neq L_2$  then  $L_1$  and  $L_2$  both incident with a point  $P$*

$$P = L_1 \times L_2 \iff \langle P, L_1 \rangle = 0 \wedge \langle P, L_2 \rangle = 0$$

*$P$  will be normalized to its canonical representative when needed.*

It is clear that the duality concept applies in the following manner for finding the line connecting two points.

**Definition 17** *Given  $P_1 \neq P_2$  then  $P_1$  and  $P_2$  are both incident with a line  $L$*

$$L = P_1 \times P_2 \iff \langle L, P_1 \rangle = 0 \wedge \langle L, P_2 \rangle = 0$$

*$L$  will be normalized to its canonical representative when needed.*

### 2.4.6 The affine plane and $L_\infty$

It will prove useful to distinguish those points having a third coordinate of zero from the others. We see that these points form a line with coordinates  $(0 : 0 : 1)$  and we will denote this line  $L_\infty$ . The rest of the points form a two-dimensional vectorspace over  $\mathbb{F}_q$  namely the plane  $z = 1$  in  $\mathbb{F}_q \times \mathbb{F}_q \times \mathbb{F}_q$ . We denote this part of the plane the affine part of the plane.

**Definition 18** *The line  $L_\infty$  is defined to be the line  $(0 : 0 : 1)$ , consisting of the points  $(x:y:0)$*

**Definition 19** *The affine part of  $\Pi$  is defined to be points  $(x : y : 1)$ , namely the points of  $\Pi$  - the points of  $L_\infty$*

**Remark 3** *One can picture these projective planes as an “ordinary” two-dimensional plane , the affine part of the plane, equipped with an extra line consisting of vanishing points for any line through origo. Introducing this concept we can slowly bring on the ordinary slope equivalence for parallel lines, determined by common incidence with the different points incident with  $L_\infty$  and we have brought the notion of direction into the game. We now see that all affine points are associated with exactly one line in each possible direction. This way we can use the point on  $L_\infty$  as an iterator for parametric production of lines from a point, because degree-1 polynomials and constants behave like they do in a field, in our case a finite field. It is very convenient because we can now construct the line cover of a single point in linear time. That is of matter since we in general cannot effort to use quartic space in the planes base numbers. So for instance Pisinger and Illes machine used a compact representation using difference sets, but paid the price of linear time as well to add values to the difference sets. The notion of coordinates enable both line intersection and point line incidence as constant time operations, without extra consumption of memory.*

## 2.5 Main theorems

### 2.5.1 The theorem of Bezout

This theorem is very important, since it put an upper bound on the covering number of a set of points obtained as the zero set of a polynomial. Note that a line in a projective plane can be seen as the zero set of a degree-1 polynomial.

**Theorem 3** *Two non-zero polynomials  $f(x, y)$  and  $g(x, y)$  having no non-trivial common divisors, have at most  $\deg(f) * \deg(g)$  common zeroes.*

**Remark 4** *This formulation of the theorem of Bezout holds for non algebraically closed fields. An algebraically closed field is a field where all polynomials factorize into degree-1 polynomials. The complex numbers is an example of an algebraically closed field. in this case a stronger version of thm. 3 applies, that is replacing the phrase “at most” with “exactly”.*

### 2.5.2 Theorems of conic sections

A conic section can be obtained as the zero set of an irreducible degree-2 polynomial, so the points of a union of  $k$  conic sections can be obtained as the zero set of the product of  $k$  irreducible degree-2 polynomials. By using thm.3 we see that there are at most  $2k$  points incident with any line of  $\Pi$ . Hence we obtain the covering number  $2k$  for these configurations.

**Theorem 4** *The union of  $c \log q$  suitably chosen conics form a blocking set. [Ughi]*

**Theorem 5** *The  $c$  in theorem 4 can be tightened so that  $c < \frac{2}{\log 2}$  when then plane observed is a Galois plane. [Abbot-Liu]*

**Theorem 6** *A union of  $c$  conic sections in  $PG(2, q)$ ,  $q$  odd, never generates a blocking set when  $c$  is constant and  $q$  is large enough. [Ughi]*

**Remark 5** *It is evident from thm.6, that unions of conic sections cannot produce blocking sets with constant cover number. So Erdős' question cannot be confirmed using conic sections. But the results of Pisinger and Illes (see [Pisinger-Illes]) using meta-heuristics to tighten upper bounds on covering numbers, seems to hit the barrier  $c$  as in thm.5. If this upper bound also proves to be the lower bound, then it will be nice to have explicit fast construction schemes for solutions.*

### 2.5.3 The theorem of Boros and Bruen-Fisher

These two theorems represent exceptions from the general pattern observed, that the upper bound on covering numbers seem to be logarithmic in the base of the planes. These results are related to the additive group in the underlying fields. The additive group in a finite field is also a  $\mathbb{F}_p$  vectorspace. So using the homomorphism  $x \rightarrow x^p - x$  in various forms to make the structure collapse. These results are brought in because they represent known upper bounds. These constructions cannot be made using conic sections. They are hard to find in a random search, and they represent a kind of commonness along with  $\mathbb{F}_p$  always being a subfield of  $\mathbb{F}_q$  so this is highly specialized property, that is easy to miss for a generic solver, like that of Pisinger and Illes, that used 0-1 matrices as representation of the field. In my opinion coordinates are most useful for exactly these case.

**Theorem 7**  *$PG(2, 3^s)$  has property  $B(5)$  [Bruen-Fisher]*

**Theorem 8**  *$PG(2, p^s)$  has property  $B(p + 2)$  [Boros]*

**Remark 6** *Though the thm.8 yield solutions better than conic section coverings when the power of  $p$  is large enough, there will always exist a prime number larger than  $p^n$ , making the conic section covering of thm.5 the better solution.*

## 2.6 Height functions

The purpose of height functions is to formalize the counting of points incident with lines. I will introduce 3 height functions. The local height of a set of points on a specific line, and the global height of a set points are relaxations of covering number of blocking sets, to give meaning to covering numbers for non-blocking sets. We use heights to monitor the fitness of the selected as it grows from on the global level, and when considering a point for inclusion, things only happen locally, on the lines incident with the considered point. In the implementation the height functions are used to give constraints and to give guidelines for both point inclusion and point removal from a selected set.

### 2.6.1 Height of lines

I will describe the global height in terms of the local height. This local height  $H$  of  $l$  is defined relative to a set of points  $S$  and will be denoted  $H_S(l)$ . This notation covers counting the size of  $S \cap PCOV(l)$ , and it is of course bounded by the number of points on a line.

**Definition 20** *Height of a line  $l$  with respect to a set of points  $S$  is defined as*

$$H_S(l) =_{def} |S \cap PCOV(l)|$$

We use the  $PCOV$  functor because we do a straight forward set intersection on sets of points in  $\Pi$  (see def.3).

### 2.6.2 Heights of sets of points

Let me define the height function  $H$  that goes from a set of points,  $S$ , into the natural numbers, as the maximum number of points incident with any line.

**Definition 21** *The global height of a set of points  $S$  is defined in terms of the local height in def.20.*

$$H(S) =_{def} \max_{l \in \Pi} (H_S(l))$$

**Remark 7** *Note that  $H(S) \leq q + 1$  with equality if and only if  $S$  contains a line. In other words, the height is the maximum number of points from  $S$  incident with a line of  $\Pi$ . This way the blocking set  $B$  has  $H(B)$  equal to the covering number of  $B$ . Understanding the growth of the height function is of major interest when selecting the points of a blocking set with small cover number.*

### 2.6.3 The line profile of a set of points

Let us consider a set of points  $S$  in the plane. The line profile of the set  $S$  is an equivalence relation on the lines of the projective plane, so that the lines  $l_1$  and  $l_2$  are equivalent if and only if  $H_S(l_1) = H_S(l_2)$ . The profile count the number of lines in each equivalence class. You might find it awkward to bring in such a definition as this, but please remember that the notion of generalisation is a useful tool. This particular profile help us keep track of what happen to the line cover of a set of selected points, when we add in some extra. What I do is exactly building sets from scratch, so I need some kind of view of the fitness of my sets. Further more this particular profile show the shape of the distribution of the point-line incidence, so looking at it help to distiguish a blocking set consisting of relatively many or relatively few points. In terms of this kind of profile in order to have a minimal blocking set with a minimal, we would like to see a distribution that skew to the left as much as it can, and have the right hand tail as long to the left as as possible. The more the profile is skewed to the right the poorer is the prospect, and that is both concerning size and covering numbers.

**Definition 22** *A tangent to a set of points  $S$  is a line incident with exactly 1 point in  $S$ .*

**Definition 23** *A secant to a set of points  $S$  is a line incident with exactly 2 points in  $S$*

**Definition 24** *An  $i$ -secant of a set of points  $S$  is a line  $l$  where  $H_S(l) = i$*

**Definition 25** *Given a set of points  $S$  the line profile of  $S$  is defined as a list of pairs  $(i, n_i)$  where  $n_i$  is the size of the set of  $i$ -secants of  $S$ .*

**Example 2** *As an example a conic section has the profile*

- 0:  $\frac{q(q-1)}{2}$  Non hit lines
- 1:  $q + 1$  The tangents of the conic
- 2:  $\frac{q(q+1)}{2}$  The secants of the conic

*To explain the numbers in the profile we start with the secants. There are  $q + 1$  points of the conic section, since no three points are colinear we have  $q$  lines in each point connecting to the  $q$  other points, hence we have  $\sum_{i=1}^q i$  secants. The last line in each point is then obviously the tangent, hence we get  $q + 1$  tangents. Of course the rest of the lines are the uncovered lines.*

While I am at the apparently ordinary geometrics, let me please define what I mean, when I speak of asymptotes to a set, that could bear the risk of being misleading, when used in intuitive ways. An **asymptote** to a set of points  $S$  is a tangent to  $S$ , that is incident with  $S$  in a point that is incident with  $L_\infty$ .

**Remark 8** *The profile is a distribution of the lines by the number of their intersection points with  $S$ . It is worthwhile to notice that the profile of a set is invariant under linear transformation due to the properties of points and lines under full rank linear transformations. Those are the  $3 \times 3$  matrices with a non zero determinant transforming points straight forward, and lines as their homogeneous degree-1 form, or as usual line coordinates, through the inverse matrix used for points. (see 3.2).*

### 2.6.4 The height of points

Considering  $H(\{p\})$  is not very interesting since we know that the line cover of a single point consists of  $q + 1$  lines, so let us instead consider a set  $S$  of selected points and a point  $p \in \Pi - S$ . Then I will define another useful height function  $h_S(p)$  mapping points outside of  $S$  into the natural numbers  $0..q$  as

**Definition 26** *Let  $S$  be a set of points in  $\Pi$  and  $p$  a point in  $\Pi - S$ . Then the height of  $p$  relative to  $S$  is defined to be the maximal number of points in  $S$  incident with a line through  $p$ .*

$$h_S(p) =_{\text{def}} \max_{l \in LCOV(p)} (H_S(l))$$

**Remark 9** *The notion of point height is interesting because if  $h_S(p) < H(S)$ , then  $H(S \cup \{p\}) = H(S)$ . This gives a formal description of points that can be added to a set without violating the global height, and hence be used constructively to select low height points for inclusion.*

### 2.6.5 Heights of sets of points on a line

In order to understand line covers in general I think it is interesting to investigate basic geometric configurations, as that of the line cover of a single point. Let's put  $k$  points,  $p_1..p_k$  on a line  $l$  and see what the line profile looks like (see def.25). Since the line cover of a single point is  $q + 1$  different lines, and  $p_i$  are stacked on  $l$  then  $LCOV(\{p_i\}) \cap LCOV(\{p_j\}) = l$  for all  $i \neq j$ . This means that we get  $qk$  lines with a height of 1 and 1 line with a height of  $k$ . This configuration is useful when increasing local height on  $qk$  lines with at most 1 in a controlled manner.

### 2.6.6 Heights of zero sets of polynomials

Having some homogenous polynomial we decompose it into its irreducible components. If none of these irreducible components are of degree 1, then no line is contained in the zero set of the polynomial. Using the theorem of Bezout (see thm.3) we see that the zero set of this class of polynomials will have a height  $H$  that is less than or equal to the degree of the polynomial. It is obvious that the height  $H$  is equal to  $q + 1$  if the polynomial has at least one degree-1 factor.

### 2.6.7 Heights of irreducible factors

Let us for a moment focus on some irreducible factor of a polynomial. We then notice that the line cover of a zero of multiplicity  $i$  is the same as the line cover of any other point namely  $q + 1$  lines through the point. This means that we can place an upper bound on the height of any zero set of a polynomial with irreducible factors of degree  $\geq 2$ , namely the sum of the degrees of the irreducible factors, since we can use the product of each of the irreducible factors in the power of 1 to obtain the points.

**Remark 10** *This observation makes it interesting to investigate zero sets of irreducible polynomials as basic building blocks to generate blocking sets, because we can control the growth of the height function, using Bezout.*

## 2.7 Elliptic curves

Like the conic sections being the roots of 2. degree irreducible polynomials, we can consider the roots of irreducible 3. degree polynomials in  $\mathbb{F}_q[X][Y]$  to generate sets of points with heights of at most 3. In my implementation I have been concentrated on the perturbations of the curve  $Y^2 = X^3$ , namely  $Y^2 = X^3 + aX + b$ , where  $a, b \in \mathbb{F}_q$ . Turning this polynomial into a homogenous 3. degree polynomial in  $\mathbb{F}_q[X][Y][Z]$  we obtain  $Y^2Z = X^3 + aXZ^2 + bZ^3$ . The zero sets of these polynomials give us what I will refer to as elliptic curves in our projective planes, in most cases when  $\text{char}(\mathbb{F}_q) \notin \{2, 3\}$ .

**Remark 11** *I have selected to experiment with elliptic curves in particular, because they are known to carry a group structure based upon line intersection. So it seemed natural to me to check out these relatively few typical perturbations. This strategy was in fact the first strategy, that for sure performed below the Abbott-Liu bound.*

### 3 Analysis

This section is a in which I relate some of my findings to math, and give proof to some of the most important results. This way consolidating the construction of the first of my conic section solvers, that is based upon the fact that  $(0,0,0)$  is not the only solution to the equation  $(x^2 + y^2 + z^2)$ , and that the the affine fibres of this, what we in  $\mathbb{R}^3$  would consider a norm, in fact consist of conic sections, meaning that  $x^2 + y^2 + -1$  is irreducible, was a great relief to me, because it was important to have at least one to transform around, since conic sections have line cover properties, as shown in ex.2, then it was a great joy to find a system of conic sections, that even could produce covering numbers less than the Abbott-Liu upper bound. I have decided to include the observations on the norm map mapping into squares, because it uncover a weird relation between the cross product and the dot product. These proofs are not just specific to the zero norm conic, but all conic sections, due to the transformability of homogenous polynomials into between those that decompose into irreducible factors of exactly the same degrees. Note that in the characteristic of  $p$  we can use what is known as freshmans rule on  $p$ th powers, meaning that  $(a + b)^p = a^p + b^p$ . This make the norm break in the characteristic of 2. But there are other spooky things happening in these planes like the common tangent incidence, that introduce a ghost point, that could be swapped with any other point of the conic section, still having the same line profile. But let us pick it up and kick it ...

The fibres of a map  $F$  that goes from the domain  $A$  to the domain  $B$  are the equivalence classes of original sets of  $b \in B$ . We say that  $a \in A$  belong to the fibre of  $b \in B$  with respect to  $F$  if and only if  $F(a) = b$

#### 3.1 The map $N(x) = \langle x, x \rangle$

The map  $N(x) = \langle x, x \rangle$  is the dotproduct of a point with itself. I have made experiments with the configurations of points under this map. The map goes from  $\mathbb{F}_q \times \mathbb{F}_q \times \mathbb{F}_q$  into  $\mathbb{F}_q$ .

What I discovered was that the fibres of  $\mathbb{F}_q$  has some interesting geometric properties. The fibres contain systems of conic sections for odd  $q$ . When  $q$  is congruent to 1 modulo 4 we obtain a hyperbolic system, and when  $q$  is congruent with 3 modulo 4 we obtain an ecliptic system.

The method i used was to generate a frequency table at first to do mining for regularities. This frequency table simply counts the number of points in each fibre. As a data structure this table is used to sort all the points so we actually get our hands directly on the fibres for closer examination.

##### 3.1.1 The fibre of $N(x) = 0$ , the radicals

**Definition 27** *The fibre of 0 under  $N$  will be called the radicals of the plane. That is the set of projective coordinate sets  $(a : b : c)$  where  $a^2 + b^2 + c^2 = 0$ . We speak of radical points and radical lines, refering to the radical property of the coordinate sets.*

**Lemma 1** *There is only one radical point incident with a radical line*

**Proof** The radical line  $r_1$  has at least 1 radical point namely  $r_1$ . Let us assume by contradiction that there exist another radical point  $r_2$  incident with the line  $r_1$ , but then the radical line  $r_2$  is also incident with  $r_1$ , hence we have two different lines through the points  $r_1$  and  $r_2$ , and this can not be true in a projective plane, since a line incident with 2 different points is uniquely determined.  $\square$

**Corollary 1** *Given two different radical points  $r_1$  and  $r_2$  then  $\langle r_1, r_2 \rangle \neq 0$ .*

**Theorem 9** *When  $\text{char}(\mathbb{F}_q) \neq 2$  the radicals form a conic section.*

**Proof** Consider three different radicals,  $(x_1 : y_1 : z_1)$ ,  $(x_2 : y_2 : z_2)$  and  $(x_3 : y_3 : z_3)$ . Assume that they are all incident with one line. Under this assumption the the matrices having, these points as coloumns or rows will have a determinant equal to zero.

$$\det \left( \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix} \right) = 0 \Rightarrow \det \left( \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{bmatrix} \right) = 0 \Rightarrow$$

Using that  $x_1^2 + y_1^2 + z_1^2 = 0$ ,  $x_2^2 + y_2^2 + z_2^2 = 0$  and  $x_3^2 + y_3^2 + z_3^2 = 0$  we simplify the determinant into

$$\det \begin{pmatrix} 0 & x_1x_2 + y_1y_2 + z_1z_2 & x_1x_3 + y_1y_3 + z_1z_3 \\ x_1x_2 + y_1y_2 + z_1z_2 & 0 & x_2x_3 + y_2y_3 + z_2z_3 \\ x_1x_3 + y_1y_3 + z_1z_3 & x_2x_3 + y_2y_3 + z_2z_3 & 0 \end{pmatrix} = 0 \Rightarrow$$

$$(1+1)(x_1x_2 + y_1y_2 + z_1z_2)(x_2x_3 + y_2y_3 + z_2z_3)(x_1x_3 + y_1y_3 + z_1z_3) = 0$$

It is clear from cor.1 that the last three factors never are zero, and that the first factor is zero if and only if the characteristic is 2, hence the contradiction is obtained.  $\square$

**Remark 12** *The radical points are the zero set of  $x^2 + y^2 + z^2$ . This zero set obviously does not contain any lines when  $\text{char} \neq 2$ , so the polynomial  $x^2 + y^2 + z^2$  is irreducible, and can be shown to have a zero set of size  $q + 1$ . Hence we identify the radical points as a conic section, when  $\text{char} \neq 2$ , and as a line when  $\text{char} = 2$ , ie.  $x^2 + y^2 + z^2 = (x + y + z)^2$ .*

**Theorem 10** *The tangents of a radical conic section are the radical lines themselves.*

**Proof** The line cover of a radical point consist of  $q + 1$  lines.  $q$  of the lines connect to the other  $q$  radical points of the radical conic section, leaving one line in each radical point not connecting to other radical points. Using lem.1 we see that the radical line is the tangent.  $\square$

**Theorem 11** *Given two radical points  $r_1$  and  $r_2$  in  $PG(2, q)$ ,  $q$  odd.*

$$N(r_1 \times r_2) = - \langle r_1, r_2 \rangle^2$$

**Proof** Let  $r_1 = (x_1 : y_1 : z_1)$  and  $r_2 = (x_2 : y_2 : z_2)$  be radical points.

$$r_1 \times r_2 = (y_1z_2 - y_2z_1 : x_2z_1 - x_1z_2 : x_1y_2 - x_2y_1) \Rightarrow$$

$$\begin{aligned} N(r_1 \times r_2) &= (y_1z_2 - y_2z_1)^2 + (x_2z_1 - x_1z_2)^2 + (x_1y_2 - x_2y_1)^2 = \\ &= x_1^2(y_2^2 + z_2^2) + y_1^2(x_2^2 + z_2^2) + z_1^2(x_2 + y_2) - (1+1)(y_1z_2y_2z_1 + x_2z_1x_1z_2 + x_1y_2x_2y_1) \end{aligned}$$

Since both  $r_1$  and  $r_2$  are radical we get

$$\begin{aligned} &x_1^2(y_2^2 + z_2^2) + y_1^2(x_2^2 + z_2^2) + z_1^2(x_2 + y_2) + x_1^2x_2^2 - x_1^2x_2^2 + y_1^2y_2^2 - y_1^2y_2^2 + z_1^2z_2^2 - z_1^2z_2^2 = \\ &= (x_1^2 + y_1^2 + z_1^2)(x_2^2 + y_2^2 + z_2^2) - (x_1^2x_2^2 + y_1^2y_2^2 + z_1^2z_2^2) = \\ &= -(x_1^2x_2^2 + y_1^2y_2^2 + z_1^2z_2^2) \end{aligned}$$

So we can reduce  $N(r_1 \times r_2)$  to

$$\begin{aligned} N(r_1 \times r_2) &= -(x_1^2x_2^2 + y_1^2y_2^2 + z_1^2z_2^2) - (1+1)(y_1z_2y_2z_1 + x_2z_1x_1z_2 + x_1y_2x_2y_1) = \\ &= -(x_1x_2 + y_1y_2 + z_1z_2)^2 \end{aligned}$$

$\square$

**Corollary 2** *Let  $s$  be a secant in a radical conic section. if  $-1$  is a square in  $\mathbb{F}_q$  then  $N(s)$  is a square in  $\mathbb{F}_q$ .*

$-1$  is a square when the underlying field has a fourth unit root in the multiplicative group. This is the case when  $p^n \equiv 1 \pmod{4}$ . We notice that there are two radicals on  $L_\infty$  namely  $(\sqrt{-1} : 1 : 0)$  and  $(-\sqrt{-1} : 1 : 0)$  thus having a hyperbolic nature.

When  $p^n \equiv 3 \pmod{4}$  we will have no radical points on  $L_\infty$  since  $N(1 : 0 : 0) = 1$  and  $N(k : 1 : 0) = 0$  has no solutions when  $-1$  doesn't have square roots in the underlying field. This give these radical points an elliptic nature in projective planes built over Galois fields of sizes prime powers congruent to 3 modulo 4.

**Theorem 12** In  $PG(2, 2^n)$  there exist sets of size  $q + 2$  having  $\frac{(q+2)(q+1)}{2}$  secants.

**Proof** Consider the zero set of  $hyp(x, y, z) = xy - kz^2$ ,  $k \neq 0$ , then the tangents will be  $(\frac{\partial hyp}{\partial x} : \frac{\partial hyp}{\partial y} : 0) = (y : x : 0)$ . Hence the the point  $(0:0:1)$  is incident with any tangent and thereby incident with tangents only. It is then obvious that by adding  $(0:0:1)$  to the zero set of  $hyp$  we obtain a set of  $q + 2$  points and the  $q + 1$  former tangents as secants, making all lines in the line cover secants, and we count the size of the line cover by counting the number of pairs of  $q + 2$  points, hence we get  $\frac{(q+2)(q+1)}{2}$  as the number of secants to this set.  $\square$

**Corollary 3** In  $PG(2, 2^n)$  there exist  $q + 2$  conic sections having the same line cover, it is easily seen by leaving any one of the points in a non colinear configuration as described above out, yielding  $q + 2$  different conic sections with the excluded point as mutual intersection point of the tangents of the conic section.

**Remark 13** From the proof of thm.12 we note that  $(0 : 0 : 1)$  is the common point of incidence for the tangents of zero sets of  $xy - kz^2$ , so the  $q - 1$  conic sections, one for each  $k \neq 0$  in  $\mathbb{F}_{2^n}$ , share all of their tangents. This fact is used in the proof of the covering number 6 formula given in sec.3.6, for counting the lines having a third coordinate of zero.

### 3.1.2 $N(x)$ when $q = 1 \pmod{4}$

When  $p^n = 1 \pmod{4}$  we have seen that  $-1$  has square roots and the radical points on  $L_\infty$  are the points  $(\sqrt{-1} : 1 : 0)$  and  $(-\sqrt{-1} : 1 : 0)$ . The fibre of 1 consists of  $2q - 1$  points where  $(1 : 0 : 0)$  and  $(0 : 1 : 0)$  form the intersection with  $L_\infty$ , and the rest are the affine roots of the lines  $(\sqrt{-1} : 1 : 0)$  and  $(-\sqrt{-1} : 1 : 0)$ . All other fibres' affine points form conic sections through the points  $(\sqrt{-1} : 1 : 0)$  and  $(-\sqrt{-1} : 1 : 0)$ . The points on  $L_\infty$  are members of the fibres of those  $k$  where  $k - 1$  is a square in  $\mathbb{F}_q$ .

### 3.1.3 $N(x)$ when $q = 3 \pmod{4}$

When  $p^n = 3 \pmod{4}$  then the fibre of 1 consists of only 3 points, that is the canonical base points  $(0 : 0 : 1)$ ,  $(0 : 1 : 0)$  and  $(1 : 0 : 0)$ . Then we have  $\frac{q+1}{2}$  fibres with  $q + 3$  points each and  $\frac{q-1}{2}$  fibres consisting of  $q + 1$  points each. If we intersect the fibres with the affine part of the plane we get  $q - 1$  fibres that are conic sections consisting of  $q + 1$  points and the fibre of 1 reduces to  $(0 : 0 : 1)$ . If we intersect the fibres with  $L_\infty$  we have 2 points in the fibres of  $k \in \mathbb{F}_q$  where  $k - 1$  is a square in  $\mathbb{F}_q$ .

### 3.1.4 $N(x)$ in char 2

The frequency table reveals that there are  $q + 1$  points in each fibre except for the fibre of 1 that has  $q + 2$  points. If we restrict the frequency table to the affine part of the plane we see that all fibres contain  $q$  points. This means that  $L_\infty$  has a point in each fibre except for the fibre of 1 where it has two points. Closer examination reveals that the affine points of the fibres form parallel lines, this means that these lines intersect in exactly one point. This point lies within the fibre of 0 and is the point  $(1 : 1 : 0)$

## 3.2 The linear group over $\mathbb{F}_q \times \mathbb{F}_q \times \mathbb{F}_q$

In a three dimensional vector space over  $\mathbb{F}_q$  we use  $GL(3, \mathbb{F}_q)$  to cover transformations sending homogenic polynomials of  $\mathbb{F}_q[X][Y][Z]$  into homogenic polynomials of the same degree. We see that cover numbers are invariant under linear transformations. If a line  $L$  is incident with  $n$  points  $P_i$  then the images of  $P_i$  are incident with the image of  $L$  in exactly  $n$  instances. So this group is quite useful and I represent it by  $3 \times 3$  matrices with a non-zero determinant over  $\mathbb{F}_q$ .

Now having coordinates we notice that the columns of a  $3 \times 3$  matrix are the images of the canonical base vectors. A projective transformation is uniquely determined by its map of four points in general position. Points in general position are points, where no three of them are incident with a common line.

### 3.2.1 $PGL(2, \mathbb{F}_q)$

We are not solving problems in the three dimensional vector space over  $\mathbb{F}_q$  but on an isomorphic image of the lines through origo, so the group we utilize in the projective planes to send lines into lines is the projective linear group.  $PGL(2, \mathbb{F}_q)$  is the quotient group of  $GL(3, \mathbb{F}_q)$  modulo the centre of  $GL(3, \mathbb{F}_q)$ .

#### Definition 28

$$PGL(2, \mathbb{F}_{p^n}) = GL(3, \mathbb{F}_{p^n}) / Z(GL(3, \mathbb{F}_{p^n})).$$

We can think of this factorizing as our normalizing procedure of the points, where we multiply the coordinates with a non-zero constant to get the last non-zero coordinate to 1. The centre of  $(GL(3, \mathbb{F}_q))$  is isomorphic to the cyclic multiplicatice group of  $\mathbb{F}_q$ , and consists of diagonal matrices with equal non-zero entries in the diagonal.

We should at this point reason about the size and the structure of this group. The easiest way to determine the size of the group is to count the number of sets of points in general position. The first point can be selected freely among all  $q^2 + q + 1$  points. The second point can be selected among  $q^2 + q$ , and the third point has to be selected among the  $q^2$  points not incident with the line through the first two points. The fourth and last point will be selected outside of the three lines connecting the first three points, giving  $q^2 - 2q + 1$  possible choices. Putting all together we see that the size of the group is  $(q^2 + q + 1)(q + 1)q^3(q - 1)^2$ . We should notice that  $GL(2, \mathbb{F}_q)$  is a subgroup, because we can consider the linear transformations fixing  $L_\infty$ , then any affine point stays affine under the transformation.

### 3.2.2 Targeting points or lines

If we have a triangle that we want to hit with another triangle we construct a matrix  $T$  with the target points as coloumns then the canonical base maps into the target triangle. To construct a map from the source triangle into the canonical base we invert the matrix  $S$  with the source points as coloumns. This way we can easily calculate the map as  $S^{-1}T$ .

## 3.3 Conic sections

A conic section is defined as a set of  $q + 1$  points where no three points are colinear. In  $PG(2, q)$  the points of a conic section can be generated as the zero set of a homogeneous irreducible degree-2 polynomial in  $GF(q)[X, Y, Z]$ .

### 3.3.1 Maximal sized sets of height 2

In this section we shall see that conic sections can be used as some sort of macro point, having large line coverings. In fact I will prove that the line cover of a conic section is maximal among height-2 sets. First two motivating lemmas then the concluding proof.

**Lemma 2** *Let  $C$  be a conic section in  $PG(2, q)$ ,  $q$  odd. Let  $p$  be a point in  $\Pi - C$ .*

$$H(C \cup \{p\}) = 3$$

**Proof** It can be shown that the tangents of a conic section does not have common intersection, when  $q$  is odd, hence  $p$  is incident with at least one secant of  $C$  thus lifting the height to 3.  $\square$

**Lemma 3** *Let  $C$  be a conic section in  $PG(2, 2^n)$ . Let  $t$  be the common intersection point of the tangents of  $C$ . Let  $p$  be a point in  $\Pi - (C \cup \{t\})$*

$$H(C \cup \{t\} \cup \{p\}) = 3$$

**Proof** Let  $C_t$  denote  $C \cup \{t\}$  and let  $c \in C_t$ . Then the  $q + 1$  lines incident in  $c$  connect to the  $q + 1$  other points of  $C_t$ , so the line incident with both  $c$  and  $p$  also connect to some other point in  $C_t - \{c\}$ , and this line obviously incident with 3 points in  $C_t \cup \{p\}$ .  $\square$

**Theorem 13** *A conic section in  $PG(2, q)$  is a configuration of height 2, providing maximal line cover.*

**Proof** The larger a height-2 set is the larger the linecover become. From the lemmas lem.2 and lem.3 we see that there is a limit for how large a height-2 set can be. Futhermore it is trivial to see that adding by a common intersection of all  $q + 1$  tangents we do not extend the lines covered by the conic section. this way it is sufficient to use the conic section.  $\square$

**Remark 14** *We should take notice, that the line cover of a conic section is more than half of the lines in the plane, because this gives a first indication of a possible upper bound of  $2\log_2(q^2 + q + 1)$  for the covering number of blocking sets built as unions of conic sections.*

### 3.3.2 Linear transformations of conic sections

In  $PG(2, q)$  we can speak about the invertible  $3 \times 3$  matrices with coefficients in  $\mathbb{F}_q$ . These matrices form the group of linear transformations of a 3-dimensional vectorspace over  $\mathbb{F}_q$ . This group modulo its centre gives us the group determining all possible linear transformations of a projective plane. The linear group of the projective plane,  $PGL(2, GF(q))$ . Having 4 non colinear points as source and 4 non colinear points as target uniquely determines one element of this group. So the image of a conic section through a linear transformation of full rank is another conic section.

### 3.3.3 Adding conic number two

Lets assume that we have a conic section  $C$  and a linear transformation  $T$ . Now we have seen that the size of  $LCOV(C)$  is slightly larger than half of  $\Pi$ , so the question is if we can continue the process of reducing the size of  $\Pi - L$  to less than half in terms of making unions of conic sections, because if this is the case, then we can claim a  $2\log_2(q)$  upper bound for  $coveringnumber(\Pi_q)$ . Before speculating further it will be proper to see if there exist a  $T$  so that  $|LCOV(C \cup T(C))| < \frac{q^2+q+1}{4}$ .

In my research I have been traversing the linear group to try to find the best next conic section, to see if there is a simple solution. It turns out that it is possible to construct an optimal  $T$ , in fact it turns out that there are many  $T$  that produce maximal line cover, and among those are transformations that is fixing both 1 and 2 lines.

Having investigated the line profile of the best chosen second conic section, i found that there exist a conic section equally good within the fibres of the  $N$  map. This is very nice since we then don't have to traverse the linear group in this case, but it is enough to test  $q - 2$  other known conic sections. Still we don't know if it is possible to continue to find optimal conic sections in the fibres of  $N$  but never the less we can find series of conic sections that more than halves the rest of uncovered lines we will be very well off towards construction of the theoretic result of Ughi, Abbot and Liu (see thm.4 and thm.5).

## 3.4 Systems of conic sections

I will describe two different ways to generate systems of conic sections in a way that we can use the multiplicative group in  $GF(q)$  as iterator for what I will refer to as orbits of conic sections. As a matter of fact we can dissect the plane into large equally sized parts of of conic sections and have 3 lines or a line and a point left. This is crucial information, when constructing solvers based on conic sections.

### 3.4.1 Disjoint conic orbits

From the analysis of the fibres of the  $N$  map in various  $PG(2, q)$  we yield the following for  $q \neq 2^n$ .

**Theorem 14**  *$\Pi$  can be constructed by  $q - 1$  disjoint conic sections and a line and a point.*

**Proof** when  $q = 3 \pmod{4}$  we select the conic sections as the affine part of fibres of  $N$  except the fibre of 1. We select  $(0:0:1)$  as single point and  $(0:0:1)$  as single line. when  $q = 1 \pmod{4}$  we use the fibres of  $x^2 + y^2 - z^2 = 0$  instead of  $N$ , and do as above.  $\square$

This system of conic sections is the perfect abstraction of concentric affine circles around origo as inner limit, and  $L_\infty$  as outer "limit". I will refer to this kind of system as "ecliptic orbits".

### 3.4.2 Conic orbits with 2 fixpoints

In this construction we use that a finite field is an integral domain. This means that  $ab = 0 \iff a = 0 \vee b = 0$ . So we can construct the affine part of the plane outside of the two axis by using  $xy = k$ ,  $a, b, k \in GF(q)$ . Homogenizing  $xy = k$  give us  $XY - kZ^2$  and from this polynomial we see that the roots on  $L_\infty$  are the points  $(1 : 0 : 0)$  and  $(0 : 1 : 0)$ . to describe these conic sections as a system, one might think of the canonical hyperbolas having the lines  $x = 0$  and  $y = 0$  as asymptotes. I will refer to this kind of system as “hyperbolic orbits”.

**Remark 15** *We get hyperbolic orbits from the affine norm fibres when  $q = 1 \pmod{4}$ , but using  $xy = k$  we do not have to distinguish  $q$  by 4th unit roots. My experiments have shown that “hyperbolic orbits” are more interesting than “ecliptic orbits” with respect to the generating of small covering numbers. (See sec.5.3.1, rem.21) The existance of both “ecliptic” and “hyperbolic” systems is an interesting property. It imply that it is possible to constuct an ecliptic-hyperbolic coordinate system, like the one used in astro-physics. It has been seen before, that changing the coordinate system can be most useful.*

### 3.5 Covering number better than $2 \log_2(q^2 + q + 1)$

Using a system of hyperbolic orbits i have observed the following result (see sec.5.3) :

**Result 1** *The zero set of the polynomial given by*

$$\prod_{i=1}^n (XY - k_i Z^2), \text{ where } k_i \neq k_j \wedge k_i \neq 0$$

*form a blocking set in  $PG(2, q)$  when selecting  $n < \log_2(q^2 + q + 1)$  suitable  $k_i$  in  $GF(q)$ .*

And what has happened.

**Result 2** *Hyperbolas having y-axis and x-axis as tangents.*

- *We have reduced the conic mining problem to a one dimensional problem over  $\mathbb{F}_q$ .*
- *The  $q - 1$  affine points of each of the conic sections are parametrized as  $P_k(x) = (x : kx^{-1} : 1), x \in GF(p^n) - \{0\}$ .*
- *for all  $k \neq 0$  solutions on  $L_\infty$  are  $(1:0:0)$  and  $(0:1:0)$ , covering the axis parallel lines.*

This observation is reflected in the fast algorithm that terminates in time  $O(q^3 \log_2(q))$ , producing covering numbers less than  $2 \log_2(q^2 + q + 1)$ . And the very same algrithm surprisingly returns the covering number 6 for any  $q = 2^n$ . The empirical results seem to indicate that the upper bound stated by Abbott and Liu of  $2 \log_2 q$  can be lowered, keeping the blocking sets in terms of unions of conic sections.

### 3.6 Cover number 6 in characteristic 2

Using the results of 2 one can ask how many of these  $k \neq 0, 1$  provides maximal line cover along with 1, and **very** surprising **all**  $k \neq 0, 1$  do ! - And additional **the last conic section is uniquely determined as  $1 + k$** , so what we get here is a very fast algorithm to produce covering sets with covering number of 6. when the underlying field is  $GF(2^n)$ .

**Theorem 15** *The zero set of  $(XY + Z^2)(XY + kZ^2)(XY + (1 + k)Z^2)$  form a blocking set with cover number 6*

So given the parametric production of the set, it is produced in time proportional to the size of the set namely the time of picking  $(0 : 1 : 0)$  and  $(1 : 0 : 0)$  and three times  $q - 1$  points yielding a time complexity of  $\Theta(3q - 1)$ .

**Proof** The case  $PG(2, 2)$  is degenerate, since it is impossible to cover all lines because 3 colinear points are all the points of a line, meaning that we cant construct a blocking set (see def. 4). The case  $PG(2, 4)$  can be covered by 2 hyperbolas. So we are only concerned about  $PG(2, 2^n)$ , where  $n > 2$ . We divide the proof into two cases. First we show that the lines with coordinate sets containing one or two zeroes are covered by the three hyperbolas, then we show that the rest of the lines are also covered.

### 1. Case

We have the points  $(1 : 0 : 0)$  and  $(0 : 1 : 0)$  in the zero set of the three hyperbolas, hence we cover lines with first and second coordinate of zero. The point  $(0 : 0 : 1)$  (origo) is the common intersection point of the tangents in the points  $(1 : 0 : 0)$  and  $(0 : 1 : 0)$ , that is the  $y$ -axis and the  $x$ -axis, both have incidence with origo. The lines incident with  $(0 : 0 : 1)$  are the lines  $(a : b : 0)$  and since the  $y$ -axis is accounted for we can consider  $x$ -values that are non zero. Let us focus on the hyperbola  $y = x^{-1}$ . If the zero set of  $xy - z^2$  have incidence with  $(a : b : 0)$  where both  $a, b$  are nonzero (none of the axis), then  $ax + bx^{-1}$  have roots for non-zero  $x$ . When  $x$  is non zero we obtain the degree-2 equation  $x^2 = \frac{b}{a}$ , and then we see that this always have solutions, since squaring is an automorphism in char 2. We can generalize this argument for any  $xy - kz^2$   $k \neq 0$ , because we notice that we get the corresponding degree-2 polynomials  $x^2 = \frac{kb}{a}$ . This way we see that all zero sets of these  $xy - kz^2$  share the lines through origo as tangents. Making origo the common intersection point of the tangents.

### 2. Case

Consider the projective line  $(a:b:1)$ . We then consider the affine points and parameterize the three hyperbolas as follows  $y = x^{-1}$ ,  $y = kx^{-1}$  and  $y = (1 + k)x^{-1}$ . If  $P : (x : y : 1)$  is a point of either three hyperbolas and  $(a:b:1)$  then  $ax + by + 1 = 0$  and if we restrict  $x$  to be non-zero we have  $ax^2 + x + b = 0$ ,  $ax^2 + x + bk = 0$  or  $ax^2 + x + b(1 + k) = 0$ .

We have now transformed the problem into existence of roots of 2.degree polynomials over  $\mathbb{F}_{2^n}$ .

Consider the map  $p : x \rightarrow ax^2 + x$ , where  $a \neq 0$  and consider  $\mathbb{F}_{2^n}$  as a  $\mathbb{F}_2$  vector space of dimension  $n$ .

$$p(x) = 0 \iff x = 0 \vee x = a^{-1}$$

$$p(x + y) = a(x + y)^2 + (x + y) = ax^2 + x + ay^2 + y = p(x) + p(y)$$

We then see that  $p$  is a homomorphism with a kernel of order 2. Then  $p(\mathbb{F}_{2^n})$  can be used as the kernel of a homomorphism from  $\mathbb{F}_{2^n}$  onto  $\mathbb{F}_2$ . Our question is then reduced to the presence of either  $b$ ,  $bk$  or  $b(1 + k)$  in the kernel, and it is clear that either is 1 or all three inside. Where 1 of them is in the kernel  $(a:b:1)$  is a private secant, and where all 3 are in the kernel we get the 6-secants.  $\square$

**Result 3** *There are polynomials in  $q$  over  $\mathbb{Q}$  counting the  $i$  - secants of the optimal char 2 solution. (see also table 7).*

$i$	$\#i$ - secants	Geometric interpretation of the lines
1	2	$(0 : 1 : 0)$ and $(1 : 0 : 0)$
2	$\frac{3}{4}q(q - 1) + 1$	Private secants and $(0 : 0 : 1)$
3	$q - 1$	The lines through $(0 : 0 : 1)$ other than $(0 : 1 : 0)$ and $(1 : 0 : 0)$
4	$2(q - 1)$	Axis parallel lines other than $(0 : 1 : 0)$ and $(1 : 0 : 0)$
5	0	
6	$\frac{1}{4}q(q - 1) - (q - 1)$	Shared secants

## 4 Using the multiplicative group

Some of the most interesting coverings that I have found, are those directly connected to the multiplicative group. I have been looking for properties within the multiplicative group that enable us to find systems of conic sections  $\Pi(xy - kz^2)$  where it sometime is sufficient to restrict  $k$  to belong to a subgroup of the entire multiplicative group within the underlying field. Like wise I have found, that grids of points

$\cup_{x,y \text{ insub}}(x : y : 1) \cup (0 : 0 : 1), (0 : 1 : 0), (1 : 0 : 0)$  provide blocking sets for some subgroups. Some of these grids, when reduced to minimal blocking sets by my reducer, provide us with very good solutions. How the subgroups are related to the quality of the blocking sets is still a deep mystery to me, but it is worthwhile to note the existence of this relation.

Let me state a couple of important observations about grids and restricted systems of conic sections.

**Result 4** • *It is not always possible to find covering sets related to a subgroup in terms of conic sections.*

- *There seem to be a lower limit for the size of the subgroup in order for a grid to be a blocking set.*
- *The grids are almost always covering sets, when picking subgroups of larger sizes than the lower limit, but there are exceptions.*
- *Viewing the height of the configuration of all conic sections generated from the group show a unique interleaving, so that the union of  $k$  conic sections very often have surprisingly low height compared to  $2k$ , that is the maximum for such a configuration.*

All in all this cries out for further investigation. This information might lead far in the search for contradiction to Erdős constant. At the moment, I do not have the time to carry out such an investigation, albeit I would very much like to do so. Let this be the last remarks in this section of analysis.

## 5 Covering number solvers

### 5.1 Introduction to the solvers

I have decided to present the important results graphical. I have done it in a way that clearly show if an algorithm has an asymptotical behavior, by presenting the quotient  $\lambda$  as a function of  $q$ .

$$\lambda = \frac{\gamma(B)}{\log_2(q^2 + q + 1)}$$

I have selected this quotient instead of  $\frac{\gamma(B)}{\log_2(q)}$ , because I have been interested in halving the number of uncovered lines, each time a conic section is added to the solving set, thereby serving as termination function for my conic section algorithms. And it doesn't really matter whether it is one or the other, since they both are polynomial expressions inside a logarithm.

What is interesting is that, The better algorithms all have  $\lambda$ -values significant below 1 as  $q$  get larger. This means that these algorithms in a deterministic way implement the upper bound found by Abbott and Liu (see thm.5).

What is even more interesting is that all the better algorithms, both the algorithms based on conic sections and the two best 'single point' algorithms - and not to forget Pisinger and Illes' simulated annealing all seem to hit a wall, that very well might be  $\lambda > \frac{1}{2}$  This could indicate that the projective planes build over prime fields, set the standard for the lower bound.

Any way we can use the average value of  $\lambda$  to give meaning to the quality of the various algorithms, along with the time complexity. This way I can say that my "Hyperbola cover" is the best of my algorithms both in terms of average  $\lambda$  and in terms of time complexity.

The order of the covering algorithms shown reflect my process of bringing the covering numbers down. My program have been used as a generic workbench for doing all sorts of experiments, but the experiments shown in this section is restricted to a subset of the solvers for blocking sets. A number of other experiments have lead to the "Hyperbola cover" algorithm, and most of these can still be carried out by using the right command line parameters to my main program.

In order to estimate the time complexity of the solvers we need to have an idea of the sizes of the blocking sets. This is why i have presented the graphs showing the quotient  $\beta$  as a function of  $q$  where  $\beta$  is the size of the blocking set divided by  $q \log_2 q$  in order to visualize that it is possible to get blocking sets of sizes less than  $q \log_2 q$ , so we for these algorithms safely can substitute  $O(|B|)$  with  $O(q \log_2 q)$ .

$$\beta = \frac{|B|}{q \log_2 q}$$

Most of my solvers do not create minimal blocking sets, so the  $\beta$  values shown are higher than necessary. I have written a simple reducer to “slim” the blocking set to be minimal, meaning that any point is incident with a 1-secant, so discarding any point will let go a captured line. The effect of the present reducer is not very significant, but it does smoothen the  $\beta$ -graphs for the conic section solvers, as it can be seen in figure 19. Note that writing a good reducer is the same as writing a solver, that could start with all points as selected set, and “slim” it to be a good solution. I have chosen not to use reducers in general, because I do not gain insight about explicit constructions by using these, and that has been important to me all the time.

## 5.2 Small covering number solvers picking 1 point at a time

All of these solvers can be characterized by adding one point at a time. Between picking points these algorithms do substantial work, making the necessary book keeping. Stuff like keeping track of the heights of all points, as the “Least height” solver does it, takes  $O(q^3)$  time and counting primitive operations, the constant is fairly large. Updating the line heights or checking for height violations take  $O(q)$  time for each point. The algorithms in this section that produce the best covering numbers, are the algorithms having the largest time complexity.

### 5.2.1 Simpler brute

This solver iterates the points of the plane picking the first point that does not cause a global increase of height, as the next member of the solving set. When no points can be found the global height is increased by one. Starting with the empty set this process continues until all lines are covered. The termination is guaranteed by the finite number of lines to be covered. (See fig 1)

**Remark 16** *It is obvious from the graph that the covering numbers produced from this algorithm are of poor quality, very far from optimal solutions. We simply do not use information enough by using global height only. So either is this problem hard to control, or else we do not use the right means at all. The latter seems to be the case from my studies.*

### 5.2.2 Smarter brute

This algorithm has no explicit height checking but selects rounds of points increasing the height with at most 2. The points open for selection are all points incident with more than the mean of the minimal and maximal number intersections of uncovered lines. It is clear that we only get good results if the sets selected in a round is large, but of course the size of such a set is bounded by the size of a conic section (+1 for char 2 planes). (See fig.3, 4 and 5).

**Remark 17** *We see that the idea of focusing on points incident with many uncovered lines, produce much better covering numbers, but still we do not get results that match low theoretical upper bounds for the covering number problem. From fig. 5 we note that the  $\gamma$  value seem to go below  $\sqrt{q}$  when  $q$  get large enough. And we can see from fig.3 that the  $\gamma$ -values for  $q$  not a power of 2 are below the Abbot-Liu upper bound. So this algorithm might be creating logarithmic solutions, albeit not very good ones. The  $\lambda$ -values for  $q = 599,601$  and  $607$  are  $1.14, 1.08$  and  $1.25$ .*

*As a side effect of this strategy we see that the blocking sets obtained really do get small. They seem to stabilize with a  $\beta$  value of  $\frac{1}{2}q \log_2 q$  and notice some low values. The values for  $B_{256}$  and  $B_512$  are invalid because  $B$  contain a line, meaning that the algorithm can diverge. As a matter of fact this algorithm is the result of a relaxation on the original algorithm in an attempt to beat the char 2 divergence. The original strategy was only to take points with maximal incidence with uncovered lines. The divergence however did not disappear, and as the results were similar those of the earlier strategy. And it is fairly obvious that we by allowing more points on a line we obtain smaller blocking sets. We can see from these*

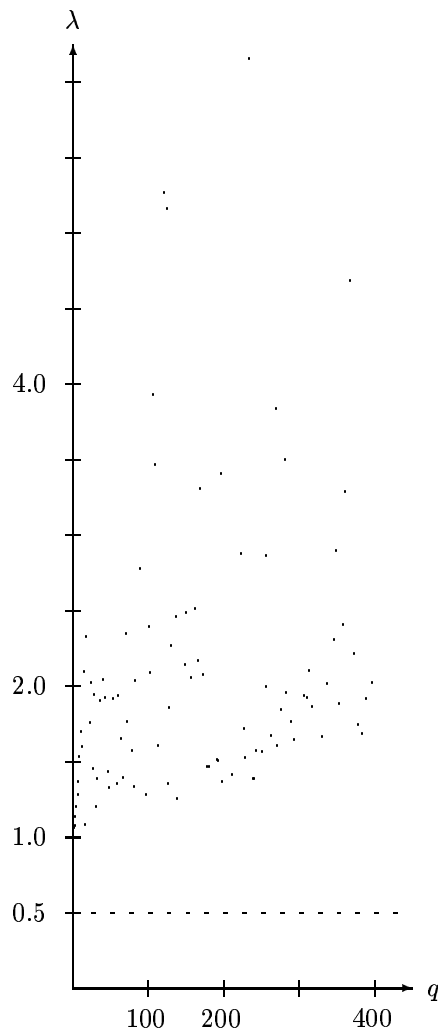


Figure 1:  $\gamma(B)/\log_2(q^2 + q + 1)$  as function of  $q$  (Simpler brute cover)

*$\beta$  values that the strategy of putting pieces of height-2 sets with high incidence on uncovered lines, that we obtain blocking sets with less than  $q \log_2 q$  points. The general picture of the  $\lambda$  is that the  $\gamma$ -values seem to be better than  $2\sqrt{q}$  and has its first point better than  $\sqrt{q}$  when  $q$  is 331.*

### 5.2.3 Elliptic cover

Introducing assumptions of coordinates. This assumption restricts the results not to apply to finite projective planes in general, but merely only to the Galois planes built over finite fields. This routine uses the zero set of the polynomials  $Y^2Z - X^3 + aXZ^2 + bZ^3$  as points open for selection. Points get selected if they are incident with an uncovered line and if it does not violate the global height. The maximal height is increased each time  $b$  equals zero. This polynomial is irreducible for most  $a$  and  $b$  and in the irreducible case the zero set is incident with at most 3 points on any line according to Bezout's theorem (see thm.3). So this routine picks points from elliptic curves with a height constraint. See Figure 6 and 7.

```

a,b,done = 0
maxheight=3
while (!done)
  do
    << solve X^3 - Y^2Z + aXZ^2 + bZ^3 = 0 >>

```

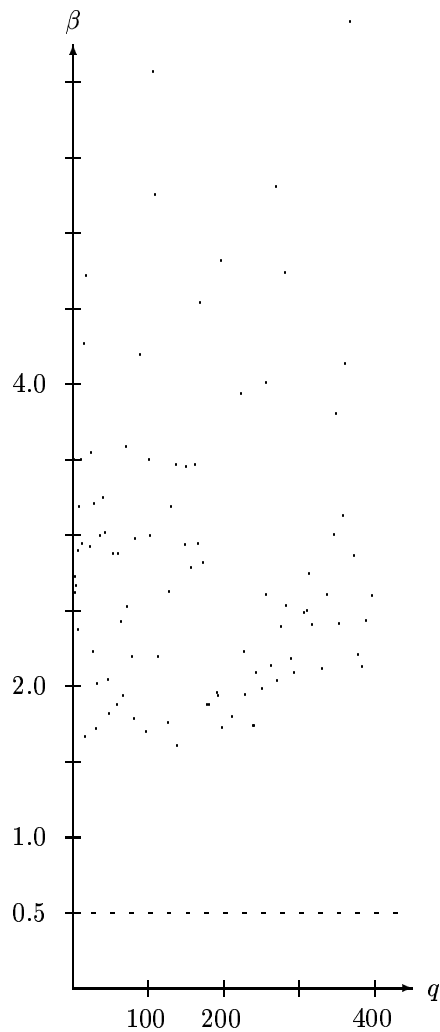


Figure 2:  $|B|/q \log_2 q$  as function of  $q$  (Simpler brute cover)

```

<< traverse zero set and select point if height <= maxheight >>
if (<< all lines are covered >>)
    done = 1;
else
    a++;
    if (a==fieldsize)
        a=0;
        maxheight++;
        b++;
        if (b==fieldsize)
            b++;
        fi
    fi
od

```

**Remark 18** *This is the first of the presented polynomial time algorithms, that for sure produce covering numbers smaller than  $2\log_2(q^2 + q + 1)$ , the Abbott-Liu bound. We should notice that the algorithm does not in any way attempt to construct conic sections, but still hit the same floor. I think it is interesting that the pure conic section solvers produce remarkably better results. The reason why the  $\beta$ -values are a bit high for this solver has a simple explanation. The algorithm does not terminate until it cannot fill in more points in the solving set, without incrementing the height. This of course increase  $\beta$ . We can see*

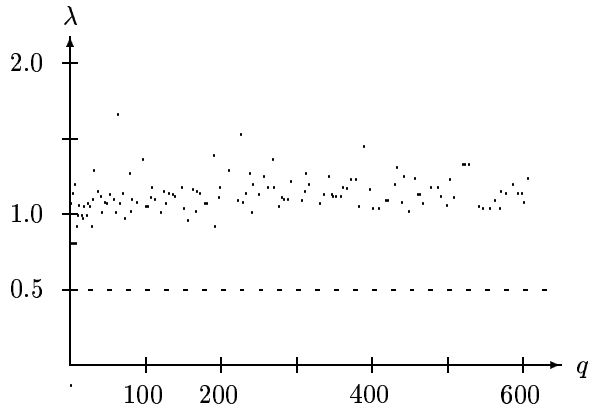


Figure 3:  $\gamma(B)/\log_2(q^2 + q + 1)$  as function of  $q$  (Smarter brute cover)

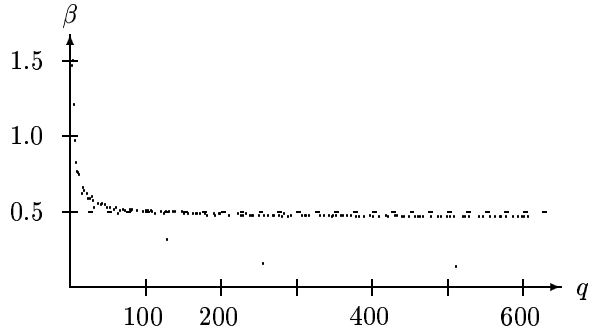


Figure 4:  $|B|/q \log_2 q$  as function of  $q$  (Smarter brute cover)

from the tables that  $\lambda$  has a value close to  $\beta$ . The elliptic curves just a sort of control the order in which the points are probed for inclusion.

#### 5.2.4 Least height cover

```

done=0;
height=2;
<< select a conic section as initial selection >>
<< initialize point heights >>
while (<< uncovered lines exist >>)
do
    << select at most height points on an uncovered line not violating height,
        maximizing the line cover >>
    if (<< no points found>>)
        height++
        << select at most height points on an uncovered line not violating height,
            maximizing the line cover >>
    fi
od

```

This algorithm starts with a conic section as selected set. In each iteration it picks points on uncovered lines having the least height relative to the selected set. See the definition of heights of points. The algorithm terminates when there are no more uncovered lines. Note that this algorithm is very slow because it does serious book keeping for each selected point.

The actual time complexity of this algorithm is  $\Theta(|B|q^3)$ , since it must update the heights of points each time a fresh point is selected, because there are  $O(|B|)$  points in the resulting covering set, and there are  $\Omega(q^2)$  points, that each require  $\Omega(q)$  time to calculate the line incidence and determine the maximum line height. When we pick a point, we pick a point having the least height. Ordering the points by height take time  $O(q^2)$  using bucketsort in  $q$  buckets, so this does not harm the time complexity since it is

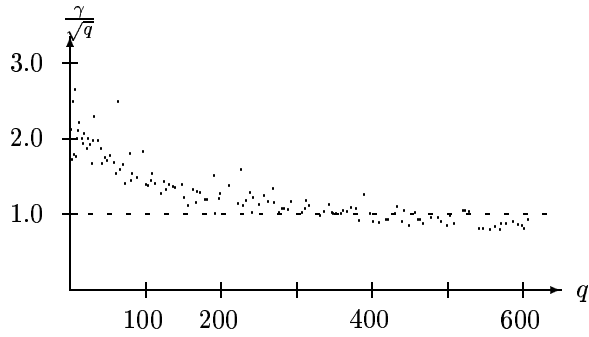


Figure 5:  $\gamma(B)/\sqrt{q}$  as function of  $q$  (Smarter brute cover)

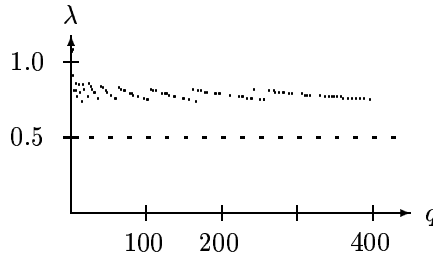


Figure 6:  $\gamma(B)/\log_2(q^2 + q + 1)$  as function of  $q$  (Elliptic cover)

dominated by updating the height map for points. We can even test  $O(q)$  side conditions for each point for free, because the number of points is limited to  $O(q^2)$ . Hence we see that picking the next point can be kept within can be done better than  $O(q^3)$ , there fore we get the tight bound on the algorithm by the experiment. In any case the algorithm is guaranteed to terminate in time  $\Theta(|B|q^3)$  with a blocking set  $B$ . To inspect the performance of this algorithm please see fig.8 and 9.

**Remark 19** *This is the second algorithm that produce covering numbers below  $2\log_2(q^2 + q + 1)$ . This algorithm select points one at a time. But it does not attempt to build conic sections, and use a totally different scheme of selecting points than the elliptic cover algorithm. So we see that it is not a coincidence that we can go below the Abbott-Liu bound. **Now we wanna know, how low below can we go!** The solutions presented by Pisinger and Illes seem to answer a part of this deep question. Note that this algorithm does not in anyway use information from coordinates. Further more it does not try in any way to build conic section pieces. So we may conclude that having coordinates are not necessary to produce logarithmic covering numbers, with low order polynomial time solvers, though coodinates might help to speed up the selection of many points at a time. The  $\beta$ -values form a smooth floor for the  $\lambda$ -values. And let me point out that  $\lambda$  is larger than  $\beta$  only for  $q \leq 103$ . We note the tendency in general of  $\beta < \lambda$  for large  $q$ . This fact is also illustrated by Pisinger and Illes results.*

*This algorithm will behave equally in projective planes built of difference sets, using point-line incidence matrices to determine the relationship between points and lines. Those are the planes that make the frame of Pisinger and Illes' work.*

### 5.2.5 Pisinger and Illes' coverings

I have recalculated the  $\lambda$ -value from Pisinger and Illes' to be the quotient  $\frac{\gamma(B)}{\log_2(q^2+q+1)}$ . And I have included the results in the tables section. I have supplied the tables with the  $\beta$ -values. This way is easy to compare the graphs and the tables. The algorithm used to produce these coverings is a sophisticated piece of A.I., please see [Pisinger-Illes] for a detailed description. The  $\lambda$ -graph is figure 10, and the  $\beta$ -graph is figure 11.

**Remark 20** *Though I'm a bit impressed by the results of this solver, albeit is has not in any way low order polynomial execution time. It is a sophisticated guessing machine. It has a flaw though, or it*

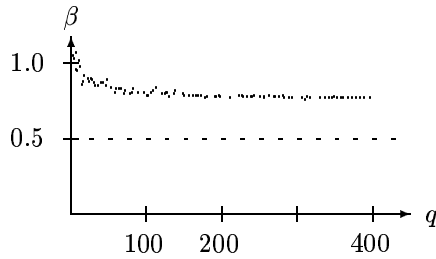


Figure 7:  $|B|/q \log_2 q$  as function of  $q$  (Elliptic cover)

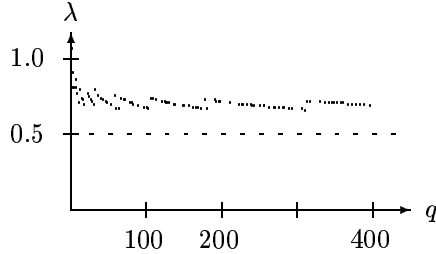


Figure 8:  $\gamma(B)/\log_2(q^2 + q + 1)$  as function of  $q$  (Least height cover)

should be investigated further, whether it is possible for this machine, to produce both covering sets as those described by Boros (see [Boros]), and further more it should be possible to produce covering sets similar to mine in char 2. Never the less I'm still impressed by the result, because the logarithmic covering number is obtained by selecting the best out of 10 executions of the guessing machine. I would suggest for instance 100 executions covering  $PG(2, 81)$  and  $PG(2, 128)$ . It is remarkable how close this algorithm come to  $\beta$ -values equal to 0.5, and that is from the upper side. The lambda-values are indeed very small, that is approximately 0.6 in the upper range. The the value goes from 0.60 to 0.66 when  $\gamma$  change from 11 to 12 for the first time. Note that the similar step when  $\gamma$  change from 10 to 11 has a lambda-value that change from 0.59 to 0.65. We see that the algorithm require more iterations, or longer time to stabilize for larger problems. Opposed to the conic section coverings presented in the next section.

### 5.3 Small covering number solvers using conic sections

Introducing the systems of conic sections, that are zero sets of irreducible degree-2 polynomials, we produce blocking sets that are easily calculated. By easy I mean that gain a lot of speedup, and that is much smaller constants for the total execution time. The speedup from the single point algorithms at least mine is really significant, albeit I haven't spent too much time on tuning the the code to the limit.

Let me at this time point out that the blocking sets generated by the first two algorithms are not minimal. We can remove points from most of the solutions found. This is done by a reducer in the last algorithm presented. Until then we will not be able to see a smooth  $\beta$ -graph, because points are added in chunks of  $q - 1$  or  $q + 1$  by these algorithms. But You can take a peek at figure 19 if you want to see a smooth  $\beta$ -graph.

#### 5.3.1 Norm fibre cover

```
eclipse      = (fibersize[1] <= 4) ? 1 : 0;
affinesize   = (eclipse) ? q+1 : q-1;
<< select the projective fibre of 0 >>
while (<< uncovered lines exist >>)
do
    << select an affine fibre maximizing line cover >>
    if (<< only 1 uncovered line >>) // the line is (0:0:1) and q=4n+1 always !
        << select a point on this line
            violating height the least >>
fi
```

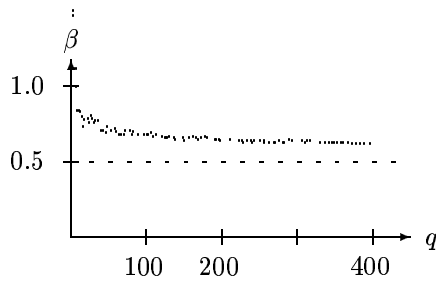


Figure 9:  $|B|/q \log_2 q$  as function of  $q$  (Least height cover)

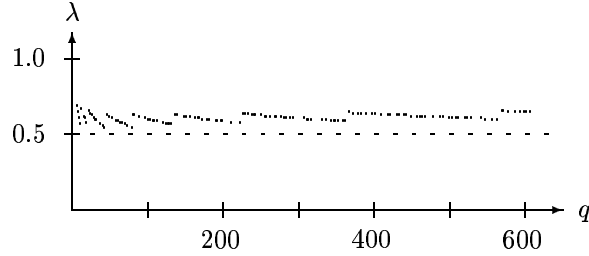


Figure 10:  $\gamma(B)/\log_2(q^2 + q + 1)$  as function of  $q$  (Simulated annealing cover)

od

This an algorithms based on selecting all the points of conic sections found in the fibres of the norm map. The algorithm starts with the fibre of 0, and then in each iteration selects the fibre that minimizes the number of uncovered lines. See Figure 12 and 13 for the hyperbola solutions and figure 14 and 15 for the eclipse solutions.

$q$	$\gamma(B)$	$\lambda$
757	12	0.63
769	14	0.73
773	12	0.63
797	12	0.62
743	13	0.68
751	13	0.68
787	15	0.78

**Remark 21** This algorithm produce good coverings, but from the graph, or the tables, we notice that there is a difference between the ecliptic coverings we get when  $q \equiv 3 \pmod{4}$  and the hyperbolic coverings we get when  $q \equiv 1 \pmod{4}$ . What we can see is that hyperbolic ones have lower  $\lambda$  values than the ecliptic ones. From this observation it is natural to seek means to always construct hyperbolic coverings. The above table show both  $\gamma(B_q)$  and  $\lambda_q$  for some hyperbola coverings and for some eclipse coverings, where the upper 3 values are from hyperbolas and the lower 3 are from eclipses. It is a fact that ecliptic systems diverge in char 2, and hyperbolic systems produce the  $\gamma$ -value of 6. This has been tested using zero sets of  $x^2 + xy + y^2 + kz^2$  to emulate the nature of the norm for odd primes numbers. When the power of 2 is odd, thus not a square number we get ecliptic systems, that are not incident with  $L_\infty$ . When the power of two on the other hand is a square we get hyperbolic systems with common incidence in two points on  $L_\infty$ . Note the very low value of  $\gamma = 10$  for  $q = 529$  this give a  $\lambda$ -value of 0.55. This value is the same as Pisinger and Illes best  $\lambda$ -value obtained when  $q = 43$ . This could indicate that the structured approach of systems of conic sections, seem to beat the simulated annealing in its present form. when  $q$  get large enough.

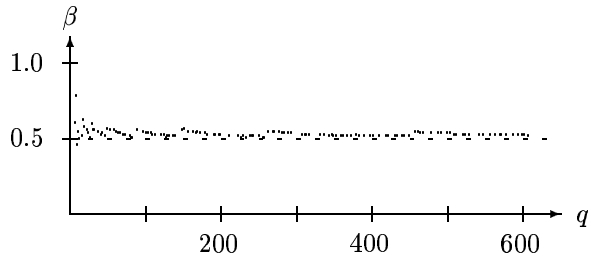


Figure 11:  $|B|/q \log_2 q$  as function of  $q$  (Simulated annealing cover)

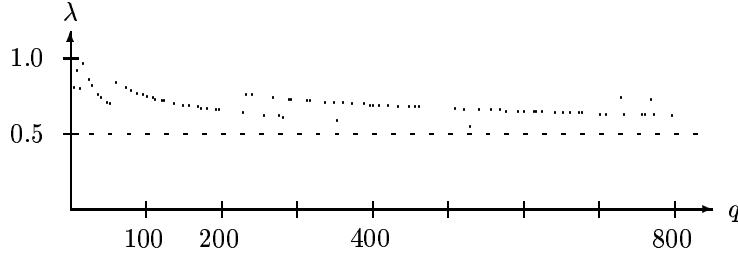


Figure 12:  $\gamma(B)/\log_2(q^2 + q + 1)$  as function of  $q$  (Norm fibre cover  $q \equiv 1 \pmod{4}$ )

### 5.3.2 Hyperbola cover

```

<< select base hyperbola >>
while (<< uncovered lines exist >>)
do
    << select q-1 affine points on hyperbola
    maximizing line cover >>
od

```

This algorithm select conic sections that are roots of  $XY - kZ^2$ . These hyperbolas have the X-axis and the Y-axis as tangents on  $L_\infty$  for all  $k \in \mathbb{F}_q$ . The algorithm starts with selecting the roots of  $XY - Z^2$ , and then add  $q - 1$  affine points corresponding to some  $k$ , that minimizes the number of uncovered lines in each iteration. (See fig.18 and 19)

**Remark 22** *This algorithm produce even better coverings. We observe that we for blocking sets in char 2 planes, we get the constant covering number of 6 for  $q > 4$ . A proof of this covering number is given in subsection 3.6. For other  $q$  than  $2^n$  we see very good coverings, nearly as good as the coverings produced by Pisinger and Illes. However these solutions are not minimal covering sets, albeit they have nice low  $\beta$ -values. Now we will see what happens to  $\lambda$  and  $\beta$  when we apply a simple reduction scheme to remove superfluous points from the blocking sets.*

### 5.3.3 Hyperbola cover with reducer

```

<< select base hyperbola >>
while (<< uncovered lines exist >>)
do
    << select q-1 affine points on hyperbola
    maximizing line cover and the number of 2-secants>>
od
while << there exist a non-empty set of points P not incident with any 1-secant >>
do
    << select a p in P incident with the most maxheight-secants
    and the least 2-secants >>
od

```

This algorithm is practically identical to the previous one. Except that it along with maximizing line cover, it also maximizes the number of 2.secants as a second priority. When the hyperbola solver

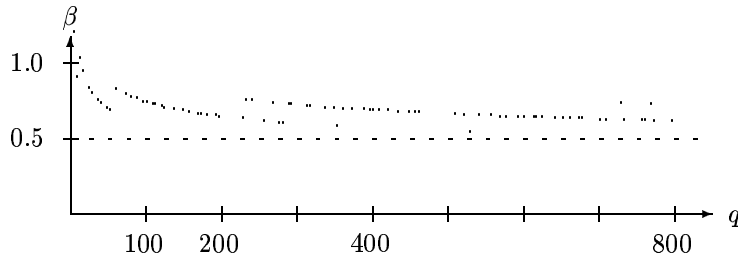


Figure 13:  $|B|/q \log_2 q$  as function of  $q$  (Norm fibre cover  $q = 1 \pmod{4}$ )

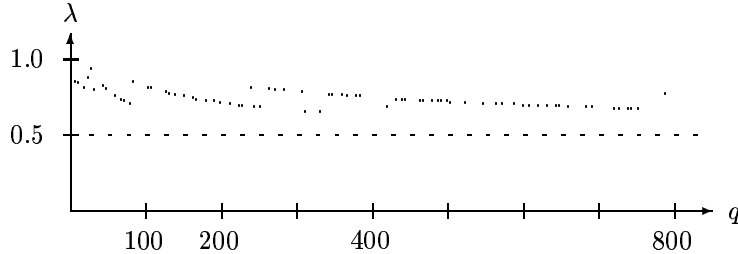


Figure 14:  $\gamma(B)/\log_2(q^2 + q + 1)$  as function of  $q$  (Norm fibre cover  $q = 3 \pmod{4}$ )

terminate, the resulting blocking set is slimmed into being minimal, using a very simple heuristic for removing superfluous points. The reducer cannot remove points on 1-secants, and seek to remove points incident with lines with maximal height, having the least 2-secants. The  $\lambda$  and  $\beta$  graphs have the figure numbers 20 and 21 respectively.

**Remark 23** *Note how the  $\beta$  values dropped from an average of 0.7 to an average of 0.6. The reducer did not make the  $\lambda$  values drop significantly. We can see from the graphs that the height increase is delayed, meaning that the first instance of  $\gamma(B) = 12$  is moved from  $q = 211$  without the reducer to  $q = 241$ . The first instance of  $\gamma(B) = 14$  is moved from  $q = 661$  to  $q = 719$ .*

## 6 Conclusions

First of all I think, that I have made a good choice studying the projective Galois planes using algebraical geometry. It has made it a lot easier to understand various aspects, and made it easier to prove things because of the mathematical connection. The coordinatized framework make construction from polynomials straight forward, thus instantly giving rich information about how to construct a large amount of subsets of points with a bounded height. In general for irreducible polynomials these sets have size approximately  $q + 1$ , so it is fairly large sets.

The proposal made by Erdős is unfortunately not closed by this piece of work, but I think we are getting closer to make this come true. It is clear to me that the problem should be attacked in a context projective planes built over prime fields. I know for a fact, that should I work out better deterministic algorithms, that in a constructive manner give witness to a low upper bound, I would really have to think hard.

### 6.1 structure

In general the results observed are independant of the characteristic of the underlying field, of course there are geometric constructions like that of Boros for odd primes and mine for char 2, that use the special properties of the additive group structure, but Boros construction is of academic interest only when the prime number is large and the power is small.

Another major structural property, that finite fields have, is contained within the multiplicative group. So we take a look at the empirical results for lets say  $q = 569, 571$  and  $q = 601, 603$ . These are pairs of

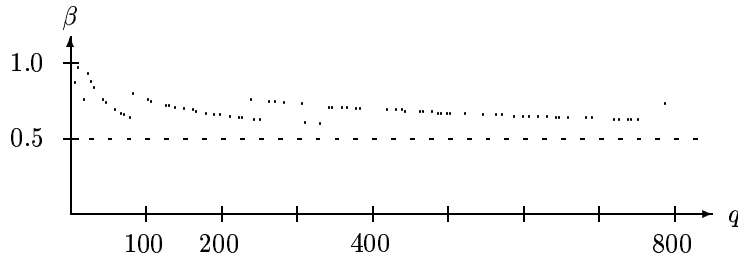


Figure 15:  $|B|/q \log_2 q$  as function of  $q$  (Norm fibre cover  $q = 3 \pmod{4}$ )

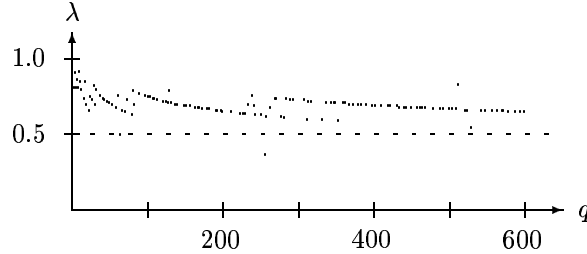


Figure 16:  $\gamma(B)/\log_2(q^2 + q + 1)$  as function of  $q$  (Norm fibres reduced)

primes with the minimal difference of 2. One of them is equal to 1 modulo 4 and the other 3. If we take a look at both the best of my results and Pisinger and Illes' results, there is no significant difference in neither  $\beta$  nor  $\lambda$ .

The only thing that really link a pair of primes like 601 and 603 is that the multiplicative groups are cyclic with 600 and 602 elements, that are equally large. Of course there is a difference in the exact occurrences of unit roots, and these pairs often differ with both magnitude of and the number of primefactors in  $q - 1$ . The only thing that they have in common is the divisability with 2, and this is a property of any power of any odd prime number.

When we look at the graphs I cant help getting associations to a lower bound for  $\gamma$  given  $q - 1$ , and  $p$  cyclicity, obtained by an adversary agument like one that can be obtained for binary search given well ordering,

It does seem like we can use  $p$ -cyclicity in the constructions of Boros, and working with conic sections the  $q - 1$ -cyclicity is quite evident.

We can observe equivalence classes of conic sections in the hyperbola solver based on the number of uncovered lines there would be left, after adding a hyperbola.

As an example we can take the hyperbolas used for hyperbola cover. We start by selecting two points, thus covering  $2q - 1$  lines, leaving  $q(q - 1)$  lines uncovered. Now adding any union of conic sections from the hyperbola system to the selected set, will leave a number of uncovered lines that is a multiple of  $q - 1$ . This fact give a rigid indication that the  $q - 1$ -cyclicity can be used for construction purposes. This  $q - 1$  divisability made it possible for my solvers to actually show the distribution of uncovered lines while probing the conic section, and that was due to the possibility of making bucket sort i  $q$  buckets in time  $O(q)$ , using a frequency table. These distributions seem to follow a general pattern while the hyperbolas are selected. I have of course only seen the patterns from the paths chosen by my algorithms.

The question is now how bad are conic section coverings compared to optimal utilisation of the information given by the cyclic structure. And in my opinion they cannot be far from optimal. If the  $\log_2 q$  lower bound hold then the solutions given by the hyperbola solver select at most one conic section to many. That is the same as saying that the minimal covering number is at most 2 less, where i think Pisinger and Illes have results at most one above the optimum for large  $q$ .

Why does it seem a hard task to build sets like Boros construct ? There are at least two possible answers to that one is, that there are the actual number of configurations of points that satisfies Boros property is small compared to the number of conic sections that satisfy the Abbott-Liu property. An other reason i can think of is that it might be easy to violate the Boros property in a local decision based

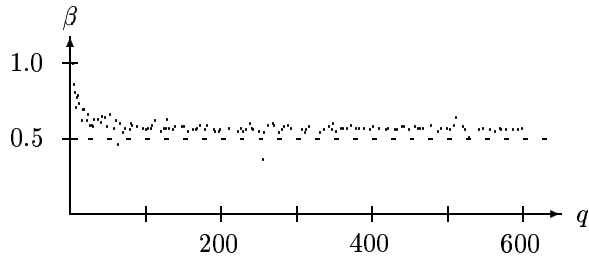


Figure 17:  $|B|/q \log_2 q$  as function of  $q$  (Norm fibres reduced)

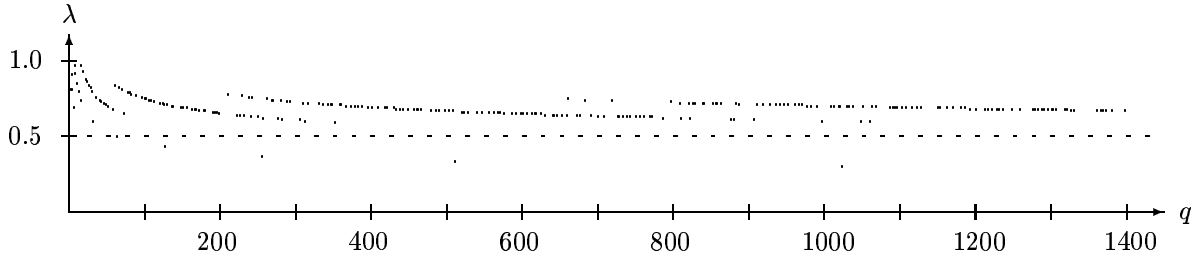


Figure 18:  $\gamma(B)/\log_2(q^2 + q + 1)$  as function of  $q$  (Hyperbola cover)

on heights in an unrecoverable manner.

What characterize good solutions is the number of lines in the cover with low height are higher. Having an average of many tangents in the points make give better line cover pr. point. Not that having the most tangents make the best blocking set. We keep in mind that the line cover of a line is all of the lines of the plane, and further each point of the line is incident with  $q$  private lines, and only share the line with the other points. So trying to build blocking sets out of this criterion could very well make algorithms diverge.

On the other hand it seem to characterize Pisinger and Illes results that the number of 1-secants to the solving set is large. But the solving set of size 529 from the norm fibre solver does not have many 1-secants. There are only a  $q$ -linear number, in this case 530.

If i should make a second attempt to bring the upper bound for the covering numbers for blocking sets down, I would probably use genetic algorithms or maybe even simulated annealing using more knowledge of the various structure at hand to speed up computation. I would also study how the multiplicative group is connected to better solutions, because I am positive, that the last information, that we can squeeze out of these projective Galois planes is hidden within the multiplicative relation in the underlying field. And I do not think, that it has been used to the full extend. It still seems like a combinatorical problem to find the right conic sections, albeit it does not seem as tough as solving the problem in terms of single points. Note that there exist  $q$  for which a conic section solver is better than the others. So it does not seem like there exist a simple algorithm, that find the best solution in terms of conic sections, and still we do not know if optimal solutions can be found in terms of conic sections. If we knew that, it we would have solved the problem of Erdős, due to the proof of Ughi, that say that a fixed number of conic sections, never form a blocking set, if we let  $q$  grow (see thm.6).

## 6.2 Does Erös' constant hold

I do not think that it is possible, that there can exist a constant  $\gamma_E$  so any projective plane has property  $B(\gamma_E)$ . Let me put this into perspective by some of observations that can done by inspecting the figures.

The  $\beta$ - and  $\lambda$ -values are close related. We see that for the solvers producing better than the Abbott-Liu bound the  $\beta$ -graphs look like a smoothed version of the  $\lambda$ -graph. We see from the Smart Brute solver that the  $\beta$ -value 0.5 is reached relatively fast. This solver produce  $\beta$ -values below 0.5, when  $q$  get large as a matter of fact it has dropped to 0.47 when  $q$  we can see that while too greedy making small blocking sets we loose the grip on the height.

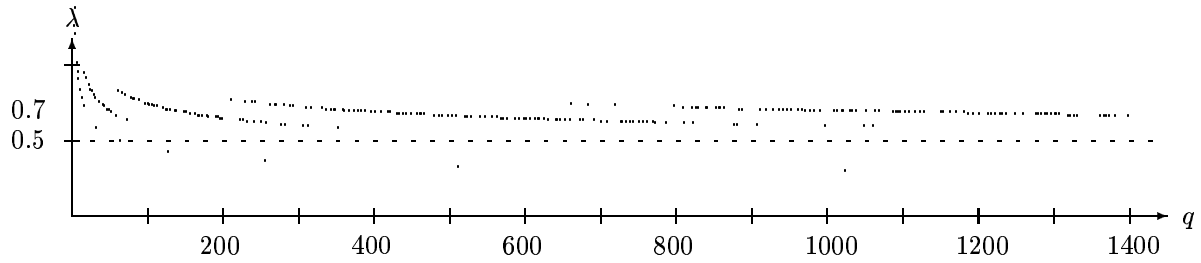


Figure 19:  $|B|/q \log_2 q$  as function of  $q$  (Hyperbola cover)

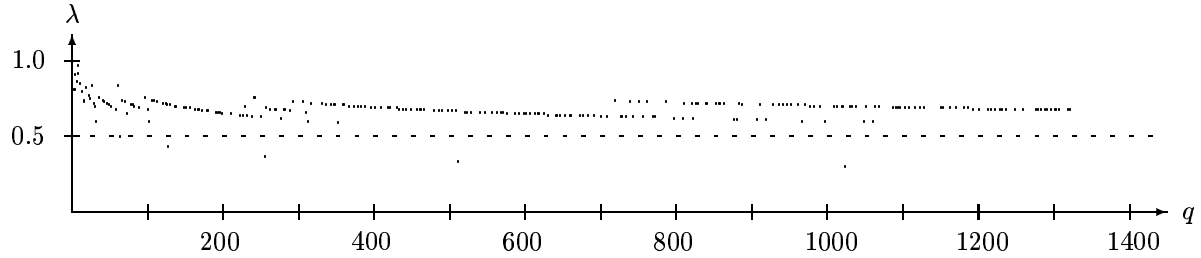


Figure 20:  $\gamma(B)/\log_2(q^2 + q + 1)$  as function of  $q$  (Hyperbola cover reduced)

The thing that is remarkable from studying Pisinger and Illes results, is that the only  $\beta$ -value found below 0,5 is that of  $q = 3^2$  where the algorithm find a blocking set with  $\gamma(B) = 3 + 1$  that match Bruen-Fisher, even with a smaller size than the sets obtained by the construction proved Boros, that is 13 compared to 18 (see the Tables and [Szönyi 2]p.102). It seems that the Abbott-Liu limit could be lowered, as my conic section solvers based on simple systems of conic sections can keep under a  $\lambda$  value under 0.75, and seem to be better when  $q$  get larger.

My major guess is that projective planes built over primefields cannot contain blocking sets with a covering number less than  $\log_2(q)$ , so that  $\log_2 p < B(PG(2, p))$  where  $p$  is a prime. I think that if there exist a proof of this, it will relate to the multiplicative group of  $\mathbb{F}_p$ . But the bottom line of this project is that constant cover of finite projective planes is still open.

## 7 Tables

### 7.1 General information of solutions

The tables 1 and 2 display some general information

- $q$  is the size of the underlying finite field.
- $n$  is the number of lines in  $\Pi_q$
- $B$  is the found blocking set, and  $|B|$  is the number of points in  $B$
- $\gamma(B)$  denotes the cover number of the blocking set
- $\lambda$  denote the quotient  $\frac{\gamma(B)}{\log_2(n)}$
- $\beta$  denote the quotient  $\frac{|B|}{q \log_2(q)}$

### 7.2 Lineprofiles of solutions

The tables 3 to 6 show the line profiles of the found blocking sets, thus showing the equivalence classes of  $i$  - secants of the found blocking set.

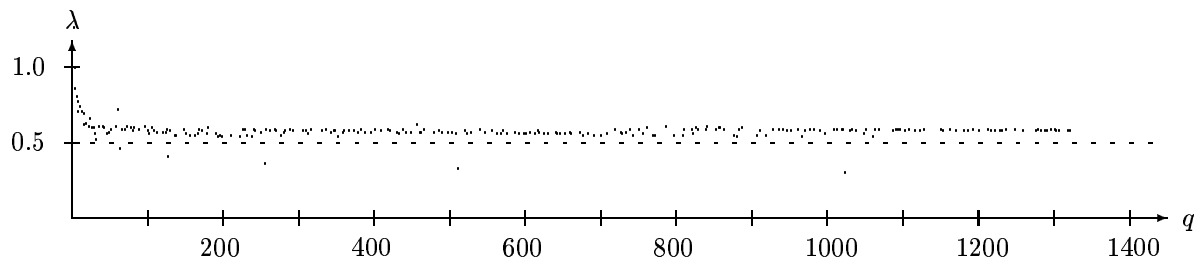


Figure 21:  $|B|/q \log_2 q$  as function of  $q$  (Hyperbola cover reduced)

### 7.3 Lineprofiles in char 2

Table 7 show the line profiles for  $PG(2, 2^n)$  where  $2 \leq n \leq 12$  and show polynomials that count the number of  $i$ -secants of the generated blocking set.

### 7.4 Distributions of $i$ -secants

The figures after the tables of the line profiles illustrate the line profiles of selected entries from the tables. The figure zoom into an  $y$ -range of the maximum frequency, and the  $x$ -range is the covering number of the found blocking set.

## 8 Implementation

### 8.1 Language selection

I have selected  $C++$  as the implementation language. The main reason is portability, since i have been working on a number of different architectures. Another reason is that  $C++$  supports overloading. This means that the abstract multiplication and addition of the finite fields can be implemented with '\*' and '+' in the sourcecode. This makes debugging and testing easier, and makes the source more readable outside of the field class. Another significant reason to select  $C++$  is the egcs compiler, that is shipped with linux distributions. This  $C++$  compiler produces very good code and is ansi compliant.

To keep the overview the various parts of the project are modularised into a hierarchy of classes. The  $3 \times 3$  matrix class is build on top of the plane class, and the plane class is build on top of the finite field class. This strategy makes testing and debugging a much easier task, rather than having a huge bowl of spaghetti, albeit the Plane class is a bit messy at present time of writing.

### 8.2 Primering class

In order to build the tower of  $\mathbb{F}_q$  we need an implementation of  $\mathbb{F}_p$ . This is the primering of  $\mathbb{F}_q$ , ie. the subring generated additively by 1. It is implemented as the ring  $\mathbb{Z}/p\mathbb{Z}$  with the straight forward addition and multiplication of  $\mathbb{Z}$  modulo  $p$ . In order to generate  $\mathbb{F}_q$  we can say that we extend  $\mathbb{F}_p$  with an element that has order  $q - 1$  with respect to multiplication.

```

struct PrimeRing {
private:
    static bool          is_initialized;
    static size_t       mod_table_size;
    static PrimeType*   mod_table;
    static PrimeType*   sub_table;
    static size_t       exp_table_size;
    static PrimeType*   exp_table;
    static size_t       log_table_size;
    static PrimeType*   log_table;
    static PrimeType*   i;

public:
    // Constants
    static PrimeType    PRIME;      // Logical const FIXME
    static const PrimeRing MIN;
    static const PrimeRing MAX;
    static const PrimeRing SIGMA;
    // Initialization
    static void         init(ostream& = silent);
    static void         term();
    // Constructors
                                PrimeRing();
                                PrimeRing(int);

    // Conversion
    int                 intOf() const;
    void               ofInt(int);
    // Observers
    int                 domainSize() const;
    static size_t      memSize();
    // Arithmetic
    PrimeRing&         operator++();
    PrimeRing          operator++(int);
    PrimeRing&         operator--();
    PrimeRing          operator--(int);
    PrimeRing          operator-() const;
    PrimeRing&         operator+=(PrimeRing);
    PrimeRing&         operator-=(PrimeRing);
    PrimeRing&         operator*=(PrimeRing);
    friend PrimeRing   operator+(PrimeRing, PrimeRing);
    friend PrimeRing   operator+(int, PrimeRing);
    friend PrimeRing   operator+(PrimeRing, int);
    friend PrimeRing   operator-(PrimeRing, PrimeRing);
    friend PrimeRing   operator-(int, PrimeRing);
    friend PrimeRing   operator-(PrimeRing, int);
    friend PrimeRing   operator*(PrimeRing, PrimeRing);
    friend PrimeRing   operator*(int, PrimeRing);
    friend PrimeRing   operator*(PrimeRing, int);
    // Comparison
    friend bool        operator==(PrimeRing, PrimeRing);
    friend bool        operator!=(PrimeRing, PrimeRing);
    friend bool        operator<(PrimeRing, PrimeRing);
    friend bool        operator<=(PrimeRing, PrimeRing);
    friend bool        operator>(PrimeRing, PrimeRing);
    friend bool        operator>=(PrimeRing, PrimeRing);
    // Io
    friend ostream&    operator<<(ostream&, PrimeRing);
    friend istream&    operator>>(istream&, PrimeRing&);
    // Functions
    static PrimeRing   exp(PrimeRing);
    static PrimeRing   log(PrimeRing);
    // Information
    static void        output_statistics(ostream&);
    static void        output_mod_table(ostream&);
    static void        output_exp_table(ostream&);
    static void        output_log_table(ostream&);
};

```

**Remark 24** *It is always important to consider the time-space trade off when implementing things. This is of course also the case here. We can implement constant time operations by using  $2 \times 2$  tables using quadratic space for each operator. By paying linear space it is however the case, that we still can obtain constant time operation for the multiplication and division (see sec.8.3.3). It might seem silly to have the prime ring in a separate class, since prime fields simply are ordinary integers modulo a prime number, but the point is encapsulation of the properties of these integers, in an environment that is a sort of aware of the  $p$ , and  $p - 1$  cyclicity, that is used in the boot-strapping process of the resulting extension field of this the prime ring, that is the smallest non trivial sub field in all fields of characteristic  $p$ , or should we say  $p$ -cyclic addition.*

### 8.3 Finite field class

The finite fields are implemented as splitting fields over  $\mathbb{Z}/p\mathbb{Z}$ , where  $p$  is some prime. The next step is to build  $\mathbb{F}_q$  from  $\mathbb{F}_p$ . This is done by finding an irreducible  $n$ 'th degree polynomial  $P_n$  in  $\mathbb{F}_p[X]$  by building an list of irreducible LC1 polynomials of degrees less than  $n$  over  $\mathbb{F}_p$ . When  $P_n$  has no roots in  $\mathbb{F}_p$  and none of the irreducible polynomials of degree less or equal to  $\frac{n}{2}$  divides it by polynomial division using the arithmetics of  $\mathbb{F}_p$ , then  $P_n$  is irreducible. When  $P_n$  is irreducible in  $\mathbb{F}_p[X]$  then it represent a maximal ideal in  $\mathbb{F}_p[X]$ , and then the factorial ring  $\mathbb{F}_p[X]/(P_n)$  becomes a field. So the polynomials of degree less than  $n$  in  $\mathbb{F}_p[X]$  represents the elements of  $\mathbb{F}_q$ . The actual representation is an integer. The enumeration is done as follows : Take an element  $k$ , this is represented as a polynomial over  $\mathbb{F}_p$ , but it is also a polynomial over  $\mathbb{Z}[X]$ , and here we evaluate it in  $p$ , thus yielding an integer between 0 and  $p^n - 1$ . This way we can get back and forth between finite field elements and integers.

```

struct PolyField {
    friend struct          LongPoly;
    typedef list<PolyField> IrreducibleList;
    typedef list<PolyField>::iterator IrreducibleIterator;
    enum OutputFormat     { Verbose, Terse };
private:
    static bool           is_initialized;
    static int            prime_width;
    static int            poly_width;
    static OutputFormat   output_format;
    static size_t         poly_size;
    static size_t         coeff_table_size;
    static PrimeRing**   coeff_table;
    static size_t         shift_table_size;
    static DomainType**  shift_table;
    static size_t         exp_table_size;
    static DomainType*   exp_table;
    static size_t         log_table_size;
    static DomainType*   log_table;
    static int*           orders;
    static int            idx;
    static int*           divisors_of_n;    // for the galois group
    static size_t         num_div_of_n;
    static int*           divisors_of_mg;   // for mult group of fp
    static size_t         num_div_of_mg;
    // Output functions
    static void           output(ostream&, int, PrimeRing*);
    static void           output_verbose(ostream&, int, PrimeRing*);
    static void           output_terse(ostream&, int, PrimeRing*);
public:
    // Constants
    static int            DOMAIN_SIZE;     // Logical const FIXME
    static const PolyField MINUS_ONE;
    static const PolyField ZERO;
    static const PolyField ONE;
    static const PolyField MAX;
    static const PolyField SIGMA;
    static LongPoly      MAX_IDEAL;       // Logical const FIXME
    // Initialization and termination
    static void           init(ostream& = silent);
    static void           term();
    // Constructors
    PolyField();
    explicit PolyField(int _idx);
    explicit PolyField(const int*);
    // Conversion
    int                   intOf() const;
    void                  ofInt(int);
    string                stringOf() const;
    int                   num_divisors_of_mulgroup(){ return num_div_of_mg; }
    int                   *divisors_of_mulgroup(){ return divisors_of_mg; }
    // Observers
    bool                  isSquare() const;
    int                   degree() const;
    int                   order() const;
    int                   subfield() const;
    // Arithmetic
    PolyField             operator-() const;
    PolyField&            operator+=(PolyField);
    PolyField&            operator-=(PolyField);
    PolyField&            operator*=(PolyField);
    PolyField&            operator/=(PolyField);
    friend PolyField      operator+(PolyField, PolyField);
    friend PolyField      operator-(PolyField, PolyField);
    friend PolyField      operator*(PolyField, PolyField);
    friend PolyField      operator/(PolyField, PolyField);
    // Functions
    friend PolyField      sqrt(PolyField);
    friend PolyField      pow(PolyField, int);
    friend int            sigpow(PolyField);
    // Evaluation
    int                   operator()(int) const;
    // Element access
    PrimeRing             operator[] (int) const;
    // Comparison
    friend bool           operator==(PolyField, PolyField);
    friend bool           operator!=(PolyField, PolyField);
    // Io
    static void           setOutputFormat(OutputFormat);
    friend ostream&       operator<<(ostream&, PolyField);
    friend ostream&       operator<<(ostream&, const LongPoly&);
    // Statistics
    static void           output_field_properties(ostream& = cout);
    static void           output_coeff_table(ostream&);
    static void           output_shift_table(ostream&);
    static void           output_log_table(ostream&);
    static void           output_exp_table(ostream&);
};

```

### 8.3.1 Initialisation

In order to boot-strap this field, we will have to use elements represented as polynomials over the prime ring, meaning that the coefficients are elements of the prime ring. All arithmetic computation is carried out in terms of polynomials. This is by no means constant time operators, but it is the only way, that we can construct generators of the cyclic multiplication or tables for fast lookup as static data to the class. The polynomials have coefficients in the prime ring. The steps of the initialisation is as follows:

1. Collect a list of all irreducible polynomials with a degree less than  $n$ , when we want to construct a field with  $p^n$  elements.
2. Traverse the polynomials over  $\mathbb{F}_p$  of degree  $n$  starting with  $x^n$ , in order to find a polynomial with no divisors in the irreducibles list. This is the irreducible degree- $n$  polynomial, that we select to use as prime divisor in the polynomial ring over  $\mathbb{F}_p$ .
3. Sieve out a generator for the multiplicative group, using the cyclic nature of the multiplication. Start flagging all elements but zero as taken. Scan over the taken array always picking the next non-taken element. Save this element as the last taken. The algorithm terminates when all is taken, Meaning that the last taken element was capable of generating all of the multiplicative group. This element is saved as a static constant. This is done using straight forward polynomial multiplication focusing on the remainder by division with our irreducible degree- $n$  polynomial.
4. Construct the log and exp tables, by mapping each element but zero back and forth to it's power of the generator.

### 8.3.2 Addition and subtraction

There is no real fast way other than using tables for the addition. The next to best solution is the straight forward addition of the polynomials. This will take up static space  $o(np^n)$ , if we want the polynomials explicitly represented. We will in this case obtain an execution time of  $o(n)$  in terms of additions in the prime ring. Having the prime ring as a class we can safely use space  $o(2p)$  to obtain this addition without divisions at this leaf level. In the present implementation the addition is implemented with a time complexity of  $o(n^2)$ , but without taking space for the explicit polynomials. This is obtained by utilizing the implicit representation using integers. This is cause to some slowdown for large powers of small prime numbers compared to primes in the same order of magnitude. As the fields observed within this project are relatively small, it would be more adequate to use tables, at least for the addition, because the additive group has the nature of an  $n$  dimensional vector space over  $\mathbb{F}_p$ .

### 8.3.3 Multiplication and division

I have selected to carry out multiplications and divisions in terms of a generator of the multiplicative group of the field, by paying a constant overhead compared to simple lookup using quadratic space. This is done by using the bijection from non zero elements to their power of the selected generator. This way we may use, a bit misleading perhaps,  $a * b = \exp(\log(a) + \log(b))$ , when  $a, b$  are non zero. The misleading part is that log map from elements into integers modulo  $p^n - 1$  and exp map from integers into values in the field. So the overhead paid for a single multiplication is that of three lookup operations on arrays and a single integer addition, so this operation is truly constant using  $o(2p^n)$  space.

### 8.3.4 Square roots and powers

Square roots and powers are multiplicative notions. In order for an element to be a square it has to have an even multiplicative order, so naturally with a generator view, we see that the square elements of a field are those that are an even power of the generator, if the multiplicative group has an even order. This is always the case when  $p$  is odd, and never the case when  $p = 2$ . In the “ $p$  odd” case we can use the log table to determine if an element is a square. If  $p = 2$  every element is a square, because squaring is an automorphism on groups of odd order. This means that the `isSquare()` function is implemented in

constant time. The squareroot of  $k$  where  $k = \sigma^{2i}$  is equal to  $+\sigma^i$  when  $p$  is odd, and when  $p = 2$  then  $\sqrt{k} = \sigma^i$ ,  $k = \sigma^{2i}$  and  $\sqrt{k} = \sigma^{\frac{i+2^n-1}{2}}$ ,  $k = \sigma^{2i+1}$ . The presence of the exp and log tables ensure constant time operation in either case. Power of elements is also implemented using the generator tables. When  $k = \sigma^i$  then  $k^m = \sigma^{im} = \sigma^{im \bmod p^n - 1}$ . Again we do constant time integer computation, using the log to lookup  $i$  execute the adequate integer computation, and then looking up the resulting value using the exp table.

**Remark 25** *All the standard arithmetic operators have been overladed with the primefield class, making trouble for the compiler and more fun for the programmer. Note that there cannot be a well defined comparison operators other than  $=$  and  $\neq$ . The root and power functions are overloading “sqrt(),exp(),log()” and “pow()” functions of the math library. The objects are streamable, so it is possible to print elements as either polynomials, or as vectors of coefficients within square brackets, having the leading coefficient leftmost. Picture an integer translated into base  $p$ .*

## 8.4 Plane class

The plane class implements coordinates for points in  $PG(2, q)$ , as a 3-tuple of elements of  $\mathbb{F}_q$ . It implements a number of handy operations such multiplications with a constant, addition of two 3-tuples, primarily for use with the matrix operations, but also to generate parametric production of points incident with a line.

```

extern int    plane_size;
extern int    field_size;
extern int    line_size;

extern int*   point_by_norm;
extern int*   norm_map;

extern KTYPE* norm_table;
extern int*   freq_table;
extern int*   k_kind;
extern int*   p_kind;

extern int*   tang;
extern int    tang_map[];
extern int*   tang_bit;

extern int*   conic;

void plane_Init();
int isPointColinear(const int points[], int count,int point);
int isColinear(const int points[], int count);
void rootsOf(int p,int roots[]);
int ellipticRoots(int a,int b,int roots[]);
int ellipticRoots2(int a,int b,int c,int d,int roots[]);
int radicalRoots(int roots[]);
int rad_2Roots(int roots[]);
int rad2_tangents(int roots[]);

struct Point {
    KTYPE x,y,z;
public:

static const Point I;
static const Point J;
static const Point K;

    Point():x(KTYPE(0)),y(KTYPE(0)),z(KTYPE(0)){
    Point(KTYPE _x,KTYPE _y,KTYPE _z):x(_x),y(_y),z(_z) {}
    Point(int p){
        int radix=KTYPE::DOMAIN_SIZE;
        int rad2 = radix*radix;
        assert(0 <= p <= (rad2+radix));
        int i=p % radix,j,k;
        if (p<rad2) {
            k=1; j=p / radix;
        } else if ((p==rad2)<radix) {
            k=0; j=1;
        } else {
            k=0; j=0; i=1;
        }
        x=KTYPE(i);
        y=KTYPE(j);
        z=KTYPE(k);
    }

    int intOf(){
        int radix=KTYPE::DOMAIN_SIZE;
        if (z != KTYPE(0)) {
            KTYPE i(x/z);
            KTYPE j(y/z);
            return i.intOf()+radix*j.intOf();
        }
        else if (y != KTYPE(0)){
            KTYPE i(x/y);
            return radix*radix+i.intOf();
        }
        else return radix*(radix+1);
    }
};

int roots(int root_result[])

friend bool isOrtho(Point l,
    return ((l.x*r.x+l.y*r.y+l
}
friend Point operator+(Point
friend Point operator-(Point
friend Point operator*(Point
Point& operator+=(Point r)
Point& operator-=(Point r)
friend Point operator* (Point
friend Point operator* (KTYPE

Point& operator*=(KTYPE c)
friend Point operator/ (Point

Point& operator/=(KTYPE c)

Point& normalize(){
    if (z==KTYPE(0))
        if (y==KTYPE(0)) { x=KTY
            else { x/=y;y=KTYPE(1);
        } else { x/=z;y/=z;z=KTYPE(1
}

friend bool operator==(Point
friend bool operator!=(Point

friend KTYPE norm(Point r){
friend KTYPE norm_c2(Point r
friend Point crossProduct(Po
    { return Point(l.y*r.
friend KTYPE dotProduct(Poin

friend ostream& operator <<
    { out << "(" << p.x
};

```

### 8.4.1 Point - line incidence

The plane class has a dot product and a cross product, are the functions directly linked to point-line incidence. A point  $p$  is incident with a line  $l$  iff  $dotProduct(l, p) = 0$ , where 0 of course is the zero of the underlying field. The dotProduct operation cost 3 multiplications and 2 additions in the underlying field, as the values of the coordinates are of the underlying field. The line incident with two different points, or dually speaking the point of common incidence with two different lines are computed using

the crossProduct, that take time 3 multiplications and one addition for each of the three coordinates in the resulting vector. This way assuming constant time operations in the field beneath, this operation is also constant in execution time. These two operations are carried out again and again, in inner loops of the solver algorithms, so these operators need to be fast. We cannot effort neither logarithmic nor linear time for these particular operations.

### 8.4.2 Parametric line construction

Consider the line  $l$  in the projective plane then given  $l = (x : y : z)$  it is one of three kinds.

1. It is equal to  $L_\infty$
2. It is a line through  $(0:1:0)$
3. It is a line with a “slope” in  $\mathbb{F}_q$

In the first case the points are given by the enumeration, that is  $L_\infty$  consist of the points enumerated in the range  $q^2..q^2 + q$ , where the affine points are enumerated in the range  $0..q^2$ .

In the second case all the points of these lines has the same value of  $x$ , which make it easy to generate the  $q$  affine points as  $(x:k:1)$  where  $x$  is constant and  $k$  takes every value of  $\mathbb{F}_q$ .

In the last case we take the point  $p_1 = l \cap (-1 : 0 : 1)$  and use the vanishing point  $p_\infty = l \cap L_\infty$  as slope. This way we generate the  $q$  affine points as  $p_1 + k * p_2$ , where  $k$  takes every value of  $\mathbb{F}_q$ , then  $p_\infty$  is the infinite point of the line. It's worth to mention, that this routine is of a time complexity that is linear in the field size, meaning that we pay constant time for each point on the line, which is optimal not having tables. Any way this is how the `Point.roots()` work. Given a point  $p$  then `p.roots(int[])` returns the line cover of  $p$  in the array in the argument. This routine is also used every where in the solvers, when measuring the the height of points, updating frequency tables and all sorts of things.

### 8.4.3 Miscellaneous polynomials

This is exactly some of the messy part of this class. I have been looking for any kind of relevant regularity and structure, so I have messed the class up a bit, introducing maps of tangents and secants for a base conic section, and put the norm function in tables, because the solvers, using the fibres of the norm, required fast acces to the systems of conic sections. I have implemented some generators of zero sets for some polynomials. That is the polynomials  $x^2 + y^2 - 1$ , that yields the radicals of the plane. The polynomials  $x^3 - y^2 + ax + b$ , that generates smooth elliptic curves in  $char \notin \{2, 3\}$  are implemented, and the polynomials  $x^r + y^r - 1$  that are irreducible when  $r$  is a prime number different from  $p$  have been implemented. I have implemented the  $N$  function that maps a point  $p$  into  $\langle p, p \rangle$ , as a table. I refer to this function as the “norm” of  $p$ , though it is not a norm in a metric sense, since there are  $q + 1$  points other than  $(0:0:0)$  that has a “norm” of zero. In the plane class I have implemented some static tables to map points to their norm, and I made tables ordering the points by their “norm”. This makes it easy to get all points of a certain norm. The distribution of points by norm is of interest since the different “norm” classes represent certain geometric configurations (see ??).

### 8.4.4 Matrix class

The matrix class builds upon the plane class and the finite field class. It implements  $3 \times 3$  matrices, and a bunch of support methods and friends. I have implemented determinant and inversion for the invertible matrices, that are the matrices with non-zero determinants. The class has of course matrix addition and subtraction along with multiplication. The invertible matrices form the general linear group, and is used to move sets of points and lines around in the plane.

```

struct Matrix {
public:
    Point r,s,t;
    static const Matrix UNIT;

    Matrix(Point _r,Point _s,Point _t):r(_r),s(_s),t(_t) {};
    Matrix():r(KTYPE(0),KTYPE(0),KTYPE(0)),
             s(KTYPE(0),KTYPE(0),KTYPE(0)),
             t(KTYPE(0),KTYPE(0),KTYPE(0)) {};
    KTYPE det(){
        return r.x*(s.y*t.z-s.z*t.y)+s.x*(t.y*r.z-t.z*r.y)+t.x*(r.y*s.z-r.z*s.y);
    }
    KTYPE trc(){
        return r.x+s.y+t.z;
    }

    int invert();
    friend Matrix inverse(const Matrix& m)
    {
        Matrix _m(m); _m.invert(); return _m;
    }
    friend Matrix operator*(const Matrix& m,KTYPE c) { return Matrix(m.r*c,m.s*c,m.t*c); }
    friend Matrix operator*(KTYPE c, const Matrix& m) { return Matrix(m.r*c,m.s*c,m.t*c); }
    Matrix& operator*=(KTYPE c) {r*=c; s*=c; t*=c; return *this;}
    friend Matrix operator*(const Matrix& l,const Matrix& r) {
        return Matrix(
            Point(l.r.x*r.r.x+l.s.x*r.r.y+l.t.x*r.r.z,
                l.r.y*r.r.x+l.s.y*r.r.y+l.t.y*r.r.z,
                l.r.z*r.r.x+l.s.z*r.r.y+l.t.z*r.r.z
                ),
            Point(l.r.x*r.s.x+l.s.x*r.s.y+l.t.x*r.s.z,
                l.r.y*r.s.x+l.s.y*r.s.y+l.t.y*r.s.z,
                l.r.z*r.s.x+l.s.z*r.s.y+l.t.z*r.s.z
                ),
            Point(l.r.x*r.t.x+l.s.x*r.t.y+l.t.x*r.t.z,
                l.r.y*r.t.x+l.s.y*r.t.y+l.t.y*r.t.z,
                l.r.z*r.t.x+l.s.z*r.t.y+l.t.z*r.t.z
                )
        );
    }

}

friend ostream& operator << (ostream& out,Matrix m) {
    out << "[[\" << m.r.x << '\t' <<m.s.x << '\t' << m.t.x << "\t]" << endl
        << " [\" << m.r.y << '\t' <<m.s.y << '\t' << m.t.y << "\t]" << endl
        << " [\" << m.r.z << '\t' <<m.s.z << '\t' << m.t.z << "\t]" << endl;
    return out;
}
};

inline Point operator* (const Matrix& m,Point p) {
return Point(
    p.x*m.r.x+p.y*m.s.x+p.z*m.t.x,
    p.x*m.r.y+p.y*m.s.y+p.z*m.t.y,
    p.x*m.r.z+p.y*m.s.z+p.z*m.t.z
);
}
}

```

## References

- [Bruen-Fisher] Bruen, A.A. and Fisher, J.C.: Blocking sets and complete arcs, Pacific Journal of Math. 53 (1974), 73-84.
- [Ughi] Ughi, E.: On  $(k,n)$ -blocking sets which can be obtained as a union of conics, Geom. ded. 26 (1988), 241-245
- [Abbot-Liu] Abbott, H.L. and Liu, A.: Property of  $B(s)$  and projective planes, Ars Combinatoria 20 (1985), 217-220.
- [Illes-Szőnyi-Wettl] Illés, T., Szőnyi, T. and Wettl, F.: Blocking sets and maximal strong representative systems in finite projective planes, Mitt. Math. Sem. Giessen 201 (1991), 97-107
- [Pisinger-Illes] Illes, T. and Pisinger D.: Upper bounds on the covering number of Galois-planes, with small order Diku Rapport Nr. 97/13 (1997) ISSN 0107-8283

- [Szönyi 1] Szönyi, T.: Note on the existence of large minimal blocking sets in Galois planes, *Combinatorica* 12 (2) (1992), 227-235
- [Szönyi 2] Szönyi, T.: Blocking sets in finite planes and spaces, *Ratio Mathematica* numero 5 (1992), 93-106
- [Boros] Boros, E.:  $PG(2, p^s)$ ,  $p > 2$  has property  $B(p+2)$ , *Ars Combinatoria* 25 (1988), 111-114
- [Albert-Sandler] An Introduction to Finite Projective Planes, Athena series printed by Holt, Rinehart and Winston, Inc (1968) Library of Congress Catalog Card Number: 68-20076
- [Jensen] Jensen, C.U.: *Klassisk Algebra Lecture notes for Mat 3 Algebra* (1997) ISBN 87-7834-228-7
- [Schmidt] Schmidt, A.L.: *Algebraisk talteori, Lecture notes on algebraic numbertheory* (1998) ISBN 87-7834-251-1
- [Kárteszi] Kárteszi F.: *Introduction to finite geometries*, (Akadémiai Kiadó, Budapest 1974).
- [Johan P] Hansen, J.P.: *Algebra Lecture notes for M21 at University of Aarhus volume 1-3* (1992).
- [Artin M.] Artin M.: *Algebra* Prentice Hall (1991) ISBN 0-13-004763-5
- [Cormen,Leiserson and Rivest] T.H. Cormen, C.E. Leiserson and R.L. Rivest: *Introduction to Algorithms*, The MIT Press (1990) ISBN 0-262-53091-0
- [Skyum] Skyum, S.: *Algebraiske algoritmer, Lecture notes from the dAlg course at University of Aarhus* (1994)
- [Stoustrup] Stoustrup B.: *The C++ Programming Language - Second Edition*, Addison-Wesley Publishing Company (1991) ISBN 0-201-53992-6
- [Brandt and Nissen] Brandt, J. and Nissen K.: *Q.E.D. - en introduktion til det matematiske bevis*, Abacus (1995) ISBN 87-89182-52-9
- [Pierce] Pierce, B.C.: *Basic Category Theory for Computer Scientists*, second printing (1993) Foundation of Computing Series, The MIT Press, ISBN 0-262-66071-7
- [Edwards and Penney] C.H. Edwards Jr. and D.E. Penney: *Calculus and Analytic Geometry*, Third Edition (1990). Prentice Hall International Editions ISBN 0-13-111006-3
- [Bentley] Bentley, J.L.: *Programming Pearls*, 20th Printing (1997) Addison-Wesley Publishing Company, ISBN 0-201-10331-1
- [Artin E.] Artin, E.: *Geometric Algebra*, Wiley Classics Library Edition (1988) ISBN 0-471-60839-4
- [Cucker and Shub] Cucker, F. and Shub, M.(Eds.): *Foundation of Computational Mathematics, Selected Papers of a Conference, held at IMPA in Rio de Janeiro, January 1997*, Springer Verlag, ISBN 3-540-61647-0
- [Ginsberg] Ginsberg, M.: *Essentials of Artificial Intelligence*, Morgan Kaufmann Publishers (1993) ISBN 1-55860-221-6
- [Scott] Scott, W.R.: *Group Theory*, Dover Publications, Inc. (1987), ISBN 0-486-65377-3
- [Katz] Katz, V.J.: *A History of Mathematics, an Introduction*, HaperCollins College Publishers (1993), ISBN 0-673-38039-4
- [Schroeder] Schroeder, M.R.: *Number Theory in Science and Communication*, Springer series in information sciences vol. 7, third edition (1997) ISBN 3-540-62006-0