COUNTING SOLUTIONS TO SOME POLYNOMIAL EQUATIONS OVER FINITE FIELDS

STICLARU, Gabriel
Faculty of Mathematics and Informatics
Ovidius University, Romania
gabrielsticlaru@yahoo.com

Abstract

We study some equations over finite fields. To solve, we use C++ and Computer Algebra Software like Singular. For C++ we use free, high-performance library, for big numbers, like GMP and NTL. Based on our simulations, we can conjecture, prove and test polynomials to represent the number of solutions.

Keywords: *equations over finite fields, Gröbner basis, varieties polynomial count, Computer Algebra Systems (Singular)*

AMS Classification: 13P10, 13P15, 12Y05

1. Introduction

Solving polynomial equations in finite fields plays a fundamental role in many important areas such as coding theory, cryptology, design and analysis of computer hardware. To find efficient algorithms to solve such equations is a central issue both in mathematics and in computer science.

We discuss the following problems: let f be a polynomial with coefficients in a finite field $\mathbb{K}$. We try to count or estimate in an efficient manner the number of solutions to the equation $f = 0$ over the base field $\mathbb{K}$.

The main focus will be on Gröbner bases and how the theory of Gröbner bases can be used to solve polynomial equations. Gröbner Bases are routinely available in current mathematical software systems like Mathematica, MatLab, Maple. Special Computer Algebra Software, like CoCoa, Macaulay, Singular, are to a large extent devoted to the applications of Gröbner Bases. All the basic operations such as addition, subtraction, inverse, multiplication are implemented.

Finally we presents some computational results obtained by experiments: theoretical aspects or conjectures with brief explanations. To compare, we tried to use different methods to solve the same sample by the same computer.

The computations were carried out on a DualCore Intel, 2200 MHz, 2 GB RAM, running Windows and using the computer algebra system Singular.

2. The algebraic preliminaries

In this section we recall all the definitions and results, which we need later.

For notational simplicity, we use $K[x]$ to denote the polynomial ring in n variables $K[x_1, \ldots, x_n]$, x^a for $x_1^{a_1} \cdots x_n^{a_n}$ and $|a| = a_1 + \ldots + a_n$

Definition 1. A subset $I \subset K[x]$ is an ideal if it satisfies:

1. $0 \in I$.
2. If $a, b \in I$, then $a + b \in I$.
3. If $a \in I$ and $b \in K[x]$, then $a \cdot b \in I$.

The two most important examples of polynomial ideals for our purposes are the following:

- The set of polynomials that vanish in a given set $V \subset \mathbb{K}^n$, i.e.,

$$I(V) = \{f \in K[x] : f(a) = 0, \forall a = (a_1, \ldots, a_n) \in \mathbb{V}\}$$

- The ideal generated by a finite set of polynomials $F = \{f_1, \ldots, f_m\}$, defined as (F) or $< F >$,

$$(f_1, \ldots, f_m) = \{f : \ f = g_1 f_1 + \ldots + g_m f_m, g_1, \ldots, g_m \in K[x]\}$$

Note that the variety depends only on the ideal of F, i.e. $V(F) = V(< F >)$.

An ideal is finitely generated if it can be written as in the above form, for some finite set of polynomials $\{f_1, \ldots, f_m\}$. An ideal is called principal if it can be generated by a single polynomial. The intersection of two ideals is again an ideal.

Theorem 1. (Hilbert Basis Theorem) Every polynomial ideal in $K[x]$ is finitely generated.

We need to define the concept of an algebraic variety as the zero set of a set of polynomial equations (zero dimensional).

Definition 2. Let K be a field, and let $f_1, \ldots, f_m$ be polynomials in $K[x]$. Let the set V be $V(f_1, \ldots, f_m) = \{a = (a_1, \ldots, a_n) \in \mathbb{K}^n \mid \ f_1(a) = \ldots = f_m(a) = 0\}$. We call $V(f_1, \ldots, f_m)$ the affine variety defined by $f_1, \ldots, f_m$. The set of polynomials that vanish in a given variety $I(V)$ is an ideal called the ideal of V. Finite unions and intersections of algebraic varieties are again algebraic varieties.

Definition 3. Let $I \subset K[x]$ be an ideal, and let $f, g \in K[x]$. We say f and g are congruent modulo I, written $f \equiv g \ mod \ I$, if $f - g \in I$.

It is easy to show that this is an equivalence relation (reflexive, symmetric, and transitive). Thus, this partitions $K[x]$ into equivalence classes, where

two polynomials are "the same" if their difference belongs to the ideal. This allows us to define the quotient ring.

Definition 4. The ring $K[x]/I$ is the set of equivalence classes for congruence modulo I.

Quotient rings are useful when considering a polynomial function f over the algebraic variety defined by $f_1(a) = \ldots = f_m(a) = 0$. Notice that if we define the ideal $I = (f_1, \ldots, f_m)$, then any polynomial g that is congruent with f modulo I takes exactly the same values in the variety.

Definition 5. A monomial ordering on $K[x]$ is a relation $\succ$ such that:

1. The relation $\succ$ is a total ordering.
2. $u = x^a \succ v = x^b$, $w = x^c$, then $x^{a+c} \succ x^{b+c}$ (it is multiplicative, $uw \succ vw$).
3. The relation $\succ$ is a well-ordering (every nonempty subset has a smallest element).

We can write $u \succ v$ or $v \prec u$. There are several term orderings of interest in computational algebra. Among them, we mention:

- Lexicographic: $u \succ_{lp} v$ if the left-most nonzero entry of $a - b$ is positive.
- Graded lexicographic: $u \succ_{Dp} v$ if $|a| > |b|$, or if $|a| = |b|$ and $u \succ_{lp} v$ (sort first by total degree, then lexicographic).
- Graded reverse lexicographic: $u \succ_{dp} v$ if $|a| > |b|$, or if $|a| = |b|$ and the right-most nonzero entry of $a - b$ is negative.

Example 1. In one variable, there is only one term order: $1 \prec x \prec x^2 \prec x^3 \prec \cdots$.

For $n = 2$, we have:

- degree lexicographic order

  $1 \prec x_1 \prec x_2 \prec x_1^2 \prec x_1x_2 \prec x_2^2 \prec x_1^3 \prec x_1^2x_2 \prec \cdots$

- purely lexicographic order

  $1 \prec x_1 \prec x_1^2 \prec x_1^3 \prec \cdots \prec x_2 \prec x_1x_2 \prec x_1^2x_2 \prec \cdots$

If we consider the monomials $u = x^4y^3z^9$ and $b = x^3y^{10}z^3$, then $|a| = 4 + 3 + 9 = 3 + 10 + 3 = |b|$. If the variables are ordered as (x, y, z), we have $u \succ_{lp} v$, $u \succ_{Dp} v$, $v \succ_{dp} u$. Notice that $x \succ y \succ z$ for all three orderings.

Definition 6. Every polynomial $f \in K[x]$ has an initial monomial, denoted by $in_\succ(f)$. For every ideal I of $K[x]$ the initial ideal of I is generated by all initial monomials of polynomials in I, $in_\succ(I) = (in_\succ(f) : f \text{ is in } I)$.

A monomial w is called standard, if it does not belong to the initial ideal $in_\succ(I)$. The set of standard monomials is a $K-$basis for the residue ring

$K[x]/I$, i.e., modulo the ideal I, every polynomial f can be written uniquely as a $K-$linear combination of standard monomials. Given f, there is an algorithm (the division algorithm) that produces this representation (called the normal form of f) in $K[x]$.

Example 2. If $n = 3$ and $in_\succ(I) =< x_1^2, x_2^3, x_3^4 >$, the number of standard monomials is 24. If $in_\succ(I) =< x_1^2, x_2^3, x_1x_3^4 >$, then the number of standard monomials is infinite.

Definition 7. A finite subset G of an ideal I is a Gröbner basis (with respect to the term order $\succ$) if $\{in_\succ(g) : g \text{ is in } G\}$ generates $in_\succ(I)$.

There are many such generating sets, for instance, we can add any element of I to G to get another Gröbner basis. There are several algorithms to effectively compute Gröbner bases, but the first is Buchberger's algorithm, developed by Bruno Buchberger in 1965.

Here, we are interested in polynomial systems that have only a finite number of solutions (the "zero-dimensional" case).

Definition 8. A reduced (standard) Gröbner basis satisfies:

1. For each g in G, the coeff of $in_\succ(g)$ is 1
2. The set $\{in_\succ(g) : g \text{ is in } G\}$ minimally generates $in_\succ(I)$ (nothing can be removed)
3. No trailing term of any g in G lies in the initial ideal $in_\succ(I)$.

Theorem 2. Fixing an ideal I in $K[x]$ and a term order $\succ$, there is a unique reduced Gröbner basis for I.

Definition 9. An ideal $I = (F)$ is zero-dimensional if the associated variety $V(I)$ is a finite set. If G is a Gröbner basis of I, $V(F)$ is empty if and only if $G = \{1\}$.

Theorem 3. (Fundamental Theorem of Algebra) The number of standard monomials equals #V(I), where the zeroes are counted with multiplicity.

When I is a zero-dimensional ideal the quotient ring $K[x]/I$ is a finite dimensional vector space, with its dimension being equal to the number of standard monomials (monomials that are not in the initial ideal $in(I)$).

Example 3. For $I = (xy - x, xy + y) \subset K[x, y]$, lexicographic ordering with $x \succ y$, a Gröbner basis is $G = \{x + y, y^2 - y\}$ and $in(I) = (x, y^2)$. We can directly see from this that I is zero-dimensional:

- there are two standard monomials, 1 and y
- a basis for the quotient space is finite, given by $\{1, y\}$
- we have two solutions (0,0) and (-1,1) for the system $xy - x = 0$, $xy + y = 0$.

The ideal I must be zero dimensional and for K we can take any field, infinite, where the characteristic is zero (like $\mathbb{Q}$ or $\mathbb{R}$) or finite, where the characteristic is finite (like $\mathbb{Z}_5$).

3. **Solving equations over finite fields**

The finite field with p^n elements, where p is a prime number, is denoted $GF(p^n)$ and is also called the Galois field. $GF(p)$ also denoted $\mathbb{F}_p$, is simply the ring of integers modulo p. That is, one can perform operations (addition, subtraction, multiplication) using the usual operation on integers, followed by reduction modulo p. First we shall deal with the case where $\mathbb{K} = \mathbb{F}_p$. Let $\mathbb{K}^*$ the multiplicative group of all non - zero elements in $\mathbb{K}$, $a, b \in \mathbb{K}^*$ and any prime $m < n$. It is easy to show that the number of solutions for equation: $x_1 \cdot x_2 \cdot x_3 = 1$ over $\mathbb{K}$ is $(p-1)^2$. One may ask what happens for the equation $x_1 \cdot x_2 \cdot x_3 \cdot (x_1 + x_2 + x_3) = 1$.

The equations to be considered here are those of the type:

$$x_1 \cdot \dots \cdot x_n \cdot (x_1 + \dots + x_n) = a \tag{1}$$

$$x_1 \cdot \dots \cdot x_n \cdot (x_1 + \dots + x_n) \cdot y_1 \cdot \dots \cdot y_m \cdot (y_1 + \dots + y_m) = b \tag{2}$$

To optimize the second type, we decompose the equation into more accessible system with two polynomial equations of the first type:

$$\begin{cases} x_1 \cdot \dots \cdot x_n \cdot (x_1 + \dots + x_n) = a \\ y_1 \cdot \dots \cdot y_m \cdot (y_1 + \dots + y_m) = a^{-1}b \end{cases} \tag{3}$$

For any prime p we denote by $N(p)$ the number of solutions of the equation (**2**) in $\mathbb{K}^{m+n}$. For $a, b \in \mathbb{K}^*$, consider the varieties

$$V_1(a) = \{(x_1, \dots, x_n) \in \mathbb{K}^n \mid \ x_1 \cdot \dots \cdot x_n \cdot (x_1 + \dots + x_n) = a\}$$

and

$$V_2(a) = \{(y_1, \dots, y_m) \in \mathbb{K}^m \mid \ y_1 \cdot \dots \cdot y_m \cdot (y_1 + \dots + y_m) = a^{-1}b\}.$$

Let $N_1(a) = \#V_1(a)$, $N_2(a) = \#V_2(a)$ and note that obviously one has

$$N(p) = \sum_{a \in \mathbb{K}^*} N_1(a) N_2(a). \tag{4}$$

Note that equation (1) is symmetric and we can find only solutions with $x_1 \leq \dots \leq x_n$ and multiply every case according to symmetry.

Example 4. Let the equation $x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5 \cdot (x_1 + x_2 + x_3 + x_4 + x_5) = 1$ over $\mathbb{K} = \mathbb{F}_{11}$. For some solutions we show this multiplier λ (arrangements with repetitions).

$x_1 = x_2 = x_3 = x_4 = x_5 = 2$, $\lambda = A(5) = \frac{(5)!}{5!} = 1$.

$x_1 = x_2 = x_3 = x_4 = 1 < x_5 = 2$, $\lambda = A(4,1) = \frac{(4+1)!}{4!1!} = 5$.

$x_1 = x_2 = x_3 = 1 < x_4 = x_5 = 6$, $\lambda = A(3,2) = \frac{(3+2)!}{3!2!} = 10$.

$x_1 = x_2 = x_3 = 1 < x_4 = 2 < x_5 = 5$, $\lambda = A(3,1,1) = \frac{(3+1+1)!}{3!1!1!} = 20$.

$x_1 = x_2 = 1 < x_3 = x_4 = 2 < x_5 = 7$, $\lambda = A(2,2,1) = \frac{(2+1+1)!}{2!1!1!} = 30$.

$x_1 = x_2 = 1 < x_3 = 2 < x_4 = 4 < x_5 = 6$, $\lambda = A(2,1,1,1) = \frac{(2+1+1+1)!}{2!1!1!1!} = 60$.

$x_1 = 1 < x_2 = 2 < x_3 = 3 < x_4 = 4 < x_5 = 9$, $\lambda = A(1,1,1,1,1) = \frac{(1+1+1+1+1)!}{1!1!1!1!1!} = 120$.

Remark 1. Equation (2) is based on equation (1). To solve equation (1), we search only solutions with ascendent components $x_1 \leq x_2 \leq \ldots \leq x_n$ and multiply each case with this number, "arrangements with repetitions", $A(k_1, \ldots, k_m) = \frac{(k_1+\ldots+k_m)!}{k_1!\ldots k_m!}$.

We give three different methods to solve this type of equations.

1. The algebraic method, based on a symbolic approach via Gröbner bases
2. Backtracking implementation using a Computer Algebra Software
3. Backtracking implementation in C++ with additional package like NTL or GMP.

More precisely, we give two examples how to use Singular and another how to use C++ to count zeros of polynomials over finite fields.

SINGULAR is a computer algebra system designed for polynomial computations, with special emphasis on the needs of commutative algebra, algebraic geometry, and singularity theory. It is a free software, available for various platforms. To obtain SINGULAR, download it from its homepage [**?**]

The components of SINGULAR include a precompiled C/C++ program, referred to as the SINGULAR kernel, several libraries, and the on-line help system. The core algorithms for polynomial computations are implemented in the kernel. Each library is a text file consisting of a collection of procedures written in the SINGULAR user language. This language is interpreted, not compiled. It resembles C++ and the user may enlarge the set of commands available by adding his own procedures.

3.1. *Algebraic method based on a symbolic approach via Gröbner bases*

This method is based on the study of the quotient algebra A of the polynomial ring $K[X]$ modulo the ideal $I(f)$. More precisely, let I be the ideal generated by $f = x_1 \cdot \cdots \cdot x_n \cdot (x_1 + \ldots + x_n) - a$ and $V(I)$ be the variety over $\mathbb{K}$ associated to I. The problem is then to find $V(I)$.

When $\mathbb{K}$ is a finite field of order p, one can always add to the existing set of equations the so-called field equations (Fermat) $f_i = x_i^p - x_i = 0$, $1 \leq i \leq p$ and obtain $n+1$ equations. Choosing a term ordering $\prec$ in $\mathbb{K}[x]$ (e.g., lexicographic, where $x_1 \prec \cdots \prec x_n$) we obtain the Gröbner basis $\mathbb{G}$ for the new ideal $\mathbb{J} = (f, f_1, \ldots, f_n)$. We can directly see that $\mathbb{G}$ is zero-dimensional and we can compute a standard basis for the quotient space, the quotient algebra $\mathbb{A} = \mathbb{K}[x]/\mathbb{G}$. Cardinality of this basis give us the number of solutions for the equation $f = 0$, over $\mathbb{K}$.

The Singular libraries provide several commands for solving systems of polynomial equations (based on a symbolic approach via Gröbner bases). In the example below, we show some of these commands at work. The monomial ordering used was Grevlex (Graded reverse lexicographical order), usually most efficient in computations.

Example 5. Let $f = xyz(x + y + z) - 1 = 0$ over finite field $\mathbb{K} = \mathbb{F}p$.

The following program, written in Singular user language, computes the number of solutions for the equation (1)

proc eq1(int n, int p, int a)
"USAGE: eq1(n,p,a); n, p, a integers
RETURN: number of solutions for f=x1...xn(x1+...+xn)-a=0
NOTE: n is the number of variables, p is any prime
"
{
ring R=p,(x(1..n)),dp; //finite field K=Z/p, ring R=K[x1,...,xn], global ordering degrevlex
int i;
poly f1=1;
poly f2=0;
for (i=1;i<=n;i++){
f1=f1*x(i);
f2=f2+x(i);
}
poly f=f1*f2-a;
ideal J=f;
for (i=1;i<=n;i++){
J=J,x(i)^p-x(i);
}
ideal G=std(J); //compute a standard basis G of J in R
int ndim=dim(G); //the (global) dimension of V(J) must be 0
int ns=vdim(G); //vector space dimension of the ring, modulo the ideal G
return (ns); //number of solutions
}

Example of usage:

Consider the ideal $\mathbb{J} \subset \mathbb{K}[x, y, z]$ given by $J = (f, x^p - x,\ y^p - x,\ z^p - x)$.where $f = xyz(x + y + z) - 1$ and $\mathbb{K} = \mathbb{F}_{53}$

eq1(3,53,1); //- 2616

Example 6. The following program, written in Singular user language, solves equation (2).

proc eq2(int n, int m, int p, int b)
"USAGE: eq2(n,m,p,b); n, m, p, b integers
RETURN: number of solutions for f=x1...xn(x1+...+xn)y1...ym(y1+...+ym)-b=0
NOTE: n the number of variables for the first set
m the number of variables for the second set
p is any prim

```
"
LIB "general.lib";
int c;
ring R=0,x,dp;
number N, n1, n2;
N=0;
rtimer=0; // The real time of each command is not printed
int t=rtimer;// initialize t by rtimer
for (int a=1;a<=p-1;a++)
{
n1=eq1(n, p,a);
c=invers_Zp(p,a)*b;
n2=eq1(m, p, c);
N=N+n1*n2;
}
write("","Number of solutions: "+string(N));
int hh,mm,ss;
t=rtimer-t;
hh= t div 3600;
t=t mod 3600;
mm=t div 60;
ss=t mod 60;
write("","time: "+"hh:"+string(hh)+" mm:"+string(mm)+" ss:"+string(ss));
}
proc invers_Zp(int p, int a)
{
int aux;
for (int i=1;i<=p-1;i++){
aux=i*a % p;
if (aux==1) {return (i);}
}
}
```

Example of usage: Consider the ideal $\mathbb{J} \subset \mathbb{K}[x_1, x_2, y_1, y_2, y_3]$ given by $J = (f,\ x_1\hat{}p - x_1,\ x_2\hat{}p - x_2,\ y_1\hat{}p - y_1,\ y_2\hat{}p - y_2,\ y_3\hat{}p - y_3)$ where $f = x_1x_2(x_1 + x_2)y_1y_2y_3(y_1 + y_2 + y_3) - 1$ and $\mathbb{K} = \mathbb{F}_{29}$

eq2(2,3,29,1);

Number of solutions: 572292

time: hh:0 mm:0 ss:19

The equation can be solved within one minute. When p, n or m becomes much bigger, the running time will increase polynomial.

eq2(2,3,61,1); // $\mathbb{K} = \mathbb{F}_{61}$

Number of solutions: 12535140

time: hh:0 mm:44 ss:8

Example 7. To count with Singular the number of solutions over Galois fields, is necessary to make few modifications in the program.

With the following procedure, we can solve equation (1) over $GF(p^q)$, for $f = xyz(x + y + z) - 1$ for different values p and q.

Due to a different internal representation, the arithmetic operations are faster if we implement the ring $GF(p^q)[x, y]$ as follows:

"ring R = (pˆq,g), (x,y), dp", where g is a generator.

```
proc gf(int p, int q)
{
ring gf=(pˆq,g),(x,y,z),dp; // //
int pq=pˆq;
poly f=x*y*z*(x+y+z)-1;
ideal J=f,xˆpq-x,yˆpq-y,zˆpq-z,
ideal G=std(J);
vdim(G); // number of solutions
}
```

Example of usage:

gf(5,2); // - 564

We obtain the same result with commands:

eq1(3, 29,1); //- 744

gf(29,1); //- 744

Remark 2. To optimize, we can test two alternatives: keep the number of variables as low as possible or keep the degree as low as possible. If we add a new variable u=xyz, our function will be: $f = u(x + y + z) - 1$, but in the ring $\mathbb{K}[x, y, z, u]$, with additional equation u-xyz=0. Based on our simulation, the time increase.

Remark 3. When the characteristic of base ring is zero ($\mathbb{K}$ infinite field, like $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$) Singular has a "solver", to compute all the solutions. We solve the system $x + y + z = 6$, $xy + xz + yz = 11$, $xyz = 6$ over $\mathbb{Q}$ with "solver.lib" from Singular:

```
LIB "solve.lib";
ring R=0,(x,y,z),dp;
ideal I=x+y+z-6, xy+xz+yz-11,xyz-6;
ideal G=std(I);
dim(G); // zero-dimensional
vdim(G); // the number of solutions (counted with multiplicities)
def RR=solve(G,10,0,"nodisplay"); // solutions should not be displayed
setring RR; // "solve" created a ring, in which a list SOL of solutions is
stored.
size(SOL); // number of different solutions
SOL[1]; // first solution (1,2,3)
```

3.2. *Bactracking implementation in Singular*

From the results, we can see that the previous method, based on Gröbner bases, need more time when the number of variables or the characteristic of the field increase. Therefore, computing the Gröbner basis will be a hard work, given information not necessary, like a basis for the quotient space.

In order to solve the equations, we have implemented backtracking with Singular user language.

Unlike backtracking implementation in C++, here we can use arithmetic of finite fields with ring declaration.

3.3. *C++ with NTL or GMP*

Experimental results show that for large parameters, the interpreter is slowly. Another point of view is to use C++. For large numbers we should look to a big number library such as NTL or GMP.

NTL (Number Theory Library)

NTL is a portable C++ library providing tools for arbitrary integer and floating point arithmetic and Polynomial Arithmetic (has one of the fastest polynomial arithmetic capability). Freely available, is portable, and thus can be used in virtually any platform with minimal alterations.

A complete reference to the NTL library and how to install it in several different platforms; as well as some examples on how to use the various library components can be found at [5].

ZZ library is provided to perform large integer arithmetic. Here is a quick example on how to use the ZZ interface:

```
#include "NTL/ZZ.h"
ZZ a, b;  // two big integers a and b
```

GMP (the Gnu Multiple Precision arithmetic library)

GMP is a free library for arbitrary precision arithmetic, operating on signed integers, rational numbers, and floating point numbers. There is no practical limit to the precision except the ones implied by the available memory in the machine GMP runs on. GMP has a rich set of functions, and the functions have a regular interface.

The main target applications for GMP are cryptography applications and research, Internet security applications, algebra systems, computational algebra research, etc.

Below is a sample in C++ with GMP (functions for performing integer arithmetic start with the prefix mpz_, with mpq_ for rational numbers, etc.)

```
#include "gmp.h"
mpz_t a,b; // big integers a and b
```

A complete reference to the GMP library can be found at [6].

4. **Computational results obtained from simulations**

This section presents computational results from experimentation with our programs. We can prove some theoretical aspects extracted from the dates or make conjectures. Based on this databases, we derive additional knowledge and try to answer to many questions, like these:

- What can we say about the solutions of this type of equations?

- Compare the number of solutions for different equations. Are these equivalent?
- Count or estimate the number of points of varieties over finite fields.
- We can say that $V(f)$ has polynomial count with count polynomial P? This means that we can find a polynomials P, to give an explicit formula for the number of solutions of the equation $f = 0$, in $\mathbb{K} = \mathbb{F}_p$ for many (infinite) primes. We have some conjectures which would imply these curious polynomials P.

Problem 1. For $f = x_1 \cdot x_2 \cdot (x_1 + x_2) - a = 0$ over finite fields $\mathbb{K} = \mathbb{F}_p$ where p is any prime of the form $p = 6k + 5$. Then, these equations are equivalent, for every $a = 1, \ldots, p - 1$.

To prove we need a lemma.

Lemma 1. Let G be a group of order m, d be relatively prime to m and $a \in G$. Then the equations $x^d = a$ has the unique solution $x = a^u$ where $du = 1 \bmod m$.

Proof: Because m and d are relatively prime, then exist integers u,v with $du + mv = 1$, so $du = 1 \bmod m$. Because $a^m = 1$ for any element a of the group, we have $a = a^{du+mv} = a^{du}a^{mv} = a^{du}$. For the uniqueness, we note that the previous sentence shows that the map $x \longmapsto x^d$ is onto. Since G is finite, it must also be $1 - 1$.

Corollary 1. Let $\mathbb{K} = \mathbb{F}_p$ be finite field with $p = 6k + 5$ prime. The equation $x^3 = a$ has the unique solution in $\mathbb{K}$.

Proof: Because $\mathbb{K}$ is field, $\mathbb{K}^* = \mathbb{K} - \{0\}$ is group of order $m = p - 1 = 6k + 4 = 2(3k + 2)$ relatively prime with 3.

Theorem 3. Let $f = x_1 \cdot x_2 \cdot (x_1 + x_2) - a = 0$ over $\mathbb{K} = \mathbb{F}_p$ where p is any prime of the form $p = 6k + 5$. These equations have the same number of solutions, for $a = 1, \ldots, p - 1$.

Proof: Let consider the equation $x_1 \cdot x_2 \cdot (x_1 + x_2) = a$ and x be the unique solution for $x^3 = a$. We have $xx_1 \cdot xx_2 \cdot (xx_1 + xx_2) = a$ and with the substitution $y_k = xx_k$ we obtain $y_1 \cdot y_2 \cdot (y_1 + y_2) = 1$.

Theorem 4. The number of solutions for $x_1 \cdot x_2 \cdot (x_1 + x_2) = 1$ is odd.

Proof: If $x_1 = x_2$ we obtain $2x^3 = 1$ with one solution. Because the equation is symmetric, if (x,y) with $y <> x$ is solution, then (y,x) is solution.

Remark 4. Let p=6k+5 prime, then the number of zeros of the equation $x_1 \cdot x_2 \cdot (x_1 + x_2) = a$, a arbitrary in $\mathbb{K}^*$, seems to be $p - 2$, so the variety $V(f)$, $f = x_1 \cdot x_2 \cdot (x_1 + x_2) - a$ has polynomial count with count polynomial $P = t - 2$.

We try to find an explicit formula for the number of solutions of the equation (2), for $n = 3$, $m = 5$, $b = 1$:

$$x_1 \cdot x_2 \cdot x_3 \cdot (x_1 + x_2 + x_3) \cdot x_4 \cdot x_5 \cdot x_6 \cdot x_7 \cdot x_8 \cdot (x_4 + x_5 + x_6 + x_7 + x_8) - 1 = 0.$$

Special thanks are due to Alexandru Dimca who proposed the previous equation (for some properties and proof, see [1] **Lemma 5.1**).

Conjecture 1. For above equation and polynomial $P(t) = t^7 - 9t^6 + 36t^5 - 82t^4 + 119t^3 - 110t^2 + 60t - 15$, it seems that $N(p) = P(p)$ for all primes of the form $p = 4k + 1$. Indeed, the equality is true for the first 20 primes numbers of this type.

p	**N(p)**	**p**	**N(p)**
5	11160	97	73603528860864
13	30575400	101	98029806010200
17	237920544	109	168252493546728
29	12579682248	113	217173059359200
37	74188920024	137	848026439513424
41	155950465680	149	1534557528742968
53	989657318520	157	2219843271599304
61	2708356179720	173	4402141408351080
73	9757738115280	181	6054740677904760
89	39954467578608	193	9519114805088640

Remark 5. For some equation we founded estimators as interpolation polynomials for equations of type (1).

The polinomials $P(t) = t^2 - 3t$, and $Q(t) = t^4 - 5t^3 + 10t^2 - 10t$ are a good estimators for equation (1) for n=3 and $n = 5$, respectively.

References

1. Dimca, A., *Tate Properties, Polynomial-Count Varieties, and Monodromy of Hyperplane Arrangements*, arXiv:1012.1437v2, 13-16, 2011.
2. Decker, W., Lossen, C., *Computing in Algebraic Geometry: A Quick Start Using Singular*, Springer, 169-199, 2006.
3. Greuel, G., Pfister, G. A., *Singular Introduction to Commutative Algebra*, Springer, Berlin/Heidelberg, 571-606, 2008.
4. Singular manual: *A Computer Algebra System for Polynomial Computations*, Centre for Computer Algebra, University of Kaiserslautern, http://www.singular.uni-kl.de.
5. NTL library: http://www.shoup.net/ntl/
6. GMP library: http://gmplib.org/