Zhuang-Zi: A New Algorithm for Solving Multivariate Polynomial Equations over a Finite Field

Jintai Ding[1], Jason E. Gower[1] and Dieter S. Schmidt[2]

Department of Mathematical Sciences[1]
Department of Electrical & Computer Engineering and Computer Science[2]
University of Cincinnati
Cincinnati, OH 45220
USA
ding@math.uc.edu, gowerj@math.uc.edu, dieter.schmidt@uc.edu

Abstract. We present the Zhuang-Zi algorithm, a new method for solving multivariate polynomial equations over a finite field. We describe the algorithm and present examples, some of which cannot be solved with the fastest known algorithms.

Keywords: multivariate polynomials, Hidden Field Equation, Berlekamp algorithm

1 Introduction

Solving a single variable polynomial equation or a set of multivariate polynomial equations has always been at the center of the development of mathematics. It is not only inspired by our curiosity, but also by the ubiquitous role these simple but fundamental problems play in all branches of science.

Though the Babylonians did not invent the notion of an "equation," they found the first algebraic solution to problems, which gave rise to what we call a single variable quadratic equation today. The first known solution of this problem is given in the Berlin papyrus from the Middle Kingdom of Egypt (circa 2160–1700 BC) [Smi52].

Though the first great success is surely solving the single variable quadratic equation, the next successes came much later with Ferro solving the single variable cubic equation and Ferrari solving the single variable quartic in the 16th century. However Galois' theory put an end to the hope of finding an elegant algebraic formula for higher order single variable equations.

The situation is very different in the multivariate case. The real great success is the Gröbner basis method [Buc65], which comes from the ideas of modern algebraic geometry. Solving polynomial equations over the integers is also a very interesting direction, for example Fermat's last theorem, but it is a completely different story which we will omit here.

A very different and modern direction is to solve multivariate equations over a finite field. Recently much effort in this area has been inspired by the appearance of multivariate public key cryptography. For the single variable case, there are very efficient algorithms, such as the Berlekamp algorithm, for factoring polynomials of relatively low degree. For the multivariate case, one can also use an extension of the Gröbner basis.

The idea of a public key cryptosystem was first suggested by Diffie and Hellman, but the first practical construction was RSA. Public key cryptography allows any two parties to communicate securely over an open communication channel, like the Internet, and it now plays a fundamental role in our communication systems. However recent developments in quantum computing, in particular Peter Shor's polynomial-time integer factorization algorithm, shows that a quantum computer can be used to break RSA. Thus there has been great interest in constructing other public key cryptosystems.

One alternative is to use multivariate polynomials, and in particular quadratic polynomials. The security of such constructions is suggested by the proven theorem that solving a set of multivariate polynomial equations over a finite field is, in general, an NP-hard problem [GJ79]. Nevertheless, this result is not enough to guarantee the security of such a cryptosystem.

Recent research in multivariate public key cryptography has stimulated a search for new methods for solving multivariate polynomial equations over a finite field, along the line of Gröbner bases. Examples include XL [CKPS00] and the enhanced Gröbner bases methods F_4 and F_5 of Faugère [Fau99,Fau02]. Inspired by the work in this area, we propose a new algorithm to solve a set of multivariate polynomial equations over a finite field.

2 Background

Let k be a finite field with q elements and suppose we have m polynomials $f_0, f_1, \ldots, f_{m-1} \in k[x_0, x_1, \ldots, x_{n-1}]$. We wish to find all $(a_0, a_1, \ldots, a_{n-1}) \in k^n$, such that

$$\begin{aligned} f_0(a_0, a_1, \ldots, a_{n-1}) &= 0 \\ f_1(a_0, a_1, \ldots, a_{n-1}) &= 0 \\ &\vdots \\ f_{m-1}(a_0, a_1, \ldots, a_{n-1}) &= 0 \end{aligned} \tag{1}$$

We may as well work in the ring

$$k[x_0, x_1, \ldots, x_{n-1}]/(x_0^q - x_0, x_1^q - x_1, \ldots, x_{n-1}^q - x_{n-1}),$$

though for convenience we will abuse notation and write $k[x_0, x_1, \ldots, x_{n-1}]$. The key idea of our new algorithm is to shift perspectives from the space of polynomials $k[x_0, x_1, \ldots, x_{n-1}]$ with coefficients in the small field k, to a space of polynomials $K[X]$ with coefficients in some suitably chosen extension field K.

To simplify matters, let us assume that $m = n$. Choose any irreducible polynomial $g(y) \in k[y]$ of degree n. Then $K = k[y]/(g(y))$ is a degree n field extension of k. Let ϕ be the standard k-linear map that identifies K with the n-dimensional vector space k^n, i.e., $\phi : k^n \longrightarrow K$, defined by

$$\phi(a_0, a_1, \ldots, a_{n-1}) = a_0 + a_1 y + \cdots + a_{n-1} y^{n-1} \tag{2}$$

Let $f : k^n \longrightarrow k^n$ be the polynomial map defined by $f = (f_0, f_1, \ldots, f_{n-1})$. We can lift f up to the extension field K using ϕ to create a map $F : K \longrightarrow K$ defined by

$$F = \phi \circ f \circ \phi^{-1}.$$

Using the Lagrangian interpolation formula, we can think of F as a polynomial in $K[X]$, where X is an intermediate. In fact, F has a unique representation in the quotient space $K[X]/(X^{q^n} - X)$. For any given f, the corresponding F can be calculated by solving a set of linear equations. The following theorem tells us the exact form of this representation.

Theorem 1. *Using the notation as defined above, for a linear polynomial map* $f = (f_0, f_1, \ldots, f_{n-1})$ *we have*

$$F(X) = \sum_{i=0}^{n-1} \beta_i X^{q^i} + \alpha \mod (X^{q^n} - X),$$

for some $\beta_i, \alpha \in K$. *If* f *is a quadratic polynomial map, then*

$$F(X) = \sum_{i=0}^{n-1} \sum_{j=i}^{n-1} \gamma_{ij} X^{q^i + q^j} + \sum_{i=0}^{n-1} \beta_i X^{q^i} + \alpha \mod (X^{q^n} - X),$$

for some $\gamma_{ij}, \beta_i, \alpha \in K$. *Representations for higher order polynomial maps are similarly described. In the case of* $q = 2$, *the formulas are slightly different.*

In this paper, we will identify the map F with its corresponding representation given in Theorem 1.

It is now clear that we can move freely between multivariate functions and single variable functions, and we will do so in order to solve the original system of equations. This is the basic idea of Matsumoto-Imai, Patarin, Kipnis and Shamir [MI88,Pat00,KS99], and is also the basis of our algorithm. Given a system of equations such as (1), the basic strategy will be to lift the associated polynomial map f to the map F in the extension field K. The roots of the representation of F given in Theorem 1 correspond exactly with the solutions to the original system of equations defined over k. Once we have the roots in K, we can descend down to k^n with ϕ^{-1}. It remains to develop techniques for reducing the degree of F, which if successful, will allow us to use efficient algorithms for solving single variable polynomial equations.

We note a fundamental difference between our algorithm and others which is that ours can be used only with finite fields and cannot be used with fields of characteristic zero. The reason for this is that the lifting from the multivariate system to a single variable equation works very well for a finite field, but not for characteristic zero cases. However, in the case of finite fields, our algorithm unifies the two problems of solving single variable and multivariate polynomial equations into a single problem. We have named this algorithm after Zhuang-Zi, an ancient Chinese philosopher who we believe was one of the first to propose the idea of shifting from a local view of problems to a global view.

The remainder of this paper is organized as follows. We will explain the basic algorithm, including a method to reduce the degree of F, and present a toy example in order to show how the algorithm works. We then present more meaningful examples and conclude with a discussion of future work.

3 The Zhuang-Zi Algorithm

We will start with the standard case of $m = n$, where we have the same number of variables and equations. The Zhuang-Zi algorithm takes the polynomials $f_0, f_1, \dots, f_{n-1} \in k[x_0, x_1, \dots, x_{n-1}]$ and a positive integer D as its input, where D is the upper bound on the degree of a polynomial equation which can be solved efficiently. When successful the algorithm returns all n-tuples $(a_0, a_1, \dots, a_{n-1}) \in k^n$ such that $f_i(a_0, a_1, \dots, a_{n-1}) = 0$, for $i = 0, 1, \dots, n-1$.

Step 1: Choose any degree n irreducible polynomial $g(y) \in k[x]$ and define $K = k[y]/(g(y))$. Let $\phi : k^n \longrightarrow K$ be as defined in (2). Define $f = (f_0, f_1, \dots, f_{n-1})$, lift this to K by $F = \phi \circ f \circ \phi^{-1}$, and compute the polynomial representation of $F(X)$ modulo $X^{q^n} - X$. If the $\deg(F(X)) \leq D$, then go to the last step; otherwise continue to the next step.

Step 2: Let $G = \mathrm{Gal}(K/k)$ be the Galois group of K over k consisting of the Frobenius maps $G_i(X) = X^{q^i}$, for $i = 0, 1, \dots, n-1$. Calculate

$$F_i(X) = G_i \circ F(X) = F(X)^{q^i} \mod (X^{q^n} - X),$$

for $i = 0, 1, \dots, n-1$. Note that $F_0(X) = F(X)$. If there exists an F_i such that $\deg(F_i(X)) \leq D$, then go to the last step; otherwise continue to the next step.

Step 3: Let N be the number of monomials that appear in any $F_i(X)$. For each $F_i(X)$ create a row vector in K^N, where the entries are the coefficients of $F_i(X)$ listed in decreasing order, and construct an $n \times N$ matrix using these row vectors. Then use Gaussian elimination to produce a new set of t basis polynomials $S = \{S_0(X), S_1(X), \dots, S_{t-1}(X)\}$. In other words eliminate the monomials in the order of the highest degree first. Label the elements of S so that $S_{t-1}(X)$ is the element of lowest degree. If $\deg(S_{t-1}(X)) \leq D$, then go to the last step; otherwise continue to the next step.

Step 4: It must be that the polynomial of minimal degree in S has degree greater than D. For each $i = 0, 1, \dots, t-1$ and $j = 0, 1, \dots, n-1$ compute

$$X^{q^j} S_i(X) \mod (X^{q^n} - X).$$

As before, apply Gaussian elimination to the matrix associated with this set of polynomials to produce a set S' of new basis polynomials. Let $S'_{t'-1}(X)$ be the polynomial in S' of minimal degree. If $\deg(S'_{t'-1}(X)) \leq D$, then go to the last step; otherwise replace S with S' and repeat this step.

Step 5: At this point we have a polynomial $G(X)$ with $\deg(G(X)) \leq D$. Solve $G(X) = 0$ with an appropriately chosen fast polynomial equation solver or factorization algorithm, such as the Berlekamp algorithm, to obtain a set $W = \{\alpha \in K \mid G(\alpha) = 0\}$. The solutions of $F(X) = 0$ will be the subset $\{\alpha \in W \mid F(\alpha) = 0\}$.

Remark 1. Since the complexity of any factorization algorithm or polynomial equation solver depends on the degree of the given polynomial and the size of the field, so too does the complexity of the Zhuang-Zi algorithm. Improvements in the area of factorization algorithms or equation solvers will translate directly into an improvement for the Zhuang-Zi algorithm, and may require modifications to the degree-reduction step if a special form other than minimal degree is desired. In fact, several such algorithms can be run in parallel to speed up the overall performance of the Zhuang-Zi algorithm.

Remark 2. If f consists of quadratic polynomials then the only powers of X that may arise in Steps 1–3 are either of the form X^{q^i} or $X^{q^i+q^j}$. Each application of Step 4 then produces monomials of the form $X^{a_i q^i}$, $X^{a_i q^i + a_j q^j}$, $X^{a_i q^i + a_j q^j + a_k q^k}$, etc. Clearly then all possible monomials will have been generated after nq steps, since there are only nq possible monomials in $K[X]/(X^{q^n} - X)$.

Remark 3. If no k-linear combination of the polynomials $f_0, f_1, \ldots, f_{n-1}$ is zero in $k[x_0, x_1, \ldots, x_{n-1}]$ (modulo $x_0^q - x_0, x_1^q - x_1, \ldots, x_{n-1}^q - x_{n-1}$), then no K-linear combination of the polynomials $F_0, F_1, \ldots, F_{n-1}$ is zero in $K[X]$ (modulo $X^{q^n} - X$), since such a linear combination of the F_i will have degree at most $q^n - 1$.

Remark 4. The Zhuang-Zi algorithm can be modified for use in the case when there are fewer equations than variables ($m < n$). Let $\pi : k^m \longrightarrow k^n$ be the injection defined by

$$\pi(a_0, a_1, \ldots, a_{m-1}) = (a_0, a_1, \ldots, a_{m-1}, 0, \ldots, 0).$$

We replace $f = (f_0, f_1, \ldots, f_{m-1})$ with $\pi \circ f$, and proceed as before. It should be noted that if m is too small then we know that there will be a large number of solutions (roughly q^{n-m-1}), and therefore the polynomials in the ideal generated by the $F_i(X)$ will have high degree. If the Zhuang-Zi algorithm is unable to generate a polynomial of sufficiently small degree, then we may never reach the factorization step.

Remark 5. If there are more equations than variables ($m > n$) then again the Zhuang-Zi algorithm can be modified for use. Suppose $m = nr + l$, for some $0 \leq l < n$, and divide $f_0, f_1, \ldots, f_{m-1}$ into r sets of n polynomials and one set of

l polynomials. If m is a multiple of n, then lift each of the r set of polynomials to a polynomial in $K[X]$ as before. If $l \neq 0$, then we lift the last set of l polynomials as in the previous remark. In either case we can apply the algorithm starting with this set of r (or $r+1$) polynomials.

4 Examples

We present an illustrative example to see how the Zhuang-Zi algorithm works in practice. We then present two non-trivial examples where Zhuang-Zi succeeds and Gröbner bases fail.

4.1 A Toy Example

In order to show how the algorithm works we first present a simple example of two quadratic equations in two variables with coefficients in the field $k = GF(2^2)$. We define the polynomial map $f : k^2 \longrightarrow k^2$ by its components

$$\begin{aligned} f_0(x_0, x_1) &= x_0^2 + x_1 + 1 \\ f_1(x_0, x_1) &= x_1^2 + x_0 x_1 + 1 \end{aligned}$$

in $k[x_0, x_1]$.

The nonzero elements in the field $k = GF(4)$ form a multiplicative group, which is generated by an element that we denote by a. The addition and multiplication table for elements in $GF(4)$ can be written in terms of a as follows.

$+$	0	1	a	a^2
0	0	1	a	a^2
1	1	0	a^2	a
a	a	a^2	0	1
a^2	a^2	a	1	0

$*$	0	1	a	a^2
0	0	0	0	0
1	0	1	a	a^2
a	0	a	a^2	1
a^2	0	a^2	1	a

One irreducible polynomial of degree two with coefficients in k is

$$g(y) = y^2 + y + a^2.$$

The mapping $\phi : k^2 \longrightarrow K$ is defined by

$$\phi(x_0, x_1) = x_0 + x_1 y = X,$$

while $\phi^{-1} : K \longrightarrow k^2$ is defined by (using matrix notation)

$$\phi^{-1}(X) = (X, X^4) \begin{pmatrix} 1+y & 1 \\ y & 1 \end{pmatrix} = (x_0, x_1)$$

With this notation, the polynomial map $F = \phi \circ f \circ \phi^{-1}$ is given by

$$F(X) = yX^8 + yX^5 + X^4 + X^2 + X + y + 1.$$

Since this is a trivial example, we could factor $F(X)$ directly and obtain

$$F(X) = y(X + y)(X^3 + X^2 + 1)(X^4 + (y + 1)X^3 + aX^2 + (ay + 1)X + a^2)$$

from which we can see that $X = y$ is the only solution of the equation $F(X) = 0$ in K. If we write $y = 0+1\cdot y$, we see that this solution corresponds to the solution $(x_0, x_1) = (0, 1) \in k^2$ of the system $f_0 = f_1 = 0$. In general we would expect the degree of $F(X)$ to be much larger, and so we have designed our method to reduced the degree to a reasonable level that can be handled by an efficient polynomial factorization algorithm such as the Berlekamp algorithm.

In order to illustrate how the algorithm works, we compute the functions $F_i(X) = F(X)^{q^i}$ for $i = 0, 1$, noting that here $n = 2$ and $q = 4$. So we have

$$\begin{aligned}
F_0(X) &= F(X) = yX^8 + yX^5 + X^4 + X^2 + X + y + 1\\
F_1(X) &= F^4(X) = X^8 + (y + 1)X^5 + X^4 + (y + 1)X^3 + X + y
\end{aligned}$$

We then create a 2×9 matrix with ij^{th} entry equal to the coefficient of X^{8-j} in $F_i(X)$, where $i = 0, 1$ and $j = 0, 1, \ldots, 8$. This matrix is brought into row echelon form via the Gaussian algorithm, which produces the new polynomials $S_0(X)$ and $S_1(X)$.

$$\begin{aligned}
S_0(X) &= X^8 + (a^2y + 1)X^4 + a^2yX^2 + (a^2y + 1)X + a^2 + 1\\
S_1(X) &= X^5 + (y + a^2)X^4 + (y + a)X^2 + (y + a^2)X + y + a
\end{aligned}$$

In order to reduce the degree further, we now multiply $S_0(X)$ and $S_1(X)$ each by X and X^4, and then add these additional four polynomials to the given set of polynomials. Once again Gaussian elimination is applied and produces the set of six polynomials $S_0(X), S_1(X), \ldots, S_5(X)$.

$$\begin{aligned}
S_0 &= X^{12} + (ay + 1)X^3 + (ay + a)X^2 + a^2yX + a\\
S_1 &= X^9 + (y + 1)X^3 + (a^2y + a^2)X^2 + (a^2y + a^2)X + ay + a^2\\
S_2 &= X^8 + aX^3 + (a^2y + a^2)X^2 + (y + a)X + a^2y\\
S_3 &= X^6 + aX^3 + (a^2y + 1)X^2 + a^2yX + a^2y + a\\
S_4 &= X^5 + (ay + a)X^3 + (ay + 1)X^2 + (ay + 1)X + a^2\\
S_5 &= X^4 + (ay + 1)X^3 + (a^2y + a)X^2 + a^2X + y + a^2
\end{aligned}$$

Since the degree reduction was not significant, we repeat the process. Multiplying each $S_i(X)$ by X and X^4, adding these polynomials to the associated matrix, and then applying Gaussian elimination reduces the set of eighteen polynomials to eleven polynomials $S_0(X), S_1(X), \dots, S_{10}(X)$.

$$\begin{aligned}
S_0(X) &= X^{13} + (y+a)X + a \\
S_1(X) &= X^{12} + (y+1)X + ay + a \\
S_2(X) &= X^{10} + a^2yX + a^2y \\
S_3(X) &= X^9 + (y+1)X + a^2y \\
S_4(X) &= X^8 + (a^2y+1)X + a^2y \\
S_5(X) &= X^7 + (ay+a)X + a^2y + a^2 \\
S_6(X) &= X^6 + (a^2y+1)X + y + a \\
S_7(X) &= X^5 + yX + y \\
S_8(X) &= X^4 + (ay+1)X + ay \\
S_9(X) &= X^3 + (a^2y+1)X + 1 \\
S_{10}(X) &= X^2 + (y+1)X + y
\end{aligned}$$

The factorization of $S_{10}(X) = (X+1)(X+y)$ shows that spurious solutions can appear, and must be screened for and discarded. If the process were to be repeated, the set of fourteen polynomials $S_0(X), S_1(X), \dots, S_{13}(X)$ will be generated and the spurious solution will disappear.

$$\begin{aligned}
S_0 &= X^{14} + ay + a, \\
S_1 &= X^{13} + a^2y + 1, \\
S_2 &= X^{12} + ay + 1, \\
S_3 &= X^{11} + ay, \\
S_4 &= X^{10} + a, \\
S_5 &= X^9 + a^2y + a^2, \\
S_6 &= X^8 + y + a, \\
S_7 &= X^7 + a^2y + a, \\
S_8 &= X^6 + a^2y, \\
S_9 &= X^5 + a^2, \\
S_{10} &= X^4 + y + 1, \\
S_{11} &= X^3 + ay + a^2, \\
S_{12} &= X^2 + y + a^2, \\
S_{13} &= X + y.
\end{aligned}$$

As expected each polynomial has the factor $X+y$. The set of polynomials as given is now invariant when our process is again applied, which must eventually occur since we are working in a finite field.

Of course this simple example could have been solved more easily by finding the Gröbner basis $\{x_0, x_1 + 1\}$ for the equations

$$\begin{aligned} f_0(x_0, x_1) &= 0 \\ f_1(x_0, x_1) &= 0 \\ x_0^4 - x_0 &= 0 \\ x_1^4 - x_1 &= 0 \end{aligned}$$

4.2 Generating Non-Trivial Examples

The Zhuang-Zi algorithm works of course for linear systems of equations and can give insight into how subspaces of k^n are represented in the bigger field K. Nevertheless, the main application is to the nonlinear multivariate problem where Gröbner bases methods do not succeed. As mentioned earlier, the Zhuang-Zi algorithm requires that we work in a finite field, whereas Göbner bases do not. When a Gröbner basis is computed in a finite field, it is accomplished usually by augmenting the original set of equations with those defining the finite field.

Examples that can be solved easily by the Zhuang-Zi algorithm, but only with great difficulties via Gröbner bases, can be constructed easily. The idea is to select a function $F(X) : K \longrightarrow K$ of low enough degree, so that it can be factored easily, while the corresponding mapping $f : k^n \longrightarrow k^n$ must be complicated. The degree of the components $f_0, f_1, \ldots, f_{n-1}$ of f depends on which powers of X have been selected in F. Terms in F of the form X^{q^i} give rise to linear terms in the components of f, $X^{q^i+q^j}$ leads to quadratic terms, $X^{q^i+q^j+q^l}$ leads to cubic terms, and so on. By keeping the exponents $i, j, l, \ldots$ in $X^{q^i}, X^{q^i+q^j}, X^{q^i+q^j+q^l}, \ldots$ small, we can choose a polynomial F of low enough degree so as to be easily factored, while the corresponding components of f will be quadratic, cubic, or higher degree polynomials in $k[x_0, x_1, \ldots, x_{n-1}]$. This idea is reminiscent of what has been suggested for the HFE public key encryption scheme [Pat96]. We now generate such an example.

Let $k = GF(2^3)$ and let $K = k[y]/(g(y))$ be a degree n extension of k, for some irreducible $g(y) \in k[y]$. We use a polynomial of low degree in $K[X]$:

$$F(X) = X^{72} + a_1X^{65} + a_2X^{64} + a_3X^{16} + a_4X^9 + a_5X^8 + a_6X^2 + a_7X + a_8, \quad (3)$$

where the coefficients a_j, for $j = 1, \ldots, 8$, are chosen at random from k, treated as a subfield of K via the standard embedding. With $q = 8$, all powers of X in (3) can be written in the form $X^{8^i+8^j}$ or X^{8^i}, and so it is clear that (3) $f = \phi^{-1} \circ F \circ \phi$ is a quadratic polynomial map from k^n to k^n. As in the previous example, it is helpful to write $\phi^{-1} : K \longrightarrow k^n$ using matrix notation

$$\mathsf{A}\,\mathsf{X} = \mathsf{x},$$

where $\mathsf{X} = (X^{8^0}, X^{8^1}, \ldots, X^{8^{n-1}})^T$, $\mathsf{x} = (x_0, x_1, \ldots, x_{n-1})^T$, and A is an $n \times n$ matrix with entries from K that can easily be found by writing each X^{8^i} as a polynomial in y with coefficients in $k[x_0, x_1, \ldots, x_{n-1}]$.

The polynomial (3) can be factored easily by the Berlekamp algorithm implemented in a computer algebra system like Magma [CAG05]. Depending on the coefficients $a_1, \dots, a_8$ of F, and on the value of n, F may have zero, one, or more linear factors in X. Each linear factor $X + \alpha$ with $\alpha \in K$ gives rise to a solution of the corresponding polynomial equations $f_i(x_0, x_1, \dots, x_{n-1}) = 0$, $i = 0, 1, \dots, n-1$. Finding the corresponding solutions directly with the help of a good Gröbner bases program such as Faugère's F_4 version in Magma [Fau99] requires exponential time with increasing n as seen in Figure 1.

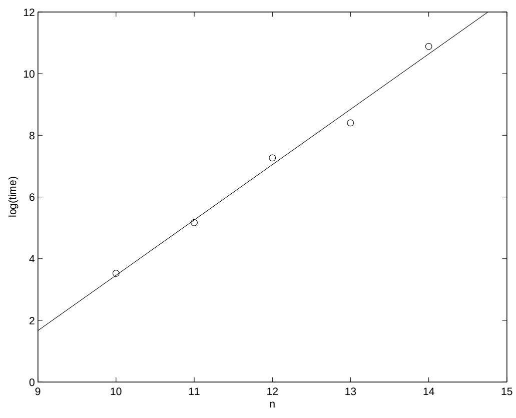

Fig. 1. Computing time for finding a Gröbner basis for n quadratic polynomials in n variables

The quadratic polynomials components of f have too many terms to be displayed here. It could be said that this is an unfair comparison, since we are solving $F(X) = 0$ directly and forcing Faugère's algorithm F_4 to work on a system with a huge number of terms. However, the Gröbner bases algorithm in Magma is very efficient, and removing even a few of the terms in (3) made it much more difficult to find examples where F_4 fails even for large n. It is easy to see why this is to be expected. In principle our algorithm should be better if the degree d of F is fixed due to the complexity estimate $O(nd^3)$ for the Berlekamp algorithm. On the other hand the complexity of the Gröbner bases algorithm is expected to be exponential in n, the number of variables.

This first non-trivial example shows that the Zhuang-Zi algorithm, even in its most simple form using only Step 1 and the Berlekamp algorithm, sometimes has an advantage over the best Gröbner bases algorithms.

4.3 A Non-Trivial Example

We now give a non-trivial example where $F(X)$ is of very high degree and therefore cannot be solved with the simplest form of the Zhuang-Zi algorithm as discussed in the previous section.

Let $q = 4$, $k = GF(q)$, as in Section 4.1, take $g(y) \in k[y]$ to be the irreducible polynomial

$$g(y) = y^{12} + y^{11} + ay^{10} + ay^9 + y^8 + y^7 + y^5 + a^2y^4 + ay^3 + a^2y^2 + ay + a,$$

and define $K = k[y]/(g(y))$, a degree $n = 12$ extension of k.

Let $F(X) \in K[X]$ be the polynomial

$$\begin{aligned} F(X) = &a^2X^{17664} + X^{5440} + aX^{5376} + X^{4416} + aX^{4096} + aX^{1360} \\ &+ X^{1344} + X^{1280} + a^2X^{1024} + a^2X^{336} + aX^{320} + a^2X^{276} \\ &+ X^{85} + aX^{84} + aX^{64} + aX^{21} + X^{20} + a \end{aligned}$$

It is easy to check that each exponent of X in $F(X)$ is a sum of powers of four, and that the exponent with the most powers of four is $5440 = 4^3 + 4^4 + 4^5 + 4^6$. Therefore the components of $f = \phi^{-1} \circ F \circ \phi$ will be of degree four. As before, there are too many terms in each $f_i(x_0, x_1, \ldots, x_{n-1})$ to be displayed here.

The degree of F prevents us from directly solving $F(X) = 0$ using the Berlekamp algorithm. Also, the F_4 implementation in Magma failed to find a Gröbner basis for $f_0, f_1, \ldots, f_{n-1}$ due to the fact that memory requirements exceeded the available resources on our PC (1.73 GHz, 1 GB of RAM) after approximately 620 seconds. However, the Zhuang-Zi algorithm found the polynomial

$$S(X) = X^{276} + aX^{85} + a^2X^{84} + a^2X^{64} + a^2X^{21} + aX^{20} + a$$

after approximately 30 seconds. Approximately 170 seconds later the solutions $\{1, a\}$ of $F(X) = 0$ were returned after using Magma's factorization algorithm.

Other similar examples whose solutions can not be obtained from the Gröbner bases algorithm, and which require several iterations of the full algorithm, can be generated in this way. We omit them here due to space constraints.

5 Discussion

What we propose here is not just a new algorithm, but a new way to look at the problem of solving a set of multivariate polynomial equations over a finite field. We lift the problem to an extension field where it becomes a single variable problem, and then use existing efficient techniques for solving a polynomial equation in a single variable over a finite field. In this way we actually unify the multivariate case and the single variable case. This also means that sometimes it can be beneficial to view a single variable polynomial equation over a given finite field as a set of multivariate polynomial equations over a smaller finite field. We believe that this is an approach that merits further investigation.

Our experiments above have shown that there are cases where the Zhuang-Zi algorithm will succeed in finding a solution to a set of polynomial equations, whereas Gröbner bases algorithms will fail due to space and/or time limitations.

There are many interesting directions for further research. One question is the complexity of the Zhuang-Zi algorithm for a set of n nonlinear equations in n variables. In general the complexity will be exponential in n, though for certain types of equations the Zhuang-Zi algorithm will work much better. A very good testing ground is the class of equations that arise in connection with the HFE multivariate public key cryptosystem, which we are currently investigating.

Another important task is to make a systematic comparison of the Zhuang-Zi algorithm with other algorithms, for example with the new Gröbner basis algorithms like F_4 and F_5, will give a better understanding of its complexity. We do not expect that the Zhuang-Zi algorithm will be better in a general way, but rather, we believe that the algorithms can be complimentary to each other.

One interesting point of the Zhuang-Zi algorithm is that it is actually closely related to the XL algorithm. If in Step 3 we do not look for a polynomial of lowest degree but instead we look for a polynomial of the form

$$\sum_{i=0}^{n-1} A_i X^{q^i} + B,$$

then the Zhuang-Zi algorithm is equivalent to the XL algorithm. It would be interesting to consider how to combine the two algorithms together.

If $F(X)$ is known to have a large number of solutions (larger than the threshold for factorization degree D), then the Zhuang-Zi algorithm cannot succeed. One strategy that can be employed is to add randomly chosen polynomials $f_n, \ldots, f_{n'-1}$ to the original set of polynomial $f_0, f_1, \ldots, f_{n-1}$. It is very likely that the resulting new system of equations will have fewer solutions, and that Zhuang-Zi may be successful. If not, we can start over with a new set of randomly chosen polynomials.

A much more general consideration is of the so-called hard cases, those equations that are generically hard to solve using any kind of general algorithm. We believe our approach may provide some insight into how to look at this problem. Among other applications, any result in this direction will have a very strong connection and impact on the provable security of multivariate public key cryptosystems.

There is considerable room to improve and optimize the implementation of the algorithm. For example, the algorithm relies very much on how the big field K is implemented. Also, we expect that it is possible to speed up the Gaussian elimination process by using sparse matrices. If the Zhuang-Zi algorithm is viewed as a philosophy for solving equations, then it is clear that there is great flexibility to create efficient variants in the degree reduction steps, variants tailored to fit with specific efficient polynomial equation solvers and factorization algorithms.

References

[Buc65] B. Buchberger. Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal. Dissertation, Universität Innsbruck, 1965

[CAG05] University of Sydney Computational Algebra Group. The MAGMA Computational Algebra System for Algebra, Number Theory and Geometry. http://magma. maths.usyd.edu.au/magma/, 2005.

[CKPS00] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In B. Preenel, editor, *Advances in Cryptology–Eurocrypt 2000*, LNCS, volume 1807, pages 392–407, Springer, 2000.

[Fau99] Jean-Charles Faugère. A New Efficient Algorithm for Computing Gröbner Bases (F_4). *Journal of Pure and Applied Algebra*, 139:61–88, June 1999.

[Fau02] Jean-Charles Faugère. A New Efficient Algorithm for Computing Gröbner Bases Without Reduction to Zero (F_5). In *International Symposium on Symbolic and Algebraic Computation–ISSAC 2002*, pages 75–83. ACM Press, July 2002.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.

[KS99] Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In M. Wiener, editor, *Advances in Cryptology–Crypto 1999*, LNCS, volume 1666, pages 19–30, Springer, 1999.

[MI88] T. Matsumoto and H. Imai. Public Quadratic Polynomial-Tuples for Efficient Signature Verification and Message Encryption. In C. G. Guenther, editor, *Advances in Cryptology–Eurocrypt 1988*, LNCS, volume 330, pages 419–453, Springer, 1988.

[Pat96] J. Patarin. Hidden Field Equations (HFE) and Isomorphism of Polynomials (IP): Two New Families of Asymmetric Algorithms. In U. Maurer, editor, *Advances in Cryptology–Eurocrypt 1996*, LNCS, volume 1070, pages 33–48, Springer, 1996. Extended Version: http://www.minrank.org/hfe.pdf.

[Pat00] J. Patarin. Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88. *Designs, Codes and Cryptography*, 20:175–209, 2000.

[Smi52] David E. Smith. *History of Mathematics*, volumes 1 and 2. New York, Dover, 1951, 52.

Appendix: Zhuang Zi

Zhuang Zi, also known as Zhuang Zhou, Chuang Tzu, Chuang Tse, and Chuang Chou (369 BC – 286 BC), was a Chinese philosopher, who lived during the Warring States era. He also lived in a time, which is called a time of the Hundred Schools of Thoughts, during which most major schools of Chinese Philosophy were originated. Zhuang is actually his family name and Zhou the first name. The name Zhuang Zi is used as a way to show respect for him. He categorically belongs to the school of Taoism.

One the most famous stories about him and his philosophy is from his book also known as Zhuang Zi.

Once upon a time, I, Zhuang Zhou, dreamt I was a butterfly, fluttering here and there. I was conscious only of my being as a butterfly, unaware that I was Zhou. Soon I awaked, and there I was, myself again. Now I am confused and do not know whether I was then a man dreaming I was a butterfly, or whether I am now a butterfly, dreaming I am a man.

For the first author, the idea of transition between the vector space on a small finite field and the equivalent large field is very much inspired by such an idea (and surely by previous works of other colleagues like Matsumoto, Imai, Patarin, Kipnis, Shamir, and etc.), and is just like Zhuang Zhou and the butter fly. This is the origin of the name for this algorithm.