A solver for General Unilateral Polynomial Matrix Equation with Second-Order Matrices Over Prime Finite Fields

Filipp Burtyka
Junior Research Fellow, Institute of Computer Technologies and Information Security of Southern Federal University, 2 Chekhova St., Taganrog, Russia
E-mail: bbfilipp@yandex.ru

Abstract. The paper firstly considers the problem of finding solvents for arbitrary unilateral polynomial matrix equations with second-order matrices over prime finite fields from the practical point of view: we implement the solver for this problem. The solver’s algorithm has two step: the first is finding solvents, having Jordan Normal Form (JNF), the second is finding solvents among the rest matrices. The first step reduces to the finding roots of usual polynomials over finite fields, the second is essentially exhaustive search. The first step’s algorithms essentially use the polynomial matrices theory. We estimate the practical duration of computations using our software implementation (for example that one can’t construct unilateral matrix polynomial over finite field, having any predefined number of solvents) and answer some theoretically-valued questions.

1. Introduction
Let \( K \) be a field and let \( M_n(K) \) denote the ring of all \( n \times n \) matrices with entries in \( K \). The field \( K \) is meant to be \( \mathbb{C} \) or \( \mathbb{F}_p \), for some \( p \in \mathbb{N} \) (when \( p \) is a prime number, then field is called prime).

The unilateral matrix polynomial (UMP) of \( n \)-th order and \( d \)-th degree over \( K \) is the expression of the form

\[
F(X) = X^d + F_{d-1} \cdot X^{d-1} + \cdots + F_2 \cdot X^2 + F_1 \cdot X + F_0,
\]

where \( F_i \in M_n(K), i = 0, \ldots, d - 1 \) are coefficients and \( X \in M_n(K) \) is (unknown) variable.

The set of all such UMP for given \( n \) and \( K \) we denote by \( \langle M_n(K) \rangle[X] \). The root (or solvent) of UMP \( F(X) \) is a matrix \( S \in M_n(K) \), such that \( F(S) = 0 \), where \( 0 \) is zero \( n \)-th order matrix. The set of all UMP’s roots we denote \( \text{Ker}(F(X)) \).

Remark 1. This paper considers only left UMPs, while right unilateral matrix polynomials exist looking like \( F(X) = X^d + X^{d-1} \cdot F_{d-1} + \cdots + X^2 \cdot F_2 + X \cdot F_1 + F_0 \). Our below discussion without loss of generality can be extended to them.

The most important problems related to UMP \( F(X) \) are (i) whether \( F(X) \) has solvents and (ii) how to find these solvents. The main challenges are non-commutative nature of UMPs, the presence of zero-divisors in \( M_n(K) \) and as a consequence anomalous (in contrast to usual scalar polynomials) total number of these solutions. Consider two examples, where \( K \) is \( \mathbb{C} \).
Example 1 (The unresolvable UMP).

\[ F(X) = X^2 - \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \in (M_2(\mathbb{C}))[X] \]

The root of \( F(X) \) can be only the matrix, having exclusively zero eigenvalues (because the square of this matrix has only zero eigenvalues), i.e. nilpotent matrix. But the square of any nilpotent 2 × 2 matrix is the zero matrix.

Example 2 (The UMP having infinity roots).

\[ F(X) = X^2 - \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \in (M_2(\mathbb{C}))[X] \]

The roots are all nilpotent 2 × 2 matrices.

The following theorem summarizes the general results about the total number of solutions for \( K = \mathbb{C} \) from a plenty of works devoted to analysis of UMP over \( \mathbb{C} \).

**Theorem 1.**

(i) There exist unresolvable UMPs \( F(X) \in (M_n(\mathbb{C}))[X] \).

(ii) There exist UMPs \( F(X) \in (M_n(\mathbb{C}))[X] \), having infinity number of solvents.

(iii) If \( F(X) \in (M_n(\mathbb{C}))[X] \) has finite number of solvents, then \( |\text{Ker}(F(X))| \leq C_n^{n} \cdot d \).

An important question is: what can we say about the total number of solvents for given UMP \( F(X) \in (M_n(F_p))[X] \), where \( p \) is prime number? This problem hasn’t been studied yet in literature for the case of arbitrary \( F(X) \). May be this was caused by the fact that until 2014 UMP-related applications consider only \( \mathbb{C} \) case (it was applications from mechanics and control theory, see, for example [34, 35]). But today it has important practical applications in homomorphic cryptography [17, 18], because finding solvents of UMP over \( \mathbb{F}_p \) in fact simulates cryptanalitic attack on some homomorphic encryption. In particular, we interested in problem: to what complexity class belongs the problem of finding UMP’s solvents? For investigation this problem the first step is to determine, if for any number \( t \in [0, p^{2n}] \cap \mathbb{N} \) there exist UMP, having exactly \( t \) solvents? Another aim is to simulate in practice the process of solving UMP problem by attacker.

In this work we partially solve both tasks. We discuss algorithms for computing all solvents of \( F(X) \in (M_2(F_p))[X] \) belonging to the class of matrices \( E_{2}(F_p) \) with all eigenvalues in \( F_p \) (the last class of matrices denoted by \( E_{2}(F_p) \)), and estimate the complexity of these algorithms.

Our results are essentially based on the thorough study of computing the roots and estimating their number for second-degree UMP \( F(X) \in (M_2(\mathbb{C}))[X] \) presented in [4]. We make an adaptation of methods (informally) described in [4] for \( \mathbb{C} \) to \( F_p \).

The paper is organized as follows: Section 2 reviews previous works devoted to finding UMP’s solvents, Section 3 gives necessary theoretical background concerning polynomial matrices, Sections 4 and 5 discuss algorithms for finding UMP’s solvents, Section 6 shows experimental results and, finally Section 7 is conclusion.

2. Related works

2.1. The works concerning the UMPs over \( \mathbb{C} \)

The major part of works dealing with matrix polynomials is devoted to calculating solvents of UMP \( F(X) \in (M_n(\mathbb{C}))[X] \). Several researchers [1, 2, 5, 10, 7] presented the result formalized in Theorem 1. This result was obtained using polynomial matrix theory [8] (see section 3). The detailed case study for second-order matrices was done in [4, 3].
In [10] the author thoroughly studies the properties of block-companion matrix. It can be composed by coefficients of UMP \( \mathcal{F}(X) \in (M_n(\mathbb{C}))[X] \) and gives an insight how matrix polynomials solvents are built inside. Also the conditions for the number of solvents to be infinite are stated and some cases of finite (positive) solvents number are considered.

Theorem 1 states if \( \mathcal{F}(X) \in (M_n(\mathbb{C}))[X] \) has the finite number of solvents \( t \), then \( t \leq C_n^d \).

But for arbitrary \( n \) it’s unknown if \( \mathcal{F}(X) \in (M_n(\mathbb{C}))[X] \) exists having exactly \( t \) roots. In [5] the author solves this problem for \( n = 2 \). He shows for \( n = 2 \) and \( \forall d, t \leq C_{2,d}^d \)-th degree UMP exists with exactly \( t \) solvents. To achieve this result the author builds UMP of special construction. Such UMP has only diagonalizable (recall that a square matrix \( A \) is called diagonalizable if it is similar to a diagonal matrix, i.e., if there exists an invertible matrix \( P \) such that \( P^{-1}AP \) is a diagonal matrix) solvents, and their number could be exactly \( t \).

Finally there is a plenty of works proposing numerical algorithms for computing solvents of UMP \( \mathcal{F}(X) \in M_n(\mathbb{C})[X] \). For example, in [1, 22, 23, 24, 25] the authors propose how to adapt numerical methods of computing matrix polynomials solvents.

2.2. The works concerning the UMPs over \( \mathbb{F}_p \)

To the best of our knowledge the problem of computing and estimating the number of roots for arbitrary given \( \mathcal{F}(X) \in M_n(\mathbb{F}_p)[X] \) is not studied in literature. But anyway some case studies have been done.

Equation \( X^n = 0 \) over \( M_n(\mathbb{F}_p) \) was investigated in the following works [21, 11, 12]. It’s well known that solutions of this equations are nilpotent \( n \times n \) matrices. It’s proved in [21] the total number of \( n \times n \) nilpotent matrices is \( p^{n^2-n} \).

The study of solving \( X^n = B \) over \( M_n(\mathbb{F}_p) \) was done in [13, 14], where \( B \in M_n(\mathbb{F}_p) \) – some arbitrary matrix.

Also there are works researching the question how to find matrices \( S \in M_n(\mathbb{F}_p) \) being solutions of given scalar polynomial \( f(x) \in \mathbb{F}_p[x] \) (see, for example, [15]).

3. Theoretical background on polynomial matrices and solvents of matrix polynomials

Each UMP can be associated with a polynomial matrix (called \( \lambda \)-matrix) \( \mathcal{F}(\lambda) \) which is obtained from \( \mathcal{F}(X) \) through replacing matrix-variable \( X \in M_n(\mathbb{K}) \) by scalar \( \lambda \in \mathbb{K} \), i.e. \( \mathcal{F}(\lambda) = \lambda^d + F_{d-1} \cdot \lambda^{d-1} + \cdots + F_0 \in (M_n(\mathbb{K})[\lambda]) \).

Below definitions has been formulated in [10] for field \( \mathbb{C} \), but here we rewrite them for arbitrary field \( \mathbb{K} \).

**Definition 1 ([10]).** The latent root (or eigenvalue) of \( \lambda \)-matrix \( \mathcal{F}(\lambda) \) is \( \alpha \in \mathbb{K} \) such that \( \det(\mathcal{F}(\alpha)) = 0 \).

**Definition 2 ([10]).** The latent vector \( \vec{v} \in \mathbb{K}^n \) (or eigenvector) of \( \mathcal{F}(\lambda) \), corresponding to eigenvalue \( \alpha \in \mathbb{K} \), is a vector such that \( \mathcal{F}(\alpha) \cdot \vec{v} = \vec{0} \).

**Definition 3 ([10]).** Given \( \mathcal{F}(\lambda) \) of second degree and its eigenvalue \( \alpha \in \mathbb{K} \), let \( \mathcal{F}' \) denote the derivative of \( \mathcal{F} \). If vectors \( \vec{v}_1, \vec{v}_2 \in \mathbb{K}^n \), with \( \vec{v}_1 \neq \vec{0} \), not necessary distinct or linearly independent satisfy

\[
\mathcal{F}(\alpha)\vec{v}_2 + \mathcal{F}'(\alpha)\vec{v}_1 = \vec{0}
\]

we say \( \vec{v}_1, \vec{v}_2 \) is a Jordan chain (or generalized eigenvectors) of length 2 of \( \mathcal{F}(\lambda) \) corresponding to the eigenvalue \( \alpha \).

**Definition 4 ([10]).** Let there is \( \mathcal{F}(\lambda) \) and the matrix \( V \) over \( \mathbb{K} \) of dimension \( n \times k \), \( V = [\vec{v}_1 \ldots \vec{v}_k] \), where \( \vec{v}_1, \ldots, \vec{v}_k \) is a Jordan chain of \( \mathcal{F}(\lambda) \) corresponding to eigenvalue \( \alpha \). The pair \( (V, J) \), where \( J \) – Jordan block of size \( k \) with \( \alpha \) in the main diagonal, is an eigenpair of \( \mathcal{F}(\lambda) \).
Theorem 2 ([10]). If the matrix $T = [V_1 \ldots V_l]$ is a nonsingular matrix of order $n$ and $J_0 = \text{diag}(J_1, \ldots, J_l)$ is also matrix of order $n$, then $(V_i, J_i), i = 1, l$ are eigenpairs of $F(\lambda)$ if and only if $S = T \cdot J_0 \cdot T^{-1}$ is a root of UMP $F(X) \in (M_n(\mathbb{K}))[X]$.

Theorem 2 shows the internal structure of solvents of given matrix polynomial $F(X) \in (M_n(\mathbb{K}))[X]$. It states that any solvent of $F(X)$ can be composed by some subsets of $F(\lambda)$ eigenpairs. Of course if the basic field $\mathbb{K}$ is not algebraically closed, than some part of eigenpairs are matrices over extension of $\mathbb{K}$. The simple corollary of Theorem 2 is if $S$ is a root of $F(X)$, then all its eigenvalues and eigenvectors are eigenvalues and eigenvectors of $F(\lambda)$.

Theorem 2 says how to compute all $F(X)$ solvents: find all eigenpairs and try to combine them in a different ways to get all different solvents of $F(X)$.

The next theorem was proved in [2], but it can be viewed as a direct consequence of Theorem 2. It shows how to get all diagonalizable solvents of $F(X)$ over $\mathbb{K}$.

Theorem 3 ([2]). Let $F(\lambda) \in (M_n(\mathbb{K}))[\lambda]$ has eigenvalues $\lambda_1, \ldots, \lambda_n \in \mathbb{K}$ and their corresponding eigenvectors form linear independent system $\{\vec{v}_1, \ldots, \vec{v}_n\}$. Then

$$S = V \cdot \text{diag}(\lambda_1, \ldots, \lambda_n) \cdot V^{-1} \in M_n(\mathbb{K})$$

is root of $F(X)$, where $i$-th column of $V \in M_n(\mathbb{K})$ is $\vec{v}_i$. Any diagonalizable root of $F(X)$ can be built in this way.

Denote as $D_2(\mathbb{K})$ the set of all diagonalizable $2 \times 2$ matrices with entries from $\mathbb{K}$.

Since in this work we are going to deal only with $2 \times 2$ matrices, let’s see how the roots of $F(X) \in (M_2(\mathbb{F}_p))[X]$ belonging to $Eig_2(\mathbb{F}_p)$ will look like. We have two types of solvents:

- Diagonalizable solvent is $S = V \cdot \begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \cdot V^{-1}$, where $V = [\vec{v}_1 \ \vec{v}_2] \in M_2(\mathbb{F}_p)$, $\alpha_i \in \mathbb{F}_p$, $(\alpha_i, \vec{v}_i), i = 1, 2$ are two eigenpairs of $F(\lambda)$, $\vec{v}_1, \vec{v}_2$ are linear independent.

- Non-diagonalizable solvent is $S = V \cdot \begin{pmatrix} \alpha & 1 \\ 0 & \alpha \end{pmatrix} \cdot V^{-1}$, where $V = [\vec{v}_1 \ \vec{v}_2] \in M_2(\mathbb{F}_p)$ is a Jordan chain of $F(\lambda)$ corresponding to eigenvalue $\alpha \in \mathbb{F}_p$, $\vec{v}_1, \vec{v}_2$ are linear independent (latent) vectors, $\alpha$ has algebraic multiplicity $\geq 2$. Recalling theorem 2 we get $\vec{v}_1, \vec{v}_2$ are solutions of the following equations system:

$$\begin{cases} F(\alpha) \ \vec{v}_1 = \vec{0} \\ F(\alpha) \ \vec{v}_2 + F'(\alpha) \ \vec{v}_1 = \vec{0}. \end{cases}$$

4. Finding solvents by exhaustive search and by solving corresponding system of scalar polynomial equations

Since the number of solvents for UMP over $\mathbb{F}_p$ is always finite, one can find all of them using exhaustive search. The complexity of evaluating $d$-th degree UMP $F(X) \in (M_2(\mathbb{F}_p))[X]$ at some point $S \in M_2(\mathbb{F}_p)$ is $O(d)$, if one uses Horner’s rule. And since the total number of $2 \times 2$ matrices is $p^4$, the complexity of brute force search is $O(d \cdot p^4)$ arithmetic operations in $\mathbb{F}_p$.

Another technique is just write in symbolic form the unknown solvent’s entries and construct a system of usual polynomial equations that then solve using such techniques as Grobner basis, linearisation, e t.c. This was done in [31], where one can find tables for number of monic UMP’s, having various numbers of solvents. In particular, in Table 1 from [31] one can see that for every $t \in [0, 16] \cap \mathbb{N}$. Such experiments was carried out for all $\mathbb{F}_2, \ldots, \mathbb{F}_{31}$ and for matrix dimensions $2, \ldots, 7$ and degrees $2, 7$ (for larger fields, dimensions and degrees our computer failed to finish).

This allows to propose the following hypothesis:

Hypothesis 1. For any prime field $\mathbb{F}_p$ and any $t \in [0, p^n] \cap \mathbb{N}$ there exist UMP $F(X) \in (M_n(\mathbb{F}_p))[X]$ having exactly $t$ solvents.
5. Finding solvents using λ-matrixes
Now let’s proceed to algorithms for finding solvents using polynomial matrices theory. Almost immediately from the definition of λ-matrices and latent roots follows Algorithm 1.

5.1. The overview of algorithm for finding all solvents
We recall there are two types of solvents – diagonalizable and non-diagonalizable ones (see Section 3). Following [4] solvents of each of these types are search in separately ways. Because our algorithm for \( \mathbb{F}_p \) is adaptation of algorithm from [4] for \( \mathbb{C} \) we also will adhere to this strategy: algorithm \( \text{FindDiagonalizableSolvents}_2(\mathcal{F}(X)) \) described below finds actually \( \text{Ker}(\mathcal{F}(X)) \cap \mathcal{D}_2(\mathbb{F}_p) \) and algorithm \( \text{FindNonDiagonalizableSolventsHavingJNF}_2(\mathcal{F}(X)) \) also described further finds actually \( \text{Ker}(\mathcal{F}(X)) \cap (\text{Eig}_2(\mathbb{F}_p) \setminus \mathcal{D}_2(\mathbb{F}_p)) \).

In the case, when solvents finding problem is considered over \( \mathbb{C} \) every matrix has JNF, therefore we can find non-diagonalizable solvents exploiting this fact. While all matrixes from \( \mathcal{D}_2(\mathbb{F}_p) \) still have JNF, the remaining matrices are divided into two sets: the set of matrices, having JNF \( (\text{Eig}_2(\mathbb{F}_p)) \) and the set of matrices, not having JNF.

\[
\text{Ker}(\mathcal{F}(X)) = \left( \text{Ker}(\mathcal{F}(X)) \cap \mathcal{D}_2(\mathbb{F}_p) \right) \cup \left( \text{Ker}(\mathcal{F}(X)) \cap (\text{Eig}_2(\mathbb{F}_p) \setminus \mathcal{D}_2(\mathbb{F}_p)) \right) \cup \\
\left( \text{Ker}(\mathcal{F}(X)) \cap (\text{M}_2(\mathbb{F}_p) \setminus \text{Eig}_2(\mathbb{F}_p)) \right)
\]

But since unlike \( \text{M}_2(\mathbb{C}) \) eigenvalues of matrix from \( \text{M}_2(\mathbb{F}_p) \) may not be in \( \mathbb{F}_p \), for now we are forced to work with \( \text{Eig}_2(\mathbb{F}_p) \) instead of entire \( \text{M}_2(\mathbb{F}_p) \). We recall \( \text{Eig}_2(\mathbb{F}_p) \) is a big class of matrices, according to [19]:

\[
|\text{Eig}_2(\mathbb{F}_p)| = \frac{1}{2} \cdot p^4 + p^3 - \frac{1}{2} \cdot p^2.
\]

Keeping in mind \( |\text{M}_2(\mathbb{F}_p)| = p^4 \) we get fraction \( \frac{|\text{Eig}_2(\mathbb{F}_p)|}{|\text{M}_2(\mathbb{F}_p)|} \) decreases to a limiting value \( \frac{1}{2} \) as \( p \) approaches infinity. And this means we take into account more than half of matrices from \( \text{M}_2(\mathbb{F}_p) \) in our current work.

So, the overall algorithm is Algorithm 1.

Algorithm 1: \( \text{FindRootsByPolynomialMatrix}_2(\mathcal{F}(X)) \)

| Input: UMP \( \mathcal{F}(X) \in \text{M}_2(\mathbb{F}_p[\lambda]) \) |
| Output: Set of all solvents \( \mathcal{F}(X) \) |
1 \( D := \text{FindDiagonalizableSolvents}_2(\mathcal{F}(X)) \)
2 \( J := \text{FindNonDiagonalizableSolventsHavingJNF}_2(\mathcal{F}(X)) \)
3 \( N := \text{FindSolventsNotHavingJNF}_2(\mathcal{F}(X)) \)
4 Return \( D \cup J \cup N \)

5.2. Diagonalizable and Non-diagonalizable solvents having JNF
We search such solvents using techniques from [4] and summarize this in [38].

5.3. Handling matrices not having JNF
Algorithm \( \text{FindSolventsNotHavingJNF}_2(\mathcal{F}(X)) \) mentioned in Algorithm 1 is just subsequent substitution in \( \mathcal{F}(X) \) of all matrixes not having JNF. The set \( \text{Eig}_2(\mathbb{F}_p) \) can be prepared in advance, before running \( \text{FindSolventsNotHavingJNF}_2(\mathcal{F}(X)) \). It can be done through
exhaustive search, but we propose slightly more computationally effective algorithm. Let’s see at Algorithm 2.

**Algorithm 2: GetMatricesNotHavingJNF\(_2(\text{deg2irreducibles}(F_p))\)**

**Input:** list \(\text{deg2irreducibles}(F_p)\) of all monic irreducible polynomials over \(F_p\)

**Output:** Set of matrixes not having Jordan Normal Form over \(F_p\)

1. \(\mathcal{N} := \emptyset\)
2. for each irreducible polynomial \(x^2 + p_1x + p_0 \in \text{deg2irreducibles}(F_p)\) do
3.   for \(d = 0\) to \(p - 1\) do
4.     \(a := -(p_1 + d)\)
5.     for \(b = 0\) to \(p - 1\) do
6.       if \(b \neq 0\) then
7.         \(c := -(d^2 + p_1d + p_0) \cdot b^{-1}\)
8.         \(\mathcal{N} := \mathcal{N} \cup \{\begin{pmatrix} a & b \\ c & d \end{pmatrix}\}\)
9.       end
10.  else
11.     for \(c = 0\) to \(p - 1\) do
12.       \(\mathcal{N} := \mathcal{N} \cup \{\begin{pmatrix} a & b \\ c & d \end{pmatrix}\}\)
13.     end
14. end
15. end
16. Return \(\mathcal{N}\)

Algorithm 2 uses, in turn list of irreducible polynomials of secon degree over \(F_p\). Such list of polynomials can be found, for example, at [31] or [33] for tables with polynomials or approaches for fast generating.

6. Experimental results

Using NTL library by Victor Shoup [30] we implement both exhaustive search and algorithms for finding solvents using polynomial matrices theory.

From our experiments we found that for \(F_p\), \(p = 2, 3, ..., 104729\) (first 10000 prime numbers) there exist UMP’s having any number of solvents from \([0, p^4] \cap \mathbb{N}\) which acknowledge Hypothesis 1 for these \(p\). Also, were carried out experiments for these \(p\) in order to found UMPs having exactly \(t\) diagonalizable solvents for each \(t \in [0, p^4] \cap \mathbb{N}\). The experiments results that for each \(p\) there exist such \(t\) that there is no UMPs having \(t\) diagonalizable solvents. For example, Table 1 presents experimental data about the number of UMPs from \((M_2(F_2))[X]\) with each number of solvents in \([0, 8] \cup \mathbb{N}\) (\(M_2(F_2)\) has only 8 diagonalizable matrices).

Another observation is that for sufficiently large degrees of UMPs (different for each \(F_p\)) pure exhaustive search algorithm finished faster then algorithm based on \(\lambda\)-matrices, while for smaller degrees algorithm based on \(\lambda\)-matrices beats pure exhaustive search. Below we give experimental results for our implementation that runs on laptop with CPU AMD Phenom(tm) II P960 Quad-Core 1.80 GHz and 4.00 Gb RAM.
Table 1. Number of UMPs from \((M_2(\mathbb{F}_2))[X]\) having exact numbers of diagonalizable solvent.

|                | \(\text{deg} = 2\) | \(\text{deg} = 3\) | \(\text{deg} = 4\) | \(\text{deg} = 5\) |
|----------------|---------------------|---------------------|---------------------|---------------------|
| without solvents | 171                 | 2736                | 43776               | 700416             |
| with 1 solvent   | 66                  | 1056                | 16896               | 270336             |
| with 2 solvents  | 0                   | 0                   | 0                   | 0                   |
| with 3 solvents  | 18                  | 288                 | 4608                | 73728              |
| with 4 solvents  | 0                   | 0                   | 0                   | 0                   |
| with 5 solvents  | 0                   | 0                   | 0                   | 0                   |
| with 6 solvents  | 0                   | 0                   | 0                   | 0                   |
| with 7 solvents  | 0                   | 0                   | 0                   | 0                   |
| with 8 solvents  | 1                   | 16                  | 256                 | 4096               |

Table 2. Ranges of time it takes to get the solvents of UMPs with various parameters using exhaustive search.

|                | \(p = 2\) | \(p = 3\) | \(p = 5\) | \(p = 7\) |
|----------------|-----------|-----------|-----------|-----------|
| \(\text{deg} = 2\) | 0..1 ms   | 0..5 ms   | 10..52 ms | 38..230 ms |
| \(\text{deg} = 3\) | 1..14 ms  | 1..16 ms  | 17..88 ms |           |
| \(\text{deg} = 4\) | 1..25 ms  | 3..39 ms  |           |           |
| \(\text{deg} = 5\) | 2..43 ms  |           |           |           |

Table 3. Ranges of time it takes to get the solvents of UMPs with various parameters using Lambda-matrices based solver.

|                | \(p = 2\) | \(p = 3\) | \(p = 5\) | \(p = 7\) |
|----------------|-----------|-----------|-----------|-----------|
| \(\text{deg} = 2\) | 4..14 ms  | 4..30 ms  | 6..369 ms | 8..741 ms |
| \(\text{deg} = 3\) | 2..150 ms | 5..190 ms |           |           |
| \(\text{deg} = 4\) | 2..251 ms |           |           |           |
| \(\text{deg} = 5\) | 3..328 ms |           |           |           |

7. Conclusion

The solver for UMP problem over \(\mathbb{F}_p\) was proposed and implemented. The solver’s algorithms based on \(\lambda\)-matrices theory and beat the pure exhaustive search algorithm for sufficiently small degrees and relatively big \(p\) (for example for \(p > 19\) and \(\text{deg}\) of UMP less than ten).

Using the solver the Hypothesis 1 was checked for \(\mathbb{F}_2, ..., \mathbb{F}_{104729}\), \(n = 2\) and was confirmed for all these parameter values.

The technique using in [5] for proof that for each number \(t\) from 0 to \(C_n^k\) there exist UMP over \(\mathbb{C}\) having exactly \(t\) solvents and that is the construction of UMP having only diagonalizable solvents now is obviously not applicable to the case of \(\mathbb{F}_p\) while the statement itself seems to be valid.
Acknowledgments
The work was supported by the RFBR grant no. 16-37-00125 mol_a.

References
[1] Dennis J E, Traub J F and Weber R P 1976 The algebraic theory of matrix polynomials. SIAM Journal on Numerical Analysis 13(6) pp 831–845.
[2] Dennis J E, Traub J F and Weber R P 1978 Algorithms for solvents of matrix polynomials SIAM Journal on Numerical Analysis 15 (3) pp 523-533
[3] Gelfand S I 2004 On chisle reshenij kvadratnogo uravnenija [On the number of quadratic equation’s solutions] Obshhematematiceskij seminar “Globus”. [All-mathematical workshop “Globus”]. 1 pp 124–133 (In Russian)
[4] Wilson R L 1999 Polynomial equations over matrices. Rutgers University preprint.
[5] Slusky M 2010 Zeros of 2 × 2 matrix polynomials. Communications in Algebra. 38(11) 4212–4223.
[6] Hellenbrand K 2009 Solutions of Polynomial Equations over Matrices.
[7] Shavarovskii B Z 2007 Finding a complete set of solutions or proving unsolvability for certain classes of matrix polynomial equations with commuting coefficients. Computational Mathematics and Mathematical Physics., 47(12) pp 1902–1911.
[8] Gantmakher F 1959 The theory of matrices – Volume 1 (Chelsea, New York)
[9] Fuchs D and Schwarz A 1996 A Matrix Vieta Theorem. E.B. Dynkin Seminar, Amer. Math. Soc. Transl. Ser. 2 . p 196.
[10] Pereira E 2003 On solvents of matrix polynomials. Applied numerical mathematics. 47 pp 197–208.
[11] Kirillov A A 1995 O chisle reshenij uravnenija $X^2 = 0$ v treugol’nyh matricah nad konechnym polem [On the number of solutions to the equation $X^2 = 0$ in triangular matrices over a finite field]. Functional Analysis and Its Applications. 29(1) pp 64–68. (In Russian)
[12] Ekhad S B and Zeilberger D 1996 The number of solutions of $X^2 = 0$ in triangular matrices over $GF (q)$. The Electronic Journal of Combinatorics 3, #R2.
[13] Hodges J H 1958 The matrix equation $X^2 – I = 0$ over a finite field. American Mathematical Monthly. 65(7) pp 518–520.
[14] Acosta-De-Orozco M T and Gomez-Calderon J 1993 On the matrix equation $X^n = B$ over finite fields. International Journal of Mathematics and Mathematical Sciences. 16(3) pp 539–544.
[15] Hodges J H 1958 Scalar polynomial equations for matrices over a finite field. Duke Math. Journal 25(3) 291–296
[16] Hua J 1999 Polynomial equations for matrices over finite fields. Bulletin of the Australian Mathematical Society. 59(1) pp 59–64.
[17] Burtyka F B 2014 Paketnoe simmetrichnoe polnost’ju gomomorfnoe shifrovanie na osnove matrichnyh polinomov [Batch symmetric fully homomorphic encryption using matrix polynomials]. Trudy Instituta sistemnogo programmirovanija RAN [The Proceedings of Institute for System Programming RAS] 26(5) pp 99–115.
[18] Burtyka F B and Makarevich O B 2014 Symmetric fully homomorphic encryption using decidable matrix equations. Proceedings of the 7th International Conference on Security of Information and Networks. ACM.
[19] Kaylor L and Offner D 2014 Counting matrices over a finite field with all eigenvalues in the field. Involve, a Journal of Mathematics. 7(5) pp 627–645.
[20] Osavsky G 2003 The Number of 2 by 2 Matrices over $Z/pZ$ with Eigenvalues in the Same Field. Mathematics magazine. 76(4) pp 314–317
[21] Gerstenhaber M 1961 On the number of nilpotent matrices with coefficients in a finite field Illinois Journal of Mathematics. 5 pp 330-333
[22] Higham N J and Kim H M 2001 Solving a quadratic matrix equation by Newton’s method with exact line searches. SIAM Journal on Matrix Analysis and Applications. 23(2) pp 303-316.
[23] Peichang G 2014 The Newton-Shamanskii method for solving a quadratic matrix equation arising in quasi-death problems arXiv preprint arXiv:1406.1075
[24] Fernando M and Pereira E 2010 A fixed point method to compute solvents of matrix polynomials Mathematica Bohemica 135(4) pp 355–362
[25] Seo J-H and Kim H M 2008 Solving matrix polynomials by Newton’s method with exact line searches Journal of the Korea Society for Industrial and Applied Mathematics 12(2) pp 55–68
[26] Von Zur Gathen J and Panario D 2001 Factoring polynomials over finite fields: A survey Journal of Symbolic Computation 31.1-2: pp 3–17.
[27] Aho A V and Hopcroft J E 1974 The design and analysis of computer algorithms. Pearson Education India
[28] Knauth D E 1975 Son of seminumerical algorithms. ACM SIGSAM Bulletin 9(4) pp 10–11
[29] Hart W and Novocin A 2011 Practical divide-and-conquer algorithms for polynomial arithmetic Proceedings of International Workshop on Computer Algebra in Scientific Computing, Springer Berlin Heidelberg

[30] Shoup V 2010 Number theory library (NTL) for C++ Available at Shoups homepage http://shoup.net/ntl

[31] Burtyka F B 2015 On complexity of Boolean matrix polynomials solving Matem. Mod. 27:7 pp 25–30 (In Russian)

[32] Lidl R and Niederreiter H 1997 Finite fields (Vol. 20). Cambridge university press.

[33] Lopez-Ruiz R 2004 An approach to the problem of generating irreducible polynomials over the finite field GF(2) and its relationship with the problem of periodicity on the space of binary sequences. arXiv preprint nlin/0401011 https://arxiv.org/pdf/nlin/0401011

[34] Gohberg I, Lancaster P and Rodman L 2005 Matrix polynomials (pp. 237–266). Birkhuser Basel.

[35] Kratz W and Stickel E 1987 Numerical solution of matrix polynomial equations by Newton’s method. IMA journal of numerical analysis, 7(3) pp 355-369.

[36] Higham N J, Kim H M 2000 Numerical analysis of a quadratic matrix equation. IMA Journal of Numerical Analysis, 20(4), pp 499–519.

[37] Ugalde E S 2015 Computation of invariant pairs and matrix solvents (Doctoral dissertation, Universit de Limoges).

[38] Burtyka F B 2017 Algorithms for computing solvents of unilateral second-order matrix polynomials over prime finite fields using lambda-matrices. Proceedings International conference on Computer Simulation in Physics and beyond (CSP2017)