Algorithms for computing solvents of unilateral second-order matrix polynomials over prime finite fields using lambda-matrices

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 considers algorithms for finding diagonalizable and non-diagonalizable roots (so called solvents) of monic arbitrary unilateral second-order matrix polynomial over prime finite field. These algorithms are based on polynomial matrices (lambda-matrices). This is an extension of existing general methods for computing solvents of matrix polynomials over field of complex numbers. We analyze how techniques for complex numbers can be adapted for finite field and estimate asymptotic complexity of the obtained algorithms.

1. Introduction
Let $\mathbb{K}$ be a field, $M_n(\mathbb{K})$ be the ring of $n \times n$ matrices over $\mathbb{K}$ (it can be, for example, the field of complex numbers $\mathbb{C}$ or finite field $\mathbb{F}_q$, where $q = p^f$, $p \in \mathbb{N}$ is prime). The unilateral matrix polynomial (UMP) of $n$-th order and $d$-th degree over $\mathbb{K}$ is the following expression:

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

where $F_i \in M_n(\mathbb{K})$, $0 \leq i \leq d$ are coefficients, $F_d \neq 0$, $X \in M_n(\mathbb{K})$ is variable.

Let $(M_n(\mathbb{K})[X])$ be the set of all UMP for given $n$ and $\mathbb{K}$. The root (or solvent) of UMP $\mathcal{F}(X)$ over $\mathbb{K}$ is a matrix $S \in M_n(\mathbb{K})$ such that $\mathcal{F}(S) = 0$ (we denote by $0$ the zero matrix of appropriate order). The solvents set of $\mathcal{F}$ we denote by $\text{Ker}(\mathcal{F}(X))$.

A Monic UMP (MUMP) is such UMP that $F_{\deg(\mathcal{F})} = I$, where $I$ is identity $n \times n$ matrix.

In this work we consider only MUMP, more particular left MUMP. Right UMP looks like $\mathcal{F}(X) = X^d + X^{d-1} \cdot F_{d-1} + \cdots + X^2 \cdot F_2 + X \cdot F_1 + F_0$. All below analysis is true and for them.

An important task for $\mathcal{F}(X)$ is computing $\text{Ker}(\mathcal{F}(X))$. Difficulties occur because of noncommutativity and the presence of zero divisors. Unlike scalar polynomials when $\mathbb{K}$ is $\mathbb{C}$ it can be true that $|\text{Ker}(\mathcal{F}(X))| = \infty$. And when $\mathbb{K}$ is $\mathbb{F}_p$ one may get $|\text{Ker}(\mathcal{F}(X))| > \deg(\mathcal{F}(X))$.

For example, solvents of $\mathcal{F}(X) = X^2 - \left( \begin{array}{cc} 0 & 0 \\ 0 & 0 \end{array} \right) \in (M_2(\mathbb{K})[X]$ are all nilpotent $2 \times 2$ matrices. So, for $\mathbb{K} = \mathbb{C}$ we have $|\text{Ker}(\mathcal{F}(X))| = \infty$, and for $\mathbb{K} = \mathbb{F}_p$ there is $|\text{Ker}(\mathcal{F}(X))| = p^2$.

In this paper we consider algorithms for computing solvents of $\mathcal{F}(X) \in M_2(\mathbb{F}_p)[X]$, where $p$ is prime, that belong to the class of matrices $\text{Eig}_2(\mathbb{F}_p) \subset M_2(\mathbb{F}_p)$. These matrices have a Jordan normal form (JNF) over $\mathbb{F}_p$. From the paper [11] we get $|\text{Eig}_2(\mathbb{F}_p)| = \frac{1}{2} \cdot p^4 + p^3 - \frac{1}{2} \cdot p^2$. Since $|M_2(\mathbb{F}_p)| = p^4$, we have $|\text{Eig}_2(\mathbb{F}_p)| > \frac{1}{2}|M_2(\mathbb{F}_p)|$. So, the class of held matrices is quite large.
Algorithms for computing solvents of MUMP over \( \mathbb{F}_p \) having arbitrary form and degree have not been considered in literature, even for \( n = 2 \). We undertook an analysis of existing works [2, 8, 4, 5] devoted to MUMP over \( \mathbb{C} \). These works describe a general approach to compute solvents. They use the following fact: any matrix in \( M_2(\mathbb{C}) \) has JNF. However this approach was not presented in algorithmic form. This paper analyzed how it can be generalized for \( \mathbb{F}_p \) and described the algorithms.

Our research has important practical application in cryptography. Complexity analysis of computing \( \text{Ker}(\mathcal{F}(X)) \) is necessary to estimate security level of one recently proposed homomorphic cryptosystem based on MUMP over \( \mathbb{F}_p \) [9, 10].

**The known works devoted to UMPs over \( \mathbb{C} \)**
The major part of works dealing with matrix polynomials is devoted to calculating solvents of UMP \( \mathcal{F}(X) \in (M_n(\mathbb{C}))[X] \). Several researchers [1, 2, 5, 8, 6] presented the result formalized in Theorem 1.

**Theorem 1.**
(i) There exist unresolvable UMPs \( \mathcal{F}(X) \in (M_n(\mathbb{C}))[X] \).
(ii) There exist UMPs \( \mathcal{F}(X) \in (M_n(\mathbb{C}))[X] \), having infinity number of solvents.
(iii) If \( \mathcal{F}(X) \in (M_n(\mathbb{C}))[X] \) has finite number of solvents, then \( |\text{Ker}(\mathcal{F}(X))| \leq C_{n,d} \).

This result was obtained using polynomial matrix theory [7] (see section 3). The detailed case study for second-order matrices was done in [4, 3].

In [8] 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 \( d, t \leq C_{2,d}^2 \) the \( d \)-th degree UMP exists with exactly \( t \) solvents. To achieve this result the author builds UMP having only diagonalizable (recall that a square matrix \( A \) is called diagonalizable if there exists an invertible matrix \( P \) such that \( P^{-1}AP \) is a diagonal matrix) solvents, and their number is 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, 12, 13, 14, 15] the authors propose how to adapt numerical methods for computing matrix polynomials solvents.

**2. Theoretical background**
For each \( \mathcal{F}(X) \in (M_n(\mathbb{F}_p))[X] \) there is \( \lambda \)-matrix \( \mathcal{F}(\lambda) \). We get it by replacing matrix variable \( X \in M_n(\mathbb{F}_p) \) by scalar variable \( \lambda \in \mathbb{F}_p \), i.e.

\[
\mathcal{F}(\lambda) = \lambda^d + F_{d-1} \cdot \lambda^{d-1} + \cdots + F_0 \in (M_n(\mathbb{F}_p[\lambda])).
\]

Below definitions were given in [8] for \( \mathbb{C} \). We rewrote them for \( \mathbb{F}_p \).

**Definition 1** [8]. Derivative of \( \mathcal{F}(\lambda) \) is \( \lambda \)-matrix \( \mathcal{F}'(\lambda) \) such that \( \mathcal{F}'(\lambda) = d \cdot I \cdot \lambda^{d-1} + (d - 1) \cdot F_{d-1} \cdot \lambda^{d-2} + \cdots + F_1 \).

**Definition 2** [8]. Eigenvalue (EVaL) of \( \mathcal{F}(\lambda) \) is such \( \alpha \in \mathbb{F}_p \) that \( \det(\mathcal{F}(\alpha)) = 0 \).

**Definition 3** [8]. Eigenvector (EVec) of \( \mathcal{F}(\lambda) \), corresponding to \( \text{EVec} \) \( \alpha \), is such \( \vec{v} \in \mathbb{F}_p^n \) that \( \mathcal{F}(\alpha) \cdot \vec{v} = 0 \).
ConstructEigenValues

In this section we consider solvents (3). Let $\ker D$.

4.1. Algorithm for computing diagonalizable solvents

Definition 4 ([8]). Let $\alpha \in \mathbb{F}_p$ be EVal and $\vec{v} \in \mathbb{F}^n_p$ be corresponding EVect of $F(\lambda)$. The pair $(\alpha, \vec{v})$ is eigenpair (EP) of $F(\lambda)$.

Definition 5 ([8]). Let there be $F(\lambda) \in (M_2(\mathbb{F}_p))[\lambda]$ having EVal $\alpha \in \mathbb{F}_p$. Vectors $\vec{v}_1, \vec{v}_2 \in \mathbb{F}^n_p$, $\vec{v}_1 \neq \vec{0}$ satisfying the system

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

are called a Jordan chain of length 2 $F(\lambda)$ corresponding to EVal $\alpha$.

3. The general form of solvents, having JNF

According to analysis done for $C \in \mathbb{C}$, we came to conclusion that there are two types of solvents of $F(X) \in (M_2(\mathbb{F}_p))[X]$ from class $Eig_2(\mathbb{F}_p)$:

- **Diagonalizable solvents**
  \[ S = V \cdot \begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \cdot V^{-1}, \]  
  where $V = [\vec{v}_1 \ \vec{v}_2] \in M_2(\mathbb{F}_p)$ – matrix with columns $\vec{v}_1$ and $\vec{v}_2$ respectively, $(\alpha_i, \vec{v}_i), i = 1, 2$ – EP of $F(\lambda)$, $\vec{v}_1, \vec{v}_2$ are linear independent.

- **Non-diagonalizable solvents**
  \[ S = V \cdot \begin{pmatrix} \alpha & 1 \\ 0 & \alpha \end{pmatrix} \cdot V^{-1}, \]  
  where $V = [\vec{v}_1 \ \vec{v}_2] \in M_2(\mathbb{F}_p)$, $\vec{v}_1, \vec{v}_2$ – Jordan chain $F(\lambda)$ satisfying (2), EVal $\alpha \in \mathbb{F}_p$ has algebraic multiplicity $\geq 2$, $\vec{v}_1, \vec{v}_2$ are linear independent.

4. Algorithms of computing solvents of MUMP based on $\lambda$-matrices

4.1. Algorithm for computing diagonalizable solvents

In this section we consider solvents (3). Let $\ker_D(F(X))$ be the set of all such solvents. We need algorithm ConstructEigenValues, computing all EVal of $F(\lambda)$. It uses algorithm FindRoots, whose input is $d(\lambda) \in \mathbb{F}_p[\lambda]$. It computes all different roots of $d(\lambda)$.

**Algorithm 1: ConstructEigenValues($F(\lambda)$)**

| Input: $\lambda$-matrix $F(\lambda) \in (M_2(\mathbb{F}_p))[\lambda]$ | Output: $A = \{\alpha_1, ..., \alpha_r\}, \alpha_i \in \mathbb{F}_p, \alpha_i \neq \alpha_j$ |
|---|---|
| 1 $d(\lambda) := \text{det}(F(\lambda))$; | 2 $A := \text{FindRoots}(d(\lambda))$; |
| 3 return $A$; | |

Further we need to compute $\ker(F(\alpha_i))$ for $\forall \alpha_i \in A$. For $M_2(\mathbb{F}_p)$ there are only two cases: 1) $\ker(F(\alpha_i)) = \mathbb{F}_p^2$, 2) $\ker(F(\alpha_i)) = \text{Lin}(\vec{v}_i)$, where $\text{Lin}(\vec{v}_i)$ is a linear subspace with basis $\vec{v}_i \in \mathbb{F}_2^2 \setminus \{\vec{0}\}$. We divide eigenvalues into two types in accordance with dim($\ker(F(\alpha_i))$). It helps us to build algorithm for finding $\ker_D(F(X))$ in such a way that all repetitive solvents will be excluded. It will decrease the complexity.

Algorithm ConstructEigenPairs computes EP of $F(\lambda)$. Its output is a set $A_0 = \{\alpha_{0,1}, ..., \alpha_{0,t_0}\} \subseteq A$, $\alpha_{0,i} \neq \alpha_{0,j}$, containing all EVal such that dim($\ker(F(\alpha_{0,i}))$) = 2. Also it computes pairs $(\vec{v}_i, A_1), ..., (\vec{v}_j, A_s)$, dim($\text{Lin}(\vec{v}_i, \vec{v}_j)$) = 2. The set $A_k = \{\alpha_{k,1}, ..., \alpha_{k,t_k}\} \subseteq A$, $\alpha_{k,i} \neq \alpha_{k,j}$ consists of all EVal such that $\ker(F(\alpha_{k,i})) = \text{Lin}(\vec{v}_k)$. Sets $A_0, A_1, ..., A_s$ haven’t any intersection and $\bigcup_{k=0}^{s} A_k = A$. 

3
Lemma 1. For any invertible matrices $V \in M_2(\mathbb{F}_p)$, $V = [\bar{v}_1 \; \bar{v}_2]$ and nonzero constants $\beta, \gamma \in \mathbb{F}_p$ we have $V \cdot \text{diag}(\alpha_1, \alpha_2) \cdot V^{-1} = V' \cdot \text{diag}(\alpha_1, \alpha_2) \cdot V'^{-1}$ where $V' = [\beta \cdot \bar{v}_1 \; \gamma \cdot \bar{v}_2]$.

Algorithm 2: ConstructEigenPairs$_2(\mathcal{F}(\lambda))$

Input: $\lambda$-matrix $\mathcal{F}(\lambda) \in M_2(\mathbb{F}_p)[\lambda]$  
Output: Set of pairs $\{(\cdot, \mathcal{A}_0), (\tilde{v}_1, \mathcal{A}_1), \ldots, (\tilde{v}_s, \mathcal{A}_s)\}$

1. $\mathcal{P} := \emptyset$;  $\mathcal{A}_0 := \emptyset$;
2. $\{\alpha_1, \ldots, \alpha_r\} := \text{ConstructEigenValues}(\mathcal{F}(\lambda))$;
3. for $i = 1$ to $r$ do
   4. if $(\mathcal{F}(\alpha_i) = 0)$ then
   5. $\mathcal{A}_0 := \mathcal{A}_0 \cup \{\alpha_i\}$;
   6. end
   7. else
   8. Compute $\tilde{v}_i$, such that $\ker(\mathcal{F}(\alpha_i)) = \text{Lin}(\tilde{v}_i)$;
   9. foreach $(\tilde{v}, \Lambda) \in \mathcal{P}$ do
   10. if $\text{Lin}(\tilde{v}, \tilde{v}_i) = 1$ then
   11. $\Lambda := \Lambda \cup \{\alpha_i\}$;
   12. goto 3;
   13. end
   14. $\mathcal{P} := \mathcal{P} \cup \{(\tilde{v}_i, \{\alpha_i\})\}$;
   15. end
16. end
17. if $|\mathcal{A}_0| > 0$ then
18. $\mathcal{P} := \mathcal{P} \cup \{(\cdot, \mathcal{A}_0)\}$
19. end
20. return $\mathcal{P}$;

According to Lemma 1 one should not use all EP $(\alpha_{0,i}, \tilde{v})$, $\tilde{v} \in \mathbb{F}_p^2$ for constructing solvents (3). Only some subset is enough to exclude repetitions.

Lemma 2. Let $\tilde{e}_1, \tilde{e}_2 \in \mathbb{F}_p^2$ be a basis. $\mathbb{F}_p^2$ is a union of disjoint subspaces (excluding $\bar{0}$):

$$\mathbb{F}_p^2 = \text{Lin}(\tilde{e}_1') \cup \text{Lin}(\tilde{e}_2') \cup \text{Lin}(\tilde{e}_3') \ldots \cup \text{Lin}(\tilde{e}_{p+1}')$$

$\tilde{e}_1' = \tilde{e}_1, \tilde{e}_2' = \tilde{e}_2, \tilde{e}_3' = \tilde{e}_1 + \tilde{e}_2, \tilde{e}_4' = \tilde{e}_1 + 2 \cdot \tilde{e}_2, \ldots, \tilde{e}_{p+1}' = \tilde{e}_1 + (p - 1) \cdot \tilde{e}_2.$

To exclude repetitions during computing by formula (3) we should use only EP $(\alpha_{0,k}, \tilde{e}_i')$, $1 \leq i \leq p+1$. Similarly we assume that in pair $(\tilde{v}_i, \mathcal{A}_i)$ there is $\tilde{v}_i \in \mathcal{E}$, where $\mathcal{E} = \{\tilde{e}_1', \ldots, \tilde{e}_{p+1}'\}$. Finally we get algorithm ConstructAllDiagonalizableSolvents, computing $\ker_D(\mathcal{F}(X))$.

The complexity of ConstructAllDiagonalizableSolvents is $O(d \cdot (\log d + p) + |\ker_D(\mathcal{F}(X))|)$. $O(d \cdot \log d)$ is complexity of computing det$(\mathcal{F}(X))$, $O(d \cdot p)$ is complexity of finding EVal $\mathcal{F}(\lambda)$, using deterministic procedure, he number of diagonalizable matrices in $Eig_2(\mathbb{F}_p)$ is $\mathbb{F}_p^2/(p^2-1) + p$. Then in the worst case the complexity is $O(d \cdot (\log d + p) + p^3)$. Also one can use probabilistic algorithm to find all EVal - then complexity in average is $\approx O(d \cdot (\log d + \log p) + |\ker_D(\mathcal{F}(X))|).$
Algorithm 3: ConstructAllDiagonalizableSolvents₂(\(\mathcal{F}(X)\))

**Input:** \(\mathcal{F}(X)\)  
**Output:** \(\ker_D(\mathcal{F}(X))\)

1. \(D := \emptyset\);
2. \((\cdot, A_0), (\vec{v}_1, A_1), \ldots, (\vec{v}_s, A_s) := \text{ConstructEigenPairs}(\mathcal{F}(\lambda))\);
3. foreach \((i, j), \text{such that } 0 \leq i < j \leq s\) do
   4. foreach \(\alpha \in A_i, \beta \in A_j\) do
      5. if \(\alpha = \beta\) then
         6. \(D := D \cup \{\text{diag}(\alpha, \beta)\}\);
      7. else if \(i = 0\) then
         8. if \(j = 0\) then
            9. foreach \(\vec{b}_1, \vec{b}_2 \in E, \text{such that } \vec{b}_1 \neq \vec{b}_2\) do
               10. \(V = [\vec{b}_1 \ \vec{b}_2]\);
               11. \(D := D \cup \{V \cdot D \cdot V^{-1}\}\);
            12. end
            13. end
            14. else
               15. foreach \(\vec{b} \in E\) do
                  16. \(V = [\vec{b} \ \vec{v}_j]\);
                  17. \(D := D \cup \{V \cdot D \cdot V^{-1}\}\);
               18. end
               19. end
            20. end
         9. else
            10. \(V = [\vec{v}_i \ \vec{v}_j]\);
            11. \(D := D \cup \{V \cdot D \cdot V^{-1}\}\);
         12. end
      7. end
   4. end
   5. return \(D\);

4.2. Algorithm for non-diagonalizable solvents

Now we consider the algorithm for finding solvents (4). The set of all these solvents we denote by \(\ker_{\mathcal{N}}(\mathcal{F}(X))\) and let \(\ker_{\alpha, \mathcal{N}}(\mathcal{F}(X))\) be a subset of such solvents for fixed EVal \(\alpha\).

Computing of EVal is similar to Algorithm 1, but it takes into account the algebraic multiplicity of each EVal. The result is truncated set \(\mathcal{A}' = \{\alpha'_1, \ldots, \alpha'_m\} \subseteq \mathcal{A}\) of EVal, having multiplicity \(> 1\).

Let’s fix some \(\alpha \in \mathcal{A}'\) and consider how to solve (2). We have the following cases.

- If \(\mathcal{F}(\alpha) = 0, \mathcal{F}'(\alpha) = 0\), then solvents are all matrices of the form \(S = N + \text{diag}(\alpha, \alpha)\), where \(N \in M_2(\mathbb{F}_p)\) is arbitrary nilpotent matrix. \(|\ker_{\alpha, \mathcal{N}}(\mathcal{F}(X))| = p^2\).
- If \(\mathcal{F}(\alpha) \neq 0, \mathcal{F}'(\alpha) = 0\) then \(|\ker_{\alpha, \mathcal{N}}(\mathcal{F}(X))| = 0\).
- If \(\mathcal{F}(\alpha) = 0, \mathcal{F}'(\alpha) \neq 0\) then we have two cases:
  - a) if \(\det(\mathcal{F}'(\alpha)) \neq 0\) then \(|\ker_{\alpha, \mathcal{N}}(\mathcal{F}(X))| = 0\);
  - b) if \(\det(\mathcal{F}'(\alpha)) = 0\) then for \(\vec{v}_1 \in \ker(\mathcal{F}'(\alpha))\) the second equation of (2) becomes an identity \(0 \cdot \vec{v}_2 = 0\). Then \(V\) can be constructed using \(\vec{v}_1 \in \ker(\mathcal{F}'(\alpha)) \setminus \{\vec{0}\}, \vec{v}_2 \in \mathbb{F}_p^2 \setminus \ker(\mathcal{F}'(\alpha))\).
and we obtain \(|\ker_{\Lambda^N}(F(X))| < p^2\).

- If \(F(\alpha) \neq 0, F'(\alpha) \neq 0\) then \(V\) consists of \(\vec{v}_1 \in \ker(F'(\alpha))\) and \(\vec{v}_2\) that is a solution of a linear system \(F(\alpha) \cdot \vec{x} = -F'(\alpha) \cdot \vec{v}_1\) and we obtain \(|\ker_{\Lambda^N}(F(X))| < p^2\).

The above is summarized into algorithm ConstructNonDiagonalizableSolvents for computing \(\ker_{\Lambda^N}(F(X))\). Its complexity in the worst case is \(O(d \cdot (\log d + \log p) + p^3 \cdot d)\).

### Algorithm 4: ConstructNonDiagonalizableSolvents\(_2(F(X))\)

**Input:** \(\lambda\)-matrix \(F(\lambda) \in M_2(\mathbb{F}_p)[\lambda]\)

**Output:** \(\ker_{\Lambda^N}(F(X))\)

1. \(\mathcal{N} := \emptyset;\)
2. \(\mathcal{A}' = \text{ConstructEigenValuesOfMultiplicityGreaterThanOne}(F(\lambda));\)
3. foreach \(\alpha \in \mathcal{A}'\) do
   4. \(J := \begin{pmatrix} \alpha & 1 \\ 0 & \alpha \end{pmatrix};\)
   5. if \(F(\alpha) = 0\) and \(F'(\alpha) = 0\) then
      6. foreach \(N \in M_2(\mathbb{F}_p), \text{ such that } N^2 = 0\) do
         7. \(\mathcal{N} := \mathcal{N} \cup \{N + \text{diag}(\alpha, \alpha)\};\)
      end
   end
   10. if \(F(\alpha) = 0\) and \(F'(\alpha) \neq 0\) and \(\det(F'(\alpha)) = 0\) then
      11. foreach \((\vec{v}_1, \vec{v}_2), \vec{v}_1 \in \ker(F'(\alpha)) \setminus \{\vec{0}\}, \vec{v}_2 \in \mathbb{F}_p^2 \setminus \ker(F'(\alpha))\) do
         12. \(V = [\vec{v}_1 \ \vec{v}_2];\)
         13. \(\mathcal{N} := \mathcal{N} \cup \{V \cdot J \cdot V^{-1}\};\)
      end
   end
   15. if \(F(\alpha) \neq 0\) and \(F'(\alpha) \neq 0\) then
      16. foreach \(\vec{v}_1 \in \ker(F'(\alpha)) \setminus \{\vec{0}\}, \vec{v}_2, \text{ such that } F(\alpha) \cdot \vec{v}_2 = -F'(\alpha) \cdot \vec{v}_1\) do
         17. if \(\dim(\text{Lin}(\vec{v}_1, \vec{v}_2)) = 2\) then
            18. \(V = [\vec{v}_1 \ \vec{v}_2];\)
            19. \(\mathcal{N} := \mathcal{N} \cup \{V \cdot J \cdot V^{-1}\};\)
         end
      end
   end
23. end
24. end
25. return \(\mathcal{N}\); 

## 5. Conclusion
In this work the study was done of how existing techniques and methods for finding of UMP’s solvents over \(\mathbb{C}\) using \(\lambda\)-matrices can be adapted for the prime finite field \(\mathbb{F}_p\). On the basis of this analysis we described algorithms for computing solvents of \(\text{MUMP} \in (M_2(\mathbb{F}_p))[X]\) and estimated their asymptotical complexity.

### Acknowledgments
The work was supported by the RFBR grant no. 16-37-00125 mol.a. The author is also very grateful to Professor V. P. Gerdt (LIT, JINR, Dubna), for participating in the fate of this work.
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) 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) 523–533.
[3] Gelfand S I 2004 O chisle reshenij kvadratnogo uravnenija [On the number of quadratic equation’s solutions] Obshhematematicheskij seminar “Globus”. [All-mathematical workshop “Globus”]. 1 124–133 (In Russian)
[4] Wilson R L 1999 Polynomial equations over matrices. Rutgers University preprints. https://pdfs.semanticscholar.org/57bf/9e36f6bc8ed7648b70d1ff7b1fa80d024f23.pdf
[5] Slusky M 2010 Zeros of 2 × 2 matrix polynomials. Communications in Algebra. 38(11) 4212–4223.
[6] 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) 1902–1911.
[7] Gantmakher F 1959 The theory of matrices – Volume 1. Chelsea, New York.
[8] Pereira E 2003 On solvents of matrix polynomials. Applied numerical mathematics. 47 197–208.
[9] 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 the Institute for System Programming RAS] 26(5) 99–115. (In Russian)
[10] 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.
[11] 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) 627–645.
[12] 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) 303-316.
[13] Peichang G 2014 The Newton-Shamanskii method for solving a quadratic matrix equation arising in quasi-birth-death problems arXiv preprint arXiv:1406.1075.
[14] Fernando M and Pereira E 2010 A fixed point method to compute solvents of matrix polynomials Mathematica Bohemica 135(4) 355-362.
[15] Seo Jong-Hyeon and Hyun-Min Kim 2008 Solving matrix polynomials by Newton’s method with exact line searches Journal of the Korea Society for Industrial and Applied Mathematics 12(2) 55-68.