

TDK dolgozat

Borus Gergő Gyula

DEBRECENI EGYETEM
INFORMATIKAI KAR

Véges testek feletti
egyenletrendszereket megoldó
algoritmusok implementációja

Témavezető:

DR. HORVÁTH GÁBOR

egyetemi adjunktus

Készítette:

BORUS GERGŐ GYULA

alkalmazott matematikus MSc

II. évfolyam

Debrecen

2012

Tartalomjegyzék

1. Bevezetés	1
2. Singular	2
3. Algoritmusok	3
3.1. simplesolver	3
3.2. GBsolve	6
3.3. PEsolve	7
3.4. ZZsolve	10
3.5. XLsolve	12
3.6. ffsolve	14
4. Összefoglalás	15
A. $\mathbb{F}_{64}$ két logaritmus táblázata	19
Saját munka leírása	21

1. Bevezetés

Különböző struktúrák feletti egyenletek és egyenletrendszerek megoldása az algebra egyik legősibb problémái közé tartozik. Napjainkban egyre gyakrabban kell véges testek feletti egyenletrendszereket megoldani, elsősorban a kriptográfia, a kombinatorika vagy a hibajavító kódok elméletében. Kriptográfiában például a blokktitkosítók algebrai kriptanalízise során egy megfelelő véges test feletti polinomiális egyenletrendszer megoldásaiból kapjuk a titkos kulcsot. Ezzel a módszerrel vizsgálták többek közt a jól ismert DES [4], HFE [11] és AES [16] kriptorendszereket, ez utóbbit például 8000 másodfokú egyenletre írták át 1600 változóban [6]. Ezen kriptográfiai protokollok mindegyike azt a tényt használja ki, hogy egy test feletti polinomegyenlet-rendszerek megoldhatóságának eldöntése NP-teljes probléma [12]. Napjainkban a fejlett algoritmust tartalmazó komputer algebrai programcsomagok a probléma megoldására (a kimerítő keresésen kívül) csak a Gröbner bázisok elméletének alkalmazásával adnak lehetőséget. Az elmúlt évtized során azonban több, a véges testek feletti egyenletrendszereket megoldó eljárás látott napvilágot. A témához kapcsolódó szakirodalom áttanulmányozása után kiválasztottam néhány, a gyakorlati életben jól alkalmazható eljárást. Ezen algoritmusokat – a matematikai hátterük vizsgálata és megértése után – implementáltam napjaink egy aktívan fejlesztett ingyenes komputer algebrai programcsomagjára.

A dolgozatban bemutatásra kerülő algoritmusokat a szabadon hozzáférhető (GNU GPL licencű) SINGULAR komputer algebra rendszerben programoztam le. A polinom számításokra specializált SINGULAR – hasonlóan például a csoportokra specializált GAP programcsomaghoz – a beépített algoritmusokon túl függvénykönyvtárakon keresztül bővíthető. Ez a rendszer tartalmazza a legtöbb fejlett algoritmust a kommutatív algebra és az algebrai geometria területéről. A SINGULAR csomag többféle egyenletrendszer megoldó eljárást is tartalmaz, ám ezek mindegyike csak 0 karakterisztikában működik, így véges testekre nem alkalmazhatóak. A SINGULAR hivatalos fejlesztői csoportja nyitott problémát írt ki a prím elemszámú testek feletti megoldó algoritmusok implementálására *Solve polynomial systems over $\mathbb{Z}/p$* címmel [17]. Az általam implementált algoritmusok nemcsak $\mathbb{Z}/p$, hanem tetszőleges véges test felett működnek.

A dolgozat elején a 2. fejezetben bemutatom a SINGULAR komputer algebra rendszert. Ezután a 3. fejezetben részletesen ismertetem az implementált algoritmusokat. Mindegyik

algoritmust egy szemléletes példa is követ. Elsőként a 3.1. szakaszban a kimerítő keresést tárgyalom. A 3.2. szakaszban egy, a Gröbner bázisok elméletét felhasználó algoritmust ismertetek. A 3.3. szakaszban polinomok ekvivalenciájára épülő eljárást mutatok be, mely az egyenletrendszerből készít egy többváltozós polinomot, majd annak vizsgálatával határozza meg a gyököket. Ezután 3.4. szakaszban olyan algoritmust mutatok be, mely egy nagyobb testbeli egyváltozós polinom gyökeinek meghatározására vezeti vissza a problémát. A 3.5 részben egy linearizációs technikán alapuló megoldót mutatok be, mely főleg másodfokú egyenletrendszerek esetén előnyös. Végül a 3.6. szakaszban bemutatom a fenti algoritmusokat összefogó saját fejlesztésű megoldó frontend programomat. A függvénykönyvtár forráskódját a SINGULAR telepítőjével együtt megtalálható a mellékelt CD-n, vagy letölthető a <http://web.unideb.hu/~borisz/ffsolve/> oldalról [2]. A forráskód a SINGULAR könyvtárak követelményei szerint készült, beépített dokumentációval rendelkezik.

2. Singular

A SINGULAR a TU Kaiserslautern által fejlesztett, polinom számításokra specializált komputer algebrai programcsomag [8]. A program a GNU GPL licenc alatt érhető el, így bárki hozzáférhet a programhoz és annak forráskódjához. A program fejlett algoritmusokat biztosít többek közt a kommutatív és nem-kommutatív algebra, algebrai geometria, Gröbner-bázisok és a több változós polinomok faktorizálásának területén.

A SINGULAR-ban lehetőségünk van különféle gyűrűk ($\mathbb{Z}$, $\mathbb{Z}(a, \dots)$, $\mathbb{Z}_m$) és testek ($\mathbb{Z}_p$, $\mathbb{F}_q$, $\mathbb{Q}$, $\mathbb{Q}(a, \dots)$, $\mathbb{R}$, $\mathbb{C}$) felett definiálni polinomgyűrűket, valamint ilyenek hányadosgyűrűit is. Polinomgyűrűk definiálására több lehetőségünk is van. Prím elemű testek esetén elegendő megadnunk a karakterisztikát, a változók listáját és a monomrendezést. Prímhatvány elemű test definiálására két lehetőségünk is van. Az első esetben a karakterisztika helyett a kívánt véges test elemszámát és a bővítéshez használt paraméter nevét kell megadnunk. A SINGULAR kiválasztja a megfelelő Conway-polinomot az általunk megadott paraméterben [13]. A másik lehetőség, hogy mi magunk adjuk meg a bővítéshez használt irreducibilis polinomot. Ennek a módszernek az előnye, hogy az így megadott gyűrűben működik csak véges test felett a beépített polinomfaktorizáló eljárás.

A SINGULAR rendszerben polinomok tárolására a *poly* típus, ideálokra az *ideal* adat-

struktúrája szolgál. Az *ideal* típus az ideált generátorainak listájaként tárolja. A rendszer kényelmes eszközöket biztosít a rajtuk való műveletek elvégzéséhez. Az ideálokat és polinomokat könnyen redukálhatjuk más ideálok vagy polinom szerint. Egy $\mathbb{F}_q[x_1, \dots, x_n]$ gyűrűben az $x_i - a_i$ polinommal való redukálás az $x_i = a_i$ helyettesítésnek felel meg. Ezek miatt érdemes az egyenletrendszereket és a megoldásokat is *idealként* tárolni. Az ilyen megoldásideálok $x_i - a_i$ ($i \in \{1, \dots, n\}$) alakú egyenletekből állnak, ahol $(a_1, \dots, a_n) \in \mathbb{F}_q^n$ az egyenletrendszer egy gyöke.

A SINGULAR program sokféle polinomrendszer-megoldó algoritmus implementációját tartalmazza, azonban ezek csak 0 karakterisztika esetén működnek, véges karakterisztika esetén nem. E rés befoltozására implementáltam a dolgozatban szereplő algoritmusokat.

3. Algoritmusok

Következő részekben bemutatam az általam implementált algoritmusokat. A 3.1. szakaszban a *simplesolver*, a 3.2. szakaszban a *GBsolve*, a 3.3. szakaszban a *PEsolve*, a 3.4. szakaszban a *ZZsolve*, 3.5. szakaszban az *XLsolve* algoritmust. Az algoritmusok működését példával is szemléltetem. A példában $\mathbb{F}_4 = \{0, 1, \alpha, \alpha^2\}$ a négyelemű test, ahol $\alpha^2 + \alpha + 1 = 0$ és az $\mathbb{F}_4[x_1, x_2, x_3]$ gyűrűben keressük az

$$I = (x_1^2x_2 + \alpha x_1x_2^2 + \alpha^2, x_1^2x_2x_3^2 + \alpha x_1, \alpha^2x_1x_3 + \alpha^2)$$

ideál polinomjainak közös gyökeit.

3.1. simplesolver

A *simplesolver* egy kimerítő keresésen alapuló megoldáskereső algoritmus. A kimerítő keresés azon az elven működik, hogy minden lehetséges megoldás n -est kipróbál. Egy q elemű test feletti n változós egyenletrendszer esetén ez q^n behelyettesítést jelent. Azt az ötletet felhasználva javítottam a kimerítő keresésen, hogy egy egyváltozós egyenletet viszonylag könnyen meg tudunk oldani (pl. Berlekamp-algoritmussal [1]), emiatt a *simplesolver* a megoldandó egyenletrendszerben keres egy egyváltozós egyenletet, majd ennek lineáris tényezőit felhasználva egyszerűsíti az egyenletrendszert. Minden ilyen lépés után a lineáris tényezők számával megegyező új egyenletrendszert kapunk, azonban ezekben már a

változók száma eggyel kevesebb.

Annak érdekében, hogy mindig találjunk egyváltozós egyenletet és ennek mindig legyen legalább egy lineáris tényezője, az egyenletrendszerhez hozzávesszük az alaptest $x_i^q - x_i$ alakú testegyenleteit. Ezen egyenleteknek minden testelem gyöke. Emiatt ha más egyváltozós egyenlet nem szerepel a rendszerben, akkor az adott változó helyébe az összes lehetséges értéket behelyettesítjük. Tehát legrosszabb esetben q^n helyettesítést végzünk.

Algoritmus.

Bemenet: E , az egyenletrendszer, mint ideál.

Kimenet: S , a megoldásideálokat tartalmazó lista.

1. E -hez hozzávesszük az $x_i^q - x_i$ alakú egyenleteket minden $i = 1 \dots n$ -re. Legyen S' egy üres ideálból álló halmaz.
2. Minden $I \in S'$ esetén hajtsuk végre a következő lépéseket:
 - a) redukáljuk E -t I -vel, dobjuk el az azonosan 0 polinomokat és jelölje ezt az új egyenletrendszert E' ,
 - b) legyen f az E' egy egyváltozós polinomja, lineáris tényezőit jelölje $f_1, \dots, f_r$,
 - c) álljon S_I azon $J = I \cup f_i$ ($i = 1, \dots, r$) ideálokból, hogy E redukálva J -vel nem tartalmaz 0-ad fokú polinomot.
3. Legyen ezután $S' = \bigcup_{I \in S'} S_I$.
4. Ha $S' \neq \emptyset$ és még van olyan változó, amire nem kerestünk megoldást, akkor lépünk vissza a 2. lépéshez.
5. Legyen $S = S'$.

Példa. Legyen $\mathbb{F}_4 = \{0, 1, \alpha, \alpha^2\}$ a négyelemű test, ahol $\alpha^2 + \alpha + 1 = 0$. Tekintsük az $\mathbb{F}_4[x_1, x_2, x_3]$ gyűrűt lexikografikus monomrendezés mellett és az

$$I = (x_1^2x_2 + \alpha x_1x_2^2 + \alpha^2, x_1^2x_2x_3^2 + \alpha x_1, \alpha^2x_1x_3 + \alpha^2x_1)$$

ideált. Első lépésben az ideál generátoraihoz hozzávesszük az $x_i^4 - x_i$ alakú polinomokat, így az

$$\left(x_1^2x_2 + \alpha x_1x_2^2 + \alpha^2, x_1^2x_2x_3^2 + \alpha x_1, \alpha^2x_1x_3 + \alpha^2x_1, x_3^4 - x_3, x_2^4 - x_2, x_1^4 - x_1\right)$$

ideált kapjuk. Kiválasztjuk az első egyváltozós polinomot és faktorizáljuk:

$$x_3^4 - x_3 = x_3(x_3 - 1)(x_3 - \alpha)(x_3 - \alpha^2).$$

A polinom gyökeit behelyettesítjük, majd a kapott rendszerre ismételjük meg a fenti lépéseket. A négy gyök behelyettesítése után az alábbi ideálokat kapjuk:

$$\begin{aligned} I_{*,*,0} &= \left(x_1^2x_2 + \alpha x_1x_2^2 + \alpha^2, \alpha x_1, \alpha^2x_1, x_2^4 - x_2, x_1^4 - x_1\right), \\ I_{*,*,1} &= \left(x_1^2x_2 + \alpha x_1x_2^2 + \alpha^2, x_1^2x_2 + \alpha x_1, x_2^4 - x_2, x_1^4 - x_1\right), \\ I_{*,*,\alpha} &= \left(x_1^2x_2 + \alpha x_1x_2^2 + \alpha^2, \alpha^2x_1^2x_2 + \alpha x_1, \alpha x_1, x_2^4 - x_2, x_1^4 - x_1\right), \\ I_{*,*,\alpha^2} &= \left(x_1^2x_2 + \alpha x_1x_2^2 + \alpha^2, \alpha x_1^2x_2 + \alpha x_1, x_1, x_2^4 - x_2, x_1^4 - x_1\right). \end{aligned}$$

Ezután mindegyik ideál generátorai közül kiválasztjuk a legalacsonyabb fokú egyváltozós polinomot. Az $I_{*,*,0}$ ideál esetén ez αx_1 . Ennek egyetlen gyökét behelyettesítve az $(\alpha^2, x_2^4 - x_2)$ ideált kapjuk. Innen látható, hogy nincs olyan közös gyöke az eredeti polinomoknak, melyben $x_3 = 0$. Az $I_{*,*,1}$ ideálból az $x_2^4 - x_2 = x_2(x_2 - 1)(x_2 - \alpha)(x_2 - \alpha^2)$ polinom gyökeit visszahelyettesítve a következő ideálokat kapjuk:

$$\begin{aligned} I_{*,0,1} &= \left(\alpha^2, \alpha x_1, x_1^4 - x_1\right), \\ I_{*,1,1} &= \left(x_1^2 + \alpha x_1 + \alpha^2, x_1^2 + \alpha x_1, x_1^4 - x_1\right), \\ I_{*,\alpha,1} &= \left(\alpha x_1^2 + x_1 + \alpha^2, \alpha x_1^2 + \alpha x_1, x_1^4 - x_1\right), \\ I_{*,\alpha^2,1} &= \left(\alpha^2x_1^2 + \alpha^2x_1 + \alpha^2, \alpha^2x_1^2 + \alpha x_1, x_1^4 - x_1\right). \end{aligned}$$

Az $I_{*,0,1}$ ideálnál láthatóan nincs közös gyök. Az $I_{*,1,1}$ ideálból az $x_1^2 + \alpha x_1 = x_1(x_1 + \alpha)$ polinom mindkét gyökének behelyettesítése után az (α^2) ideált kapjuk, tehát egyik sem gyök. Az $I_{*,\alpha,1}$ ideálnál az első polinomot választjuk: $\alpha x_1^2 + x_1 + \alpha^2 = \alpha(x_1 + \alpha)(x_1 + 1)$. Az $x_1 = \alpha$ helyettesítésekor az (α) ideált kapjuk, ilyenkor nincs közös gyök. Az $x_1 = 1$ után viszont az (0) ideált kapjuk, azaz $(x_1, x_2, x_3) = (1, \alpha, 1)$ egy közös gyöke az eredeti

polinomrendszernek. Végül az $I_{*,\alpha^2,1}$ ideálból az $\alpha^2 x_1^2 + \alpha x_1 = x_1(x_1 - \alpha^2)$ első gyökének a behelyettesítése után az (α^2) ideál adódik, így ez nem gyök. A második gyök helyettesítése után a (0) ideált kaptuk, tehát megtaláltunk az $(x_1, x_2, x_3) = (\alpha^2, \alpha^2, 1)$ gyököt.

Az $x_3 = \alpha$ illetve $x_3 = \alpha^2$ helyettesítésekhez tartozó ideálokból az αx_1 illetve x_1 polinomokat választjuk. Ezek behelyettesítése után mindkét esetben az $(\alpha^2, x_2^4 - x_2)$ ideált kapjuk, tehát ezekben az esetekben nincs közös gyök. Ezzel ellenőriztük az összes szükséges helyettesítést és megtaláltuk az összes közös gyököt: $(1, \alpha, 1)$ és $(\alpha^2, \alpha^2, 1)$.

3.2. GBSolve

A GBSolve egy Gröbner-bázis számításra épülő megoldóeljárás.

Egy Gröbner-bázisokat használó eljárás az alábbi módon működik. Először kiszámítjuk az egyenletrendszer által generált ideál redukált Gröbner-bázisát pl. a SINGULAR *std* parancsával. Amennyiben az ideál nulla dimenziós, akkor a kapott bázis tartalmazni fog egyváltozós polinomot. Ezt megoldjuk, majd ennek gyökeit visszahelyettesítve egy olyan egyenletrendszert kapunk, amiben a változók száma eggyel kevesebb és szintén tartalmaz egyváltozós polinomot. Ezt az eljárás ismételve megkapjuk az egyenletrendszer megoldásait [7]. A GBSolve ennél speciálisabb módszert használ, faktorizált Gröbner-bázist számol. Ennél a módszernél a köztes lépésekkor megpróbáljuk faktorizálni a bázisba belépő polinomokat. Ha ez sikerül, akkor annyi új részproblémánk lesz, ahány tényezőt sikerült találnunk. Faktorizált Gröbner-bázist a SINGULAR *facstd* parancsával tudunk számolni.

Példa. Legyen $\mathbb{F}_4 = \{0, 1, \alpha, \alpha^2\}$ a négyelemű test, ahol $\alpha^2 + \alpha + 1 = 0$. Tekintsük az $\mathbb{F}_4[x_1, x_2, x_3]$ gyűrűt lexikografikus monomrendezés mellett és az

$$(x_1^2 x_2 + \alpha x_1 x_2^2 + \alpha^2, x_1^2 x_2 x_3^2 + \alpha x_1, \alpha^2 x_1 x_3 + \alpha^2 x_1)$$

ideált. Az ideál Gröbner-bázisa az $(x_3 + 1, x_2^2 + x_2 + 1, x_1 + \alpha x_2^2)$ polinomokból áll. A kapott bázis olyan tulajdonságú, hogy az első polinom csak x_n -től függ, a következő polinom a bázisban már függ x_{n-1} -től és x_n -től is, és így tovább. Ezt felhasználva a bázis első polinomjának gyökeit behelyettesítjük a többi polinomba. Ezzel olyan polinomrendszerekhez jutunk, amikben az első polinom csak x_{n-1} -től függ, a második csak x_{n-1} -től és x_{n-2} -től függ. Ezt ismételve eljutunk a polinomok közös gyökéhez.

A példa Gröbner-bázisának kiszámolása után elvégezzük az $x_3 = 1$ behelyettesítést. Ezután második polinomot faktorizáljuk, majd gyökeit (α, α^2) behelyettesítjük a harmadik polinomba. Az $x_2 = \alpha$ helyettesítés után kapjuk, hogy $x_1 = 1$, az $x_2 = \alpha^2$ után pedig, hogy $x_1 = \alpha^2$, így megkapjuk, hogy a polinomok közös gyökei $(1, \alpha, 1)$ és $(\alpha^2, \alpha^2, 1)$. Ha a *facstd* eljárást hívjuk meg a polinomokra, akkor a SINGULAR az $(x_3 + 1, x_2 + \alpha, x_1 + 1)$ és az $(x_3 + 1, x_2 + \alpha^2, x_1 + \alpha^2)$ ideálokat adja vissza. Ezek az előbb kiszámolt két megoldásnak felelnek meg.

3.3. PEsolve

A PEsolve esetén azt a tényt használjuk fel, hogy egy q elemű test minden $a \neq 0$ elemére teljesül, hogy $a^{q-1} = 1$. Tehát egy $f \in \mathbb{F}_q[x_1, \dots, x_n]$ polinomnak pontosan akkor nincs gyöke, ha az $f^{q-1} - 1$ polinom minden helyettesítésre 0-t vesz fel.

Hasonló ötletet alkalmazva oldhatunk meg egyenletrendszereket is. Tekintsük egy polinomrendszert, mely az $f_1, \dots, f_r \in \mathbb{F}_q[x_1, \dots, x_n]$ polinomokból áll. Ezekből elkészítjük a $\prod_{i=1}^r (f_i^{q-1} - 1)$ polinomot, majd ezután vesszük a $x_i^q - x_i$ polinomokkal való osztási maradékát. Az így kapott polinomot nevezzük g -nek. A g polinom pontosan a közös gyökökön nem tűnik el. Tehát a $g(x_1, \dots, x_n)$ pontosan akkor lesz a 0 polinom, ha az egyenletrendszer *nem* megoldható [14].

A fenti algoritmussal el lehet dönteni, hogy létezik-e közös gyök. A továbbiakban ismertetem, hogy az implementációban miként keresem meg ezeket. Az algoritmussal eldöntjük, hogy az $x_1 = a_1$ helyettesítésre van-e közös gyöke polinomoknak. Ha van, akkor létezik $(a_1, \dots)$ alakú közös gyök és helyettesíték a következő változóba. Ha nincs, akkor egyetlen közös gyök sem kezdődik a_1 -gyel, ezért veszek egy másik, $x_1 = b_1$ helyettesítést és ezt vizsgálom meg. Ezt az eljárást folytatva legfeljebb $n \cdot q \cdot s$ lépésben megtalálható az összes közös gyök, ahol s a közös gyökök száma. Vegyük észre, hogy ha az $f_i(a_1, \dots, a_k, x_{k+1}, \dots, x_n)$ polinomokból készítjük el a fentebb leírt módon a g polinomot, akkor az éppen a $g(a_1, \dots, a_k, x_{k+1}, \dots, x_n)$ polinom lesz. Emiatt elegendő az eredeti polinomokból elkészíteni a g -t és a többi lépésben már közvetlenül ebbe helyettesíteni az f_i polinomok helyett. Továbbá az implementációban a g polinomot úgy számolom, hogy először az f_i -ket növekvő sorrendbe rendezem, majd az $f_i^{q-1} - 1$ polinomokat ebben a sorrendben szorzom össze. Ha bármikor menet közben a 0 polinomot kapom, akkor az eljárás megáll. Így ha az egyenletrendszernek nincs gyöke, az esetleg még azelőtt kide-

rülhet, mielőtt a hosszabb f_i polinomokkal végeznénk bármilyen (várhatóan időigényes) műveletet.

Az algoritmus a következőképpen is szemléltethető. Tekintsünk egy fa gráfot, melynek gyökerét jelölje a g polinom. Ennek a csúcsnak legyen q darab gyermeke, melyeket a g első változójának q különböző helyettesítése után kapott polinomok jelölik. Ezen csúcsok mindegyikének is lesz q gyermeke. A gráf utolsó szintjén q^n csúcs található, amely az összes különböző helyettesítésnek felel meg. Amennyiben egy csúcs a 0 polinom, az azt jelenti, hogy az aktuális helyettesítés mellett a többi változó semelyik helyettesítése esetén sem lesz közös gyöke a polinomoknak. Emiatt szükségtelen tovább behelyettesítenünk, azaz a csúcs és a hozzátartozó részfa törölhető. Tehát nem kell az egész fát felépíteni. Lényegében mélységi keresést hajtunk végre a q^n helyettesítésre.

Algoritmus.

Bemenet: $f_1, \dots, f_r \in \mathbb{F}_q[x_1, \dots, x_n]$ a megoldandó egyenletrendszer

Kimenet: S , a megoldás ideálokat tartalmazó lista

Feltesszük, hogy adott az alaptest elemeinek egy tetszőleges sorba rendezése. Amikor az *első*, a *következő* és az *utolsó* elemre hivatkozunk, akkor azt ennek a sorrendnek megfelelően értjük.

Az a_i változó *léptetése* alatt azt értjük, hogy a jelenlegi elem után következőt állítjuk be az értékének. Amennyiben az a_i -nek nincs érték megadva, akkor a léptetés után az első test elem lesz az értéke. Ha a_i az utolsó elem, akkor a léptetések töröljük a_i -t, csökkentjük i -t, majd léptetjük az új a_i -t. Az algoritmus akkor ér véget, ha valamely léptetés után $i = 0$ teljesül.

1. Legyen $S = \emptyset$ és készítsük el a

$$g(x_1, \dots, x_n) = \prod_{i=1}^r \left(f_i(x_1, \dots, x_n)^{q-1} - 1 \right) \pmod{x_1^q - x_1, \dots, x_n^q - x_n}$$

polinomot.

2. Ha a kapott g a 0 polinom, akkor nincs megoldása rendszernek, ellenkező esetben legyen $i = 1$ és hajtsuk végre az alábbi lépéseket. (Amennyiben a program a 0

kiegészítő paraméterrel lett meghívva, úgy itt megáll, és jelzi, hogy megoldható-e az egyenletrendszer.)

3. Ha $i = 0$, akkor véget ért az algoritmus, különben léptessük a_i -t és tekintsük az $x_i = a_i$ helyettesítését g -nek.
4. Ha a $g(a_1, \dots, a_i, x_{i+1}, \dots, x_n) = 0$, akkor lépünk vissza az előző pontra.
5. Ha $i = n$, akkor legyen $S = S \cup \{(a_1, \dots, a_n)\}$. Ha $i < n$, akkor növeljük i -t. Lépünk a 3. pontra.

Példa. Vegyük az $\mathbb{F}_4[x_1, x_2, x_3]$ gyűrűt. Monomrendezésnek válasszuk a lexikografikus rendezést, az $f_1, f_2, f_3 \in \mathbb{F}_4[x_1, x_2, x_3]$ pedig a következők:

$$\begin{aligned} f_1(x_1, x_2, x_3) &= x_1^2 x_2 + \alpha x_1 x_2^2 + \alpha^2, \\ f_2(x_1, x_2, x_3) &= x_1^2 x_2 x_3^2 + \alpha x_1, \\ f_3(x_1, x_2, x_3) &= \alpha^2 x_1 x_3 + \alpha^2 x_1. \end{aligned}$$

Első lépésben számoljuk ki a g polinomot.

$$\begin{aligned} g = (f_1^3 - 1)(f_2^3 - 1)(f_3^3 - 1) \bmod x_1^4 - x_1, x_2^4 - x_2, x_3^4 - x_3 = \\ x_1^3 x_2^2 x_3^3 + x_1^3 x_2^2 x_3^2 + x_1^3 x_2^2 x_3 + x_1^3 x_2 x_3^3 + x_1^3 x_2 + \\ + x_3^2 + x_1^3 x_2 x_3 + \alpha x_1^2 x_2^3 x_3^3 + \alpha x_1^2 x_2^3 x_3^2 + \alpha x_1^2 x_2^3 x_3 + \\ + \alpha x_1^2 x_2 x_3^3 + \alpha x_1^2 x_2 x_3^2 + \alpha x_1^2 x_2 x_3 + \alpha^2 x_1 x_2^3 x_3^3 + \\ + \alpha^2 x_1 x_2^3 x_3^2 + \alpha^2 x_1 x_2^3 x_3 + \alpha^2 x_1 x_2^2 x_3^3 + \alpha^2 x_1 x_2^2 + \\ + x_3^2 + \alpha^2 x_1 x_2^2 x_3. \end{aligned}$$

Helyettesítsünk $x_1 = 0$ -t, ekkor $g(0, x_2, x_3) = 0$, tehát nincs olyan megoldás, melyben az első változó 0. Ezután végezzük el az $x_1 = 1$ helyettesítést. Ekkor

$$\begin{aligned} g(1, x_2, x_3) &= x_2^3 x_3^3 + x_2^3 x_3^2 + x_2^3 x_3 + \alpha x_2^2 x_3^3 + \alpha x_2^2 x_3^2 + \\ &+ \alpha x_2^2 x_3 + \alpha^2 x_2 x_3^3 + \alpha^2 x_2 x_3^2 + \alpha^2 x_2 x_3. \end{aligned}$$

Mivel most nem tűnt el g , ezért helyettesünk a következő változóba. A $g(1, \alpha, x_3) =$

$= x_3^3 + x_3^2 + x_3$ polinom kivételével a többi helyettesítésre 0-t kaptunk, így csak ezzel a polinommal kell tovább számolnunk. Végül helyettesítsünk a harmadik változóba. Ekkor g csak a $g(1, \alpha, 1) = 1$ helyettesítésre nem tűnik el, a többire igen, így ezzel megkaptuk az $(x_1, x_2, x_3) = (1, \alpha, 1)$ gyököt.

A következő lépésben az $x_1 = \alpha$ helyettesítést elvégezve kapjuk, hogy $g(\alpha, x_2, x_3) = 0$. Ezután az $x_1 = \alpha^2$ következik,

$$g(\alpha^2, x_2, x_3) = x_2^3 x_3^3 + x_2^3 x_3^2 + x_2^3 x_3 + \alpha^2 x_2^2 x_3^3 + \\ + \alpha^2 x_2^2 x_3^2 + \alpha^2 x_2^2 x_3 + \alpha x_2 x_3^3 + \alpha x_2 x_3^2 + \alpha x_2 x_3.$$

Ezután a második változóba helyettesítés után a $g(\alpha^2, \alpha^2, x_3) = x_3^3 + x_3^2 + x_3$ polinomot kapjuk, a többi pedig eltűnik, így ennek kell az x_3 változójába helyettesíteni. A behelyettesítések után $g(\alpha^2, \alpha^2, 1) = 1$ az egyetlen eset, ahol g nem tűnik el, azaz megtaláltuk az $(x_1, x_2, x_3) = (\alpha^2, \alpha^2, 1)$ gyököt.

Mivel az összes szükséges helyettesítést elvégeztük, az algoritmus futása véget ér, megtaláltuk az f_1, f_2, f_3 polinomok összes közös gyökét: $(1, \alpha, 1)$ és $(\alpha^2, \alpha^2, 1)$.

3.4. ZZsolve

A ZZsolve algoritmusban a többváltozós polinomrendszereket egy nagyobb test feletti egyváltozós egyenletre vezetjük vissza [10, 15], amit hatékonyan tudunk megoldani (pl. Berlekamp-algoritmussal).

Legyen $\mathbb{F}_q$ a q elemű test és tegyük fel, hogy $f_1, \dots, f_m \in \mathbb{F}_q[x_1, \dots, x_n]$. Tegyük fel először, hogy ugyanannyi polinom van, mint változó, azaz $n = m$. Választunk egy n -ed fokú $g(y)$ irreducibilis polinomot az $\mathbb{F}_q[y]$ gyűrűből és ennek segítségével elkészítjük az $\mathbb{F}_{q^n} \cong \mathbb{F}_q[y] / (g(y))$ testet. Legyen $\phi: \mathbb{F}_q^n \rightarrow \mathbb{F}_{q^n}$, $\phi(a_1, \dots, a_n) = a_1 + a_2 y + \dots + a_n y^{n-1}$. Ez a leképezés az $\mathbb{F}_q$ feletti n -dimenziós lineáris teret megfelelteti az $\mathbb{F}_{q^n}$ testtel. Továbbá legyen $f: \mathbb{F}_q^n \rightarrow \mathbb{F}_q^n$ leképezés az $f = (f_1, \dots, f_n)$ képlettel definiálva. A ϕ segítségével felemelhetjük f -et az $\mathbb{F}_{q^n}$ testbe úgy, hogy készítünk egy $F: \mathbb{F}_{q^n} \rightarrow \mathbb{F}_{q^n}$ leképezést az $F = \phi \circ f \circ \phi^{-1}$ módon. Ekkor F reprezentálható egy $\mathbb{F}_{q^n}$ feletti egyváltozós polinomként. Ennek gyökei kölcsönösen egyértelműen megfeleltethetők az $f_1, \dots, f_n$ polinomok közös gyökeivel.

Algoritmus.

Bemenet: E , az egyenletrendszer, mint ideál.

Kimenet: S , a megoldásideálokat tartalmazó lista.

1. Legyen n a változók száma és q az alaptest elemszáma. Vegyünk egy n -ed fokú irreducibilis polinomot $\mathbb{F}_q$ felett, legyen ennek egy gyöke β . Készítsük el az $\mathbb{F}_{q^n}$ testet.
2. Legyen $\phi(x_1, \dots, x_n) = x_1 + x_2\beta + \dots + x_n\beta^{n-1} = X$. A ϕ^{-1} meghatározásához oldjuk meg $M \in \mathbb{F}_{q^n}^{n \times n}$ -re a $\phi^{-1}(X) = (X, X^q, \dots, X^{q^{n-1}})M = (x_1, \dots, x_n)$ lineáris egyenletrendszert mod $x_1^q - x_1, \dots, x_n^q - x_n$.
3. Határozzuk meg az $F = \phi \circ f \circ \phi^{-1} \bmod X^{q^n} - X$ polinomot.
4. Faktorizáljuk F -et és a lineáris tényezőkhöz tartozó gyökök halmaza legyen S .

Az algoritmus leírásában feltettük, hogy a polinomok és a változók száma megegyezik. Az algoritmus használható abban az esetben is, amikor ez nem teljesül. Ha $m < n$, akkor úgy tekintjük, hogy $f_{m+1} = \dots = f_n = 0$. Amennyiben $m > n$, akkor léteznek $t, r \in \mathbb{N}$, hogy $m = t \cdot n + r$ és $0 \leq r < n$. Ilyenkor t darab n egyenletből és egy r egyenletből álló rendszerre alkalmazzuk az algoritmust. Az eredeti polinomok közös gyökei a $t + 1$ darab $\mathbb{F}_{q^n}$ -beli egyenlet legnagyobb közös osztójának lineáris tényezőinek felelnek meg.

Példa. Legyen $\mathbb{F}_4 = \{0, 1, \alpha, \alpha^2\}$ a négyelemű test, ahol $\alpha^2 + \alpha + 1 = 0$. Tekintsük az $\mathbb{F}_4[x_1, x_2, x_3]$ gyűrűt lexikografikus monomrendezés mellett és az

$$I = (x_1^2x_2 + \alpha x_1x_2^2 + \alpha^2, x_1^2x_2x_3^2 + \alpha x_1, \alpha^2x_1x_3 + \alpha^2)$$

ideált. Mivel ez három változós, így az $\mathbb{F}_{64}$ testbe fogjuk felemelni a rendszert, ezért szükségünk van egy harmadfokú irreducibilis polinomra $\mathbb{F}_4$ felett. Ehhez faktorizáljuk a megfelelő Conway-polinomot [13]:

$$y^6 + y^4 + y^3 + y + 1 = (y^3 + y^2 + y + \alpha) \cdot (y^3 + y^2 + y + \alpha^2).$$

Válasszuk a $g(y) = y^3 + y^2 + y + \alpha$ tényezőt, és legyen ennek az egyik gyöke β . Ennek segítségével el tudjuk készíteni a $\phi: \mathbb{F}_4^3 \rightarrow \mathbb{F}_{64}$, $\phi(x_1, x_2, x_3) = x_1 + x_2\beta + x_3\beta^2 = X$

leképezést. Ekkor felhasználva az $x_1^4 = x_1$, $x_2^4 = x_2$, $x_3^4 = x_3$ összefüggéseket, azt kapjuk, hogy $X^4 = x_1 + (\alpha + \alpha^2\beta)x_2 + (\alpha + \alpha\beta^2)x_3$ és $X^{16} = x_1 + (\alpha^2 + \alpha\beta)x_2 + (\alpha + \alpha^2\beta^2)x_3$, így a $\phi^{-1}: \mathbb{F}_{64} \rightarrow \mathbb{F}_4^3$ leképezést az alábbi alakban írhatjuk fel,

$$\phi^{-1}(X) = (X, X^4, X^{16}) \begin{pmatrix} \alpha^2\beta + \alpha\beta^2 & \alpha + \alpha\beta^2 & \alpha + \alpha\beta \\ \alpha^2 + \beta + \alpha^2\beta^2 & \alpha^2 + \alpha^2\beta^2 & 1 + \beta \\ \alpha^2 + \alpha^2\beta + \beta^2 & 1 + \beta^2 & \alpha^2 + \alpha^2\beta \end{pmatrix} = (x_1, x_2, x_3).$$

Az $f: \mathbb{F}_4^3 \rightarrow \mathbb{F}_4^3$, $f(x_1, x_2, x_3) = (x_1^2x_2 + \alpha x_1x_2^2 + \alpha^2, x_1^2x_2x_3^2 + \alpha x_1, \alpha^2x_1x_3 + \alpha^2)$ leképezést felhasználva az eredeti egyenletrendszer képe a 64 elemű testben modulo $X^{64} - X$:

$$\begin{aligned} F(X) = \phi \circ f \circ \phi^{-1}(X) = & \beta^{31}X^{56} + \beta^{37}X^{50} + \beta^8X^{48} + \beta^{10}X^{44} + \beta^{52}X^{41} + \\ & + \beta^{16}X^{38} + \beta^{24}X^{36} + \beta^{58}X^{35} + \beta^{37}X^{33} + \beta^{61}X^{26} + \beta^{25}X^{24} + \beta^{43}X^{20} + \\ & + \beta^{33}X^{18} + \beta^{60}X^{17} + \beta^{45}X^{16} + \beta^{40}X^{14} + \beta^2X^{12} + \beta^{19}X^{11} + \beta^6X^9 + \\ & + \beta^{18}X^8 + \beta^{22}X^6 + \beta^7X^5 + \beta^{15}X^4 + \beta^{32}X^3 + \beta^{42}X^2 + \beta^{39}X + \beta^{42}. \end{aligned}$$

Ennek két lineáris tényezője van, $X + (\alpha^2 + \alpha^2\beta + \beta^2)$ és $X + (1 + \alpha\beta + \beta^2)$. Az eredeti polinomok közös gyökeit pedig könnyen leolvashatjuk ezekről, az első $(\alpha^2, \alpha^2, 1)$, a második pedig $(1, \alpha, 1)$.

A példában az $\mathbb{F}_4$ testet bővítve készítettük el az $\mathbb{F}_{64}$ testet. A program technikai okok miatt $\mathbb{F}_2$ -t bővítve készíti el a hatodfokú Conway-polinom segítségével $\mathbb{F}_{64}$ -et, majd ebben számol tovább, felhasználva, hogy $\beta^{21} = \alpha$ (ld. A. függelék). Továbbá az algoritmus megköveteli, hogy egyszerre két különböző test feletti többváltozós polinomgyűrűben dolgozzunk. Mivel a SINGULAR egyszerre csak egy polinomgyűrűben hajlandó műveleteket végezni, ezért definiálni kellett a két polinomgyűrű között egy alkalmas homomorfizmust, mely segítségével tudunk váltani a két gyűrű között.

3.5. XLsolve

Az XLsolve az XL (eXtended Linearization) linearizációs technikát használó algoritmus. A tipikus linearizációs technika elsősorban kvadratikus egyenletek megoldására lett kifejlesztve, ahol az x_ix_j alakú monomokat új y_{ij} változókkal helyettesítjük. Az így kapott lineáris egyenletrendszer már hatékonyan megoldható. Az új változókra kapott megoldá-

sokból már ki tudjuk számolni az eredeti változók értékeit. Ennek a módszernek létezik egy módosított változata, a relinearizáció [15]. Lényege, hogy a lineáris egyenletek mellé beveszünk nem lineáris egyenleteket is melyek azt az információt hordozzák, hogy a linearizáció során bevezetett új y_{ij} változók nem függetlenek egymástól. Legegyszerűbb formájában a relinearizáció a változó 4-esek szorzásra való kommutativitására épül. Tetszőleges i, j, k, l indexek esetén $(x_i x_j)(x_k x_l) = (x_i x_k)(x_j x_l) = (x_i x_l)(x_j x_k)$, ami az új változók esetén az $y_{ij} y_{kl} = y_{ik} y_{jl} = y_{il} y_{jk}$ összefüggésnek felel meg.

Az XL lényege, hogy az egyenletrendszer minden egyenletét megszorozzuk egy bizonyos fokszámkorlát alá eső monomokkal. Az így kapott egyenletrendszerre pedig alkalmazzuk a linearizációt [5]. Az eljárás működik kvadratikusnál magasabb fokú egyenletekre is [9].

Példa. Legyen $\mathbb{F}_4 = \{0, 1, \alpha, \alpha^2\}$ a négy elemű test, ahol $\alpha^2 + \alpha + 1 = 0$. Tekintsük az $\mathbb{F}_4[x_1, x_2, x_3]$ gyűrűt lexikografikus monomrendezés mellett és az

$$I = (x_1 x_3 + \alpha^2, x_1^2 + \alpha^2 x_2^2 + \alpha, x_1^2 + x_1 x_3 + x_2^2 + \alpha^2, x_1^2 + x_1 x_3 + \alpha)$$

ideált. A polinomok szorzásához vegyük az összes, legfeljebb elsőfokú monomot $(1, x_1, x_2, x_3)$, így a következő polinomokat kapjuk:

$x_1 x_3 + \alpha^2$	$x_1^2 + \alpha^2 x_2^2 + \alpha$	$x_1^2 + x_1 x_3 + x_2^2 + \alpha^2$	$x_1^2 + x_1 x_3 + \alpha$
$x_1^2 x_3 + \alpha^2 x_1$	$x_1^3 + \alpha^2 x_1 x_2^2 + \alpha x_1$	$x_1^3 + x_1^2 x_3 + x_1 x_2^2 + \alpha^2 x_1$	$x_1^3 + x_1^2 x_3 + \alpha x_1$
$x_1 x_2 x_3 + \alpha^2 x_2$	$x_1^2 x_2 + \alpha^2 x_2^3 + \alpha x_2$	$x_1^2 x_2 + x_1 x_2 x_3 + x_2^3 + \alpha^2 x_2$	$x_1^2 x_2 + x_1 x_2 x_3 + \alpha x_2$
$x_1 x_3^2 + \alpha^2 x_3$	$x_1^2 x_3 + \alpha^2 x_2^2 x_3 + \alpha x_3$	$x_1^2 x_3 + x_1 x_3^2 + x_2^2 x_3 + \alpha^2 x_3$	$x_1^2 x_3 + x_1 x_3^2 + \alpha x_3$

Az $x_1^{e_1} x_2^{e_2} \cdots x_n^{e_n}$ alakú monomokat cseréljük le új $y_{e_1 e_2 \dots e_n}$ változókra, így a következő lineáris polinomokat kapjuk:

$y_{101} + \alpha^2$	$y_{200} + \alpha^2 y_{020} + \alpha$	$y_{200} + y_{101} + y_{020} + \alpha^2$	$y_{200} + y_{101} + \alpha$
$y_{201} + \alpha^2 y_{100}$	$y_{300} + \alpha^2 y_{120} + \alpha y_{100}$	$y_{300} + y_{201} + y_{120} + \alpha^2 y_{100}$	$y_{300} + y_{201} + \alpha y_{100}$
$y_{111} + \alpha^2 y_{010}$	$y_{210} + \alpha^2 y_{030} + \alpha y_{010}$	$y_{210} + y_{111} + y_{030} + \alpha^2 y_{010}$	$y_{210} + y_{111} + \alpha y_{010}$
$y_{102} + \alpha^2 y_{001}$	$y_{201} + \alpha^2 y_{012} + \alpha y_{001}$	$y_{201} + y_{102} + y_{012} + \alpha^2 y_{001}$	$y_{201} + y_{102} + \alpha y_{001}$

A Gauss-elimináció elvégzése és az $y_{i_1 \dots i_k}$ változók monomokká való visszaírása után a következő polinomokat kapjuk:

$$\begin{array}{cccc} x_1 + \alpha x_3 & x_1 x_3 + \alpha^2 & x_2^2 + 1 & x_1^2 + 1 \\ x_1 x_3^2 + \alpha^2 x_3 & x_2^2 x_3 + x_3 & x_1 x_2 x_3 + \alpha^2 x_2 & x_1^2 x_3 + x_3 \\ x_2^3 + x_2 & x_1 x_2^2 + \alpha x_3 & x_1^2 x_2 + x_2 & x_1^3 + \alpha x_3. \end{array}$$

A kapott egy változós polinomokból meghatározzuk, hogy $x_1 = 1$ és $x_2 = 1$. Majd ezek behelyettesítése után kapjuk, hogy $x_3 = \alpha^2$.

Az eljárás során a SINGULAR-t többször is polinomgyűrű váltására kényszerítjük: először az $\mathbb{F}_q[x_1, \dots, x_n]$ gyűrűhöz hozzávesszük az y_j változókat, és definiáljuk a köztük megtalálható összefüggéseket. Majd szükségünk van egy csak y_j -ket tartalmazó polinomgyűrűre, ahol megoldjuk az eredeti egyenletrendszer megfelelőjét. A megoldásokat visszaírjuk az átmeneti (x_i -ket és y_j -ket is tartalmazó) polinomgyűrűbe, ahol a monomrendezés módosításával kifejezhetjük y_j monomjait x_i monomjaivá. Végül a kapott megoldásokat visszahúzva az eredeti polinomgyűrűbe kapjuk az eredeti egyenletrendszer megoldásait.

3.6. ffsolve

Az ffsolve eljárás véges test feletti univerzális megoldóként működik. Heurisztikusan választja ki a megadott bemenethez legjobban illő algoritmust. Az eljárás rekurzívan működik és felhasználja azt a simplesolverben alkalmazott ötletet, hogy az esetleg előforduló egyváltozós polinomok gyökeit meghatározva csökkenthetjük az egyenletrendszert. Az egyváltozós polinom gyökeinek behelyettesítése után kapott új egyenletrendszerekre rekurzívan meghívja önmagát, mivel ilyenkor előfordulhat, hogy a kapott új rendszerekhez más-más megoldóeljárást érdemes használni. Az ffsolve alapértelmezésben a dolgozatban bemutatott algoritmusokból válogat, de lehetőség van bemenetként saját megoldóeljárást is megadni. Az ilyen eljárásoktól elvárt, hogy a bemenetet *ideal* típusként várják és kimenetként egy listát adjon, melynek elemei *ideal* típusúak.

A megfelelő megoldó eljárást kiválasztó heurisztika (a CD-n mellékelt változatban) a következőképpen működik: tegyük fel, hogy a listában kapott választható megoldó eljárások $alg_1, \dots, alg_n$. Ekkor a fenti algoritmusok közül a heurisztika véletlenszerűen választ egyet úgy, hogy az alg_i eljárást $2n+1-i$ súllyal látja el. Tehát az i -edik eljárás kiválasztási

valószínűsége $\frac{2(2n+1-i)}{n(3n+1)}$. Ezzel elérjük, hogy a lista elején található algoritmusokat preferálja a később szereplőkkel szemben, ám nem túlzó mértékben teszi azt. Az alapértelmezett lista

- lineáris egyenletrendszer esetén (GBsolve),
- másodfokú egyenletrendszer esetén (XLsolve, PEsolve, simplesolver, GBsolve, ZZsolve),
- magasabb fokú egyenletrendszer esetén (PEsolve, simplesolver, GBsolve, ZZsolve, XLsolve).

Ezt a listát a diplomamunkám alapján állítottam össze [3]. Ebben elemeztem a fenti algoritmusok elméleti futásidejét, valamint a kapott eredményeket gyakorlati módon tesztelt futásidőkkel is megerősítettem.

A későbbiekben tervezem a fenti heurisztika javítását a SINGULAR fejlesztői csoportjával folytatott eszmecsere követően. Felvettem a kapcsolatot a SINGULAR fejlesztői csoportjával, hogy a bemutatott függvénykönyvtár bekerüljön a rendszerrel szállítottak közé. A jövőben várhatóan több, a fenti heurisztikára illetve az egyes algoritmusok implementálására vonatkozó tanácsot fognak adni.

4. Összefoglalás

Készítettem egy függvénykönyvtárat a polinom számításokra specializált, szabadon hozzáférhető (GNU GPL licencű) SINGULAR komputer algebra rendszerben. Ez a rendszer tartalmazza a legtöbb fejlett algoritmust a kommutatív algebra és az algebrai geometria területéről. A SINGULAR csomag többféle egyenletrendszer megoldó eljárást is tartalmaz, ám ezek mindegyike csak 0 karakterisztikában működik. A SINGULAR hivatalos fejlesztői csoportja nyitott problémát írt ki a prím elemszámú testek feletti megoldó algoritmusok implementálására *Solve polynomial systems over $\mathbb{Z}/p$* címmel.

A dolgozatban bemutatok öt ismert, véges testek feletti egyenletrendszereket megoldó algoritmust, és azok általam írt implementációját SINGULAR környezetben. Mindegyik implementáció fut tetszőleges véges testre, nem csak a kért $\mathbb{Z}/p$ testekre, ezzel jóval általánosabb megoldást nyújtva a SINGULAR nyitott problémára. Az eljárások az alábbi matematikai elméleteket használják:

- helyettesítések ellenőrzése,
- Gröbner bázisok elmélete,
- polinomok ekvivalenciája,
- többváltozós egyenletrendszer leképezése egyváltozós egyenletté,
- linearizációs technikák.

Hivatkozások

- [1] E. R. BERLEKAMP: Factoring polynomials over finite fields. *Bell System Tech. J.*, 46. évf. (1967), 1853–1859. p. ISSN 0005-8580.
- [2] BORUS Gergő Gyula: ffsolve - singular library for solving systems of polynomial equations over finite fields. URL <http://web.unideb.hu/~borisz/ffsolve/>.
- [3] BORUS Gergő Gyula: Véges testek feletti egyenletrendszerek megoldása. Diplomamunka (Debreceni Egyetem). 2012.
- [4] Nicolas COURTOIS – Gregory BARD: Algebraic cryptanalysis of the data encryption standard. In Steven GALBRAITH (szerk.): *Cryptography and Coding*. Lecture Notes in Computer Science sorozat, 4887. köt. Springer Berlin / Heidelberg, 152–169. p. ISBN 978-3-540-77271-2. URL http://dx.doi.org/10.1007/978-3-540-77272-9_10.
- [5] Nicolas COURTOIS – Alexander KLIMOV – Jacques PATARIN – Adi SHAMIR: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in cryptology—EUROCRYPT 2000 (Bruges)*. Lecture Notes in Comput. Sci. sorozat, 1807. köt. Berlin, Springer, 2000, 392–407. p. URL http://dx.doi.org/10.1007/3-540-45539-6_27.
- [6] Nicolas COURTOIS – Josef PIEPRZYK: Cryptanalysis of block ciphers with overdefined systems of equations. In Yuliang ZHENG (szerk.): *Advances in Cryptology — ASIACRYPT 2002*. Lecture Notes in Computer Science sorozat, 2501. köt. Springer Berlin / Heidelberg, 267–287. p. ISBN 978-3-540-00171-3. URL http://dx.doi.org/10.1007/3-540-36178-2_17.

- [7] David COX – John LITTLE – Donal O’SHEA: *Ideals, varieties, and algorithms*. Undergraduate Texts in Mathematics sorozat. Third. kiad. New York, Springer, 2007, xvi+551. p. ISBN 978-0-387-35650-1; 0-387-35650-9.
URL <http://dx.doi.org/10.1007/978-0-387-35651-8>. An introduction to computational algebraic geometry and commutative algebra.
- [8] W. DECKER – G.-M. GREUEL – G. PFISTER – H. SCHÖNEMANN: SINGULAR 3-1-4 — A computer algebra system for polynomial computations. 2012.
<http://www.singular.uni-kl.de>.
- [9] Claus DIEM: The XL-Algorithm and a Conjecture from Commutative Algebra. In Pil LEE (szerk.): *Advances in Cryptology - ASIACRYPT 2004*. Lecture Notes in Computer Science sorozat, 3329. köt. Springer Berlin / Heidelberg, 146–159. p. ISBN 978-3-540-23975-8. URL http://dx.doi.org/10.1007/978-3-540-30539-2_23.
- [10] Jintai DING – Jason E. GOWER – Dieter S. SCHMIDT: Zhuang-Zi: A New Algorithm for Solving Multivariate Polynomial Equations over a Finite Field. Cryptology ePrint Archive, Report 2006/038, 2006. URL <http://eprint.iacr.org/2006/038>.
- [11] Jean-Charles FAUGÈRE – Antoine JOUX: Algebraic cryptanalysis of hidden field equation (hfe) cryptosystems using gröbner bases. In Dan BONEH (szerk.): *Advances in Cryptology - CRYPTO 2003*. Lecture Notes in Computer Science sorozat, 2729. köt. Springer Berlin / Heidelberg, 44–60. p. ISBN 978-3-540-40674-7.
URL http://dx.doi.org/10.1007/978-3-540-45146-4_3.
- [12] M.R. GAREY – D.S. JOHNSON: *Computers and intractability: a guide to the theory of NP-completeness*. Series of books in the mathematical sciences sorozat. W. H. Freeman, 1979. ISBN 9780716710448.
- [13] Derek F. HOLT – Bettina EICK – Eamonn A. O’BRIEN: *Handbook of computational group theory*. Discrete Mathematics and its Applications (Boca Raton) sorozat. Chapman & Hall/CRC, Boca Raton, FL, 2005, xvi+514. p. ISBN 1-58488-372-3.
URL <http://dx.doi.org/10.1201/9781420035216>.
- [14] Gábor HORVÁTH: The complexity of the equivalence problem over finite rings. *Glasg. Math. Journal*, 54. évf. (2012) 1. sz., 193–199. p.

- [15] Aviad KIPNIS–Adi SHAMIR: Cryptanalysis of the HFE public key cryptosystem by relinearization. In *Advances in cryptology—CRYPTO '99 (Santa Barbara, CA)*. Lecture Notes in Comput. Sci. sorozat, 1666. köt. Berlin, Springer, 1999, 19–30. p.
URL http://dx.doi.org/10.1007/3-540-48405-1_2.
- [16] NIST FIPS 197 - Advanced Encryption Standard (AES).
URL <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.
- [17] Solve polynomial systems over $\mathbb{Z}/p$.
URL <http://www.singular.uni-kl.de/index.php/open-tasks/13718.html?tid=42>.

A. $\mathbb{F}_{64}$ két logaritmus táblázata

Legyen β gyöke az $x^6 + x^4 + x^3 + x + 1 \in \mathbb{Z}_2[x]$ irreducibilis Conway-polinomnak. Ekkor β hatványai a következők.

$\beta^6 = 1 + \beta + \beta^3 + \beta^4$	$\beta^{35} = 1 + \beta^4$
$\beta^7 = \beta + \beta^2 + \beta^4 + \beta^5$	$\beta^{36} = \beta + \beta^5$
$\beta^8 = 1 + \beta + \beta^2 + \beta^4 + \beta^5$	$\beta^{37} = 1 + \beta + \beta^2 + \beta^3 + \beta^4$
$\beta^9 = 1 + \beta^2 + \beta^4 + \beta^5$	$\beta^{38} = \beta + \beta^2 + \beta^3 + \beta^4 + \beta^5$
$\beta^{10} = 1 + \beta^4 + \beta^5$	$\beta^{39} = 1 + \beta + \beta^2 + \beta^5$
$\beta^{11} = 1 + \beta^3 + \beta^4 + \beta^5$	$\beta^{40} = 1 + \beta^2 + \beta^4$
$\beta^{12} = 1 + \beta^3 + \beta^5$	$\beta^{41} = \beta + \beta^3 + \beta^5$
$\beta^{13} = 1 + \beta^3$	$\beta^{42} = 1 + \beta + \beta^2 + \beta^3$
$\beta^{14} = \beta + \beta^4$	$\beta^{43} = \beta + \beta^2 + \beta^3 + \beta^4$
$\beta^{15} = \beta^2 + \beta^5$	$\beta^{44} = \beta^2 + \beta^3 + \beta^4 + \beta^5$
$\beta^{16} = 1 + \beta + \beta^4$	$\beta^{45} = 1 + \beta + \beta^5$
$\beta^{17} = \beta + \beta^2 + \beta^5$	$\beta^{46} = 1 + \beta^2 + \beta^3 + \beta^4$
$\beta^{18} = 1 + \beta + \beta^2 + \beta^4$	$\beta^{47} = \beta + \beta^3 + \beta^4 + \beta^5$
$\beta^{19} = \beta + \beta^2 + \beta^3 + \beta^5$	$\beta^{48} = 1 + \beta + \beta^2 + \beta^3 + \beta^5$
$\beta^{20} = 1 + \beta + \beta^2$	$\beta^{49} = 1 + \beta^2$
$\beta^{21} = \beta + \beta^2 + \beta^3$	$\beta^{50} = \beta + \beta^3$
$\beta^{22} = \beta^2 + \beta^3 + \beta^4$	$\beta^{51} = \beta^2 + \beta^4$
$\beta^{23} = \beta^3 + \beta^4 + \beta^5$	$\beta^{52} = \beta^3 + \beta^5$
$\beta^{24} = 1 + \beta + \beta^3 + \beta^5$	$\beta^{53} = 1 + \beta + \beta^3$
$\beta^{25} = 1 + \beta^2 + \beta^3$	$\beta^{54} = \beta + \beta^2 + \beta^4$
$\beta^{26} = \beta + \beta^3 + \beta^4$	$\beta^{55} = \beta^2 + \beta^3 + \beta^5$
$\beta^{27} = \beta^2 + \beta^4 + \beta^5$	$\beta^{56} = 1 + \beta$
$\beta^{28} = 1 + \beta + \beta^4 + \beta^5$	$\beta^{57} = \beta + \beta^2$
$\beta^{29} = 1 + \beta^2 + \beta^3 + \beta^4 + \beta^5$	$\beta^{58} = \beta^2 + \beta^3$
$\beta^{30} = 1 + \beta^5$	$\beta^{59} = \beta^3 + \beta^4$
$\beta^{31} = 1 + \beta^3 + \beta^4$	$\beta^{60} = \beta^4 + \beta^5$
$\beta^{32} = \beta + \beta^4 + \beta^5$	$\beta^{61} = 1 + \beta + \beta^3 + \beta^4 + \beta^5$
$\beta^{33} = 1 + \beta + \beta^2 + \beta^3 + \beta^4 + \beta^5$	$\beta^{62} = 1 + \beta^2 + \beta^3 + \beta^5$
$\beta^{34} = 1 + \beta^2 + \beta^5$	$\beta^{63} = 1$

Tekintsük az $x^6 + x^4 + x^3 + x + 1 \in \mathbb{Z}_2[x]$ irreducibilis Conway-polinomot. Ez $\mathbb{F}_4$ felett $(x^3 + x^2 + x + \alpha)(x^3 + x^2 + x + \alpha^2)$ alakban bomlik irreducibilis polinomok szorzatára. Legyen β gyöke az $x^3 + x^2 + x + \alpha$ polinomnak. Ekkor β hatványai a következők.

$\beta^3 = \alpha + \beta + \beta^2$	$\beta^{34} = 1 + \alpha\beta + \alpha\beta^2$
$\beta^4 = \alpha + \alpha^2\beta$	$\beta^{35} = \alpha^2 + \alpha^2\beta$
$\beta^5 = \alpha\beta + \alpha^2\beta^2$	$\beta^{36} = \alpha^2\beta + \alpha^2\beta^2$
$\beta^6 = 1 + \alpha^2\beta + \beta^2$	$\beta^{37} = 1 + \alpha^2\beta$
$\beta^7 = \alpha + \alpha\beta^2$	$\beta^{38} = \beta + \alpha^2\beta^2$
$\beta^8 = \alpha^2 + \alpha\beta^2$	$\beta^{39} = 1 + \alpha^2\beta + \alpha\beta^2$
$\beta^9 = \alpha^2 + \beta + \alpha\beta^2$	$\beta^{40} = \alpha^2 + \alpha^2\beta + \beta^2$
$\beta^{10} = \alpha^2 + \beta + \alpha^2\beta^2$	$\beta^{41} = \alpha + \alpha\beta + \alpha\beta^2$
$\beta^{11} = 1 + \alpha\beta^2$	$\beta^{42} = \alpha^2$
$\beta^{12} = \alpha^2 + \alpha^2\beta + \alpha\beta^2$	$\beta^{43} = \alpha^2\beta$
$\beta^{13} = \alpha^2 + \beta + \beta^2$	$\beta^{44} = \alpha^2\beta^2$
$\beta^{14} = \alpha + \alpha\beta$	$\beta^{45} = 1 + \alpha^2\beta + \alpha^2\beta^2$
$\beta^{15} = \alpha\beta + \alpha\beta^2$	$\beta^{46} = 1 + \alpha\beta$
$\beta^{16} = \alpha^2 + \alpha\beta$	$\beta^{47} = \beta + \alpha\beta^2$
$\beta^{17} = \alpha^2\beta + \alpha\beta^2$	$\beta^{48} = \alpha^2 + \alpha\beta + \alpha^2\beta^2$
$\beta^{18} = \alpha^2 + \alpha\beta + \beta^2$	$\beta^{49} = 1 + \beta^2$
$\beta^{19} = \alpha + \alpha\beta + \alpha^2\beta^2$	$\beta^{50} = \alpha + \beta^2$
$\beta^{20} = 1 + \beta + \beta^2$	$\beta^{51} = \alpha + \alpha^2\beta + \beta^2$
$\beta^{21} = \alpha$	$\beta^{52} = \alpha + \alpha^2\beta + \alpha\beta^2$
$\beta^{22} = \alpha\beta$	$\beta^{53} = \alpha^2 + \beta^2$
$\beta^{23} = \alpha\beta^2$	$\beta^{54} = \alpha + \alpha\beta + \beta^2$
$\beta^{24} = \alpha^2 + \alpha\beta + \alpha\beta^2$	$\beta^{55} = \alpha + \alpha^2\beta + \alpha^2\beta^2$
$\beta^{25} = \alpha^2 + \beta$	$\beta^{56} = 1 + \beta$
$\beta^{26} = \alpha^2\beta + \beta^2$	$\beta^{57} = \beta + \beta^2$
$\beta^{27} = \alpha + \beta + \alpha\beta^2$	$\beta^{58} = \alpha + \beta$
$\beta^{28} = \alpha^2 + \alpha^2\beta^2$	$\beta^{59} = \alpha\beta + \beta^2$
$\beta^{29} = 1 + \alpha^2\beta^2$	$\beta^{60} = \alpha + \beta + \alpha^2\beta^2$
$\beta^{30} = 1 + \alpha\beta + \alpha^2\beta^2$	$\beta^{61} = 1 + \beta + \alpha\beta^2$
$\beta^{31} = 1 + \alpha\beta + \beta^2$	$\beta^{62} = \alpha^2 + \alpha^2\beta + \alpha^2\beta^2$
$\beta^{32} = \alpha + \alpha^2\beta^2$	$\beta^{63} = 1$
$\beta^{33} = 1 + \beta + \alpha^2\beta^2$	

Saját munka leírása

A dolgozatban különböző véges testek feletti egyenletrendszereket megoldó algoritmusokat ismeretek, melyeket az elérhető szakirodalomból válogattam össze és tanulmányoztam az elméleti hátterüket. Mindegyik eljárás ismertetésekor hivatkozok a megfelelő cikkekre, melyben az eredeti leírás megtalálható.

A kiválasztott algoritmusok mindegyikét implementáltam a SINGULAR komputer algebra rendszerben. Az elkészült eljárásokból készítettem egy függvénykönyvtárat. A függvénykönyvtár és a benne szereplő implementációkat önállóan készítettem, azok a saját eredményeim. Az ffsolve programnak nem csak az implementációja, hanem a heurisztikája is önálló munkám eredménye.

A publikáció elkészítését a TÁMOP-4.2.2/B-10/1-2010-0024 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

