KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Aleksėjus Michalkovič

NETIESINĖS ALGEBRINĖS LYGČIŲ SISTEMOS SPRENDINIŲ SKAIČIAUS ANALIZĖ

Magistro darbas

Vadovas

prof. E. Sakalauskas

KAUNAS, 2010

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU

Katedros vedėjas

……. doc. dr. N. Listopadskis

NETIESINĖS ALGEBRINĖS LYGČIŲ SISTEMOS SPRENDINIŲ SKAIČIAUS ANALIZĖ

Taikomosios matematikos magistro baigiamasis darbas

Vadovas:

……. prof. E. Sakalauskas

2010 06 03

Recenzentas:

……. doc. G. S. Dosinas

2010 06 01

Atliko: FMMM-8 gr. stud.

……. A. Michalkovič

2010 05 25

KAUNAS, 2010

KVALIFIKACINĖ KOMISIJA

Pirmininkas:	Leonas Saulis, profesorius (VGTU)
Sekretorius:	Eimutis Valakevičius, docentas (KTU)
Nariai:	Algimantas Jonas Aksomaitis, profesorius (KTU)
	Vytautas Janilionis, docentas (KTU)
	Vidmantas Povilas Pekarskas, profesorius (KTU)
	Rimantas Rudzkis, habil. dr., vyriausiasis analitikas (DnB NORD Bankas)
	Zenonas Navickas, profesorius (KTU)
	Arūnas Barauskas, dr., vice-prezidentas projektams (UAB „Baltic Amadeus“)

Mihalkovich A. Analysis of number of solutions of an algebraic system of non-linear equations: Master's work in applied mathematics / supervisor prof. E. Sakalauskas; Department of Applied mathematics, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2010. – 49 p.

SUMMARY

Since the introduction of Diffie-Hellman key agreement protocol in 1976 computer technology has made a giant step forward. Nowadays there is not much time left before quantum computers will be in every home. However it was theoretically proven that discrete logarithm problem which is the basis for Diffie-Hellman protocol could be solved in polynomial time using such computers. Such possibility would make D-H protocol insecure. Thus cryptologists are searching for different ways to improve the security of the protocol by using hard problems. One of the ways to do so is to introduce secure one-way functions (OWF).

In this paper a new kind of OWF called the matrix power function will be analyzed. Professor Eligijus Sakalauskas introduced this function in 2007 and later used this function to construct a Diffie-Hellman type key agreement protocol using square matrices. This protocol is not only based on matrix power function but also on commutative matrices which are defined in finite fields or rings. Thus an algebraic non-linear system of equations is formed. The security of this system will be analyzed. It will be shown that we can use matrix power function in cryptography. We will also be analyzing how does the solution of the system depend on system parameters: the order of matrices and a parameter p which defines a finite group $\mathbb{Z}_p$. We will also briefly discuss the usage of this system in real life and the algebraic properties of the suggested OWF.

TURINYS

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

ĮVADAS

Kriptografija yra viena iš seniausių matematikos šakų. Pirmą kartą kriptografiniai algoritmai buvo panaudoti IV tūkstantmetyje prieš Kristų senovės Egipte. Vėliau kriptografinius algoritmus naudojo ir kitos pasaulio civilizacijos. Daugiausiai šie algoritmai buvo taikomi šifruojant karinius pranešimus. Vienas iš labiausiai žinomų senovės algoritmų – Cezario šifras – buvo naudojamas senovės Romoje I amžiuje prieš Kristų. Šifro esmę sudarė tai, kad kiekviena abecelės raidė buvo paslenkima per tam tikrą skaičių pozicijų. Dabartiniai mokslininkai traktuotų šį poslinkį kaip vartotojų slaptą raktą. Cezaris naudojo trijų pozicijų poslinkį.

Kriptografijos kaip matematinės šakos vystymas prasidėjo XIV amžiuje, kai buvo paskelbti pirmieji kriptografiniai darbai. Šiuose darbuose buvo dešifruoti senovės Egipto bei kiti šifrai. 1883 metais olandų kriptografas Auguste'as Kerckhoffs‘as savo darbe „Karinė kriptografija“ paskelbė šešis reikalavimus kurių turi prisilaikyti saugi sistema. Šie reikalavimai yra žinomi kaip Kerckhoffs‘o principas.

Šiuolaikinė kriptografija atsirado jau XX amžiuje kai buvo galutinai suformuoti tokios matematikos šakos kaip algebra, skaičių teorija, tikimybių teorija, matematinė statistika ir kitos. Atsiradus kompiuteriams reikėjo naujų šifravimo algoritmų. Nuo 1940 metų įvairių šalių mokslininkai kurdavo daugybę naujų šifravimo algoritmų. 1975 metais Whitfield‘as Diffie ir Martin‘as Hellman‘as paskelbė savo darbą „Naujos kryptys kriptografijoje“ (angl. „New directions in cryptography“). Taip atsirado asimetrinė kriptografija [6].

Vienas iš svarbiausių šiuolaikinės kriptografijos uždavinių yra saugių vienkrypčių funkcijų paieška. Dabartiniai mokslininkai skiria šiam klausimui ypatingą demėsį. Šiame darbe yra nagrinėjama viena iš naujausių vienkrypčių funkcijų – matricinio laipsnio funkcija. Ši funkcija yra panaudota netiesinės algebrinės lygčių sistemos sudarymui. Pagrindinis demėsys darbe yra skirtas šios lygčių sistemos analizei bei jos praktiniam taikymui. Nustatysime ar matricinio laipsnio funkcija gali būti panaudota kriptografijoje. Taip pat nustatysime lygčių sistemos sprendinių skaičiaus priklausomybę nuo jos parametrų: matricų eilės m bei grupės $\mathbb{Z}_p$ parametro p.

1. BENDROJI DALIS

1.1 KAI KURIOS GRUPIŲ TEORIJOS SĄVOKOS

Šiame skyrelyje apibrėšime svarbiausias magistro darbe naudojamas sąvokas. Pradėsime nuo grupės apibrėžimo.

Apibrėžimas: Aibė $\mathbb{G}$ su joje apibrėžta operacija * vadinama *grupe* jeigu yra tenkinamos šios aksiomos:

1. Operacija * yra uždara, t.y bet kuriems grupės elementams a ir b elementas $a * b \in \mathbb{G}$
2. Operacija * yra asociatyvi, t.y bet kuriems grupės elementams a, b ir c galioja lygybė $(a * b) * c = a * (b * c)$
3. Operacijos * atžvilgiu egzistuoja toks elementas e, kad bet kuriam grupės elementui a yra teisinga lygybė $a * e = e * a = a$. Toks elementas e vadinamas *neutraliuoju elementu*
4. Operacijos * atžvilgiu bet kuriam aibės $\mathbb{G}$ elementui a egzistuoja toks elementas $\tilde{a}$, kad yra teisinga lygybė $a * \tilde{a} = \tilde{a} * a = e$. Toks elementas $\tilde{a}$ vadinamas *atvirkštiniu elementu*

Pastaba: Jeigu 4 aksioma yra netenkinama, tai aibė $\mathbb{G}$ su joje apibrėžta operacija * vadinama *pusgrupe*

Pavyzdys: Sveikųjų skaičių aibė $\mathbb{Z}$ su sudėties operacija + sudaro grupę, nes yra tenkinamos visos keturios aksiomos. Tačiau natūraliųjų skaičių aibė su $\mathbb{N}$ daugybos operacija $\cdot$ sudaro tiktai pusgrupę, nes paskutinė aksioma yra netenkinama.

Apibrėžimas: Jeigu grupės $\mathbb{G}$ operacija yra komutatyvi, t.y bet kuriems grupės $\mathbb{G}$ elementams a ir b galioja lygybė $a * b = b * a$, tai grupė $\mathbb{G}$ yra vadinama *Abelio grupe*.

Pavyzdys: Sveikųjų skaičių aibė $\mathbb{Z}$ su sudėties operacija + sudaro Abelio grupę, nes sudėties operacija yra komutatyvi.

Darbe yra plačiai naudojamos žiedo bei lauko sąvokos. Apibrėžkime jas:

Apibrėžimas: Aibė $\mathbb{R}$ su dviejomis apibrėžtomis joje operacijomis + ir $\cdot$ yra vadinama *žiedu*, jeigu aibė $\mathbb{R}$ su operacija + sudaro Abelio grupe, o su operacija $\cdot$ sudaro pusgrupe.

Apibrėžimas: Aibė $\mathbb{F}$ su dviejomis apibrėžtomis joje operacijomis + ir $\cdot$ yra vadinama *lauku*, jeigu aibė $\mathbb{F}$ su kiekviena iš šių operacijų sudaro Abelio grupes.

Pastaba: Operacijas + ir · dažnai paprastumo dėlei vadina atitinkamai sudėtimi ir daugyba, nors šios operacijos bendruoju atveju nebūtinai sutampa su mums įprastomis sudėties bei daugybos operacijomis.

Kaip žinome nuo mokylos laikų dviejų lyginių skačių sandauga yra lyginis skaičius, o iš to, kad dviejų skaičių sandauga yra nelyginis skaičius išplaukia išvada, kad šie skaičiai yra nelyginiai. Lyginių (l) ir nelyginių (n) skaičių sudėties bei daugybos lentelės atrodo taip:

1.1 lentelė

Lyginių ir nelyginių skaičių sudėties lentelė

+	l	n
l	l	n
n	n	l

1.2 lentelė

Lyginių ir nelyginių skaičių daugybos lentelė

·	l	n
l	l	l
n	l	n

Matome, kad lyginius ir nelyginius skaičius galima sutapatinti atitinkamai su skaičiais 0 ir 1. Nesunku įsitikinti, kad aibė $\mathbb{G} = \{0,1\}$ su sudėties ir daugybos operacijomis sudaro lauką. Šis elementarus pavyzdys parodo, kad sveikųjų skaičių aibės elementus galima suskirstyti į dvi skirtingas klases, kurias galima trumpai pažymėti $\overline{0}$ ir $\overline{1}$. Prisiminkime, kad:

Apibrėžimas: Sveikas skaičius r vadinamas skaičiaus a *dalybos iš p liekana*, jeigu $0 \leq r < p$ ir p dalija skaičių $a - r$.

Tokiu būdu kiekvieną sveiką skaičių galima priskirti vienai iš šių klasių naudojant liekanos funkciją *mod*:

- Skaičius a priklauso klasei $\overline{0}$ jeigu $a \ mod \ 2 = 0$
- Skaičius a priklauso klasei $\overline{1}$ jeigu $a \ mod \ 2 = 1$

čia $a \ mod \ 2$ – yra dalybos iš 2 liekana.

Šias išvadas galima apibendrinti bet kokiam teigiamam skaičiui p. Tai reiškia, kad kiekvieną sveiką skaičių galima priskirti vienai iš klasių $\overline{0}$, $\overline{1}$, ..., $\overline{p-1}$.

Apibrėžimas: Klasė $\overline{a}$ $(0 \leq a \leq p-1)$ vadinama *liekanų klase*.

Taigi skaičius n priklauso liekanų klasei $\overline{a}$ jeigu $n \bmod p = a$. Dvi liekanų klasės $\overline{a}$ ir $\overline{b}$ yra lygios jeigu $a \bmod p = b \bmod p$. Be to yra teisingi tokie teiginiai:

Teorema: *Jeigu* $\overline{a} \neq \overline{b}$ *tai:*

1. $\overline{a} \cap \overline{b} = \emptyset$
2. *Egzistuoja lygiai p skirtingų liekanų klasių*

Šios teoremos įrodymą galima rasti šaltinyje [5].

Apibrėžimas: Liekanų klasės $\overline{0}$, $\overline{1}$, ..., $\overline{p-1}$ sudaro *pilnąją liekanų sistemą*

Liekanų klasių aibę moduliu p žymėsime $\mathbb{Z}_p$. Brūkšnelių virš liekanų klasių nerašysime. Aibę $\mathbb{Z}_p$ toliau vadinsime grupe laikydami, kad šioje aibėje sudėties ir daugybos operacijos yra apibrėžtos tokiu būdu:

$$a + b = (a + b) \bmod p \tag{1.1}$$

$$a \cdot b = (a \cdot b) \bmod p \tag{1.2}$$

Nagrinėkime ar aibė $\mathbb{Z}_p$ su sudėties ir daugybos operacijomis, apibrėžtomis formulėmis (1.1) ir (1.2) sudaro žiedą arba lauką. Kaip jau galėjome matyti aibė $\mathbb{Z}_2$ su sudėties ir daugybos operacijomis sudaro lauką. Dabar nagrinėkime aibę $\mathbb{Z}_4$. Sudėties ir daugybos lentelės atrodo taip:

1.3 lentelė

Grupės $\mathbb{Z}_4$ sudėties lentelė

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

1.4 lentelė

Grupės $\mathbb{Z}_4$ daugybos lentelė

·	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Matome, kad aibė $\mathbb{Z}_4$ su sudėties ir daugybos operacijomis sudaro žiedą, nes neegzistuoja atvikštinio elemento skaičiui 2. Šias išvadas galima apibendrinti tokiu būdu: grupė $\mathbb{Z}_p$ yra laukas, jeigu p yra pirminis skaičius; grupė $\mathbb{Z}_p$ yra žiedas, jeigu p yra sudėtinis skaičius.

Atkreipkime demėsį dar į tai, kad žiede $\mathbb{Z}_4$ $2 \cdot 2 = 0$. Matome, kad sudauginę du nenulinius elementus gauname nulinį elementą.

Apibrėžimas: Skaičių a vadinsime *nulio dalikliu* jeigu egzistuoja toks skaičius b, kad $a \cdot b = 0$.

Pavyzdys: Žiede $\mathbb{Z}_6$ egzistuoja tris nulio dalikliai: 2, 3 ir 4.

Kadangi grupėje $\mathbb{Z}_p$ daugyba yra asociatyvi, tai galima apibrėžti kėlimo laipsniu operaciją.

$$a^n = \underbrace{a \cdot a \cdot \ldots \cdot a}_{n} \ mod\ p \tag{1.3}$$

Apibrėžkime dar vieną svarbią sąvoką, susijusią su kėlimo laipsniu operacija.

Apibrėžimas: Natūraliųjų skaičių, mažesnių už p ir tarpusavyje pirminių su p skaičius yra vadinamas *Oilerio funkcija nuo p* ir žymimas $\varphi(p)$.

Oilerio funkciją ir kėlimo laipsniu operaciją sieja Oilerio teorema. Žymėkime $\gcd(a, p)$ skaičių a ir p didžiausią bendrą daliklį.

Teorema (Oilerio): Jeigu $\gcd(a, p) = 1$, tai $a^{\varphi(p)} = 1$.

Ši teorema taip pat leidžia lengvai surasti atvirkštinį elementą, nes$a^{-1} = a^{\varphi(p)-1}$

Sudarykime kėlimo laipsniu lentelę lauke $\mathbb{Z}_5$. Nulinio elemento nenagrinėsime, nes šiam elementui $0^a = \begin{cases} 1, & kai\ a = 0 \\ 0, & kai\ a \neq 0 \end{cases}$. Grupę $\mathbb{Z}_p$ be nulinio elemento žymėsime $\mathbb{Z}_p^*$.

1.5 lentelė

Lauko $\mathbb{Z}_5^*$ kėlimo laipsniu lentelė

^	1	2	3	4
1	1	1	1	1
2	2	4	3	1
3	3	4	2	1
4	4	1	4	1

Atkreipkime demėsį į elementus 2 ir 3. Matome, kad keliant šiuos skaičius laipsniais gaunami visi lauko $\mathbb{Z}_5^*$ elementai.

Apibrėžimas: Elementas g vadinamas *grupės generatoriumi*, jeigu kiekvieną grupės elementą galima išreikšti kaip fiksuotą elemento g laipsnį. Tokiu būdu sugeneruota grupė vadinama *cikline*.

Taigi skaičiai 2 ir 3 yra lauko $\mathbb{Z}_5^*$ generatoriai. Generatorių paieška grupėje yra labai svarbi, nes šie elementai yra naudojami Diffie-Hellman‘o rakto apsikeitimo protokolo konstravimui.

1.2 DIFFIE-HELLMAN‘O RAKTŲ APSIKEITIMO PROTOKOLAS

Diffie-Hellman‘o (D-H) protokolas – vienas iš plačiausiai praktikoje taikomų algoritmų. Savo protokolą Whitfield‘as Diffie ir Martin‘as Hellman‘as paskelbė 1976 metais. Tačiau nuo to laiko technika padarė didelį žingnį pirmyn. Praėjus jau daugiau kaip 30 metų nuo pratokolo atsiradimo paaiškėjo, kad jo esminis trūkumas yra dideli viešieji bei slaptieji raktai (praktikoje raktai yra apie 1024 bitų ilgio).

D-H protokolas remiasi diskretinio logaritmo uždaviniu, kuris yra formuluojamas taip: lauke $\mathbb{F}$ rasti tokį skaičių x, kad galiotų lygybė

$$g^x = a \tag{1.4}$$

čia g – yra lauko $\mathbb{F}$ generatorius, a – bet koks nenulinis lauko $\mathbb{F}$ elementas. Todėl yra būtina efektyviai apskaičiuoti lauko $\mathbb{F}$ generatorius. Dažnai lauko $\mathbb{F}$ vaidmenį atlieka laukas $\mathbb{Z}_p^*$. Pateiksime dvi teoremas ir jų įrodymus, leidžiančias efektyviai surasti generatorius bei patikrinti, ar duotasis skaičius yra generatorius.

Teorema: Tegu $\varphi(p) = p_1^{n_1} p_2^{n_2} \dots p_k^{n_k}$. Skaičius g yra lauko Z_p^* generatorius jeigu negalioja nei viena iš lygybių

$$g^{\frac{\varphi(p)}{p_i}} = 1, \qquad i = \overline{1,k} \tag{1.5}$$

Įrodymas: Akivaizdu, kad jeigu galioja bent viena iš lygybių (1.5), tai g – nėra generatorius. Atvikščiai, jeigu skaičiui g negalioja nei viena iš lygybių (1.5), tai padarę prielaidą, jog egzistuoja toks sveikas skaičius $\alpha < \varphi(p)$ kad $g^{\alpha} = 1$ gautume prieštarą, nes tokiu atvėju egzistuotų skaičiusβ toks, kad $g^{\alpha\beta} = 1$. Tačiau iš čia išplaukia, kad $\alpha = \frac{\varphi(p)}{\beta}$. Kadangi α – sveikas skaičius, taiβ yra lygus vienam iš skaičių p_i. Teorema įrodyta [4].

Teorema: *Tegu g yra grupės Z_p^* generatorius. Tada skaičius g^i yra grupės Z_p^* generatorius tada ir tik tada, kai $gcd(i, p-1) = 1$.*

Įrodymas: Įrodykime būtinumą, t.y tegu g^i yra grupės Z_p^* generatorius. Įrodykime, kad tokiu atvėju $\gcd(i, p-1) = 1$.

Tarkime priešingai, t.y $\gcd(i, p-1) \neq 1$. Tada egzistuoja du skaičiai $k < p-1$ ir $m < p-1$ tokie, kad $ik = (p-1)m$. Pakeliame skaičių g^i laipsniu k. Tada turime:

$$\left(g^i\right)^k = g^{ik} = g^{m(p-1)} = 1 \tag{1.6}$$

Tačiau pagal (1.6) lygybę g^i nėra grupės Z_p^* generatorius. Gavome prieštarą.

Įrodykime pakankamumą, t.y tegu $\gcd(i, p-1) = 1$. Įrodykime, kad g^i yra grupės Z_p^* generatorius.

Tarkime priešingai, t.y g^i nėra grupės Z_p^* generatorius. Tada egzistuoja toks skaičius $k < p-1$, kad $\left(g^i\right)^k = g^{ik} = 1$. Tačiau kadangi $g^{p-1} = 1$, tai egzistuoja toks skaičius $m < p-1$, kad būtų teisinga (1.6) lygybė. Tada sulyginę laipsnius gauname $ik = (p-1)m$. Tačiau ši lygybė reiškia, kad $\gcd(i, p-1) \neq 1$. Gavome prieštarą. Teorema įrodyta.

Išvada: Bendras generatorių skaičius lauke yra $\varphi(\varphi(p))$.

Pavyzdys: Nagrinėkime lauką $\mathbb{Z}_{41}$. Kadangi $\varphi(41) = 40 = 2^3 \cdot 5$, tai skaičius g yra lauko $\mathbb{Z}_{41}$ generatorius, jeigu $g^8 \neq 1$ ir $g^{20} \neq 1$. Mažiausias toks skaičius šiame lauke yra 6. Skaičius $6^3 = 11$ taip pat yra generatorius, nes $\gcd(3, 40) = 1$.

Kadangi g yra grupės generatorius, tai lygtis (1.4) visada turi bent vieną sprendinį. Paprasčiausias šio sprendinio paieškos būdas yra pilnas perrinkimas. Tačiau kai skaičius p yra didelis atlikti tokią paiešką yra pakankamai sudėtinga, nes tam prireiktų daug laiko. Trumpai apžvelgsime kitus lygties (1.4) sprendimo algoritmus [7].

Tokio sprendimo algoritmo sudėtingumas yra $\mathcal{O}(p^{\frac{1}{2}}\log p)$

- Priskirti $h = \left[p^{\frac{1}{2}}\right] + 1$
- Rasti $c = g^h$
- Sudaryti reišknio c^u kai $1 \leq u \leq h$ reikšmių lentelę ir ją surušiuoti
- Sudaryti reišknio $a \cdot g^v$ kai $0 \leq u \leq h$ reikšmių lentelę ir ją surušiuoti
- Rasti vienodus elementus iš abiejų lentelių. Šiems elementams $c^u = a \cdot g^v$
- Grąžinti $x = (hu - v) mod\ (p-1)$

Taip pat yra žinomi ir kiti lygties (1.4) sprendimo algoritmai. Šie algoritmai dažniausiai yra eksponentiniai arba subeksponentiniai. Eksponentinių algoritmų pavyzdžiai yra Shanks‘o algoritmas (angl. baby-step giant-step), kurio sudėtingumas yra $\mathcal{O}(\sqrt{p})$ ir Pohling‘o ir Hellman‘o algoritmas, kuris yra ypač efektyvus, kai skaičiaus p Oilerio funkcijos faktorizacijos $\varphi(p) = p_1^{n_1} p_2^{n_2} \dots p_k^{n_k}$ pirminiai skaičiai p_i yra pakankamai maži. Pohling‘o ir Hellman‘o algoritmo sudėtingumas yra $\mathcal{O}(\sum_{i=1}^{k} n_i(\log p + p_i))$. Subeksponentiniai algoritmai turi sudėtingumą $\mathcal{O}(c(\log p \log\log p)^d)$, čia c ir d yra tam tikros konstantos. Algoritmas yra tuo efektyvesnis, kuo arčiau c yra prie 1, o d – prie 0. Tokių algoritmų pavyzdžiai yra COS (autoriai – Don Coppersmith, Andrew Odlyzko, Richard Schroeppel) algoritmas ir skaičių lauko gardelės algoritmas, kuris šiuo metu yra plačiausiai taikomas pirminiams skaičiams $p \geq 10^{100}$.

D-H raktų apsikeitimų protokolas atrodo taip:

- Aldona ir Bronius susitaria dėl viešųjų parametrų: lauko $\mathbb{Z}_p$ ir šio lauko generatoriaus g
- Aldona pasirenka skaičių x ir apskaičiuoja $a = g^x$. Skaičių a Aldona siunčia Broniui
- Bronius pasirenka skaičių y ir apskaičiuoja $b = g^y$. Skaičių b Bronius siunčia Aldonai
- Aldona ir Bronius apskaičiuoja bendrą slaptą raktą $K = (g^a)^b = (g^b)^a$

Taigi nors šiuolaikiniai lygties (1.4) sprendimo algoritmai negali surasti sprendinio per polinominį laiką teoriškai yra įrodyta, kad naudojant kvantinius kompiuterius toks algoritmas gali butį realizuotas, o tai reikštų, jog D-H protokolas bei kiti kriptografiniai uždaviniai, paremti diskretinio logaritmo uždaviniu taptų neefektyvūs.

Dėl šių priežasčių jau nuo pačio D-H algoritmo atsiradimo Lietuvos ir užsienio kriptografai bando įvairiais būdais jį tobulinti. Mokslininkų keliami tikslai yra ne tik raktų ilgio sumažinimas, bet ir didesnio protokolo saugumo užtikrinimas. Taip 1985 metais atsirado eliptinių kreivių D-H protokolas kuris remiasi diskretinio logaritmo uždaviniu ant eliptinių kreivių. Šiuo metu laikoma, kad toks D-H

protokolas pasižymi didesniu saugumu, nes subeksponentinių algoritmų, leidžiančių efektyviai spręsti tokį uždavinį iki šiol nėra [7].

Vienas iš galimų D-H protokolo tobulinimo būdų yra naujų patikimų vienkrypčių funkcijų kūrimas. Šiame darbe nagrinėsime vieną iš neseniai pasiūlytų vienkrypčių funkcijų.

1.3 MATRICINIO LAIPSNIO FUNKCIJA

Profesorius Eligijus Sakalauskas 2007 metais pasiūlė naują funkciją kvadratinių matricų algebroje, kuria pavadino *matricinio laipsnio funkcija* [1]. Ši funkcija yra žymima

$$B = A \lhd X^1 \tag{1.7}$$

Matricos B elementai yra apskaičiuojami pagal formule (1.8)

$$b_{ij} = \prod_{k=1}^{m} a_{ik}^{x_{kj}} \tag{1.8}$$

čia m – matricų eilė.

Pastebėkime, kad matricinio laipsnio funkcija tenkina šias lygybes:

$$A \lhd (XY) = A \lhd (X \lhd Y) = A \lhd X \lhd Y \tag{1.9}$$

$$A \lhd X \lhd X^{-1} = A \lhd (XX^{-1}) = A \lhd I = A \tag{1.10}$$

Lygybės (1.9) ir (1.10) parodo, kad matricinio laipsnio operacija yra apibrėžta korektiškai.

Kaip matome matricinio laipsnio funkcija yra vienkryptė, nes ji tam tikra prasme apibendrina diskretinio logaritmo uždavinį. Tačiau įrodymo, jog ši funkcija priklauso NP-pilnųjų funkcijų klasei dar nėra. Esminiai matricinio laipsnio privalumai yra šie [2]:

1. Matricinio laipsnio funkcija yra atspari pilno perrinkimo atakai. Šis rezultatas pasiekiamas žymiai greičiau lyginant su skaičiais.
2. Funkcija yra apskaičiuojama pakankamai greitai
3. Nereikia atlikti veiksmų su labai dideliais skaičiais

Matricinio laipsnio funkcija pirmą karta buvo panaudota Eligijaus Sakalausko ir Kęstučio Lukšio darbe [1] S-blokų (angl. S-box) konstravimui. Šis darbas buvo pristatytas 2008 metų konferencijoje Varnos mieste (Bulgarija).

[1] Darbe lygybė (1.7) turi dvi interpretacijas: funkcija ir lygtis

Panagrinėkime kai kurias matricinio laipsnio funkcijos algebrines savybes. Kaip jau mateme egzistuoja neutralusis elementas $X = I$. Be to egzistuoja elementai $A = \mathbf{1}$ ir $X = \mathbf{0}$, čia $\mathbf{1}$ – matrica, kurios visi elementai yra lygūs vienetui, $\mathbf{0}$ – nulinė matrica, kuriems yra teisingos lygybės:

$$\begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix} \triangleleft X = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix} \tag{1.11}$$

$$A \triangleleft \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix} \tag{1.12}$$

Dabar paanalizuokime (1.7) funkcijos elementą b_{ij}, kuris yra apskaičiuojamas pagal lygybę (1.8). Kadangi šis elementas yra apskaičiuojamas naudojant tik daugybos bei kėlimo laipsniu operacijas, tai jeigu nors vienas iš daugiklių yra lygus 0, tai ir $b_{ij} = 0$. Gavome dvi labai svarbias išvadas:

1 Jeigu bent vienas iš matricos A elementų yra lygus 0, tai visi matricos B atsitinkamos eilutės elementai yra lygūs 0
2 Jeigu kiekvienoje matricos A eilutėje yra bent vienas 0, tai matrica B yra nulinė

Šios dvi išvados galioja kiekvienoje grupėje $\mathbb{Z}_p$. Tačiau jeigu p yra sudėtinis skaičius, tai žiede $\mathbb{Z}_p$ egzistuoja nulio dalikliai. Apibrėžkime tokią sąvoka:

Apibrėžimas: Elementų (a, b) pora vadinama *nulio daliklių pora*, jeigu $(ab)mod(p) = 0$.

Pavyzdys: Žiedas $\mathbb{Z}_8$ turi tris nulio daliklius: 2, 4, 6. (2, 4) ir (4, 6) yra nulio daliklių poros, bet (2, 6) nėra nulio daliklių pora, nes $(2 \cdot 6)mod(8) = 4$.

Pastebėkime, kad nulio daliklių poros vienoje matricos A eilutėje įtakoja matricos B elementus taip pat kaip ir nuliniai elementai.

Pastaba: Kadangi $0^0 = 1$, tai gautos išvados galioja, jeigu matrica X neturi nulinių elementų tose vietose, kad naudojant funkciją (1.7) nulinis elementas būtų keliamas nuliniu laipsniu.

Pavyzdžiai: 1) Nagrinėkime lauką $\mathbb{Z}_5$.

a) Matrica A turi nulinį elementą, o matrica X neturi

$$\begin{pmatrix} 1 & 0 & 2 \\ 4 & 3 & 1 \\ 2 & 3 & 4 \end{pmatrix} \triangleleft \begin{pmatrix} 3 & 4 & 4 \\ 2 & 1 & 3 \\ 3 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 3 & 2 \\ 3 & 3 & 3 \end{pmatrix}$$

Matome, kad visi pirmos eilutės elementai yra lygūs 0

b) Matrica A turi nulinį elementą, o matricos X elementas $x_{21} = 0$.

$$\begin{pmatrix} 1 & 0 & 2 \\ 4 & 3 & 1 \\ 2 & 3 & 4 \end{pmatrix} \triangleleft \begin{pmatrix} 3 & 4 & 4 \\ 0 & 1 & 3 \\ 3 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 0 \\ 4 & 3 & 2 \\ 2 & 3 & 3 \end{pmatrix}$$

Matome, kad $b_{11} = 3$

2) Nagrinėkime žiedą $\mathbb{Z}_6$. Šiame žiede egzistuoja nulio daliklių pora (2, 3)

a) Matrica A turi nulio daliklių pora kiekvienoje eilutėje, o matrica X neturi nulinių elementų

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix} \triangleleft \begin{pmatrix} 4 & 1 & 4 \\ 5 & 5 & 4 \\ 3 & 5 & 4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Matome, kad matrica B yra nulinė.

b) Matrica A turi nulio daliklių pora kiekvienoje eilutėje, o matrica X turi nulinį elementą

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix} \triangleleft \begin{pmatrix} 4 & 1 & 4 \\ 5 & 5 & 0 \\ 3 & 5 & 4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 3 \\ 0 & 0 & 4 \\ 0 & 0 & 0 \end{pmatrix}$$

Matome, kad $b_{13} = 3$ ir $b_{23} = 4$.

Žiede $\mathbb{Z}_p$ sudarykime visas įmanomas nulio daliklių poras. Kiekvienoje poroje išbraukime tą elementą, kuris dažniau pasitaiko porose.

Apibrėžimas: Išbrauktų elementų aibę vadinsime *nulio daliklių branduoliu*

Pavyzdys: Nagrinėkime žiedą $\mathbb{Z}_{15}$. Turime tokias nulio daliklių poras: (3, 5), (3, 10), (6, 5), (6, 10), (9, 5), (9, 10), (12, 5), (12, 10). Matome, kad dažniausiai šiose porose pasitaiko skaičiai 5 ir 10. Taigi nulio daliklių branduolys yra aibė (5, 10).

Pastebėkime, kad pašalinus nulio daliklių branduolį iš grupės $\mathbb{Z}_p$, joje jau nebeegzistuoja nulio dalikliai.

Matome, jog skirtingai nuo įprasto D-H protokolo norint konstruoti D-H protokolą, paremtą matricinio laipsnio funkcija reikalavimas, kad grupės $\mathbb{Z}_p$ parametras būtu pirminis skaičius nėra būtinas.

1.4 DIFFIE-HELLMAN‘O PROTOKOLAS, PAREMTAS MATRICINIO LAIPSNIO FUNKCIJA

2007 metais Sakalauskas panaudojo matricinio laipsnio funkciją D-H tipo protokolo konstravimui [2]. Algoritmas atrodo taip:

- Aldona ir Bronius susitaria dėl viešųjų parametrų: grupės $\mathbb{Z}_p$, matricos A ir aibės $\mathcal{R}$, kurios elementai yra komutuojančios tarpusavyje matricos
- Aldona pasirenka matricą $X \in \mathcal{R}$ ir apskaičiuoja matricą $B_X = A \lhd X$. Matricą B_X Aldona siunčia Broniui
- Bronius pasirenka matricą $Y \in \mathcal{R}$ ir apskaičiuoja matricą $B_Y = A \lhd Y$. Matricą B_Y Bronius siunčia Aldonai
- Aldona ir Bronius apskaičiuoja bendrą slaptą raktą $K = A \lhd X \lhd Y = A \lhd Y \lhd X$

Matome, kad norint nulaužti protokolą, t.y apskaičiuoti slaptą raktą K reikia spręsti (1.7) lygtį. Tačiau teoriniai metodai, leidžiantys apskaičiuoti matricą X yra nežinomi. Dar viena problema yra ta, kad ne visi (1.7) lygties sprendiniai priklauso aibei $\mathcal{R}$. Tai reiškia, kad reikia spręsti papildomą lygtį

$$MX = XM \tag{1.13}$$

čia M yra matrica, generuojanti aibę $\mathcal{R}$.

1.5 DARBO UŽDUOTIS

Šio darbo pagrindinis uždavinys yra ištirti lygčių sistemą

$$\begin{cases} A \lhd X = B \\ MX = XM \end{cases} \tag{1.14}$$

Darbo tikslai yra šie:

1. Atlikti lygčių sistemos (1.14) sprendinių skaičiaus analizę
2. Nustatyti kaip priklauso sprendinių skaičius nuo grupės $\mathbb{Z}_p$ parametro p
3. Nustatyti kaip priklauso sprendinių skaičius nuo matricų eilės m

1.6 DARBO UŽDUOTIES ANALIZĖ

Lygčių sistemą (1.14) galima spręsti dviem būdais: pirmiausia spręsti matricinio laipsnio lygtį, o vėliau tikrinti kurie iš gautų sprendinių tenkina komutatyvumo sąlyga, arba spręsti komutatyvumo lygtį, o vėliau tikrinti kurie iš gautų sprendinių tenkina matricinio laipsnio funkciją.

1.6.1 MATRICINIO LAIPSNIO LYGTIS

Nagrinėkime matricinio laipsnio lygtį. Tiesiogiai iš apibrėžimo turime, kad matricos B stulpeliai gali būti apskaičiuojami atskirai. Taigi lygties (1.7) sprendimas yra ekvivalentus lygčių sistemai, sudarytai iš m lygčių, čia m – matricų eilė

$$A \lhd X_{.j} = B_{.j}, \quad j = \overline{1,m} \tag{1.15}$$

čia taškas reiškia, kad nagrinėjamas j-asis stulpelis.

Lygties (1.15) sprendinių aibę prie fiksuotos j reikšmės pažymėkime S_j. Tada visus lygties (1.7) sprendinius galima gauti naudojant sąjungos operaciją

$$S = \bigcup_{j=1}^{m} S_j \tag{1.16}$$

Pastaba: Atliekant sąjungos operaciją reikia iš kiekvienos aibės S_j imti po vieną elementą.

Matome, kad (1.7) lyčiai išspresti turime du kartus atlikti pilną perrikimą: vieną kartą skaičiuodami aibes S_j ir antrą kartą formuodami galutinę sprendinių aibę S. Turime, kad visoms aibėms S_j apskaičiuoti reikia m kartų patikrinti p^m skirtingų variantų. Tačiau toks lygčių sistemos (1.14) sprendimo būdas yra neracionalus, nes jam atlikti prireiktų daug laiko.

1.6.2 KOMUTATYVUMO LYGTIS

Prieš nagrinėjant komutatyvumo lygtį (1.13) prisiminkime kai kurias matricų teorijos sąvokas.

Apibrėžimas: Matrica $A(\lambda)$ vadinama *λ-matrica* jeigu visi jos elementai yra daugianariai nuo λ.

Tegu λ-matrica turi rangą r. Visų i-osios eilės minorų didžiausią bendrą daliklį žymėkime $D_i(\lambda)$. Taip pat tegu $D_0(\lambda) = 1$.

Apibrėžimas: Daugianariai $E_i(\lambda) = \frac{D_i(\lambda)}{D_{i-1}(\lambda)}, i = \overline{1,r}$ vadinami *invariantiniais matricos* $A(\lambda)$ *daugianariais.*

Lygties (1.7) sprendinių skaičių galima apskaičiuoti naudojant teoremą [3]:

Teorema: *Tiesiškai nepriklausomų matricų, tenkinančių lygtį (1.7), skaičius yra lygus*

$$N = m_1 + 3m_2 + \cdots + (2k-1)m_k \tag{1.17}$$

čia $m_1, m_2, \dots, m_k$ – invariantinių matricos M daugianarių laipsniai ir $m_1 \geq m_2 \geq \cdots \geq m_k > m_{k+1} = 0$.

Pastebėkime, kad

$$\sum_{i=1}^{k} m_i = m \tag{1.18}$$

Teoremos įrodymas yra pateiktas [3] šaltinyje.

Reikia pabrėžti, kad pateiktas įrodymas parodo, kad lygybė (1.17) galioja bet kuriame uždarame algebriniame lauke, o tai reiškia kad ši lygybė galioja ir lauke $\mathbb{Z}_p$. Taip pat iš lygybės (1.17) išplaukia, kad

$$N \geq m \tag{1.19}$$

Nelygybė (1.19) tampa lygybe tada ir tik tada kai $k = 1$. Tai reiškia, kad $N = m$ tada ir tik tada kai visi invariantiniai matricos M daugianariai yra poromis pirminiai.

Tegu turime daugianarį $P(x)$. Tada matricos M ir $P(M)$ komutuoja. Kyla klausimas: kokiu atvėju galima visas matricas, komutuojančias su matrica M, išreikšti daugianariais nuo M? Atreipkime demėsį į tai, kad tokiu atvėju bet kuri matrica, tenkinanti lygtį (1.13) būtų matricų $I, M, M^2, \dots, M^{m_1-1}$ tiesinis darinys. Tačiau tada

$$N \leq m \tag{1.20}$$

Palyginę (1.19) ir (1.20) turime, kad $N = m$, t.y visas matricas, komutuojančias su matrica M, galima išreikšti daugianariais nuo M tik tuo atvėju, kai visi invariantiniai matricos M daugianariai yra poromis pirminiai. Ši lygybė tenkinama, kai visos matricos M tikrinės reikšmės yra skirtingos.

Tarkime, kad $\mathbb{Z}_p$ yra laukas ir pareikalaukime, kad visos matricos M tikrinės reikšmės yra skirtingos. Lauke $\mathbb{Z}_p$ galima sudaryti lygiai p^m skirtingų $(m-1)$-osios eilės polinomų. Taigi tiek yra ir lygties (1.13) sprendinių.

Reikia pastebėti, kad gauti rezultatai galioja ir tuo atvėju, kai $\mathbb{Z}_p$ yra žiedas, jeigu matrica M tenkina minėtą sąlygą. Tačiau priešingu atvėju skirtingų matricų, komutuojančių su M, skaičius nebūtinai yra lygus p^m.

Pavyzdys: Nagrinėkime žiedą $\mathbb{Z}_6$. Tegu $M = \begin{pmatrix} 4 & 4 \\ 0 & 4 \end{pmatrix}$. Ši matrica turi antro kartotinumo tikrinę reikšmę $\lambda = 4$. Matricų, komutuojančių su M, skaičius yra lygus 144.

Grįžkime prie lygčių sistemos (1.14) bei prisiminkime skyrelio 1.6.1 išvadas. Pradedant sprendimą nuo lygties (1.13) turime atlikti vieną pilno perrinkimo operaciją sudarant visus įmanomus $(m-1)$-osios eilės polinomus, o vėliau patikrinti ar yra tenkinama (1.7) lygybė. Šiam būdui atlikti reikia žymiai mažiau laiko todėl darbe buvo pasirinktas būtent šis būdas.

1.6.3 LYGČIŲ SISTEMOS PARAMETRŲ APRIBOJIMAI

Apibendrinkime gautas teorines išvadas ir suformuluokime reikalavimus lygčių sistemos parametrams: matricoms A ir M, o taip pat grupės $\mathbb{Z}_p$ parametrui p.

1. Parametras p turi būti arba pirminis skaičius, arba dviejų pirminių skaičių sandauga, t.y $p = p_1 p_2$.
2. Matrica A turi būti pilnai užpildyta, t.y $\forall i, \forall j\ a_{ij} \neq 0$. Taip pat matrica A neturi turėti nulio daliklių branduolio elementų, jeigu šie elementai egzistuoja.
3. Matricos M visos tikrinės reikšmės turi būti skirtingos

Pirmasis apribojimas yra reikalingas nes tokiu atvėju kai p – piminis skaičius, grupė $\mathbb{Z}_p$ yra laukas, o kai p – sudėtinis skaičius $\mathbb{Z}_p$ yra žiedas. Be to sudėtiniai skaičiai, turintys minėtą pavidalą, yra sunkiausiai faktorizuojami kai $p_1 \approx p_2$. Taip pat lengvai surandamas nulio daliklių branduolys.

Trečiojo apribojimo sąlygas išpildysime naudojant spektrinį matricos išdėstymą:

$$M = T\Lambda T^{-1} \tag{1.21}$$

čia Λ – diagonalioji tikrinių reikšmių matrica, T – tikrinių vektorių matrica.

Iš trečiojo apribojimo sąlygų taip pat išplaukia akivaizdi išvada, kad $m < p$.

1.6.4 PROGRAMINĖS ĮRANGOS PASIRINKIMAS

Darbo užduoties sprendimui buvo pasirinktas matematinis paketas MATLAB R2007b. Šis paketas yra ypač patogus tuo, kad yra specialiai skirtas darbui su matricomis. MATLAB‘o terpėje galima greitai generuoti matricas bei atlikinėti įvairiausius veiksmus su jomis (pvz. atvirkštinės matricos skaičiavimas). Tai yra pasiekiama standartinių funkcijų pagalba, kurių sąrašas yra pateiktas 1.6 lentelėje.

1.6 lentelė

Kai kurios standartinės MATLAB‘o funkcijos

Funkcija	Aprašymas
$size(A)$	Grąžina matricos A eilučių ir stulpelių skaičių
$length(A)$	Grąžina vektoriaus A ilgį
$transpose(A)$ arba A'	Transponuoja matricą A
$inv(A)$	Gražina matricos A atvirkštinę matricą, jeigu ji egzistuoja, priešingu atvėju gražina klaidą
$mod(A,p)$	Grąžina matricą A, kurios elementai priklauso grupei $\mathbb{Z}_p$
$[A;B]$	Apjungia matricas A ir B vertikaliai
$[A\ B]$	Apjungia matricas A ir B horizontaliai
$randint(n,n,p)$	Generuoja sveikųjų skaičių matricą, kurios formatas yra $n \times n$, o elementai nuo 0 iki p-1
$zeros(n)$	Generuoja n-tos eilės nulinę matricą
$eye(n)$	Generuoja n-tos eilės vienetinę matricą
$A(i,:)$	Išskiria matricos A i-tąją eilutę

Taip pat šis programinis paketas leidžia lengvai rašyti funkcijas, peržiurėti jau parašytas funkcijas ir atlikti jų testavimą. Visus duomenis ir rezultatus galima iš karto pamatyti komandiniame lange. Visa sąsaja su vartotoju gali būti vykdoma naudojant tik komandinį langą, todėl programai nereikalingas interfeisas.

2. TIRIAMOJI DALIS

2.1 LYGČIŲ SISTEMOS SPRENDINIŲ PAIEŠKOS ALGORITMAS

Sudarykime darbo užduoties sprendimo algoritmą. Šį algoritmą padalinsime į dvi dalis: pirmoji dalis bus skirta lygčių sistemos (1.14) parametrų generavimui (punktai 1–5), o atroje dalyje aprašysime lygčių sistemos (1.14) sprendimo algoritmą (punktai 6–9).

1. Pasirenkame grupės $\mathbb{Z}_p$ parametrą, tenkinantį skirelio 1.6.3 apribojimus.
2. Grupėje $\mathbb{Z}_p$ generuojame matricą A, tenkinančią skirelio 1.6.3 apribojimus.
3. Grupėje $\mathbb{Z}_p$ generuojame matricą X.
4. Apskaičiuojame matricą B naudojant lygybę (1.7).
5. Apskaičiuojame matricą M kaip m-tos eilės polinomą $P(X)$. Polinomo koeficientai priklauso grupei $\mathbb{Z}_p$ ir yra atsitiktiniai.
6. Altiekame m-tos eilės polinomo koeficientų pilną perrinkimą.
7. Prie kiekvieno polinomo koeficientų rinkinio apskaičiuojame matricą $P(M)$.
8. Tikriname, ar yra tenkinama lygybė (1.7). Jeigu lygybė yra tenkinama, tai $P(M)$ yra sprendinys.
9. Jeigu dar ne visi koeficientų rinkiniai buvo perrinkti, tai grįžtame prie 7 punkto. Priešingu atvėju turime visus lygčių sistemos (1.14) sprendinius.

2.2 MATRICINIO LAIPSNIO LYGTIS

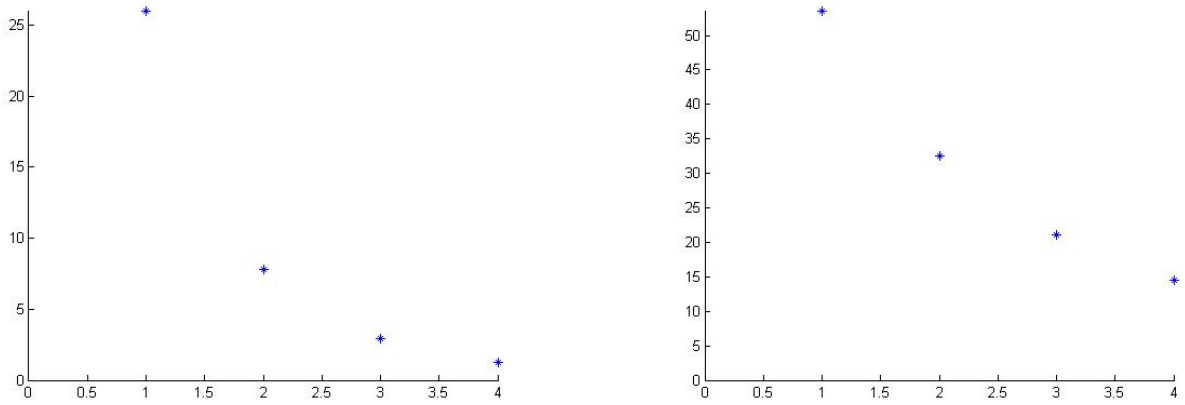
Prieš nagrinėdami lygčių sistemą (1.14) panagrinėkime kaip priklauso lygties (1.7) sprendinių skaičius nuo matricos formato. Generuokime 100 ketvirtos eilės matricų, bei nagrinėkime jų vieną, dvi, tris, keturias matricos eilutes. Matrica X šiuo atvėju yra vektorius stulpelis. Stebėkime sprendinių skaičiaus vidurkio santykį su bendru variantų skaičiumi. Šį santykį išreiškime procentais. T.y šiuo atvėju nagrinėjame lygtį (1.15) prie fiksuotos j reikšmės. Nagrinėkime du atvėjus: pirmuoju atvėju tirsime lygtį (1.15) lauke $\mathbb{Z}_5$, o antruoju atvėju – žiede $\mathbb{Z}_6$. Gautus rezultatus surašykime į lentelę 2.1.

2.1 lentelė

Sprendinių skaičiaus priklausomybė nuo matricos A eilučių skaičiaus

Eilučių skaičius	$p = 5$	$p = 6$
1	26.0160	53.6165
2	7.7856	32.6003
3	2.9200	21.1173
4	1.2880	14.5123

Grafinis vaizdas yra pateiktas 2.1a ir 2.1b paveisluose. Abscisių ašyje pavaizduotos matricos A eilučių skaičiaus reikšmės, o ordinačių ašyje – procentinė dalis.



2.1 pav. Sprendinių skaičiaus priklausomybė nuo matricos A eilučių skaičiaus, kai

a) p = 5 **b) p = 6**

Grafikai 2.1a ir 2.1b parodo formulės (1.16) praktinį neefektyvumą. Taip yra nes lygties (1.15) sprendinių skaičius prie fiksuotos j reikšmės yra didelis, todėl didėjant matricos A eilei sąjungos operacija yra atliekama labai ilgai.

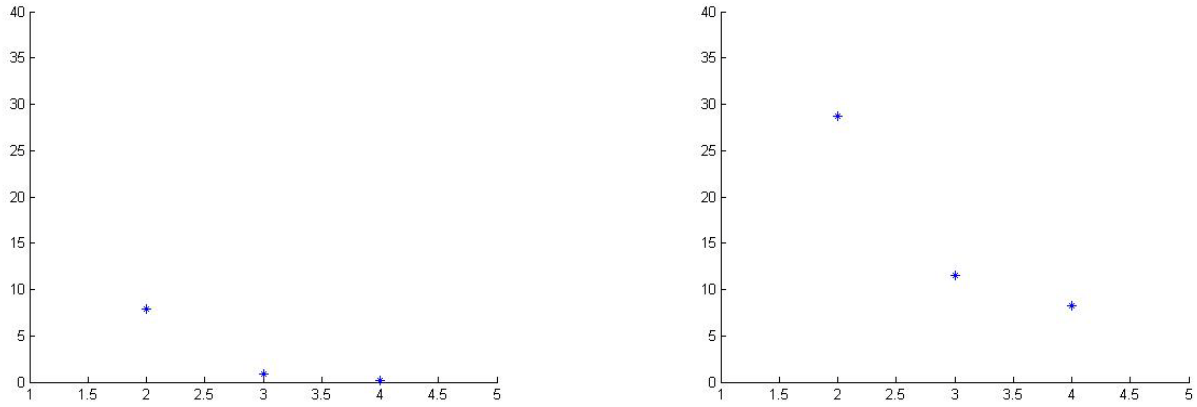
2.3 DARBO UŽDUOTIES SPRENDIMAS

Kaip jau buvo parodyta skyreliuose 1.6.1 ir 1.6.2 lygčių sistemą (1.14) pradėsime spręsti nuo komutatyvumo lygties. Šiame skyrelyje parodysime kaip priklauso nagrinėjamos lygčių sistemos sprendinių skaičius nuo esminių parametrų: matricų eilės m bei grupės $\mathbb{Z}_p$ parametro p. Taip pat nubraižysime priklausomybių grafikus ir mažiausių kvadratų metodu įvertinsime aproksimacijų parametrus. Nagrinėsime trijų tipų aproksimacijas:

1. $f(x) = ax^b$
2. $f(x) = a\exp(-b(x-c))$
3. $f(x) = a\exp(-b(x-c)^d)$

2.3.1 LYGČIŲ SISTEMOS SPRENDINIŲ SKAIČIAUS PRIKLAUSOMYBĖ NUO MATRICŲ EILĖS

Nagrinėkime kaip priklauso lygčių sistemos (1.14) sprendinių skaičius nuo matricų A ir M eilės. Išskirkime du atvėjus: kai parametras p yra pirminis skaičius ir kai p yra sudėtinis. Generuokime lygčių sistemos parametrus A ir M 100 kartų prie kiekvienos m reikšmės (nuo 2 iki 4) kai $p = 5$ ir kai $p = 6$. Spreskime gautas lygčių sistemas. Nagrinėkime sprendinių skaičiaus santykį su bendrų komutuojančių matricų skaičiumi. Nagrinėjamą santykį išreiškime procentais. Abscisių ašyje vaizduosime matricų eilės reikšmes, o ordinačių ašyje vaizduosime nagrinėjamą santykį.



2.2 pav. Sprendinių skaičiaus priklausomybė nuo matricos eilės

a) kai p = 5 **b) kai p = 6**

Nagrinėkime atvėjį, kai $p = 5$. Atlikime minėtą tyrimą 10 kartų ir sudarykime paklaidų lentelę, kuri leistų iš nagrinėjamų trijų tipų aproksimacijų išrinkti tiksliausią. Vidutinių kvadratinių paklaidų lentelė atrodo taip:

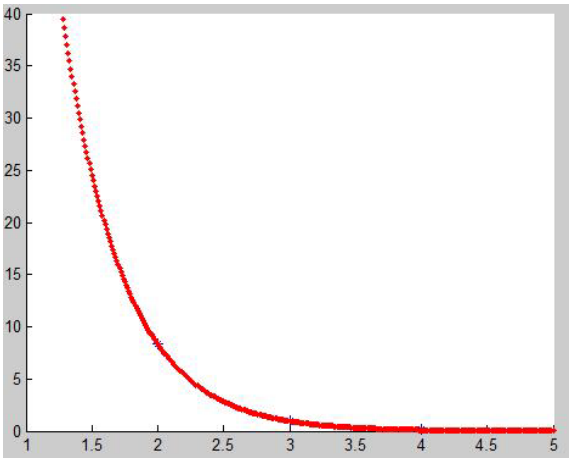
2.2 lentelė

Vidutinių kvadratinių paklaidų kai p = 5 reikšmes lentelė

Bandymo numeris	Aproksimacijos		
	ax^b	$a\exp(-b(x-c))$	$a\exp(-b(x-c)^d)$
1	5.0822e-004	8.8619e-004	0.0196
2	2.7310e-006	0.0022	0.0230
3	1.1075e-004	0.0013	0.0122
4	1.7151e-005	0.0018	0.0152
5	6.0127e-006	0.0023	0.0218
6	1.3546e-004	0.0014	0.0207
7	7.0144e-005	0.0016	0.0198
8	1.6708e-004	0.0013	0.0183
9	0.0010	5.4225e-004	0.0194
10	8.1588e-005	0.0016	0.0188
Vidurkis	2.10e-004	0.00149	0.01888

Pastaba: Lentelėje užrašas 5.0822e-004 reiškia skaičių 5.0822×10^{-4}

Matome, kad šiuo atvėju mažiausią vidutinę kvadratinę paklaidą turi pirmoji aproksimacija. Tačiau pasirinksime bei nubraižysime antrąją aproksimaciją. Tokio pasirinkimo priežastį paaiškinsime, kai išnagrinėsime antrąjį atvėjį.



2.3 pav. Sprendinių skaičiaus priklausomybė nuo matricų eilės kai $p = 5$. Aproksimacija $a\exp(-b(x-c))$

Turime tokius aproksimacijos parametrų įverčius: $\hat{a} = 8.4766$, $\hat{b} = 2.1641$. $\hat{c} = 1.9913$. Vidutinė kvadratinė paklaida yra 8.8619×10^{-4}.

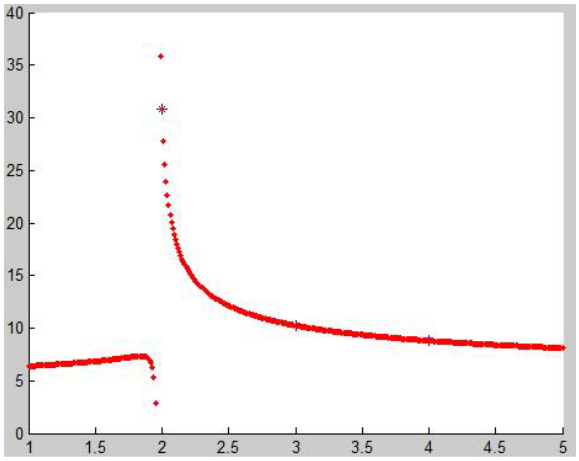
Nagrinėkime atvėjį, kai $p = 6$. Atlikime minėtą tyrimą 10 kartų ir sudarykime paklaidų lentelę, kuri leistų iš nagrinėjamų trijų tipų aproksimacijų išrinkti tiksliausią. Vidutinių kvadratinių paklaidų lentelė atrodo taip:

2.3 lentelė

Vidutinių kvadratinių paklaidų kai p = 6 reikšmes lentelė

Bandymo numeris	Aproksimacijos		
	ax^b	$a\exp(-b(x-c))$	$a\exp(-b(x-c)^d)$
1	2.8832	5.4792	2.2079e-014
2	0.9099	2.3427	5.0446e-016
3	2.9475	5.7010	1.6447e-012
4	1.3855	3.3790	4.1328e-016
5	1.6254	3.6816	3.2339e-016
6	1.8606	4.0675	1.7189e-014
7	1.3470	3.1728	1.3670e-014
8	2.1337	4.4230	2.5571e-012
9	0.4974	1.4639	6.6141e-015
10	1.9538	4.1859	8.1717e-015
Vidurkis	1.7544	3.78966	4.27e-013

Šiuo atvėju mažiausią paklaidą duoda trečioji aproksimacija. Nubraižykime šią aproksimaciją.



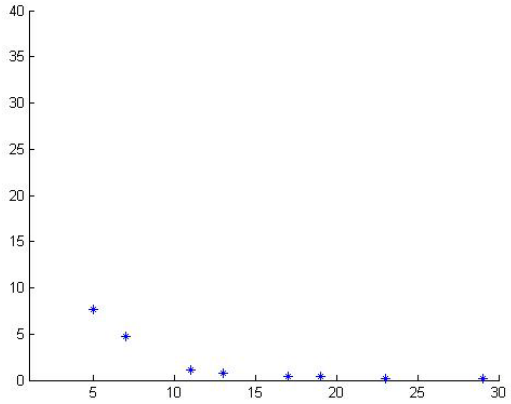
2.4 pav. Sprendinių skaičiaus priklausomybė nuo matricų eilės kai $p = 6$. Aproksimacija $a\exp(-b(x-c)^d)$

Turime tokius aproksimacijos parametrų įverčius: $\hat{a} = 2.5362$, $\hat{b} = -1.4057$. $\hat{c} = 1.9652$ ir $\hat{d} = -0.1712$. Vidutinė kvadratinė paklaida yra 6.5098×10^{-8}.

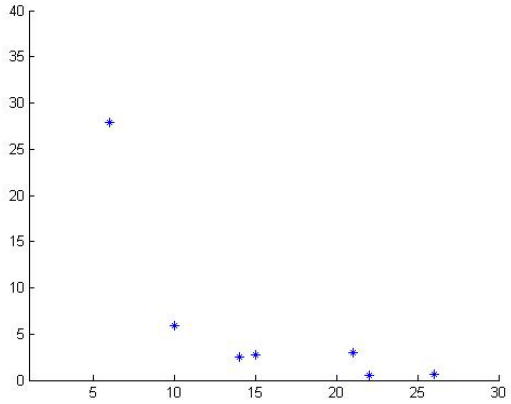
Trumpai paaiškinsime, kodėl pirmuoju atvėju buvo pasirinkta antroji aproksimacija. Kaip jau galėjome įsitikinti antruoju atvėju trečioji aproksimacija duoda mažiausią vidutinę kvadratinę paklaidą. Be to šios aproksimacijos vidutinė kvadratinė paklaida ryškiai skiriasi nuo kitų dviejų aproksimacijų. Taigi yra akivaizdu, jog norint palyginti šiuos du atvėjus tarpusavyje žymiai geriau yra naudoti būtent eksponentinį dėsningumą.

2.3.2 LYGČIŲ SISTEMOS SPRENDINIŲ SKAIČIAUS PRIKLAUSOMYBĖ NUO PARAMETRO P

Nagrinėkime kaip priklauso lygčių sistemos (1.14) sprendinių skaičius nuo parametro p. Išskirkime du atvėjus: kai parametras p yra pirminis skaičius ir kai p yra sudėtinis. Generuokime lygčių sistemos parametrus A ir M 100 kartų prie kiekvienos p reikšmės kai $m = 2$. Spreskime gautas lygčių sistemas. Nagrinėkime sprendinių skaičiaus santykį su bendrų komutuojančių matricų skaičiumi. Nagrinėjamą santykį išreiškime procentais. Abscisių ašyje vaizduosime matricų eilės reikšmes, o ordinačių ašyje vaizduosime nagrinėjamą santykį.



2.5a pav. Sprendinių skaičiaus priklausomybė nuo pirminių p, kai n = 2



2.5b pav. Sprendinių skaičiaus priklausomybė nuo sudėtinių p, kai n = 2

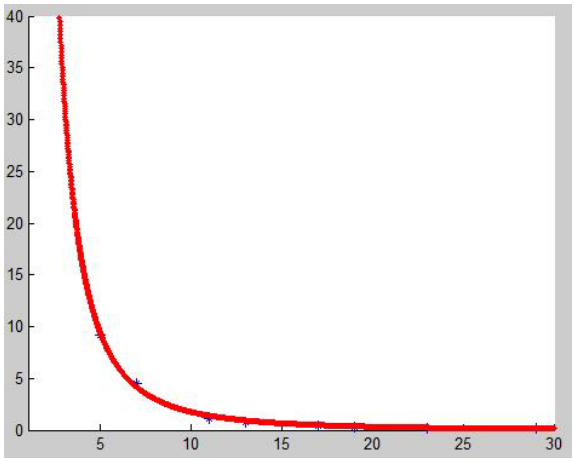
Nagrinėkime atvėjį, kai parametro p reišmės yra pirminiai skaičiai. Atlikime minėtą tyrimą 10 kartų ir sudarykime paklaidų lentelę, kuri leistų iš nagrinėjamų trijų tipų aproksimacijų išrinkti tiksliausią. Vidutinių kvadratinių paklaidų lentelė atrodo taip:

2.4 lentelė

Vidutinių kvadratinių paklaidų kai p įgyja pirmines reikšmes lentelė

Bandymo numeris	Aproksimacijos		
	ax^b	$a\exp(-b(x-c))$	$a\exp(-b(x-c)^d)$
1	0.0372	0.0099	0.1771
2	0.0042	0.0345	0.0518
3	0.0011	0.0368	0.0654
4	0.0450	0.0069	0.1839
5	0.0075	0.0264	0.1114
6	0.0027	0.0201	0.0792
7	0.0126	0.0900	0.0314
8	0.0128	0.0117	0.0755
9	0.0038	0.0644	0.0363
10	0.0053	0.0667	0.0322
Vidurkis	0.01322	0.03674	0.08442

Šiuo atvėju mažiausią paklaidą duoda pirmoji aproksimacija. Nubraižykime šią aproksimaciją.



2.6 pav. Sprendinių skaičiaus priklausomybė nuo pirminių p. Aproksimacija ax^b

Turime tokius aproksimacijos parametrų įverčius: $\hat{a} = 446.8542$, $\hat{b} = -2.4064$. Vidutinė kvadratinė paklaida yra 0.0253.

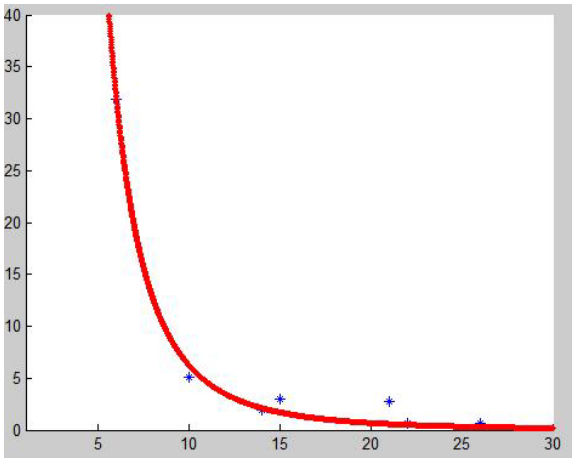
Nagrinėkime atvėjį, kai parametro p reišmės yra sudėtiniai skaičiai. Atlikime minėtą tyrimą 10 kartų ir sudarykime paklaidų lentelę, kuri leistų iš nagrinėjamų trijų tipų aproksimacijų išrinkti tiksliausią. Vidutinių kvadratinių paklaidų lentelė atrodo taip:

2.5 lentelė

Vidutinių kvadratinių paklaidų kai p įgyja sudėtines reikšmes lentelė

Bandymo numeris	Aproksimacijos		
	ax^b	$a\exp(-b(x-c))$	$a\exp(-b(x-c)^d)$
1	0.3647	0.9307	1.0589
2	0.4870	1.1123	1.0667
3	0.2786	0.7236	0.1757
4	0.3386	0.9551	1.2289
5	0.4675	1.0955	0.8327
6	0.4389	1.0504	1.2429
7	0.5395	1.1643	1.4097
8	0.4379	1.0091	1.0591
9	0.2419	0.7268	0.0776
10	0.5339	1.1348	1.8766
Vidurkis	0.41285	0.99026	1.00288

Šiuo atvėju mažiausią paklaidą duoda pirmoji aproksimacija. Nubraižykime šią aproksimaciją.



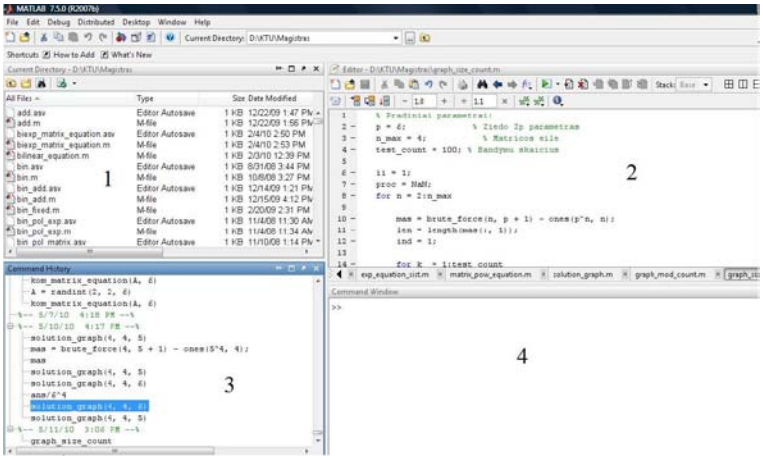
2.7 pav. Sprendinių skaičiaus priklausomybė nuo sudėtinių p. Aproksimacija ax^b

Turime tokius aproksimacijos parametrų įverčius: $\hat{a} = 9.7203 \times 10^3$, $\hat{b} = -3.2$. Vidutinė kvadratinė paklaida yra 0.3986.

2.4 PROGRAMINĖ REALIZACIJA IR INSTRUKCIJA VARTOTOJUI

2.4.1 MATLAB R2007b DARBO APLINKOS APRAŠYMAS

Kaip jau buvo minėta skyrelyje 1.6.4 darbo užduoties realizacijai buvo pasirinktas matematinis paketas MATLAB R2007b. Viena iš šios programinės įrangos pasirenkimo priežasčių yra ta, kad visa sąsaja su vartotoju yra vykdoma per komandinį langą. MATLAB‘o darbo aplinka atrodo taip



2.8 pav. MATLAB R2007b darbo aplinka

1 - Šiame lange parodomi visi vartotojo sukurti failai. Pagrindiniai failų tipai yra šie:

- M-file yra vartotojo sukurtos funkcijos
- asv failai yra vartotojo sukurtų funkcijų automatiškai išsaugotas kodas. Tai programinės įrangos apsauga nuo energijos šuolių
- Kiti failai (pvz MS Excel failai)

2 - Šis langas yra skirtas funkcijų kurimui. Lango viršuje yra panelė, kurios pagalba galima sukurti naują m-failą, atsidaryti jau sukurtą m-failą arba atlikti kitokius veiksmus (pvz. paleisti failą). Lango apačioje yra juosta, kurioje kortelių principu yra surašytos funkcijų vardai. Paspaudžius pelės kairiuoju klavišu ant funkcijos vardo atitinkamos funkcijos kodas yra parodomas lange.

3 - Šiame lange rodomi visi vartotojo atlikti veiksmai komandiniame lange

4 - Komandinis langas. Per šį langą vykdoma sąsaja su vartotoju.

2.4.2 SUKURTŲ FUNKCIJŲ APRAŠYMAS

MATLAB'o terpėje buvo sukurtos funkcijos, leidžiančios atlikti darbo užduotį. Galutinė magistrinio darbo programa yra šių funkcijų rinkinys. Šiame skyrelyje aprašysime pagrindines funkcijas, kurios buvo naudojamos užduoties sprendimui.

1. *matrix_exp(A, B, p)* – kelia matricą A matriciniu laipsniu B. Rezultatas priklauso grupei $\mathbb{Z}_p$.
   Įvestis: matricos A ir B ir grupės parametras p. Matricų A ir B formatai turi būti suderinti, p – teigiamas skaičius.
   Rezultatas: matrica $A \lhd B$
2. *brute_force(n, p)* – atlieka vektorių, kurių ilgis yra n, o elementai įgyja reikšmes nuo 1 iki p – 1, pilną perrinkimą.
   Įvestis: teigiami skaičiai n ir p.
   Rezultatas: visi vektoriai nuo $(1, 1, \dots, 1)$ iki $(p-1, p-1, \dots, p-1)$.
3. *exp_equation_sist(A, D, mas, p)* – sprendžiama lygtis (1.15), kai j reikšmė fiksuota.
   Įvestis: matrica A, vektorius-stulpelis D, pilno perrinkimo masyvas mas, grupės parametras p. Matrica A ir vektorius-stulpelis D turi būti suderinti.
   Rezultatas: sprendinių vektorius $[S_1; S_2; \dots; S_k]$, čia S_i - i-tasis sprendinys vektorius-stulpelis, k – sprendinių skaičius.
4. *exp_equation_matrix(A, B, M, p)* – sprendžiama lygčių sistema (1.14) naudojant algoritmą, kuris yra aprašytas skyrelyje 2.1 punktuose 6 – 9.
   Įvestis: kvadratinės matricos A, B ir M bei grupės parametras p. Reikalavimai parametrams yra pateikti skyrelyje 1.6.3.

Rezultatas: sprendinių skaičius

5. *solution_graph(m, n, p)* – 100 kartų generuojami lygties (1.15) parametrai: matrica A, kurios formatas yra $m \times n$ ir vektorius-stulpelis B. Atliekamas tyrimas, aprašytas skyrelyje 2.2.
   Įvestis: matricos A eilučių skaičius m, stulpelių skaičius n, grupės parametras p.
   Rezultatai: grafikas, kuris pavaizduoja sprendinių skaičiaus priklausomybę nuo matricos A eilučių skaičiaus, komandiniame lange parodomi gauti procentiniai santykiai.
6. *graph_mod_count* – 100 kartų generuojami lygčių sistemos (1.14) parametrai prie kiekvienos p reikšmės ir ieškoma gautos lygčių sitemos sprendinių. Atliekamas tyrimas, kuris yra aprašytas skyrelyje 2.3.1.
   Įvestis: nėra.
   Rezultatai: grafikas, kuris pavaizduoja sprendinių skaičiaus pryklausomybę nuo parametro p. Komandiniame lange pateikiama informacija apie pasirinktą aproksimacijos modelį, pateikiami MKM metodu gauti aproksimacijos parametrų įverčiai ir parodoma aproksimacijos vidutinė kvadratinė paklaida
7. *graph_size_count* – 100 kartų generuojami lygčių sistemos (1.8) parametrai prie kiekvienos m reikšmės ir ieškoma gautos lygčių sitemos sprendinių. Atliekamas tyrimas, kuris yra aprašytas skyrelyje 2.3.2.
   Įvestis: nėra.
   Rezultatai: grafikas, kuris pavaizduoja sprendinių skaičiaus pryklausomybę nuo matricų eilės m. Komandiniame lange pateikiama informacija apie pasirinktą aproksimacijos modelį, pateikiami MKM metodu gauti aproksimacijos parametrų įverčiai ir parodoma aproksimacijos vidutinė kvadratinė paklaida.

2.5 REZULTATŲ ANALIZĖ

Šiame skyrelyje trumpai paanalizuosime gautus rezultatus bei pasigylinsime į praktinio uždavinio taikymo ypatumus.

Rezultatai akivaizdžiai rodo, jog, kai parametras p yra pirminis skaičius, lygčių sistemos (1.14) sprendiniai sudaro mažesnę dalį visų variantų aibėje. Taip pat būtina paminėti tą faktą, jog grafikų 2.2a ir 2.2b braižymas užima daugiau laiko, negu grafikų 2.5a ir 2.5b. Reikia pabrėžti, jog didelė laiko dalis yra sunaudojama paskutinio taško braižymui (kai m = 4). Taip yra dėl to, kad reikia atlikti vsių trečios eilės daugianarių pilną perrinkimą, o, kaip mes jau žinome, šios operacijos laiko sąnaudas stipriai įtakoja pradiniai parametrai. Būtent dėl šios priežasties nebuvo nagrinėjamos penktos eilės matricos. Toks mažas nagrinėjamų taškų skaičius sudarė sunkumus skaičiuojant aproksimacijas. Ypač reikia

paminėti aproksimaciją $f(x) = a\exp\big(-b(x-c)\big)^d$, nes šiuo atvėju naudojant MKM parametrų vertinimui lygčių skaičius yra didesnis, nei taškų skaičius. Pranešimas apie tai parodomas ir MATLAB‘o komandiniame lange. Tačiau, kaip galėjome matyti, šis sunkumas nebuvo esminis ir leido efektyviai įvertinti sprendinių skaičiaus priklausomybę nuo matricų eilės kai p = 6. Taip pat didesnis nagrinėjamų taškų skaičius galėtų patvirtinti arba paneigti mūsų prielaidą apie eksponentinės aproksimacijos pasirinkimą atliekant sprendinių skaičiaus priklausomybės nuo m vertinimą kai p = 5.

Taikant šiuos rezultatus praktikoje reikia atsižvelgti į tai, kad kadangi parametro p reikšmė nėra didelis skaičius, tai nagrinėjant lygčių sistemą (1.14) lauke diskretinio logaritmo uždavinys yra lengvai sprendžiamas. Parodysime kaip galima netiesinę lygtį (1.7) paversti tiesine lygtimi.

Apibrėžkime matricos A diskretinį logaritmą

$$\log_{\mathrm{g}} A = \begin{pmatrix} \log_{\mathrm{g}} \mathrm{a}_{11} & \cdots & \log_{\mathrm{g}} \mathrm{a}_{1\mathrm{m}} \\ \vdots & \ddots & \vdots \\ \log_{\mathrm{g}} \mathrm{a}_{\mathrm{m}1} & \cdots & \log_{\mathrm{g}} \mathrm{a}_{\mathrm{mm}} \end{pmatrix} \tag{2.1}$$

čia g – lauko $\mathbb{Z}_p$ generatorius. Kaip jau buvo minėta skyrelyje 1.2 generatorius lauke galima rasti naudojant dvi pateiktas teoremas. Tokiu atvėju prisiminę matricinio laipsnio elemento apibrėžimą (1.8) turime

$$\log_g b_{ij} = \sum_{k=1}^{m} x_{kj} \log_{\mathrm{g}} a_{ik} \tag{2.2}$$

Atsižvelgę į formules (2.1) ir (2.2) turime, kad

$$(\log_g A) \cdot X = \log_g B \tag{2.3}$$

Matome, kad logaritmuodami lygybę (1.7) gauname tiesinę matricinę lygtį (2.3). Nors šiuo atvėju reikia spręsti m^2 lygčių (1.4) tačiau vertinant diskretinio logaritmo uždavinio ypatumus toks perėjimas gali būti realizuotas per polinominį laiką. Taigi, nors pirminės parametro p reikšmės duoda geresnius rezultatus, yra patartina naudoti sudėtinius skaičius, nes šiuo atvėju diskretinis logaritmavimas yra neefektyvus.

IŠVADOS

Rezultatų analizė parodė, jog, nors lygčių sistemos sprendinių skaičius sudaro mažesnę dalį visų galimų variantų aibėje kai parametro p reikšmės yra pirminės, yra patartina naudoti sudėtines parametro p reikšmes. Tokiu būdu yra išvengiama diskretinio logaritmavimo, kuris netiesinę lygčių sistemą (1.14) paverčia tiesinę.

Darbe buvo nagrinėjama, kaip priklauso lygčių sistemos sprendinių skaičius nuo dviejų sistemos parametrų: grupės $\mathbb{Z}_p^*$ parametro p bei nuo matricų A ir M eilės m. Buvo parodyta, kad sprendinių skaičiaus dalis visų galimų variantų aibėje didėjant matricų eilei m mažėja eksponentiškai, t.y $\mathcal{O}(a\exp(-b(m-c)))$, kai parametras p yra pirminis, ir $\mathcal{O}\left(a\exp\left(-b(m-c)\right)^d\right)$, kai parametras p yra sudėtinis skaičius, o didėjant parametro p reikšmei – hiperboliškai, t.y $\mathcal{O}(ap^{-b})$.

Matematinio modeliavimo būdu gauti sprendinių skaičiaus procentinės dalies įvertinimas leidžia juos ekstrapoliuoti ir parinkti tokius matricinio laipsnio funkcijos parametrus, prie kurių neįmanoma rasti sprendinius atsitiktinės paieškos būdu. Parinkę lygčių sistemos parametrų reikšmes $p = 33$ ir $m = 16$ jau gauname būtent tokį rezultatą.

Kadangi šios funkcijos determinuoti apgręžimo algoritmai nėra žinomi, tai leidžia daryti išvadą, jog matricinio lapsnio funkcija ir su ja susieta lygčių sistema (1.14) apibrėžta žiede $\mathbb{Z}_p$ gali būti panaudota kriptografijoje Diffie-Hellman‘o raktų apsikeitimų protokolo konstravimui.

DARBO UŽDUOTIES ATEITIES PERSPEKTYVOS

Darbe buvo nagrinėjami sveikųjų skaičių žiedai ir laukai. Tačiau lygčių sistema (1.14) gali būti apibrėžta ir kituose algebrinėse struktūruose. Viena iš tokių struktūrų yra kompleksiniai žiedai $\mathbb{Z}_p + i\mathbb{Z}_p$. Nors matricinio laipsnio funkcija negali būti išplėsta į kompleksinių skaičių žiedus (taip yra nes žiedas $\mathbb{Z}_p$ nėra metrinė erdvė), t.y matricos X elementai yra sveikieji skaičiai, šie žiedai yra naudingi tuo, kad sprendinių paieška yra vykdoma aibėje $\mathbb{Z}_{p^2}$.Tai reiškia, jog parinkus sistemos parametrus $p = 15$ ir $m = 11$ jau nebeįmanoma rasti sprendinius atsitiktinės paieškos būdu. Kriptografinio lygčių sistemos (1.14), apibrėžtos kompleksiniame žiede, saugumo tyrimas liko už šio darbo ribų.

PADĖKOS

Autorius dėkoja profesoriui Eligijui Sakalauskui už vertingus patarimus ruošiant šį magistro darbą, o taip pat už aprūpinimą naudingą literatūra. Taip pat autorius dėkoja doktorantui Kęstučiui Lukšiui už vertingus patarimus rašant darbo programą.

LITERATŪRA

1. E. Sakalauskas, K. Luksys, Matrix Power S-Box Construction
2. E. Sakalauskas, N. Listopadskis, P. Tvarijonas, Key Agreement Protocol Based on Matrix Power Function
3. Ф. Р. Гантмахер, Теория матриц, М.: Наука, 1966
4. И. М. Виноградов, Основы теории чисел, Москва, 1952
5. К. Айерлэнд, М. Роузен, Классическое введение в современную теорию чисел, М.: «Мир», 1987
6. http://ru.wikipedia.org/wiki/История_криптографии
7. http://ru.wikipedia.org/wiki/Дискретное_логарифмирование

PRIEDAS

Funkcija *bin.m*. Skaičiuoja skaičiaus dvejatainę išraišką. Grąžina 0 ir 1 masyvą

```
function Dig = bin(p)
    pstr = dec2bin(p);
    pbin = str2num(pstr);
    k = 1;
    while 2^(k + 1) <= p
        k = k + 1;
    end
    k = k + 1;
    for n = 0:k - 1
        pbin = pbin/10;
        if pbin - round(pbin) == 0
            Dig(k - n) = 0;
        else
            Dig(k - n) = 1;
        end
        pbin = round(pbin);
    end
end
```

Funkcija *exp_num.m*. Skaičiuoja modulinę eksponentę. Ši funkcija yra efektyvesnė už standartinę funkciją mod.

```
function answer = exp_num(a, b, p)
    B = bin(b);
    n = length(B);
    answer = 1;
    tmp(1) = a;
    for i = 2:n
        tmp(i) = mod(tmp(i - 1) ^ 2, p);
    end
    for i = 1:n
        if B(n - i + 1) == 1
            answer = mod(answer * tmp(i), p);
        end
    end
end
```

Funkcija *matrix_exp.m*

```
function M = matrix_exp(A, B, p)
    % Skaiciuoja A^B moduliu p
    [n1 n2] = size(A);
    [m1 m2] = size(B);
    for i = 1:n1
        for j = 1:m2
            M(i, j) = 1;
            for k = 1:n2
                tmp = exp_num(A(i, k), B(k, j), p);
                M(i, j) = mod(M(i, j) * tmp, p);
            end
        end
    end
```

```
end
```

Funkcija *is_in.m*. Funkcija atlieka elemento paiešką struktūroje A.

```
function answer = is_in(elem, A)
    answer = 0;
    [m, n, p] = size(A);
    if ~isempty(A)
        for i = 1:m
            for j = 1:n
                id = 0;
                    for k = 1:p
                        if A(i, j, k) == elem(1, k)
                            id = id + 1;
                        end
                    end
                if id == p
                    answer = answer + 1;
                end
            end
        end
    end
end
```

Funkcija *brute_force.m.*

```
function mas = brute_force(n, p)
    max = (p - 1)^n;

     for k = 1:n
        for j = 1:p-1
            for i = (j - 1) * max/(p - 1)^k + 1:j * max/(p - 1)^k
                for m = 0:(p - 1)^(k - 1) - 1
                    mas(i + m * max/(p - 1)^(k - 1), k) = j;
                end
            end
        end
     end

end
```

Funkcija *nulio_dalikliai.m.*

```
function zero_div = nulio_dalikliai(p)
    % Skaiciuoja ziedo Zp nulio dalikliu branduoli
    P = factor(p);
    len = length(P);
    zero_div = [];
    if len ~= 1  % Jeigu skaicius yra sudetinis
        if P(1) == P(len)   % Jeigu skaicius p = P(1) ^ len
            z = P(1) ^ ceil(len/2);
            h = p/z;
            for i = 1:h - 1
                zero_div = [zero_div i*z];
            end
        else
```

```
            M = multi_periodai(p);
            for j = 1:length(M);
                if M(j) <= sqrt(p)
                    z = p/M(j);
                    for i = 1:M(j) - 1
                        if is_in(i*z, zero_div) == 0
                            zero_div = [zero_div i*z];
                        end
                    end
                end
            end
        end
    end
end

function P = multi_periodai(n)
    N = factor(n);
    P = NaN;
    tmp = NaN;
    len = length(N);
    mas = brute_force(len, 3) - ones(2^len, len);
    index = 1;
    for i = 2:2^len
        tmp(i - 1) = 1;
        for j = 1:len
            tmp(i - 1) = tmp(i - 1) * N(j) ^ mas(i, j);
        end
        if is_in(tmp(i - 1), P) == 0
            P(index) = tmp(i - 1);
            index = index + 1;
        end
    end
end
```

Funkcija *pusgrupe.m*. Formuoja pusgrupę be nulio daliklių

```
function Z = pusgrupe(p)
    P = factor(p);
    len = length(P);
    Z = NaN;
    if len == 1
        Z = 1:(p-1);
    else
        nuliai = nulio_dalikliai(p);
        index = 1;
        for i = 1:(p-1)
            if is_in(i, nuliai) == 0
                Z(index) = i;
                index = index + 1;
            end
        end
    end
end
```

Funkcija *gen_matrix_pusgrupe.m*. Generuoja matricą pusgrupėje

```
function A = gen_matrix_pusgrupe(n, m, p)
    Z = pusgrupe(p);
    len = length(Z);
    A = NaN;
    for i = 1:n
        for j = 1:m
            ind = 1 + randint(1, 1, len - 1);
            A(i, j) = Z(ind);
        end
    end
end
```

Funkcija *exp_equation_sist.m*

```
function S = exp_equation_sist(A, D, mas, p)

    % Lygties A^X = D sprendimas, cia D ir X - yra vektoriai-stulpeliai.
Rezultatas -
    % sprendiniu vektorius S

    len = length(mas(:, 1));
    S = [];     % Sprendiniu vektorius siuo momentu yra tuscias
    ind = 0;    % Rastu sprendiniu skaicius duotuoju momentu;

    for i = 1:len
        if matrix_exp(A, transpose(mas(i, :)), p) == D
            ind = ind + 1;
            S(:, ind) = transpose(mas(i, :));
        end
    end

end
```

Funkcija kom_matrix.m. Skaičiuoja komutuojančią matricą

```
function B = kom_matrix(A, p)
    n = length(A(1, :));

% Generuojami koeficientai:
    a = 1 + randint(1, 1, p-1);
    b = 1 + randint(1, 1, p-1);
    c = 1 + randint(1, 1, p-1);

% Apskaiciuojama matrica B:
    T = mod(A * A, p);
    B = mod(mod(a * eye(n) + b * A, p) + c * T, p);

end
```

Funkcija *exp_equation_matrix.m*

```
function sk = exp_equation_matrix(A, B, M, p)
```

```
    % Lygciu sistemos
    % A^X = B
    % M*X = X*M
    % sprendimas. Rezultatas - sprendiniu skaicius sk

    n = size(A);                                          % matricos eile
    sk = 0;                                               % sprendiniu skaicius
    mas = brute_force(n(2), p + 1) - ones(p^n(2), n(2));
    len = length(mas(:, 1));
         for i = 1:len
            X = zeros(n);
            L = eye(n);
            for j = 1:n
                X = mod(X + mas(i, j) * L, p);
                L = mod(M * L, p);
            end

            if matrix_exp(A, X, p) == B
                sk = sk + 1;
            end
         end
end
```

Funkcija *solution_graph.m*

```
function avg_spr_count = solution_graph(m, n, p)

    mas = brute_force(n, p + 1) - ones(p^n, n);
    spr_count = NaN;               % rastu sprendiniu kiekis
    avg_spr_count = NaN;

    for j = 1:100

        A = gen_matrix_pusgrupe(m, n, p);             % Matricos formatas
        X = randint(n, 1, p);
        B = matrix_exp(A, X, p);

        for i = 1:m

            % Sprendziama sistema is i lygciu

            M = A(1:i, 1:n);
            K = B(1:i);
            if ~isempty(mas)
                tmp = size(exp_equation_sist(M, K, mas, p));
                spr_count(i, j) = tmp(2);
            else
                spr_count(i, j) = 0;
            end

        end

    end
    for i = 1:m

        avg_spr_count(i) = mean(spr_count(i, :)) * 100/p^n;
```

```
        % Grafiko braizymas

        h = line(0, 0, 'erase', 'none');
        axis([0 m 0 avg_spr_count(1)]);
        set(h, 'xdata', i, 'ydata', avg_spr_count(i), 'Marker', '*');
        drawnow;

    end
end
```

Funkcija *least_squares_hyp.m*. Įvertina aproksimacijos ax^b parametrus.

```
function teta = least_squares_hyp(xdata, ydata)
    start = [1, -3];
    teta = lsqcurvefit(@fun, start, xdata, ydata);
end

function y = fun(x, mas)
    y = x(1) * mas.^x(2);
end
```

Funkcija *least_squares_exp.m*. Įvertina aproksimacijos $a\exp(-b(x-c))$ parametrus.

```
function teta = least_squares_exp(xdata, ydata)
    start = [1, 1, 0];
    teta = lsqcurvefit(@fun, start, xdata, ydata);
end

function y = fun(x, mas)
    y = x(1) * exp(-x(2) * (mas - x(3)));
end
```

Funkcija *least_squares_exp_pow.m*. Įvertina aproksimacijos $a\exp\left(-b(x-c)\right)^d$ parametrus.

```
function teta = least_squares_exp_pow(xdata, ydata)
    start = [1, 1, 0, 1];
    teta = lsqcurvefit(@fun, start, xdata, ydata);
end

function y = fun(x, mas)
    y = x(1) * exp(-x(2) * (mas - x(3)).^x(4));
end
```

Failas *graph_mod_count.m*

```
% Pradiniai parametrai:

n = 2;              % Matricos eile
P = [6 ,10, 14, 15, 21, 22, 26];
%P = [5 ,7, 11, 13, 17, 19, 23, 29];
len_p = length(P);
test_count = 100;
```

```
sk_max = 0;
jj = 1;
proc = NaN;
for ii = 1:len_p

    p = P(ii);

    % Generuojami lygciu sistemos
    % A^X = B
    % MX = XM
    % parametrai

    mas = brute_force(n, p + 1) - ones(p^n, n);
    len = length(mas(:, 1));

    for k  = 1:test_count

        % Generuojami lygciu sistemos
        % A^X = B
        % MX = XM
        % parametrai

        A = gen_matrix_pusgrupe(n, n, p);
        while 1
            T = gen_matrix_pusgrupe(n, n, p);
            T_inv = mod_inv(T, p);
            if ~isempty(T_inv)
                M = mod(mod(T * diag(1:n), p) * T_inv, p);
                break;
            end
        end
        X = kom_matrix(M, p);
        B = matrix_exp(A, X, p);

        sk(k) = 0;

        for i = 1:len
            X = zeros(n);
            L = eye(n);
            for j = 1:n
                X = mod(X + mas(i, j) * L, p);
                L = mod(M * L, p);
            end

            if matrix_exp(A, X, p) == B
                sk(k) = sk(k) + 1;
            end
        end
    end

proc(jj) = (mean(sk)/len) * 100;
jj = jj + 1;

        % Grafiko braizymas
    h = line(0, 0, 'erase', 'none');
    axis([1 30 0 40]);
    set(h, 'xdata',  p , 'ydata', proc(jj - 1), 'Marker', '*');
    drawnow;

end
```

```
paklaida = 0;
teta = least_squares_hyp(P, proc);
disp('Modelis: a/x^n (raudonas grafikas).');
disp('Parametrai:')
disp(teta);
for x = 0.1:0.01:30
    y = teta(1) * x^teta(2);
    set(h, 'xdata',  x , 'ydata', y, 'Marker', '.', 'Color', 'red');
    for i = 1:len_p
        if x == P(i)
            paklaida = paklaida + (proc(i) - y)^2;
            break;
        end
    end
end
disp('Vidutine kvadratine paklaida');
disp(paklaida/len_p);

paklaida = 0;
teta = least_squares_exp(P, proc);
disp('Modelis: a*exp(-b(x - c)) (zalias grafikas).');
disp('Parametrai:')
disp(teta);
for x = 0.1:0.1:30
    y = teta(1) * exp(-teta(2) * (x - teta(3)));
    set(h, 'xdata',  x , 'ydata', y, 'Marker', '.', 'Color', 'green');
    for i = 1:len_p
        if x == P(i)
            paklaida = paklaida + (proc(i) - y)^2;
            break;
        end
    end
end
disp('Vidutine kvadratine paklaida');
disp(paklaida/len_p);

paklaida = 0;
teta = least_squares_exp_pow(P, proc);
disp('Modelis: a*exp(-b(x - c)^d) (geltonas grafikas).');
disp('Parametrai:');
disp(teta);
for x = 0.1:0.1:30
    y = teta(1) * exp(-teta(2) * (x - teta(3)) ^ teta(4));
    set(h, 'xdata',  x , 'ydata', y, 'Marker', '.', 'Color', 'yellow');
    for i = 1:len_p
        if x == P(i)
            paklaida = paklaida + (proc(i) - y)^2;
            break;
        end
    end
end
disp('Vidutine kvadratine paklaida');
disp(paklaida/len_p);
```

Failas *graph_size_count.m*

```
% Pradiniai parametrai:
p = 6;              % Ziedo Zp parametras
n_max = 4;            % Matricos eile
```

```
test_count = 100; % Bandymu skaicius

ii = 1;
proc = NaN;
for n = 2:n_max

    mas = brute_force(n, p + 1) - ones(p^n, n);
    len = length(mas(:, 1));
    ind = 1;

    for k  = 1:test_count

        % Generuojami lygciu sistemos
        % A^X = B
        % MX = XM
        % parametrai

        A = gen_matrix_pusgrupe(n, n, p);
        while 1
            T = gen_matrix_pusgrupe(n, n, p);
            T_inv = mod_inv(T, p);
            if ~isempty(T_inv)
                M = mod(mod(T * diag(1:n), p) * T_inv, p);
                break;
            end
        end
        X = kom_matrix(M, p);
        B = matrix_exp(A, X, p);

        sk(k) = 0;

        for i = 1:len
            X = zeros(n);
            L = eye(n);
            for j = 1:n
                X = mod(X + mas(i, j) * L, p);
                L = mod(M * L, p);
            end

            if matrix_exp(A, X, p) == B
                sk(k) = sk(k) + 1;
            end

        end

    end

  proc(ii) = (mean(sk)/len) * 100;
  ii = ii + 1;

            % Grafiko braizymas

        h = line(0, 0, 'erase', 'none');
        axis([1 5 0 40]);
        set(h, 'xdata',  n , 'ydata', proc(ii - 1), 'Marker', '*');
        drawnow;

end

paklaida = 0;
```

```
teta = least_squares_hyp(2:n_max, proc);
disp('Modelis: a/x^n (raudonas grafikas).');
disp('Parametrai:')
disp(teta);
for x = 1:0.01:5
    y = teta(1) * x^teta(2);
    set(h, 'xdata',  x , 'ydata', y, 'Marker', '.', 'Color', 'red');
    for i = 2:n_max
        if x == i
            paklaida = paklaida + (proc(i - 1) - y)^2;
            break;
        end
    end
end
disp('Vidutine kvadratine paklaida');
disp(paklaida/(n_max - 1));

paklaida = 0;
teta = least_squares_exp(2:n_max, proc);
disp('Modelis: a*exp(-b(x - c)) (zalias grafikas).');
disp('Parametrai:')
disp(teta);
for x = 1:0.01:5
    y = teta(1) * exp(-teta(2) * (x - teta(3)));
    set(h, 'xdata',  x , 'ydata', y, 'Marker', '.', 'Color', 'green');
    for i = 2:n_max
        if x == i
            paklaida = paklaida + (proc(i - 1) - y)^2;
            break;
        end
    end
end
disp('Vidutine kvadratine paklaida');
disp(paklaida/(n_max - 1));

paklaida = 0;
teta = least_squares_exp_pow(2:n_max, proc);
disp('Modelis: a*exp(-b(x - c)^d) (geltonas grafikas).');
disp('Parametrai:');
disp(teta);
for x = 1:0.01:5
    y = teta(1) * exp(-teta(2) * (x - teta(3)) ^ teta(4));
    set(h, 'xdata',  x , 'ydata', y, 'Marker', '.', 'Color', 'red');
    for i = 2:n_max
        if x == i
            paklaida = paklaida + (proc(i - 1) - y)^2;
            break;
        end
    end
end
disp('Vidutine kvadratine paklaida');
disp(paklaida/(n_max - 1));
```