KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Aurimas Neverauskas

Lygčių ir nelygybių simbolinio sprendimo lygiagretusis metodas

Magistro darbas

Darbo vadovas
doc. dr. Romas Marcinkevičius

Kaunas, 2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Aurimas Neverauskas

Lygčių ir nelygybių simbolinio sprendimo lygiagretusis metodas

Magistro darbas

Vadovas
doc. dr. Romas Marcinkevičius
Programų inžinerijos katedra

Recenzentas
doc. dr. Armantas Ostreika
Multimedijos inžinerijos katedra

Atliko
IFM-9/2 gr. stud.
Aurimas Neverauskas

Kaunas, 2011

Turinys

1 Įvadas 8

1.1 Dokumento paskirtis 8

1.2 Santrauka 9

2 Srities analizė 10

2.1 Pagrindiniai lygčių ir nelygybių tipai 10

2.2 Lygčių ir nelygybių spredimo algoritmų įgyvendinimo sunkumai 11

2.3 Simbolinės lygtys ir nelygybės 13

2.4 Kompiuterinės algebros sistemos 14

2.5 Kompiuterinės algebros sistema – Maxima 15

2.6 Egzistuojantys lygiagrečių skaičiavimų sprendimai 16

3 Reikalavimai sistemai 18

3.1 Panaudos atvejai 18

3.2 Funkciniai reikalavimai 20

3.3 Apribojimai sprendimui 21

4 Bendra architektūra 22

4.1 Bendras sistemos vaizdas 22

4.2 Bendravimas tarp Vartototojo PĮ ir Skaičiavimų PĮ procesų 24

4.3 Uždavinio sistemų ir rezultatų atvaizdavimas vartotojui 25

5 Vartotojo PĮ architektūra 26

5.1 Model-View-Controller paternas 26

5.2 „Svogūno“ architektūra 27

5.3 Priklausomybių injekcijos 29

5.4 Įvesties validacija 29

5.5 Skaičiavimų pabaigos nustatymas 31

6 Skaičiavimų PĮ architektūra 32

6.1 Socket‘ai – bendravimo tarp procesų būdas 32

6.2 Maxima CAS paketo panaudojimas 33

6.3 Lygiagretaus programavimo biblioteka MPI 34

7 Lygiagretaus sprendimo metodas 36

7.1 Algoritmas 36

7.1.1 Valdančiojo proceso algoritmo dalis 38

7.1.2 Skaičiavimų proceso algoritmo dalis 40

7.2 Apribojimai sprendžiamiems uždaviniams 40

8 Skaičiavimų algoritmo tyrimas 42

8.1 Naudojama techninė įranga 42

8.2 Algoritmo greičio priklausomybės 42

8.3 Lygybių ir nelygybių sistemų greičio palyginimas su Maple CAS 47

9 Išvados....50
10 Literatūra....51
11 Terminų ir santrumpų žodynas....53

Paveikslėlių sąrašas

Pav. 1: Tiesinės lygties pavyzdys 10
Pav. 2: Polinominės lygties pavyzdys 11
Pav. 3: Diophantine lygties pavyzdys 11
Pav. 4: Faktorizuojamos lygties pavyzdys (3 sprendimo žingsniai) 11
Pav. 5: Dviejų tiesių susikirtimas, lygybių atveju rezultatas taškas, nelygybių atveju, vienas iš keturių plotų 13
Pav. 6: Simbolinė lygčių sistema ir jos sprendinys 14
Pav. 7: Maxima grafinė sąsajos, kairėje sąsajos langas Windows sistemoje, dešinėje konsolė Linux sistemoje 16
Pav. 8: Panaudos atvejai 18
Pav. 9: Programų sistemos veikimo principas 22
Pav. 10: Vartotojo PĮ veikimo principas 23
Pav. 11: Skaičiavimų PĮ veikimo principas 23
Pav. 12: Bendravimas tarp Vartotojo PĮ ir Skaičiavimų PĮ procesų 25
Pav. 13: Išraiškos atvaizdavimas <img src="http://myServer/Eq2Img?\\Large c = 433^{1/5}*e^{2*i*pi/5}?"> 25
Pav. 14: MVC paterno dalys 26
Pav. 15: Sluoksninė architektūra (kairėje) ir svogūno architektūra (dešinėje) 28
Pav. 16: Skaičiavimų PĮ ir Maxima CAS bendravimo per socket'us eiga 33
Pav. 17: Algoritmo efektyvumo tikrinimo pavyzdžio sąlyga 36
Pav. 18: Neefektyvaus darbų paskirstymo algoritmo pavyzdys 37
Pav. 19: Efektyvaus darbų paskirstymo algoritmo pavyzdys 37
Pav. 20: Bendroji skaičiavimų PĮ architektūra 38
Pav. 21: Valdančiojo proceso algoritmo schema 39
Pav. 22: Skaičiavimų proceso algoritmo schema 40
Pav. 23: Lygybių ir nelygybių sistemos sprendinys (1) 43
Pav. 24: Lygybių ir nelygybių sistemos sprendinys (2) 43
Pav. 25: Sukurtos PĮ skaičiavimų trukmės priklausomybė nuo nelygybių kiekio 45
Pav. 26: Skaičiavimų trukmės priklausomybė nuo procesų kiekio 46
Pav. 27: 10-o laipsnio polinomas 46
Pav. 28: Skaičiavimų laiko priklausomybė nuo darbų kiekio 47
Pav. 29: Netiesinės lygčių sistemos sprendimo pavyzdys, viršuj sukurtos PĮ atsakymas, žemiau Maple atsakymas 47
Pav. 30: Sukurtos PĮ ir Maple nelygybių sistemos sprendimų trukmės prklausomybė nuo nelygybių kiekio 48
Pav. 31: Lygybių sistemų skaičiavimų trukmių priklausomybė nuo lygčių ir nežinomųjų kiekio 49

Lentelių sąrašas

Lentelė 1: Kompiuterinių algebrinių sistemų, sprendžiančių lygčių ir nelygybių sistemas, lentelė 15
Lentelė 2: Vartotojo PĮ MVC paterno realizacija 27
Lentelė 3: Lygčių ir nelygybių sintaksės teisingumo lentelė 30
Lentelė 4: Panaudotos MPI bibliotekos funkcijos 35
Lentelė 5: Skaičiavimų trukmės priklausomybė nuo procesų kiekio 45
Lentelė 6: Sukurtos PĮ ir Maple sistemų skaičiavimų trukmės sprendžiant nelygybių sistemas 48
Lentelė 7: Lygybių sistemų skaičiavimų trukmių priklausomybė nuo lygčių ir nežinomųjų kiekio 49

Parallel method for symbolic solution of equations and inequalities

SUMMARY

I have presented an effective way to solve symbolic systems of equations and inequalities using parallel processes and compared it to ineffective method. Also, I have performed analysis of presented algorithm, determining its performance dependencies and comparing its performance to existing software.

Also, this paper discusses architectural solutions for the application system: MVC design pattern, "Onion" architecture and Dependency Injection. These architectural patterns benefit more than standard layered architecture, software, based on these patterns, is more maintainable and changeable.

These days, computers usually have multi-core processors, but not all software use them efficiently. The main problem is to create algorithm for solving symbolic systems of equations and inequalities using parallel processes, using calculation power and decreasing calculation time. Such application system was created and analyzed in this paper.

It was determined that created software is superior to Maple CAS when task is small by input but requires a lot of calculating power (systems of inequalities). On the other hand, results differ when task consist of plenty of equations (40-50 equations in system, same number of unknowns). Created software falls behind Maple CAS in performance. The main reason, for this, is that created software spends too much time to analyze task and strings in it.

1 Įvadas

1.1 Dokumento paskirtis

Šio dokumento paskirtis yra pateikti lygčių ir nelygybių simbolinio sprendimo lygiagretaus metodo architektūrinius ir algoritminius sprendimus, atlikti programų sistemos architektūros analizę, atlikti lygiagretaus algoritmo tyrimą.

Antrajame skyriuje atliekama šio darbo srities anallizė, aprašomi pagrindiniai lygčių ir nelygybių tipai. Nustatomi skirtumai tarp skaitinių ir simbolinių išraiškų. Išnagrinėjamas galimų panaudoti kompiuterinės algebros sistemų sąrašas. Aptariami egzistuojantys lygiagrečių simbolinių skaičiavimų sprendimai.

Trečiajame skyriuje aptariami sistemai keliami reikalavimai. Nustatomi panaudos atvejai, aprašomi funkciniai reikalavimai, numatomi apribojimai. Pagal šio skyriaus reikalavimus buvo realizuota programinė įranga.

Ketvirtajame skyriuje aprašoma bendra sistemų sistemos architektūra, nustatomi ryšiai tarp sistemų. Aptariamas sistemų bendravimo principas (procesų kūrimas, bendravimas per socket‘us). Aprašoma atviro kodo sistema, skirta matematinių išraiškų atvaizdavimui.

Penktajame skyriuje aptariama Vartotojo PĮ (web programa) architektūra. Analizuojami MVC patern‘o ir „Svogūno“ architektūros, priklausomybių injekcijos (Dependency Injection) privalumai, sukurtai PĮ. Aprašomas vartotojo duomenų (lygčių ir nelygybių sistemos) validavimas ir rezultatų pateikimas (kaip tekstinės išraiškos tampa paveiksliukais).

Šeštasis skyrius skirtas Skaičiavimų PĮ (C++ MPI programa) architektūros aptarimui. Apibrėžiamas socket‘ais (kompiuterio prievadais, port‘ais) grįstas bendravimas tarp procesų. Pateikiamas Maxima CAS (kompiuterinės algebros sistema) panaudojimas kitos programos kontekste. Apžvelgiama MPI lygiagrečių skaičiavimų biblioteka ir jos panaudojimas.

Septintajame skyriuje pateikiamas lygiagretaus sprendimo metodas. Pateikiama bendra algoritmo schema, valdančiojo proceso algoritmo dalies schema, skaičiavimų procesų algoritmo schema. Pateikiami apribojimai lygčių ir nelygybių sistemoms ir pavyzdžiai.

Aštuntajame skyriuje pateikiamas skaičiavimų algoritmo tyrimas. Ištiriamos bendros algoritmo charakteristikos, nustatant priklausomybes tarp procesų kiekio, dalinių uždavinių kiekio, matuojant algoritmo vykdymo laiką. Atliekamas sprendimų greičio palyginimas su Maple kompiuterinės algebros sistema.

1.2 Santrauka

Pateiktas lygčių ir nelygybių simbolinio sprendimo lygiagretus algoritmas ir jo analizė, palyginimas su neefektyvia algoritmo realizacija. Atliktas įgyvendinto algoritmo tyrimas, nustatant jo spartos priklausomybes nuo aplinkos ir užduoties, palyginant rezultatus su esama PĮ.

Taip pat, šiame darbe aptariami sukurtos programų sistemos architektūriniai sprendimai: MVC patern‘as (design pattern), „Svogūno“ architektūra, priklausomybių injekcijos (Dependency Injections). Šie architektūriniai sprendimai yra pranašesni už standartinę sluoksninę architekūrą, jais paremta PĮ yra lengviau palaikoma ir modifikuojama.

Šiais laikais dauguma kompiuterių turi daugiabranduolius procesorius, tačiau esama PĮ jų neišnaudoja. Šio darbo tikslas yra sukurti tokią lygčių ir nelygybių simbolinio sprendimo lygiagrečiu metodu realizaciją, kuri panaudodama turimą skaičiavimų galią, sutrumpintų skaičiavimų laiką.

Atlikus tyrimus nustatyta, jog sukurtoji PĮ yra pranašesnė už Maple CAS tik tuo atveju, kai uždavinio sąlyga nėra didelė, bet reikalaujama didelės skaičiavimų galios (nelygybių sistemų sprendimas). Tačiau sprendžiant didelės apimties lygčių sistemas (40-50 nežinomųjų ir tiek pat lygčių) sukurtoji PĮ atsilieka nuo Maple CAS, kadangi daug laiko sugaištama nagrinėjant pateiktą užduotį ir skaidant ją į dalinius uždavinius.

2 Srities analizė

Šiame skyriuje apžvelgiama:

- Pagrindiniai lygčių ir nelygybių tipai: Tiesinės, polinominės, diophantine, inversinės, faktorizacinės, kompleksinių šaknų, Taylor‘o eilutės, matricinės, diferencialinės, integralinės
- Lygčių ir nelygybių sprendimo algoritmų įgyvendinimo sunkumai kyla dėl didelio kiekio lygčių ir nelygybių tipų, tipai turi atskirus atvejus. Taip pat sudėtingumo prideda simboliniai skaičiavimai, algoritmo lygiagretinimas, nelygybių sprendimas.
- Simbolinės lygtys ir nelygybės yra tokios, kur koeficientai žymimi simboliai, operacijos atliekamos ne su skaičiais, pasižymi išraiškų suprastinimu, o rezultatai nepraranda tikslumo.
- Kompiuterinės algebros sistemos: pateikiamas CAS apibrėžimas, galimybės, sąrašas. Pateikiamas tinkamos CAS pasirinkimas.
- Kompiuterinės algebros sistema – Maxima: istorija, galimybės, vartotojo sąsaja.
- Egzistuojantys lygiagrečių skaičiavimų sprendimai: įprastos CAS palaiko vieną procesą, „Parallel Fourier-Motzkin elimination“ algoritmas, esamos paskirstyto skaičiavimo sistemos.

2.1 Pagrindiniai lygčių ir nelygybių tipai

Lygtys pagal savo savybes dažniausiai skirtomos pagal tokius tipus [12]:

- **Tiesinės** lygtys ir jų sistemos – paprasčiausios visiems pažįstamos pirmojo laipsnio lygtys

$$\begin{aligned} 3x + 2y - \ \ z &= \ \ 1 \\ 2x - 2y + 4z &= -2 \\ -x + \tfrac{1}{2}y - \ \ z &= \ \ 0 \end{aligned}$$

Pav. 1: Tiesinės lygties pavyzdys

- **Polinominės** lygtys – lygtys, kurių nežinomieji turi laipsnius, didžiausias laipsnis nulemia šaknų skaičių

$$4x^5 - x^3 - 3 = 0$$

Pav. 2: Polinominės lygties pavyzdys

- **Diophantine** tipo lygtys – polinominių lygčių tipas, naudojami tik sveiki skaičiai. Nežinomųjų būna daugiau negu lygčių, tenka surasti visus tinkamus sprendinių variantus.

$$2x^5 - 5x^4 - x^3 - 7x^2 + 2x + 3 = 0$$

Pav. 3: Diophantine lygties pavyzdys

- **Inversinės** lygtys – šiam lygčių tipui priklauso lygtys su neigiamais laipsniais, logaritmais, inversinės trigonometrinės funkcijos.
- **Factorizacija** paremtos lygtys – lygtys, kurios sprendžiamos iškeliant bendrą dauginamąjį prieš skliaustus ir sprendžiant naujas lygtis atskirai.

$$\begin{aligned} &\tan x + \cot x = 2 \\ &\tan x + \cot x - 2 = 0\,, \\ &(\tan x - 1)(\cot x - 1) = 0\,. \end{aligned}$$

Pav. 4: Faktorizuojamos lygties pavyzdys (3 sprendimo žingsniai)

- Kitos lygtys – lygtys su kompleksinėmis šaknimis, Taylor‘o eilutės, matricinės lygtys, diferencialinės lygtys, integralinės lygtys ir t.t.

Nelygybes kaip ir lygtis galima suskirstyti pagal panašius tipus, tačiau nelygybės pasižymi išskirtinėmis savybėmis:

- Pagrindinis pastebimas skirtumas yra išraiškos ženklas, kai lygybė turi sprendinį griežtai apibrėžiantį ženklą „=“, nelygybės turi keturis – „<“, „>“, „≤”, “≥”.
- Remiantis ankstesniąja sąlyga, lygtys turi šaknis, o nelygybių sprendinys yra reikšmių intervalas, su ribinių reikšmių įtraukimu, priklausomai nuo ženklo.

2.2 Lygčių ir nelygybių spredimo algoritmų įgyvendinimo sunkumai

Didžiausia problema yra ta, kad nėra vienintelio algoritmo, kuris išspręstų visas lygtis ir nelygybes. Iš pradžių vykdoma uždavinio analizė, gaunami jo parametrai, pagal juos nusprendžiama kaip sistema bus sprendžiama.

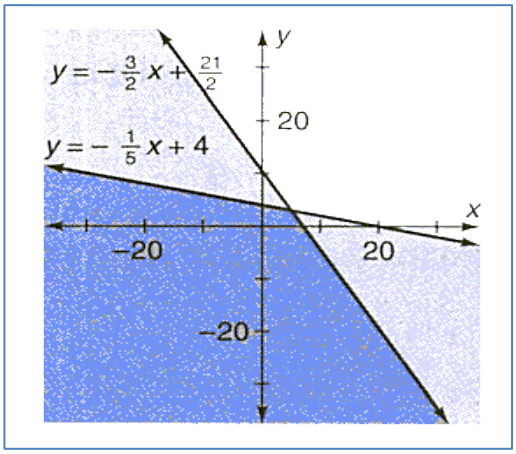
Kiekvienas lygčių ir nelygybių tipas turi skirtingus sprendimo algoritmus. Dėl šios priežasties neįmanoma apimti didelio tipų kiekio. Tiesinių lygčių sprendimui naudojami paprastesni algoritmai, tokie kaip Gauso metodas [19]. Netiesinės lygtys turi sudėtingesnius algoritmus, po kuriais slepiasi teoremos, statistika, spėjimai.

Tačiau, dažniausiai neužtenka vieno algoritmo vienam lygybių tipui. Netgi atskiri šių tipų atvejai, priklausantys nuo sistemos parametrų turi skirtingas algoritmų realizacijas. Kaip pavyzdį galima paimti polinomines lygtis ir atskirą jų šaką Diophantine lygtį. Pastaroji operuoja tik su sveikais skaičiais, šaknimis gali būti tik sveikieji skaičiai, o nežinomųjų kiekis būna didesnis už lygčių kiekį.

Sprendžiamos gali būti skaitinės ir simbolinės sistemos [18]. Skaitinės sistemos yra paprastesnės, jose koeficientai yra skaitinės reikšmės. Šias sistemas išspręsti užtenka paprastų matematinių operacijų. Sudėtingesnės yra simbolinės sistemos, jose koeficientai yra žymimi konstantomis (pvz. a, b). Kadangi neįmanoma atlikti matematinių veiksmų su tokiomis išraiškomis, tenka atlikti leksikinę analizę – veiksmus atlikti ne su skaičiais, o su išraiškomis.

Vienas didžiausių iššūkių yra lygiagrečių algoritmų realizavimas. Šie algoritmai turi veikti greičiau už savo vieno proceso algoritmo atitikmenį ir grąžinti tuos pačius rezultatus. Kadangi yra galybė sistemų tipų ir atvejų, kiekvieno algoritmo realizacija yra didžiulis uždavinys.

Nelygybių sprendimas yra daug sudėtingesnis už lygybių sprendimą. Tai yra todėl, kadangi rezultate gaunamos ne šaknys, o išraiškos. Grafiškai išsprendus dviejų tiesinių lygčių sistemą gaunamas vienas taškas (jeigų lygtimis aprašomos tiesės susikerta), nelygybių atveju gaunamas plotas apribotas tiesėmis. Vienas geriausių žinomų algoritmų, nelygygėms spręsti, yra „Fourier-Motzkin elimination“ [6] algoritmas. Šis algoritmas remiasi kintamųjų atmetimu iš sistemos. Tačiau šio algoritmo sudėtingumas yra dvigubai eksponentinis, t.y. skaičiavimo laikas ypač greitai kyla didėjant nelygybių ir nežinomųjų kiekiui.



Pav. 5: Dviejų tiesių susikirtimas, lygybių atveju rezultatas taškas, nelygybių atveju, vienas iš keturių plotų

Taigi norint įgyvendinti lygiagretų lygčių ir nelygybių sprendimą tektų realizuoti daugybę simbolinių lygčių sistemų sprendimą lygiagrečiai ir „Fourier-Motzkin elimination“ lygiagretaus algoritmo versiją simboliniams skaičiavimams. Šio uždavinio sudėtingumo sumažinimui buvo pasirinkta kompiuterinės algebros sistema „Maxima“ [16], kuri turi realizuotus nelygiagretaus skaičiavimo algoritmus simboliniams skaičiavimams, tad šie skaičiavimai bus lygiagretinami pasinaudojus MPI [15] lygiagretaus skaičiavimo bibliotekomis.

2.3 Simbolinės lygtys ir nelygybės

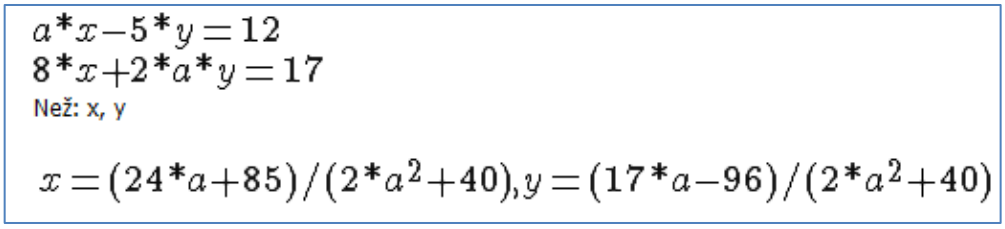
Vienas labiausiai pastebimas skirtumas tarp simbolinės ir skaitinės išraiškos yra tas, kad vietoj koeficientų yra naudojami simboliai (dažniausiai raidės: „a“, „b“ ir t.t.). Tačiau nevisi koeficientai turi būti simboliai, norint atlikti simbolinius skaičiavimus, gali ir nebūti nei vieno simbolio, kuris aprašytų koeficientą.

Veiksmai su simbolinėmis išraiškomis yra daug sudėtingesni negu su skaitinėmis išraiškomis. Simbolinės išraiškos atveju nevisada galima atlikti matematines operacijas su koeficientais, kadangi jais gali būti ne tik skaičiai bet ir simboliai, įvedamas specialus simbolio tipas, perrašomos pagrindinės operacijos šiam simbolio tipui.

Svarbi simbolinės išraiškos savybė yra išraiškos suprastinimas arba pakeitimas, pakeičiant vieną išraišką kita arba kitomis, nepakeičiant jos reikšmės.

Svarbiausia simbolinės išraiškos savybė yra tikslumas. Atlikus simbolinius skaičiavimus gaunama tiksli reikšmė, kadangi nėra paklaidos, kuri įsivelia atliekant skaitinius skaičiavimus. Kodėl skaitiniai skaičiavimai turi paklaidą, o simboliniai neturi? Skaitiniai algoritmai atlieka operacijas su real tipo skaičiais, dažnai šaknų ieško iteraciniais metodais artėdami link jų reikšmių. Tuo tarpu simboliniai skaičiavimai nedirba su real tipo skaičiais, jie remiasi išraiškų

suprastinimais ir veiksmais su simbolinėmis išraiškomis. Geras pavyzdys yra lygties sprendinys šaknis iš 3, skaitinis metodas grąžintų reikšmę 1,732050807. O simbolinis metodas gražintų rezultatą kaip sqrt(3). Jeigu šis rezultatas būtų naudojamas tolesniam kitų sistemų sprendimui, skaitinio metodo atveju tikslumas būtų toliau prarandamas, o simbolinio metodo atveju tikslumas nenukentėtų.



Pav. 6: Simbolinė lygčių sistema ir jos sprendinys

2.4 Kompiuterinės algebros sistemos

Kompiuterinės algebros sistema [3] (CAS – computer algebra system) yra programinė įranga atliekanti simbolinius matematinius skaičiavimus.

Simbolinius skaičiavimus dažniausiai sudaro:

- Išraiškos prastinimas į mažesnę ar standartinę formą
- Simbolių ar išraiškų keitimas į kitas išraiškas
- Laipsnių išskleidimas, faktorizavimas, trupmenų pertvarkymas
- Diferenciacija (diferenciacijos operacija)
- Integravimas
- Simbolinių išraiškų optimizavimas
- Tiesinių ir kai kurių netiesinių lygčių sprendimas
- Kai kurių diferencialinių lygčių sprendimas
- Ribų paieška

Išraiškas dažniausiai sudaro polinomai, standartinės funkcijos (sinusas, eksponentė), įvairos specialios funkcijos (pvz. Bassel funkcija), integralai ir t.t.

Šiame darbe domina tik tos kompiuterinės algebros sistemos, kurios palaiko lygčių ir nelygybių sprendimą. Sąrašas [2] tokių sistemų:

Lentelė 1: Kompiuterinių algebrinių sistemų, sprendžiančių lygčių ir nelygybių sistemas, lentelė

Pavadinimas	Kūrėjas	Ar komercinė?	Pradėta vystyti (data)	Pirmoji viešа realizacija (data)	Išlaiko rezultatų tikslumą	Sprendžia lygčių sistemas	Sprendžia nelygybių sistemas
Algebrator	Neven Jurkovic	Taip	1986	1999	Ne	Taip	Taip
Maple	Symbolic Computation Group, University of Waterloo	Taip	1980	1984	Taip	Taip	Taip
Mathematica	Wolfram Research	Taip	1986	1988	Taip	Taip	Taip
Maxima	MIT Project MAC and Bill Schelter	Ne	1967	1998	Taip	Taip	Taip
Microsoft Mathematics	Microsoft	Ne	-	2005	Ne	Taip	Taip
Sage	William A. Stein	Ne	2005	2005	Taip	Taip	Taip
Xcas	Bernard Parisse	Ne	2004	2008	Taip	Taip	Taip

Simboliniams skaičiavimams atlikti reikia pasirinkti labiausiai tinkamą kompiuterinės algebros sistemą. Ši sistema neturi būti komercinė, atlikti simbolinius skaičiavimus – išlaikant rezultatų tikslumą. Tarp galimų CAS yra Maxima, Sage ir Xcas.

Sage yra Python kalba realizuota CAS, tačiau tai nėra tikra algebrinė sistema, ji naudoja kitus paketus skaičiavimams atlikti, o jos priėjimas yra tik per naršyklę (panaši koncepcija į šio magistrinio darbo programą). Dėl šios priežasties ši sistema nėra panaudojama.

Xcas yra C++ kalba realizuota, vartototojo sąsaja paremta (langai), CAS. Ši sistema nėra plačiai paplitusi, o jos panaudojimas (kaip ir Sage ateveju) kitame kontekste yra beveik neįmanomas, tenka atsisakyti ir šio pasirinkimo.

Maxima yra Common Lisp [7] kalba realizuota CAS, turinti tiek konsolinę, tiek ir langų vartotojo sąsają. Įdomi šios sistemos savybė yra galimybė su konsole bendrauti per socket‘us, t.y. siuntinėti ir gauti pranešimus iš kito proceso. Dėl šios priežasties ir buvo pasirinkta ši sistema.

2.5 Kompiuterinės algebros sistema – Maxima

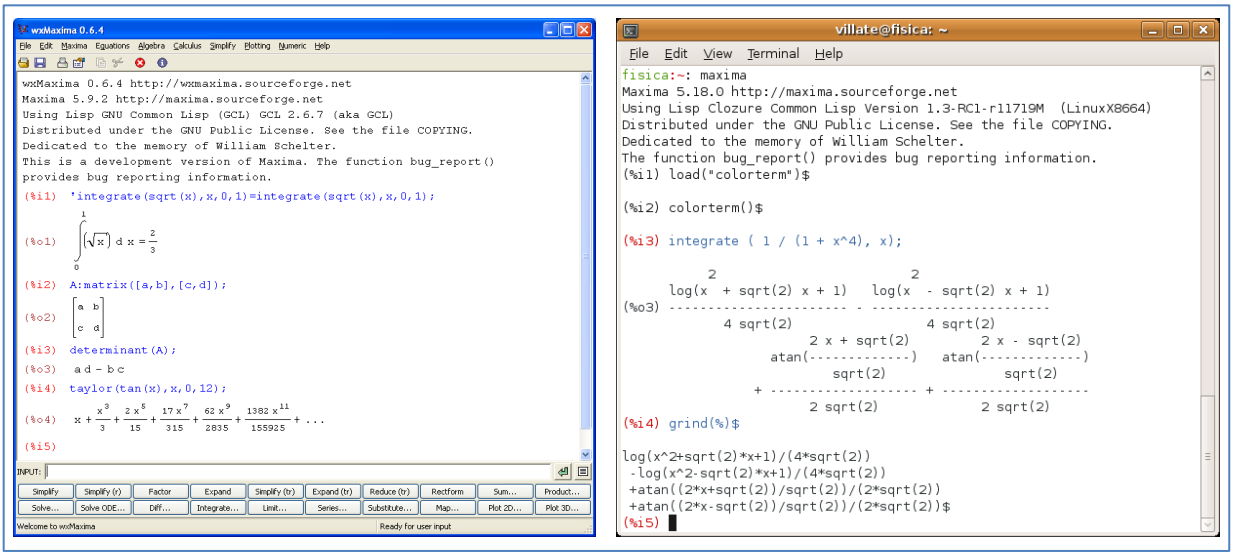
Maxima yra kompiuterinės algebros sistema, kurios pagrindą sudarto 1982 metų Macsyma [13] (Project MAC’s SYmbolic MAnipulator) versija. Maxima yra parašyta Common Lisp kalba ir veikia visose POSIX platformose tokios kaip Mac OS X, Unix, BSD, ir GNU/Linux, taipogi Microsoft Windows. Tai yra nemoka programinė įranga turinti „GNU General Public License“ licenziją.

Maxima turi užbaigtą programavimo kalbą, kurios sintaksė panaši į ALGOL, o žymėjimai į Lisp kalbos. Kadangi sistema yra parašyta Common Lisp kalba, sistemos funkcionalumą gali išplėsti pats vartotojas sukurdamas naujas funkcijas ir jas kviesdamas iš Maxima aplinkos.

Maxima yra pilnai realizuota CAS, kurios paskirtis yra atlikti simbolinius skaičiavimus, tačiau ji gali atlikti ir skaitinius skaičiavimus su sveikais ir realiais skaičiais, kurių reikšmes apriboja tik turima kompiuterio atmintis. Dėl šios priežasties realieji skaičiai gali praktiškai turėti begalinį tikslumą (skaitmenų kiekis po kablelio).

Skaičiavimams, kurie ypač naudoja daug realių reikšmių, Maxima gali sugeneruoti kitų kalbų (pvz. Fortran) kodą, kuris bus vykdomas daug greičiau negu Common Lisp.

Maxima turi daugybę grafinių sąsajų, tokių kaip „wxMaxima“, „Cantor“, „GNU TeXmacs“, „Lyx“. Tačiau šiame magistriniame darbe labiausiai domina konsolinė versija.



Pav. 7: Maxima grafinė sąsajos, kairėje sąsajos langas Windows sistemoje, dešinėje konsolė Linux sistemoje

2.6 Egzistuojantys lygiagrečių skaičiavimų sprendimai

Visos kompiuterinės algebros sistemos sugebančios simboliškai spręsti lygčių ir nelygybių sistemas, tai daro pasitelkdamos tik vieną procesą, t.y. nevykdo lygiagrečių skaičiavimų. Tai yra todėl, kad tam tikrais atvejais yra per brangu sukurti teisingą algoritmą, pigiau yra didinti skaičiavimo mašinos galią. Tačiau yra ir išimčių, tokių kaip matricų sudėtis, daugyba ir panašūs veiksmai, nereikalaujantys ypač specializuotų algoritmų, rezultatų radimui.

Tarp mokslinių straipsnių galima rasti ne tik „Fourier-Motzkin elimination“ (algoritmas nelygybėms spręsti) bet ir „Parallel Fourier-Motzkin elimination“ [1] algoritmo teorinius įgyvendinimus. Tačiau tokio lygiagretaus algoritmo įgyvendinimas užtruktų nemažai laiko,

kadangi kiekvieno nelygybių tipo atvejis turi būti nagrinėjamas atskirai, simbolinis skaičiavimas ir netiesinės nelygybės prideda dar daugiau sudėtingumo.

Egzistuoja keli paskirstytų skaičiavimų atlikimo tinkle paketai: „HPC-Grid for Maple“ [8] ir „Parallel Computing Toolkit“. Pirmasis naudoja „Maple“ CAS, o antrasis „Mathematica“ CAS. Tačiau šie paketai atlieka tik darbų paskirstymą kompiuterių tinkle, surenka rezultatus, dauguma funkcijų yra naudojama iš standartinių kompiuterinės algebros sistemų, tik kai kurios yra pakeičiamos aukštu lygiu išlygiagretintomis komandomis.

3 Reikalavimai sistemai

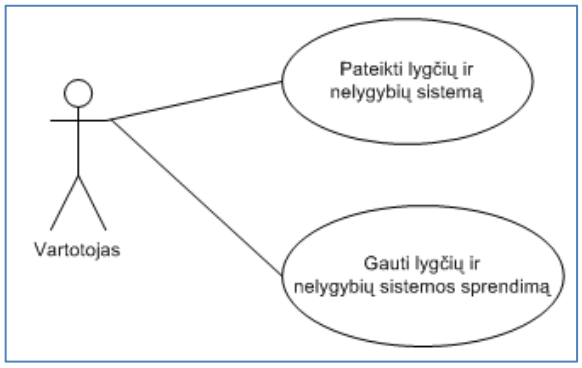
Šiame skyriuje apžvelgiama:

- Panaudos atvejai: pateikti lygčių ir nelygybių sistemą, gauti lygčių ir nelygybių sistemos sprendinį.
- Funkciniai reikalavimai: duomenų validacija, skaičiavimų būsenos sekimas, rezultatų pateikimas, skaičiavimų istorijos naršymas.
- Apribojimai sprendimui: panaudoti C++ MPI biblioteką, pritaikyti naršyklei, naudojimas be specialios PĮ ar žinių.

3.1 Panaudos atvejai

Kadangi sistema sprendžia algoritminį uždavinį, tėra tik du būtini panaudos atvejai. Pirmasis yra pateikti lygčių ir nelygybių sistemą, o antrasis yra sulaukti atsakymo. Atsakymo gavimas yra traktuojamas kaip panaudos atvejis, kadangi sprendimas užima šiek tiek laiko, vartotojas gali išeiti iš uždavinio lango, tada norėdamas pamatyti sprendinį, jis turi naviguoti atgal į tos užduoties langą.

Taip pat yra papildomų panaudos atvejų kurie yra realizuoti, bet nėra būtini. Tokie kaip: nežinomųjų generavimas iš sistemos, spręstų uždavinių sąrašo peržiūra, veiksmų atšaukimas ir panašiai.



Pav. 8: Panaudos atvejai

Panaudojimo atvejis 1: Pateikti lygčių ir nelygybių sistemą

Aktorius: vartotojas

Tikslas: skirta pateikti lygčių ir nelygybių sistemą programinei įrangai, jai suprantamu formatu

Ryšiai su kitais PA: sukelia "Gauti lygčių ir nelygybių sistemos sprendimą" panaudos atvejį

Prieš-sąlygos: vartotojas pasiruošia duomenis, kuriuos pateiks sistemai

Sužadinimo sąlyga: vartotojas pasirinko duomenų parteikimą vartotojo sąsajoje

Po-sąlyga: sistema apdoroja pateiktus duomenis

Pagrindinis scenarijus:

1. Vartotojas parengia duomenis skirtus apdoroti sistemai
2. Vartotojas pateikia duomenis sistemai
3. Sistema vykdo skaičiavimus

Alternatyvūs scenarijai:

Vartotojas pateikia netinkamus duomenis, pranešamos klaidos, skaičiavimai nevykdomi

Panaudojimo atvejis 2: Gauti lygčių ir nelygybių sistemos sprendinį

Aktorius: vartotojas

Tikslas: skirta pateikti lygčių ir nelygybių sistemos rezultatus vartotojui tinkama forma

Ryšiai su kitais PA: šis panaudos atvejis gali įvykti tik tada, kai prieš tai yra sužadinamas "Pateikti lygčių ir nelygybių sistemą" panaudos atvejis

Prieš-sąlygos: vartotojas prieš tai pateikė teisingus duomenis

Sužadinimo sąlyga: vartotojas pasirinko rezultatų peržiūra vartotojo sąsajoje

Po-sąlyga: vartotojas pasiima rezultatus

Pagrindinis scenarijus:

1. Sistema apskaičiuoja sprendinį
2. Vartotojas sistemai pateikia užklausą dėl rezultatų gavimo
3. Rezultatai pateikiami vartotojui

Alternatyvūs scenarijai:

Sistema dar nespėjo įvykdyti skaičiavimų

3.2 Funkciniai reikalavimai

Sistemai buvo iškelti keturi funkciniai reikalavimai:

1. Vartotojo duomenų validacija, nes tik tinkamai pateiktas sistemas galima išspręsti
2. Skaičiavimas turi turėti būseną. Esamos būsenos: „Sprendžiama“ ir „Išspręsta“.
3. Rezultatai pateikiami vartotojui skaitomu formatu, uždavinio detalėse rezultatus pavaizduojamas suformuotu paveikslėliu, sąraše paprastu tekstu
4. Vartotojas turi galimybę atsidaryti bet kokio anksčiau vykdyto skaičiavimo detales

Reikalavimas #:	1	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	1
Aprašymas:	Sistema privalo pranešti vartotojui apie neteisingus pateiktus pradinius duomenis				
Pagrindimas:	Užtikrinamas tinkamas programinės įrangos veikimas				
Šaltinis:	Vartotojas				
Tikimo kriterijus:	Sistema dirba tik su tinkamais duomenimis				
Užsakovo tenkinimas:	4			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2010 kovo 28 d.				

Reikalavimas #:	2	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	1
Aprašymas:	Sistema vartotojui praneša apie tikėtiną skaičiavimų ilgumą ir/ar būseną				
Pagrindimas:	Vartotojas turi informaciją apie skaičiavimų atlikimą				
Šaltinis:	Sistema				
Tikimo kriterijus:	Vartotojas žino kada tikėtis rezultatų				
Užsakovo tenkinimas:	5			Užsakovo netenkinimas:	3
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2010 kovo 28 d.				

Reikalavimas #:	3	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	2
Aprašymas:	Sistema pateikia sprendinius vartotojui priimtinu formatu				
Pagrindimas:	Vartotojas sugeba teisingai interpretuoti rezultatus				
Šaltinis:	Sistema				
Tikimo kriterijus:	Vartotojas gauna rezultatus				
Užsakovo tenkinimas:	4			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2010 kovo 28 d.				

Reikalavimas #:	4	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	2
Aprašymas:	Vartotojas gali gauti anksčiau įvykdytų skaičiavimų rezultatus				
Pagrindimas:	Vartotojui gali prireikti gauti anksčiau vykdytų sprendimų rezultatus				
Šaltinis:	Vartotojas				
Tikimo kriterijus:	Vartotojas passima ankstesnių skaičiavimų rezultatus				
Užsakovo tenkinimas:	3			Užsakovo netenkinimas:	3
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2010 kovo 28 d.				

3.3 Apribojimai sprendimui

- Sistemos kūrime panaudoti C++ MPI biblioteką
- Vartotojas sistemą turi pasiekti per naršyklę
- Vartotojas neturi įsidiegti specializuotos PĮ

4 Bendra architektūra

Šiame skyriuje apžvelgiama:

- Programų sistemą sudaro Vartotojo PĮ, Skaičiavimų PĮ ir Eq2Img atviro kodo sistema.
- Bendravimas tarp Vartototojo PĮ ir Skaičiavimų PĮ procesų naudojant socket'us. Bendravimo žingsniai iš Vartotojo PĮ pusės ir iš Skaičiavimų PĮ pusės.
- Uždavinio sistemų ir rezultatų atvaizdavimas vartotojui atliekamas naudojant Eq2Img atviro kodo sistemą, verčiant tekstą į paveikslėlį.

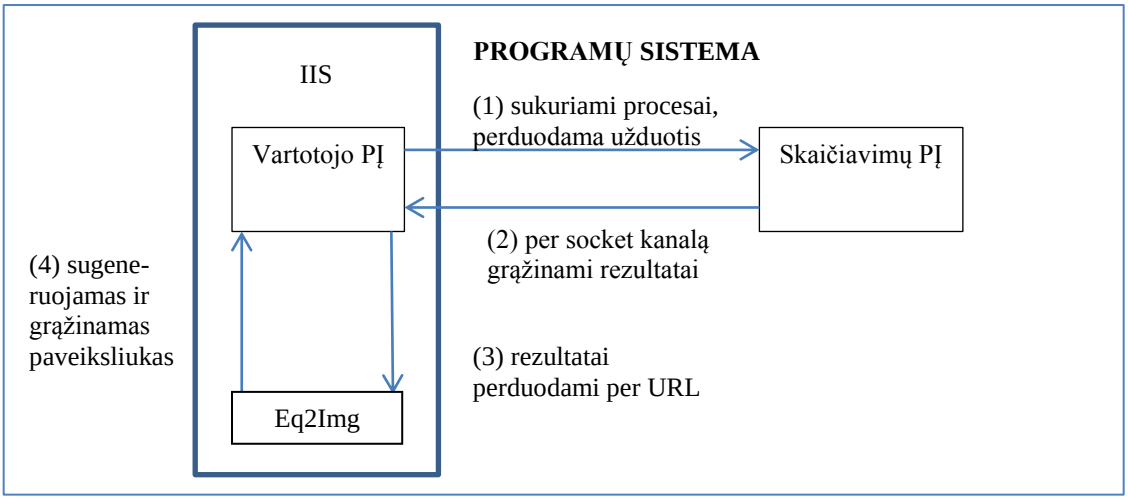
4.1 Bendras sistemos vaizdas

Šią programų sistemą sudaro trys dalys:

- Vartotojo PĮ
- Skaičiavimų PĮ
- Eq2Img [4] (tekstu užrasytų lygčių ir nelygybių vertimas į paveikslėlius)

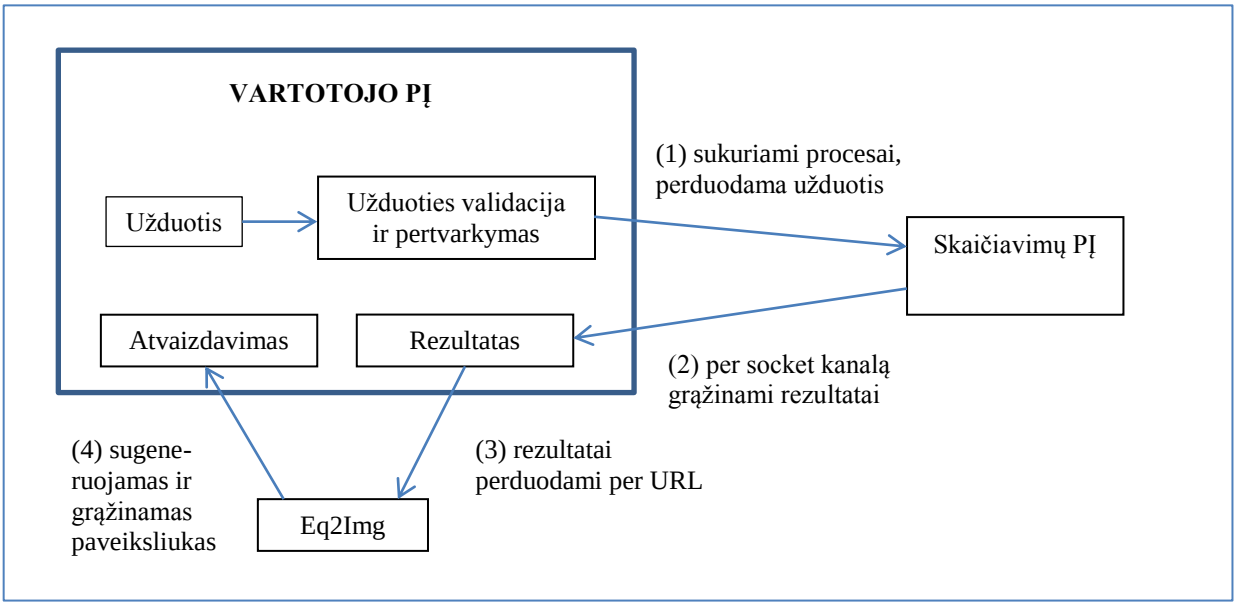
Vartotojo PĮ ir Eq2Img yra Web programos, todėl jos yra „hostinamos" IIS (Internet Information Service) aplinkoje. Ši aplinka leidžia šias programas pasiekti per naršyklę. Abi IIS programos yra parašytos ASP.NET aplinkoje, C# kalba. Vartototojo PĮ yra sukurta šio magistrinio darbo metu, o Eq2Img yra atviro kodo sistema, kuri LaTex [11] išraiškas verčia į žmonėms lengviau suvokiamus paveiksėlius.

Skaičiavimų PĮ yra realizuota C++ kalba, panaudojant lygiagretaus programavimo biblioteką MPI. Šią programą valdo Vartotojo PĮ.



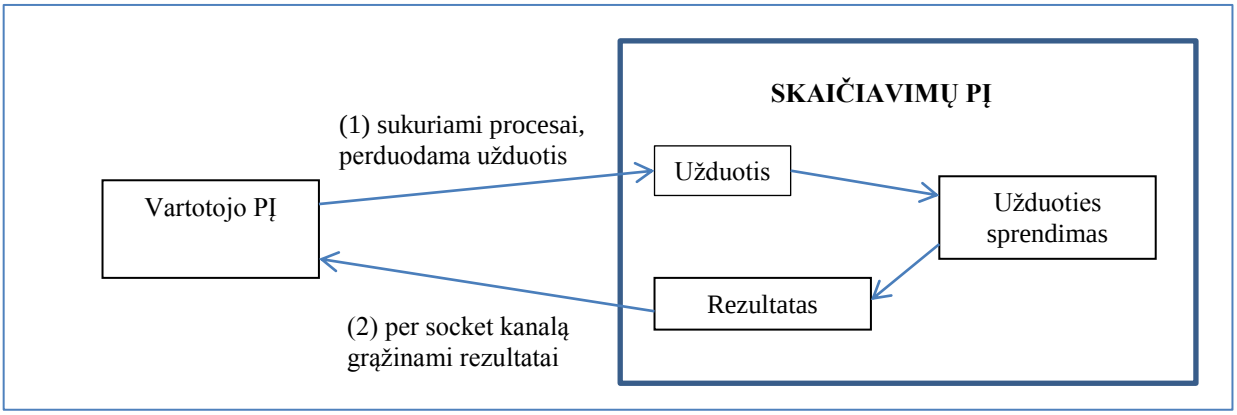
Pav. 9: Programų sistemos veikimo principas

Vartotojo PĮ suteikia vartotojui prieigą prie sistemos per naršyklę. Vartotojas sukuria naują uždavinį, įrašo sąlygą. Jeigu sąlyga neteisinga, parodomi klaidų pranešimai, kol vartotojas įrašo tinkamą sąlygą. Tada programa pertvarko pateiktą užduotį, išsaugo ją duomenų bazėje ir kuria Skaičiavimų PĮ procesus MPI aplinkoje, per argumentus perduodant užduotį. Vartotojo PĮ turi procesą, kuris socket'ų pagalba gauna rezultatą iš Skaičiavimų PĮ. Rezultatas pertvarkomas su Eq2Img ir atvaizduojamas kaip paveikslėlis.



Pav. 10: Vartotojo PĮ veikimo principas

Vartotojo PĮ sukuria Skaičiavimo PĮ procesus MPI aplinkoje. Turi būti sukurti bent du procesai, nes vienas bus valdantysis, o kiti skaičiavimų. Kadangi ši programa yra parašyta C++ kalba, užduotis jai perduodami per programos įėjimo taško (EntryPoint) argumentus. Taip pat paduodamas Vartotojo PĮ socket adresas, kuriuo grąžinti rezultatą. Gauta užduotis išsprendžiama ir valdantysis procesas išsiunčia rezultatą į Vartototojo PĮ.



Pav. 11: Skaičiavimų PĮ veikimo principas

4.2 Bendravimas tarp Vartototojo PĮ ir Skaičiavimų PĮ procesų

Socket‘ai [20] yra vienas efektyviausių būdų bendrauti tarp dviejų procesų. Socket‘ai naudoja prievadus (ports) kaip adresus, kuriais yra keičiamasi duomenimis. Įdomi socket‘ų savybė yra ta, kad jie nepriklauso nuo programavimo kalbos ir remiasi serverio – kliento architektūra. Šioje programų sistemoje socket‘ų pagalba bendrauja Vartotojo PĮ (C# kalba) ir Skaičiavimo PĮ (C++ kalba). Šiuo konkrečiu atveju bendravimas vyksta tik į vieną pusę – Skaičiavimo PĮ (klientas) siunčia rezultatus tam tikru žinomu adresu Vartotojo PĮ (serveris).

Žemiau esantis paveikslėlis parodo visą bendravimo procesą, pav. 12.

Vartototojo sąsaja turi PendingProblemService servisą, kuris sukasi visą programos veikimo laiką. Kuriant šį servisą yra sukuriamas socket‘ų serveris ir pririšamas prie tam tikro prievado numerio (diapazone 2000-2099). Suradus laisvą prievadą serveris pradeda jo klausytis, kol kas nors juo pasiųs duomenis. Turime paveikslėlyje parodytą žingsnį (1).

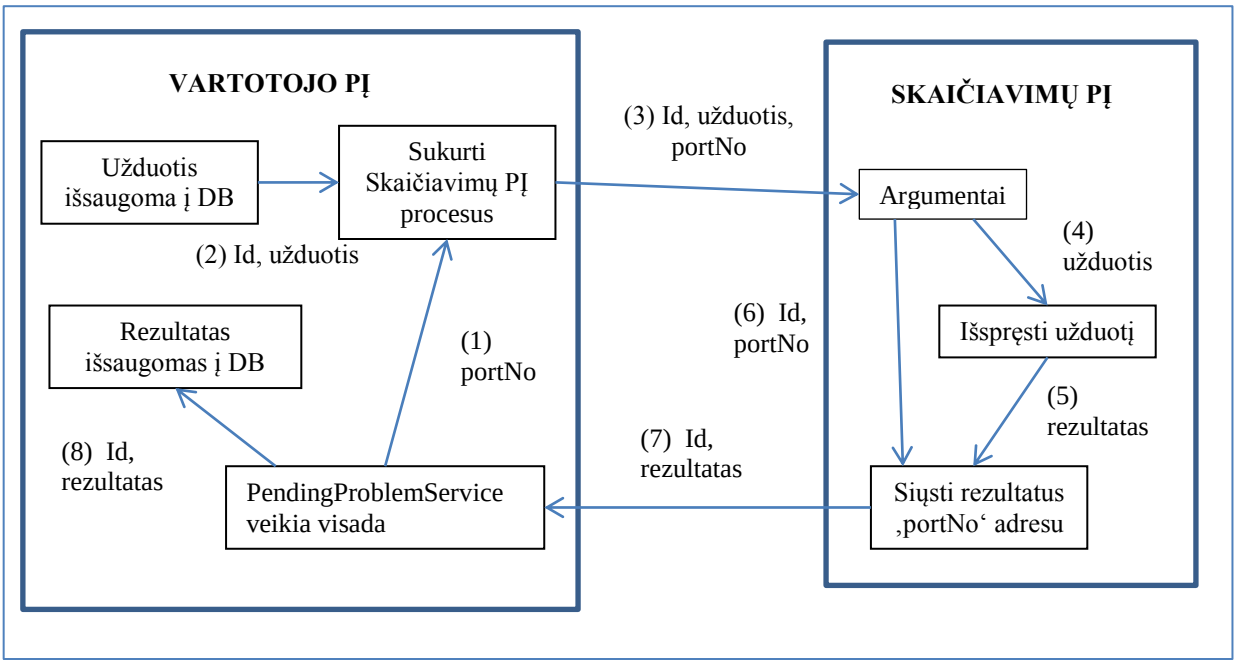
Vartotojui įvedus užduotį ji yra išsaugojama duombazėje, gaunamas užduoties Id ir pati užduotis, žingsnis (2).

Tada sukuriami Skaičiavimų PĮ procesai. Sukonstruojama tokia eilutė: `/c mpiexec -n <procesų skaičius> -noprompt <Skaičiavimų PĮ .exe failo adresas> <Id> <portNo, prievado numeris> <Užduotis>`. Tada kuriamas „cmd“ procesas su tokiu argumentu, taip pat nustatoma, kad komandinės eilutės konsolė nebūtų matoma. Paleidus šį procesą sukuriami MPI procesai, žingsnis (3). Į Skaičiavimų PĮ perduodami argumentai: užduoties Id, pati užduotis, ir serverio prievado numeris.

Žingsnyje (4) iš argumentų ištraukiama užduotis ir išsprendžiama. Tada žingsnyje (5) gaunamas rezultatas, kuris bus išsiųstas į serverį (Vartotojo PĮ).

Skaičiavimų PĮ turint rezultatą, ji susikuria socket‘ų klientą, kaip adresatą nustato žingsnyje (6) turimą ‚portNo‘, siunčia užduoties Id ir rezultatą, žingsnis (7).

PendingProblemService servisas gauna duomenis, pagal Id nustato pradinį uždavinį ir į duomenų bazę įrašo rezultatą, žingsnis (8). Rezultatui atsidūrus duomenų bazėje, netrukus atsinaujina naršyklės langas, pateikiantis rezultatą vartotojui.



Pav. 12: Bendravimas tarp Vartotojo PĮ ir Skaičiavimų PĮ procesų

4.3 Uždavinio sistemų ir rezultatų atvaizdavimas vartotojui

Skaičiavimų PĮ grąžina tekstu išreikštą rezultatą. Tačiau vartotojo toks atvaizdavimo būdas dažniausiai netenkina. Vienas iš būdų yra iš šio teksto sugeneruoti paveikslėlį. Šiai užduočiai įgyvendinti buvo panaudota atviro kodo sistema Eq2Img. Tačiau ši sistema priima tik LaTex formatu parašytą tekstą, tad rezultatas yra pakoreguojamas prieš duodant jį apdoroti Eq2Img sistemai.

Eq2Img pagal gautą išraišką sugeneruoja paveikslėlį ir išsaugo į diską, grąžindamas adresą į tą paveikslėlį. Duodant vėl tą pačią užklausą, paveikslėlis nėra iš naujo generuojamas, pasitikrinama ar toks paveikslėlis jau yra sugeneruotas, ir grąžinamas senasis. Sugeneruotos išraiškos pavyzdys yra paveikslėlyje 13.

Šis vaizdavimo būdas panaudojamas Vartotojo PĮ užduoties detalėse, kur rodoma tik viena užduotis. Užduočių sąraše, rezultatai ir užduotys rodomi paprastu tekstu, norint išvengti šimtų ar net tūkstančių kreipinių į šią sistemą.

$$c = 433^{1/5} * e^{2*i*\pi/5}$$

Pav. 13: Išraiškos atvaizdavimas <img src="http://myServer/Eq2Img?\\Large c = 433^{1/5}*e^{2*i*pi/5}?">

5 Vartotojo PĮ architektūra

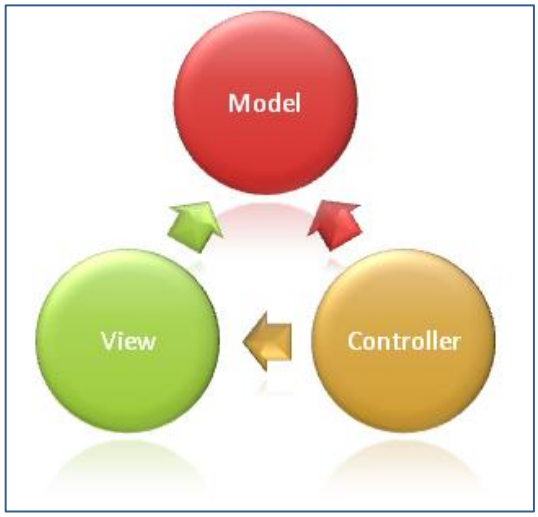
Šiame skyriuje apžvelgiama:

- Model-View-Controller paternas: pateikiamas MVC apibrėžimas, principai ir įgyvendinimas
- „Svogūno“ architektūros palyginimas su standartine sluoksnine architektūra ir „Svogūno“ "architektūros pranašumai
- Priklausomybių injekcijos (Dependency Injection) apibrėžimas ir realizacija
- Įvesties validacijos vykdymas, sintaksės teisingumo lentelė, klaidų pranešimai
- Skaičiavimų pabaigos nustatymas panaudojant Javascript kodą

5.1 Model-View-Controller paternas

Vartotojo PĮ architektūra yra sukurta ASP.NET MVC [9] framework‘o pagrindu, kuris yra MVC (modelis-vaizdas-kontroleris) paterno realizacija. Šis paternas remiasi atsakomybių paskirstymu:

- Modelis – atvaizduojamų duomenų struktūra
- Vaizdas – vartototojo sąsajos langas
- Kontroleris – valdo, kokį vaizdą rodyti ir kokius duomenis į jį paduoti



Pav. 14: MVC paterno dalys

MVC paternas pasižymi tokiomis veiklos savybėmis:

1. Vartotojas sąveikauja su vartotojo sąsaja (pvz. mygtuko paspaudimas)
2. Kontroleris reaguoja į vartotojo veiksmą, atlieka nustatytus veiksmus, nukreipia į vartotojo sąsajos langą
3. Kontroleris keičia modelio duomenis
4. Vaizdas pagal jam paduotą modelį sugeneruoja langą

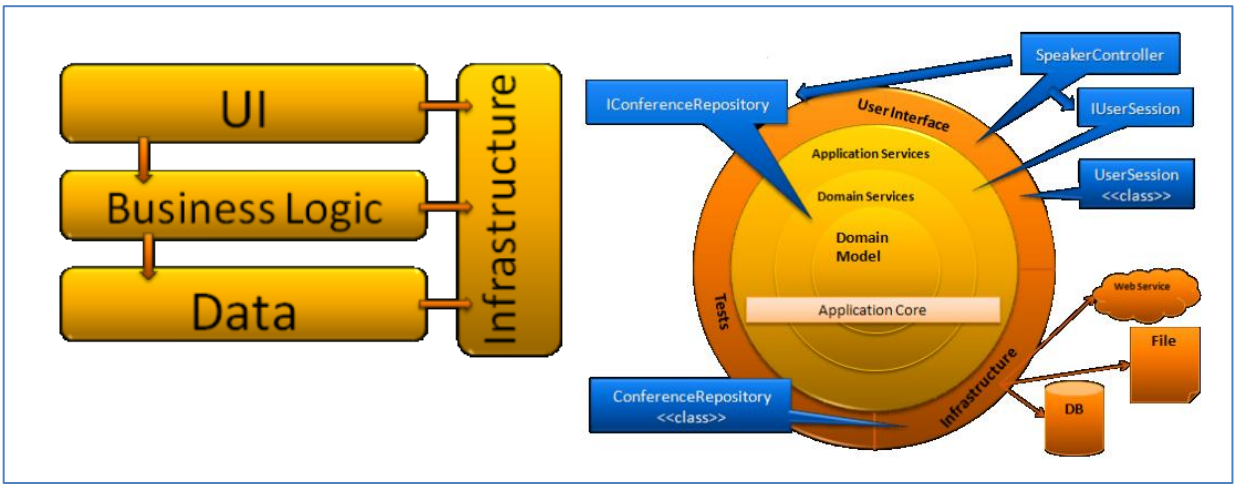
Vartotojo PĮ įgyvendinti užteko tik vieno užduoties kontrolerio ‚ProblemController', kuris turi veiksmus užduočių sąrašo rodymui (Index), užduoties sukūrimo lango rodymui (Create), užduoties detalių lango rodymui (Details) ir du papildomus metodus: GetUnknowns – iš duotos lygčių ir nelygybių sistemos rasti nežinomuosius (kuriant naują užduotį galima automatiškai gauti nežinomųjų sąrašą), IsResultCalculated – pagal užduoties ID nustatyti ar uždavinys jau išspręstas (pateikus uždavinį sprendimui, puslapis vis tikrinasi ar uždavinys yra išspręstas).

Lentelė 2: Vartotojo PĮ MVC paterno realizacija

Kontroleris	
ProblemController	
Modelis	**Vaizdas**
ProblemListModel	Index
ProblemListItemModel	
ProblemCreateModel	Create
ProblemDetailsModel	Details

5.2 „Svogūno" architektūra

„Svogūno" architektūra [10] leidžia kurti labiau palaikomas programas, kadangi ji pabrėžia sluoksnių atskyrimą programinės įrangos architektūroje. Ši struktūra remiasi interfeisais ir išorine infrastuktūra. Standartinė sluoksninė architektūra pasižymi tuo, kad kiekvienas sluoksnis priklauso nuo kito sluoksnio esančio po juo. Didžiulis sluoksninės struktūros trūkumas yra tas, kad ši struktūra pririša sluoksnius vienus prie kitų, sunku daryti architektūrinius keitimus, kaip naujų sluoksnių įterpimas.



Pav. 15: Sluoksninė architektūra (kairėje) ir „Svogūno“ architektūra (dešinėje)

Didžiausias standartinės sluoksninės (žiūr. pav.) architektūros trūkumas yra tas, kad vartotojo sąsaja ir programos logika yra pririšta prie duomenų priėjimo. Vartotojo sąsaja negali funkcionuoti be programos logikos, o programos logika negali funkcionuoti be duomenų priėjimo. Istoriškai susiklostė, kad priėjimo prie duomenų būdas keičiasi kas trejus metus, tačiau turint tokią struktūrą, neįmanoma padaryti pakeitimų, tad pasiliekama prie senų technologijų. Jeigu sluoksnių surišimas neleidžia atnaujinti sistemos naujomis technologijomis, rinkoje produktas praranda savo pozicijas. Dėl šios priežasties sistemos sensta ir tampa liktinėmis sistemomis, kol galiausiai jos yra perrašomos.

Modernesnis architektūrinis paternas yra „Svogūno“ architektūra. Pagrindinis jos bruožas yra tas, kad šios architektūros pagalba galima valdyti sluoksnių surišimą. Pagrindinė taisyklė yra ta, kad sluoksnis gali priklausyti tik nuo giliau esančių sluoksnių, bet negali priklausyti nuo sluoksnių, kurie yra arčiau išorės. Kitais žodžiais, surišimas galioja tik į centrą.

„Svogūno“ architektūra stipriai remiasi priklausomybių injekcijų principu. Programa kuriama remiantis interfeisais, o programos naudojimo metu (run-time) nustatoma, kokias klases reikia naudoti tų interfeisų vietoje.

Pagal senąjį modelį duomenų bazė yra pats giliausias sluoksnis, tačiau „Svogūno“ modelyje duomenų bazė yra išorinis objektas. Duomenų bazių iškėlimas į išorę turėtų pakeisti nuomonę, kad programa nėra kuriama ant duomenų bazės pagrindo, o ją naudoja tik duomenims pasiimti ir saugoti. Duomenų bazės atrišimas nuo programos sumažina programos palaikymo kaštus.

Kaip suprasti teiginį jog duomenų bazė yra iškelta į išorę? Kaip matoma paveikslėlyje Domain Services sluoksnyje yra aprašytas interfeisas IConferenceRepository. Infrastructure sluoksnyje yra aprašyta klasė ConferenceRepository. Tikroji realizacija yra išoriniame

„Svogūno“ sluoksnyje ir norint ją keisti, nėra jokių problemų, kadangi joks kitas sluoksnis nuo šio sluoksnio nepriklauso.

Vartotojo PĮ remiasi šiuo „Svogūno“ modeliu. Naudojami servisų (IProblemService – ProblemService) ir repositorių (duomenų priėjimas, IProblemRepository - ProblemRepository) sluoksniai, duomenų modeliai, vartotojo sąsaja.

5.3 Priklausomybių injekcijos

Priklausomybių injekcijos [5] arba įskiepijimas (Dependency injection) yra objektiškai orientuoto programavimo technika, kurios pagalba galima nuspręsti, kurias klasių realizacijas reikia naudoti. Kitais žodžiais tai yra tam tikras architektūrinis patern‘as, kurio pagalba programos dalys tampa mažiau surištos vienos su kitomis.

Paprastai priklausomybių injekcijos yra realizuojamos kaip interfeisas ir jį implementuojanti klasė, registracijos metu susiejant interfeisą su viena iš jo implementacijų. Kai programos vykdymo metu kuriama nauja klasė pagal interfeisą, patikrinamas registras ir pagal tai nusprendžiama kokios klasės objektą reikia sukurti.

Vartotojo PĮ naudoja ‚StructureMap‘ priklausomybių injekcijas servisams ir repositoriams.

5.4 Įvesties validacija

Vartotjo PĮ įvestis yra lygčių ir nelygybių sistemos bei nežinomųjų sąrašas, kuriuos suveda vartotojas. Sistema prieš vykdydama skaičiavimus turi įsitikinti, kad pateikti duomenys yra sintaksiškai taisyklingi ir neprieštarauja iš anksto numatytoms taisyklėms. Pateiktos lygtys ir nelygybės bei nežinomųjų sąrašas yra išanalizuojami, suskirtomi į lygčių, nelygybių ir nežinomųjų grupes, kurios siunčiamas į Skaičiavimo PĮ.

Sintaksės nagrinėjimas remiasi lentele, kuri aprašo koks elementas po kokio elemento gali eiti. Lentelė parodo ar po elemento aprašyto lentelės viršuje gali eiti elementas aprašytas lentelės kairėje.

Lentelė 3: Lygčių ir nelygybių sintaksės teisingumo lentelė

	Pradžia	(	)	Dvip. oper.	Vien. oper	Skaičius	Lyg., nelyg.	Funkcija	Simbolis	Nežinomasis	Pabaiga
Pradžia											
(	+	+		+	+		+	+			
)			+			+			+	+	
Dvip. oper.			+			+			+	+	
Vien. oper.	+	+		+			+				
Skaičius	+	+		+	+		+				
Lyg., nelyg.			+			+			+	+	
Funkcija	+	+		+	+		+				
Simbolis	+	+		+	+		+				
Nežinomasis	+	+		+	+		+				
Pabaiga			+			+			+	+	

Čia lentelėje: Dvip. oper. – dvipusis operatorius, Vien. oper. – vienpusis operatorius, Simbolis – koeficientas išreikštas simboliu (simbolinis skaičiavimas).

Taip pat išraiškų validavime galioja ir papildomos taisyklės, kurių viena yra ta, kad išraiška privalo turėti vieną lygybės arba nelygybės ženklą.

Susidarius nenumatytoms situacijomis vartotojui pranešama apie klaidą, nurodomas eilutės ir simbolio eilutėje numeris, ties kuriuo buvo surasta klaida. Galimų validacijos klaidų sąrašas: `Nežinoma klaida, Bloga atidarančiojo skliausto vieta, Bloga uždarančiojo skliausto vieta, Nėra atidarančiojo skliausto prieš uždarantįjį skliaustą, Bloga operatoriaus vieta, Bloga skaičiaus vieta, Skaičiuje per daug taškų, Taškas padėtas be skaitmenų, Bloga lyginimo operatoriaus vieta, Lyginimo operatorius jau buvo panaudotas, Skliaustai neuždaryti prieš lyginimo operatorių, Lygtis nėra tinkamai užbaigta, Nepanaudotas lyginimo operatorius, Skliaustai neuždaryti prieš lygties pabaigą, Sąlygoje nėra nežinomųjų, Bloga vieta funkcijai, Bloga vieta nežinomajam, Bloga vieta konstantai, Tokia konstanta neegzistuoja, Nežinomas simbolis`. Trys paskutiniai pranešimai turėtų skambėti taip: Bloga vieta simboliui, Toks simbolis neegzistuoja ir Neatpažintas ženklas.

5.5 Skaičiavimų pabaigos nustatymas

Pateikus lygčių ir nelygybių sistemas, skaičiavimas gali šiek tiek užtrukti. Vartotojas yra nukeliamas į užduoties detalių langą, bet sprendinys ten būna tuščias. Po kelių ar keliolikos sekundžių rezultatas pats atsiranda puslapyje.

Kaip buvo nustatyta, kad skaičiavimai buvo baigti, o rezultatai buvo parodyti naršyklės lange? Visų pirma skaičiavimai baigiasi tada, kai rezultatas patenka į duomenų bazę. Detalių puslapis turi realizuotą Javascript kodą, kuris jeigu nėra jokio rezultato atvaizduoto lange, kas kelias sekundes, vis dvigubinant intervalą (1 sek., 2, 4, 8, 16 ir t.t.), patikrina ar duombazėje neatsirado rezultato. Jeigu rezultatas yra duombazėje, Javascript kodas iš naujo perkrauna puslapį, jame jau matomas rezultatas.

6 Skaičiavimų PĮ architektūra

Šiame skyriuje apžvelgiama:

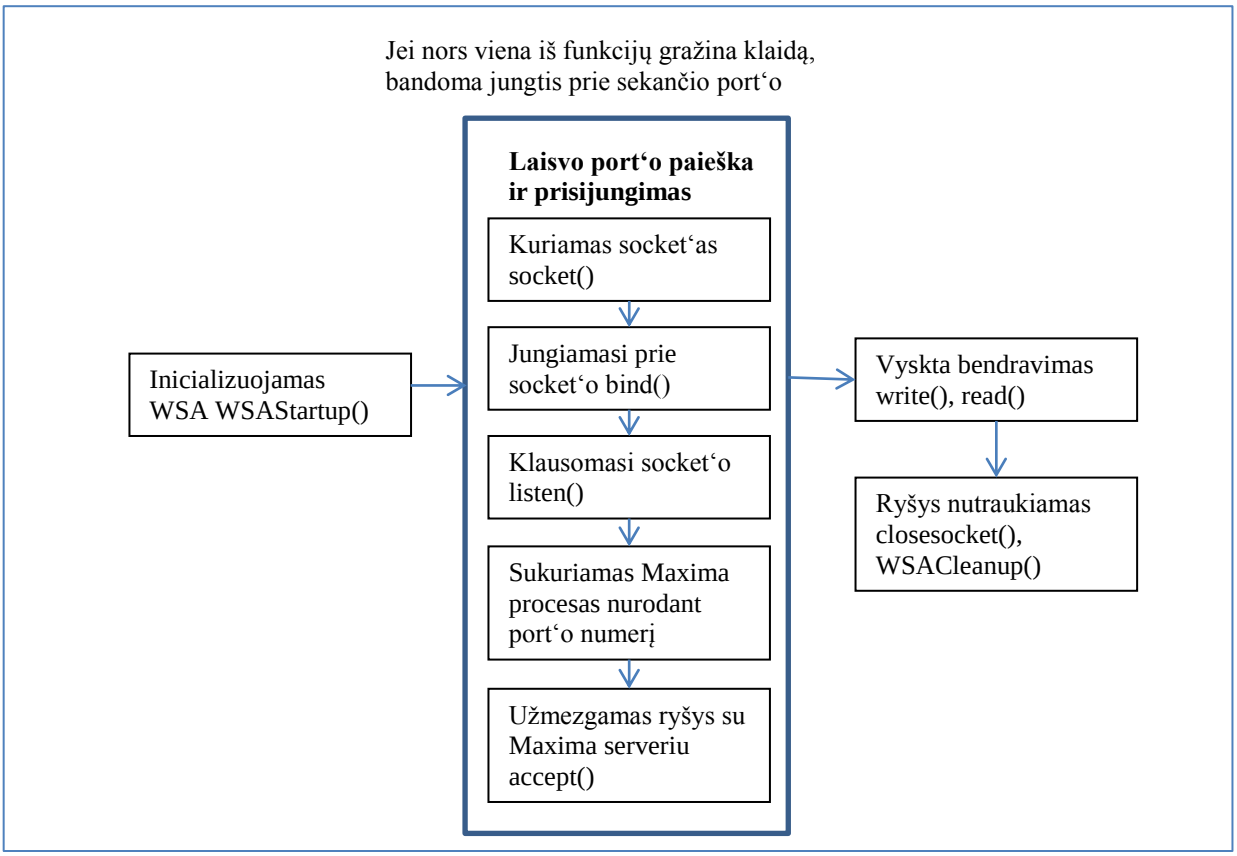
- Socket‘ai – bendravimo tarp procesų būdas: socket‘ų apibrėžimas ir panaudojimas, Skaičiavimų PĮ ir Maxima CAS bendravimas
- Maxima CAS paketo panaudojimas, Maxima argumentai ir komandos
- Lygiagretaus programavimo biblioteka MPI, naudojamos funkcijos, neblokuojančios funkcijos

6.1 Socket‘ai – bendravimo tarp procesų būdas

Šiame darbe buvo panaudotas Windows Sockets API (WSA), kurio pavadinimas trumpinamas kaip Winsock. Šis API nustato specifikaciją kaip Windows programinė įranga turėtų naudotis tinklo paslaugomis, ypač TCP/IP. Nustato standartinį bendravimo modelį tarp Windows TCP/IP kliento programos (tokios kaip FTP klientas ar naršyklė) ir TCP/IP protokolo.

Skaičiavimų PĮ atveju socketai yra naudojami išsiųsti rezultatą į Vartotojo PĮ (anksčiau aptarta) ir bendravimui su kompiuterinės algebros sistema Maxima. Čia bus detaliau apžvelgiamas antrasis variantas.

Maxima CAS turi galimybę per parametrus būti paleista kaip socket‘ų serveris nurodant port‘o (prievado) numerį. Šis porto numeris nustatomas perrenkant numerius nuo 4000 iki 4099, surandant neužimtą ir jį priskiriant paleidžiamai Maxima programai. Skaičiavimų PĮ žinodamą port‘o numerį komunikuoja su Maxima, keičiasi žinutėmis.



Pav. 16: Skaičiavimų PĮ ir Maxima CAS bendravimo per socket'us eiga

6.2 Maxima CAS paketo panaudojimas

Maxima CAS yra parašyta Common Lisp kalba, tad iškyla klausimas kaip šia programa pasinaudoti. Pasirodo šis paketas palaiko bendravimą per socket'us, tereikia nustatyti laisvą port'o numerį. Skaičiavimų PĮ atlieka šį darbą ir paleidžia naujus Maxima procesus pasinaudojant C++ funkcija „CreateProcess", nurodant paleidimo parametrus. Naudojami paleidimo parametrai [17]:

- -s <port>, sukuria Maxima serverį pasiekiamą per port'o numerį
- --very-quiet, neleidžia Maxima spausdinti jokios papildomos informacijos į išvestį, tokios kaip paleidimo pasisveikinimo žinutė, ar išraiškų numeravimas

Maxima CAS iš prigimties yra interaktyvi programa. Tai reiškia, kad bet kokiu uždavinio sprendimo momentu, ji gali paklausti vartotojo klausimo, į kurį reikia atsakyti taip arba ne (pvz. ar x yra sveikas skaičius, ar x yra teigiamas skaičius). Šio šalutinio efekto atsikratyti galima į Maxima nusunčiant tam tikras komandas. Paleidus Maxima yra nusunčiamos tokios komandos:

- display2d:false$ - neleidžiama spausdinti išplėstu formatu, pavyzdžiui laipsnius spausdinti reikia su ženklu ^, o ne virš skaičiaus kaip įprasta (šiuo atveju laipsnis atsidurs kažkur prieš reikšmę eilutėje)
- load(fourier_elim)$ - užkraunama nelygybių sprendimo biblioteka. Ji nebūna užkrauta iškart, kadangi ji naudoja labai daug resursų ir lėtina Maxima darbą
- algexact:true$ - jeigu įmanoma Maxima turėtų bandyti suprastinti išraiškas, nors tai ir būtų labai brangios operacijos
- linsolvewarn:false$, solvetrigwarn:false$, ratprint:false$ - šios komandos užtikrina, kad Maxima į rezultatus neįrašytų įspėjamųjų pranešimų apie prarandamą tikslumą ir pan.
- linel:10000 – nustatomas eilutės ilgis. Paprastai eilutės ilgis yra 80 simbolių, tad rezultatas būna pilnas naujos eilutės simbolių, šis nustatymas padeda nuo to apsisaugoti.

6.3 Lygiagretaus programavimo biblioteka MPI

MPI (Message-Passing-Interface) yra biblioteka skirta spręsti lygiagretumo reikalaujantiems uždaviniams. Pagrindinė jos savybė yra žinučių apsikeitimo mechanizmas, kuris leidžia tai atlikti tiek sinchroniškai (blokuojantis), tiek ir asinchroniškai (neblokuojantis). Dažniausiai MPI paremtos programos remiasi valdančiojo (master) proceso ir skaičiavimų (slaves) procesų modeliu. Šios dvi savybės yra panaudotos šioje sistemoje, tad yra realizuotas blokuojantis ir neblokuojantis bendravimas tarp procesų, turimas vienas valdantysis procesas (tad minimaliai turi būti kuriami bent du procesai, nes turi būti bent vienas, kuris atlieka skaičiavimus).

Skaičiavimų PĮ procesus sukuria Vartotojo PĮ sukurdama naują „mpiexec“ procesą, kuris savo ruoštu klonuoja šį procesą, kiekvienam iš jų suteikiant laipsnį (rank).

Naudojamos MPI funkcijos:

Lentelė 4: Panaudotos MPI bibliotekos funkcijos

Funkcijos pavadinimas	Funkcijos paskirtis
`MPI_Init`	Inicializuojama MPI vykdymo aplinka
`MPI_Comm_size`	Nustatomas sukurtų procesų kiekis
`MPI_Comm_rank`	Nustatomas proceso laipsnis (rank)
`MPI_Finalize`	Uždaro ir išvalo MPI vykdymo aplinką
`MPI_Send`	Vykdo blokuojantį žinutės siuntimą
`MPI_Recv`	Vykdo blokuojantį žinutės gavimą
`MPI_Irecv`	Pradeda neblokuojantį žinutės gavimą
`MPI_Test`	Tikrina ar buvo baigtas neblokuojantis žinutės siuntimas ar gavimas

Kam reikalingas neblokuojantis žinutės gavimas? Jeigu pavyzdžiui turėtume tris skaičiavimų procesus, valdantysis procesas turėtų apsispręsti griežtą eiliškumą, kokia tvarka reikėtų gauti žinutes. Taip sugaištama daug laiko tiesiog laukiant ilgai skaičiuojančio proceso, kai kiti jau pabaigė darbą ir laukia eilėje. Neblokuojančio žinutės gavimo atveju procesai greičiau atlikę savo darbą, gauna galimybę išsiųsti savo žinutes ir gali tuoj pat gauti naujo darbo.

7 Lygiagretaus sprendimo metodas

Šiame skyriuje apžvelgiama:

- Algortimo esmė, neefektyvaus darbo paskirstymo pavyzdys, efektyvaus darbo paskirstymo pavyzdys, bendroji algoritmo schema
- Valdančiojo proceso algoritmo schema ir žingsniai
- Skaičiavimų proceso algoritmo schema ir žingsniai
- Apribojimai taikomi sprendžiamiems uždaviniams

7.1 Algoritmas

Šiame darbe realizuotas lygiagretus skaičiavimų algoritmas. Algoritmo esmė yra išskaidyti pradinį uždavinį į dalis, efektyviai išdalinti dalinius uždavinius skaičiavimų procesams, neprarandant laiko surinkti dalinius rezultatus ir juos sujungti į vieną bendrą rezultatą. Esminė sėkmingos algoritmo realizacijos idėja yra negaišti laiko laukiant kol tam tikras skaičiavimų procesas baigs savo darbą, o priimti rezultatus iš to, kuris pirmas baigia. Tai yra dėl to, kad sprendžiamų dalinių uždavinių dažniausiai yra daugiau negu procesų (resursų).

Išnagrinėkime pavyzdį, kai turime tris skaičiavimų procesus ir devynis darbus su skirtingais laiko intervalais, parodančiais kiek laiko užima atlikti darbą. Realioje sistemoje nėra žinoma, kiek laiko užtruks atlikti darbą, todėl darbai yra imami iš eilės.

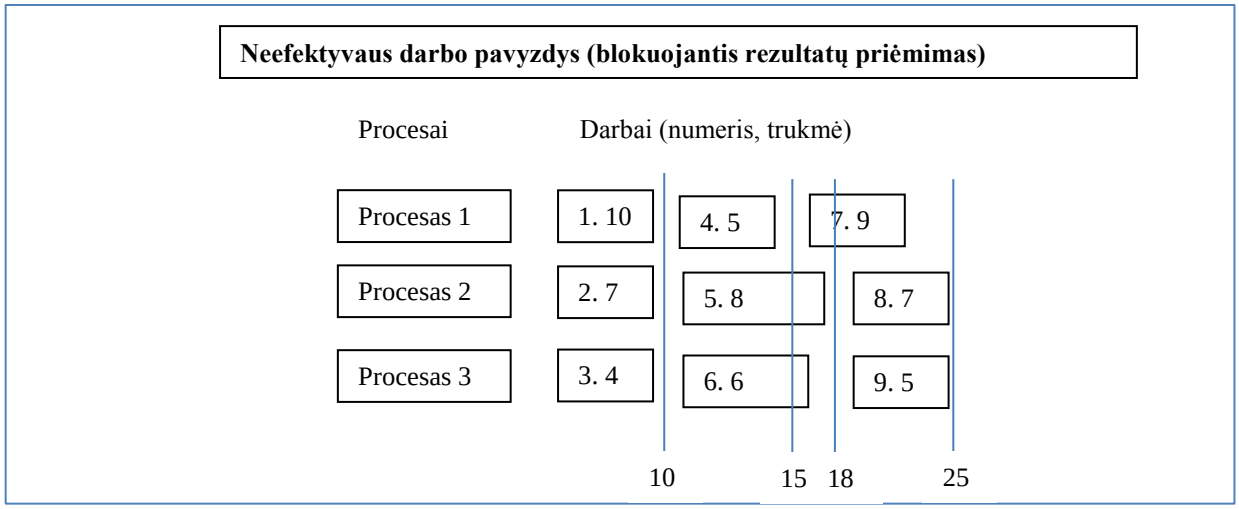
Pradinė sąlyga

Procesai	Darbai (numeris, trukmė)		
Procesas 1	1. 10	4. 5	7. 9
Procesas 2	2. 7	5. 8	8. 7
Procesas 3	3. 4	6. 6	9. 5

Pav. 17: Algoritmo efektyvumo tikrinimo pavyzdžio sąlyga

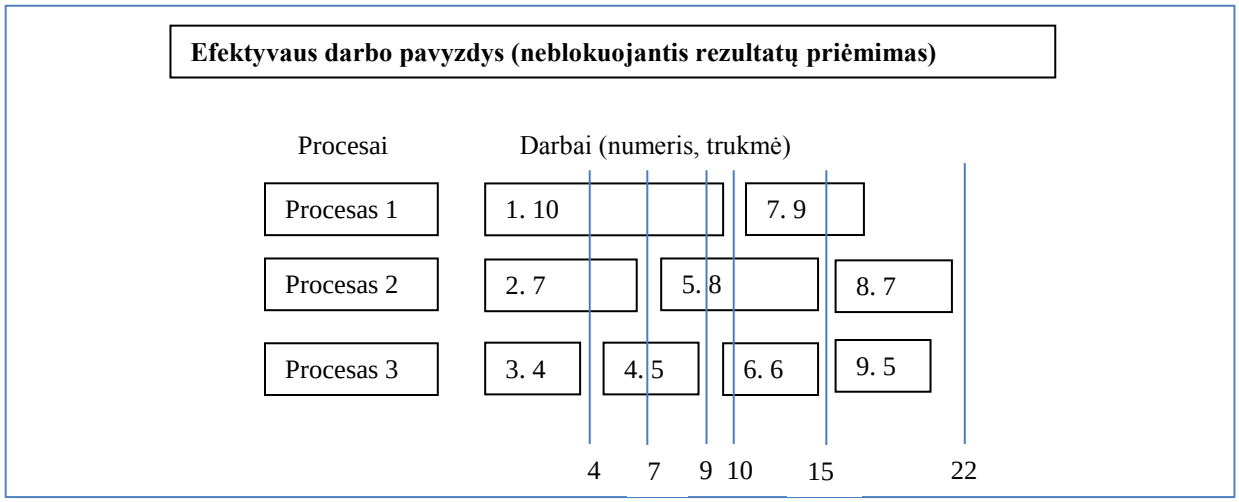
Neefektyvaus darbo pavyzdys pradeda apdoroti darbus 1, 2 ir 3, laukia rezultato iš darbo 1. Po 10 laiko vienetų rezultatas grąžinamas, tada procesui duodamas sekantis darbas 4. Tuo pačiu momentu gaunami rezultatai iš darbų 2 ir 3, nes jie jau seniai baigė darbus. Pradedami darbai 5 ir 6. Po 15 laiko vienetų darbą baigia 4 darbas, procesui priskiriamas 7 darbas. Po 18 laiko vienetų gaunamas rezultatas iš darbo 5, iškart gaunamas rezultatas ir iš darbo 6, laisviems

procesams priskiriami darbai 8 ir 9. Po 25 laiko vienetų visi procesai yra baigę darbus, nes neliko darbų eilėje. Šis pavyzdys parodo, kad atliekus darbą 2 procesas tuščiai laukė 3 laiko vienetus, atlikus darbą 3 - 6 laiko vienetus, o atlikus darbą 6 – 2 laiko vienetus. Dėl šios priežasties patiriami didžiuliai skaičiavimo laiko nuostoliai.



Pav. 18: Neefektyvaus darbų paskirstymo algoritmo pavyzdys

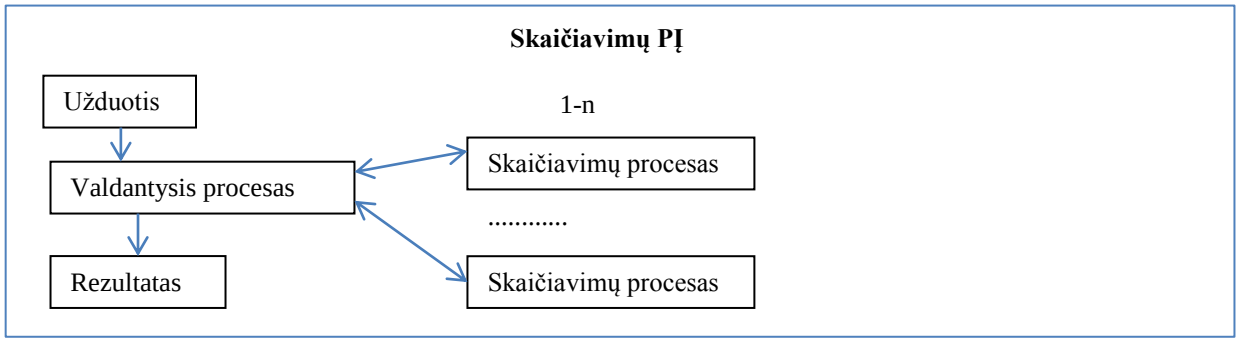
Efektyvaus darbo pavyzdys taip pat paskirto pirmus 3 darbus, tačiau po 4 laiko vienetų aptinka, kad darbas 3 yra baigtas ir procesui 3 priskiria sekantį darbą eilėje 4. Po 7 laiko vienetų baigiamas darbas 2 ir procesui 2 priskiriamas darbas 5. Taip darbai skirstomi, kol jų nelieka eilėje, ir jie visi atlikti. Kaip matome laikas sutrumpėjo nuo 25 laiko vienetų, neefektyvaus darbo pavyzdys, iki 22. Taip pat buvo sutaupyta 10 tuščiai praleistų laiko vienetų, kurie buvo sugaišti neefektyvaus darbo pavyzdyje.



Pav. 19: Efektyvaus darbų paskirstymo algoritmo pavyzdys

Šis efektyvumas gaunamas iš to, kad valdantysis procesas netikrina procesų darbų pabaigos (nelaukia kol tikrinamas procesas baigs darbą) iš eilės, bet tikrina, kuris iš jų darbą atliks pirmas ir galės imtis sekančio uždavinio.

Bendroji algoritmoji schema yra standartinė lygiagrečios programos schema. Ją sudaro vienas valdantysis procesas ir keli skaičiavimų procesai (privalo būti bent vienas). Reikia pažymėti, kad skaičiavimo procesai yra sukuriami vieną kartą ir naudojami daugelį kartų, priklausomai nuo to kiek darbų reikia atlikti.



Pav. 20: Bendroji skaičiavimų PĮ architektūra

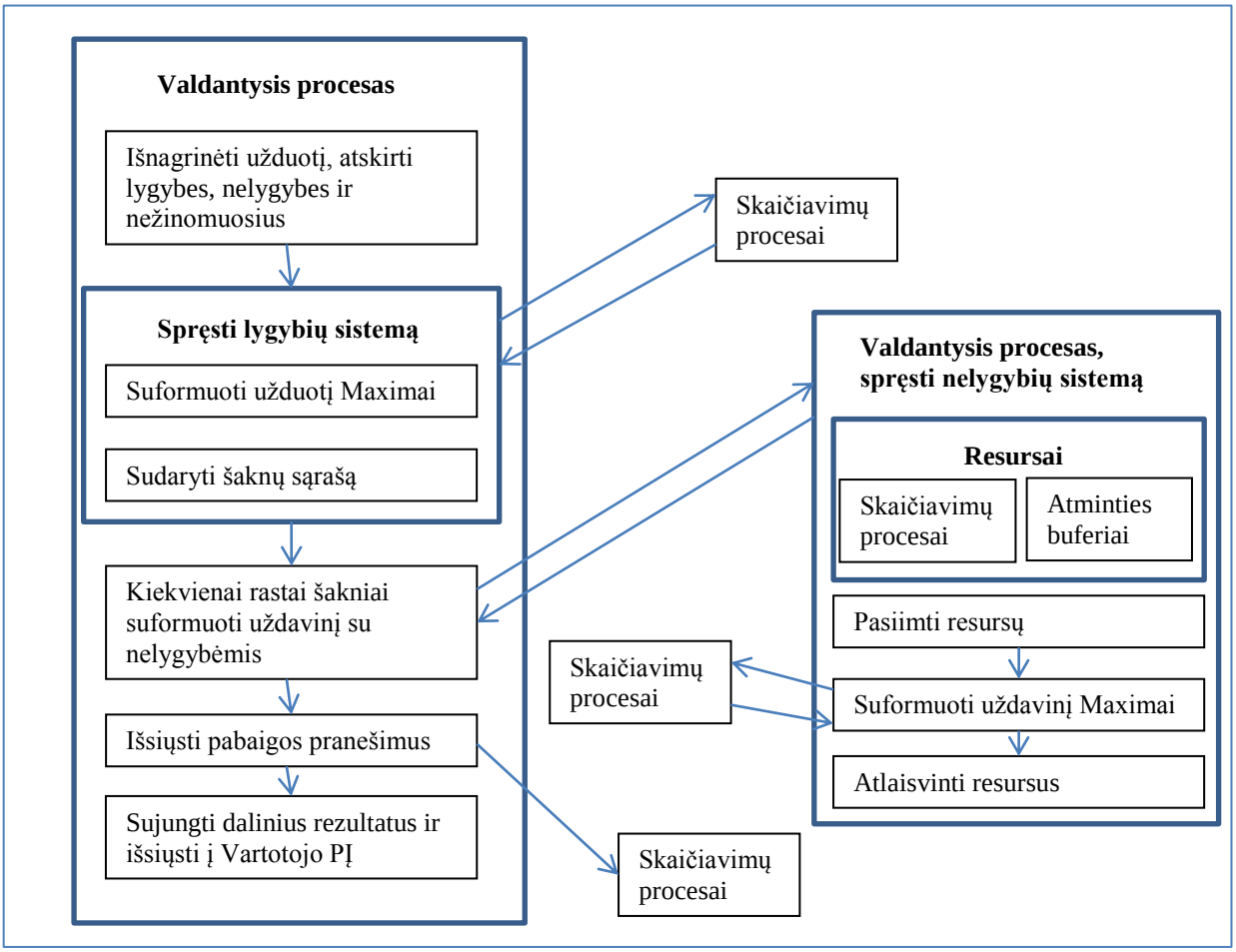
Šio algoritmo stiprioji pusė yra ta, kad jis gali dirbti su bet kokiais darbais, ne tik matematiniais uždaviniais. Tereikia sugebėti išskaidyti naują uždavinį ir surinkti rezultatą iš dalinių rezultatų. Tada pakeisti skaičiavimų kodą, kurį atlieka skaičiavimų procesai.

7.1.1 Valdančiojo proceso algoritmo dalis

Valdančiojo proceso paskirtis yra vykdyti darbų paskirstymą, išdalinti užduotis, surinkti rezultatus. Taip pat aptikus skaičiavimų procesus, kurie jau baigė savo darbą, duoti sekantį darbą esantį eilėje. Valdančiojo proceso algoritmo dalis vykdoma tokiais žingsniais:

1. Valdantysis procesas gauna užduotį, ją išnagrinėja ir išskiria lygtis, nelygybes ir nežinomuosius į atskirus poaibius.
2. Suformuoja lygčių sistemą iš esamų lygčių, pagal Maxima sintaksę. Šį uždavinį pasiunčia bet kuriam skaičiavimų procesui ir laukia atsakymo. Šiuo atveju naudojamas blokuojantis gavimas, kadangi be šių duomenų negalima tęsti darbo.
3. Gaunamos lygčių sistemos šaknys (simboliniai sprendiniai)
4. Kiekvienai rastai šakniai sudaromas naujas uždavinys pridedant esančią nelygybių sistemą. Uždavinys suformuojamas pagal Maxima sintaksę.

5. Laukiama kol atsilaisvins resursai, t.y. skaičiavimo procesas ir atminties buferis, kuriuo yra persiunčiami duomenys tarp procesų. Iš pradžių resursų pakanka, kol neišnaudojami visi laisvi procesai.
6. Naujai suformuotas uždavinys paduodamas į skaičiavimų procesą, tačiau dabar nenaudojamas blokuojantis rezultatų gavimas. Valdantysis procesas nuolatos tikrinasi, kuris procesas baigė savo darbą. Aptikus tokį procesą – rezultatas nuskaitomas iš jo buferio, procesas ir atminties buferis pažymimas kaip laisvas.
7. Žingsniai 5-6 kartojomai tol kol išsprendžiamos visos 4 žingsnyje sudarytos užduotys. Surinkti daliniai rezultatai sujungiami į vieną galutinį rezultatą.
8. Valdantysis procesas kiekvienam skaičiavimų procesui išsiunčia pabaigos žinutes, kad pastarieji užsidarytų ir neliktų jų atmintyje.
9. Galutinis rezultatas per socket'us išsiunčiamas į Vartotojo PĮ.



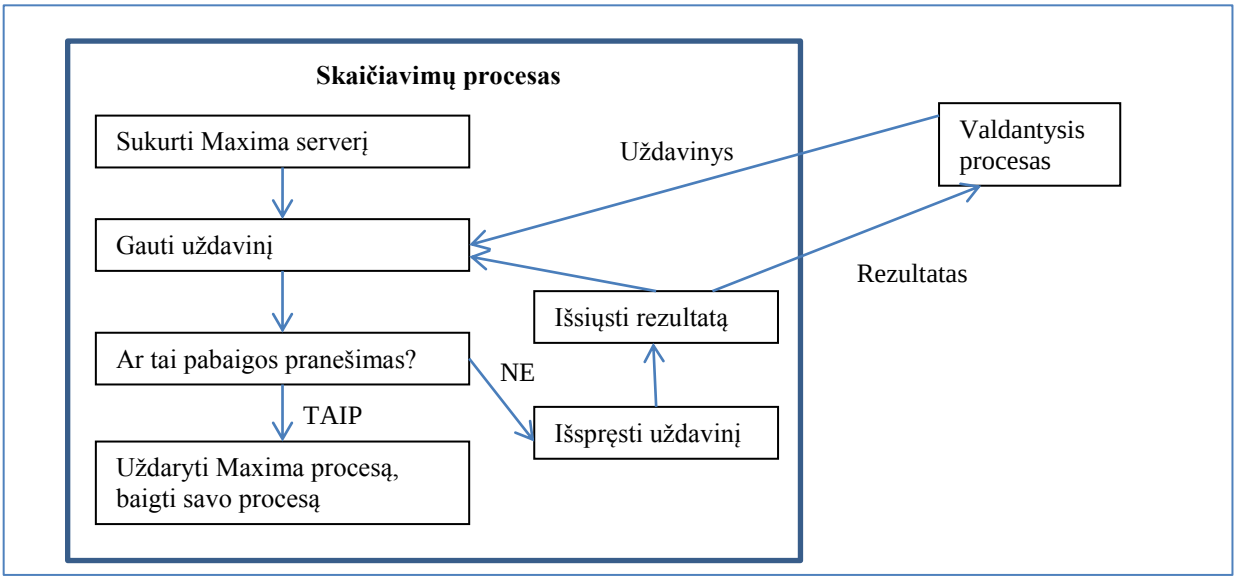
Pav. 21: Valdančiojo proceso algoritmo schema

Reikėtų pastebėti, kad valdančiojo proceso algoritmo schemoje „Skaičiavimų procesai" parodyti tris kartus, taip tik buvo lengviau atvaizduoti schemą, visi šie objektai turėtų būti vienu.

7.1.2 Skaičiavimų proceso algoritmo dalis

Skaičiavimų procesai yra sukuriami tuo pačiu metu kaip ir valdantysis procesas. Kol valdantysis procesas nagrinėja užduotį, skaičiavimo procesai pasileidžia po Maxima procesą kiekvienam ir laukia sau užduočių. Pabaigus darbą, skaičiavimų procesai, taip pat užbaigia Maxima procesą, nusiųsdami jai komandą – „quit“. Skaičiavimų proceco algoritmo dalis vykdoma tokiais žingsniais:

1. Pasileidžiamas Maxima procesas, bendravimas per socket‘us.
2. Laukiama uždavinio iš valdančiojo proceso.
3. Gavus uždavinį patikrina ar tai nėra pabaigos pranešimas.
4. Jeigu tai nėra pabaigos pranešimas, uždavinys sprendžiamas, sąlygą perduodant Maxima CAS ir gaunant rezultatą.
5. Rezultatas siunčiamas valdančiajam procesui ir vėl laukiama naujo uždavinio, kartojami žingnsiai 4-5.
6. Jeigu buvo gautas pabaigos pranešimas, tada Maxima procesas yra užbaigiamas, pats procesas taip pat baigia savo ciklą – procesas pasibaigia.



Pav. 22: Skaičiavimų proceso algoritmo schema

7.2 Apribojimai sprendžiamiems uždaviniams

Uždaviniai turi pasižymėti tokiomis savybėmis:

- Turi būti bent viena išraiška
- Visos iššraiškos privalo turėti vieną lygybės arba nelygybės ženklą
- Sistemoje privalo būti bent vienas nežinomasis

- Galimi tik sveikieji ir realūs skaičiai, kompleksinių skaičių atveju nevisada galima gauti atsakymą
- Skaičius (koeficientus) galima žymėti simboliais
- Naudojami operatoriai: +, -, *, /, ^
- Galimos funkcijos: sin, cos, log, sqrt
- Galimos konstantos: %e, %i
- Nepalaikomi integralai ir diferencialai

Pateikus užduotį, Vartotojo PĮ ją patikrins ir praneš apie rastas klaidas ir jų vietą, vartotojo pateiktoje sistemoje.

8 Skaičiavimų algoritmo tyrimas

Šiame skyriuje apžvelgiama:

- Naudojama techininė įranga, daugiabranduolė sistema
- Algoritmo greičio priklausomybė nuo nelygybių kiekio, procesų kiekio, dalinių uždavinių kiekio
- Sukurtos PĮ spartos palyginimas su Maple CAS, lygčių ir nelygybių sprendimo srityse

8.1 Naudojama techninė įranga

Skaičiavimo algoritmo tyrimas atliktas kompiuteriu, kurio programinę įrangą ir techininius parametrus sudaro:

- Windows 7 Ultimate, 64-bit Operating System
- Intel(R) Core(TM) i5 CPU 750 @ 2.67 GHz
- 4.00 GB RAM

Operacinė sistema ir jos 64 bit versija neturi jokios įtakos skaičiavimų algoritmo tyrimui, kadangi 64 bitų sistemos puikiai susidoroja ir su 32 bit programomis. Didelės įtakos neturi ir turimos atminties kiekis. Kiekvienas skaičiavimų procesas užima apie 7,5 MB darbinės atminties. Kiekvienas Maxima procesas užima apie 25 MB darbinės atminties, tad sumoje, kad vyktų skaičiavimai reikia apie 33 MB darbinės atminties per procesą.

Didžiausią įtaką skaičiavimų pajėgumams turi centrinis procesorius. Šis procesorius turi 4 branduolius, tai jam leidžia vienu metu lygiagrečiai vykdyti iki keturių procesų vienu metu, neprarandant skaičiavimų galios. Kadangi Skaičiavimų PĮ yra realizuota naudojantis MPI bibliotekomis, toks procesorius yra labai tinkamas atlikti tyrimams.

8.2 Algoritmo greičio priklausomybės

Algoritmo greitis labai priklauso nuo nelygybių sprendimo algoritmo sudėtingumo. Naudojamas standartinis nelygybių sprendimo algoritmas „Fourier-Motzkin elimination“, kurio sudėtingumas yra dvigubai eksponentinis. Pagal sudėtingumą galima nustatyti, kad ties tam tikra nežinomųjų ar lygčių kiekio riba, skaičiavimo laikas išaugs labai smarkiai.

Šio eksponentiškai augančio algoritmo laikas buvo matuojamas atsitiktinai generuojant lygčių ir nelygybių sistemas, kur nelygybių kiekis palaipsniui auga (skaičiavimai atliekami su 4 procesais):

- Išmatuojama, kiek laiko skaičiavimas užima, jeigu turimas trivialus uždavinys. Į šį laiką įeina Maxima procesų sukūrimas, žinučių siuntinėjimas. Turime lygtį:

  x=0

  Skaičiavimų trukmė: 0.409331 sekundės

- Turime sistemą su 2 nežinomaisias ir 1 nelygybe:

  +97*a-180*b<0
  -860*a+103*b=0
  Skaičiavimų trukmė: 0.584388 sekundės

  $$[a = 103*b/860, 0 < b]$$

  Pav. 23: Lygybių ir nelygybių sistemos sprendinys (1)

- Turime sistemą su 3 nežinomaisias ir 2 nelygybėmis:

  +877*a+416*b-815*c<=0
  -240*a-398*b+718*c>0
  +0*a+318*b+858*c=0
  Skaičiavimų trukmė: 0.60851 sekundės

  $$[a = 102683*c/46481, b = -143*c/53, 0 < c] or [b = -143*c/53, a < min(3957*c/530, 102683*c/46481)]$$

  Pav. 24: Lygybių ir nelygybių sistemos sprendinys (2)

- Turime sistemą su 4 nežinomaisiais ir 3 nelygybėmis:

  -441*a+720*b-153*c+185*d=0
  -561*a+693*b-510*c-925*d<=0
  +905*a+62*b+13*c-676*d>0
  +657*a-750*b+280*c-504*d<=0
  Skaičiavimų trukmė: 0.727899 sekundės

- Turime sistemą su 5 nežinomaisiais ir 3 nelygybėmis (rezultatas panašus į ankstesnį, kadangi nelygybių kiekis nesikeičia):

  +977*a+234*b-774*c+981*d+484*e=0
  +347*a+884*b+829*c-181*d-474*e<=0
  -94*a-215*b+913*c+978*d-276*e<0
  -2*a-946*b+474*c+389*d+551*e<0
  +160*a-405*b-913*c-36*d+40*e=0
  Skaičiavimų trukmė: 0.696707 sekundės

- Turime sistemą su 6 nežinomaisiais ir 4 nelygybėmis:

  -321*a+707*b+220*c-991*d+273*e-803*f<=0
  +228*a+397*b-333*c+366*d+0*e-377*f=0
  -202*a-841*b-762*c-910*d-958*e-232*f>0
  +881*a-467*b-13*c-505*d-889*e-287*f=0
  +191*a-675*b+333*c+64*d-839*e-547*f<0
  -228*a-21*b+645*c-791*d+58*e+780*f<=0
  Skaičiavimų trukmė: 0.967096 sekundės

- Turime sistemą su 7 nežinomaisiais ir 5 nelygybėmis:

  +905*a-629*b-678*c+298*d-244*e-891*f-593*g<0
  -981*a+32*b+815*c+298*d+249*e-655*f-841*g>0
  +306*a-451*b+225*c+898*d+786*e+177*f+560*g<0
  -220*a-928*b-24*c-888*d+175*e-649*f+989*g<0
  +417*a-705*b-877*c+291*d+279*e+339*f+817*g=0
  -653*a-856*b-49*c-694*d-747*e+699*f+675*g<=0
  -837*a-286*b-784*c+744*d+22*e-827*f-301*g=0
  Skaičiavimų trukmė: 1.77349 sekundės

- Turime sistemą su 8 nežinomaisiais ir 6 nelygybėmis (matome, kad turint 6 nelygybes skaičiavimų trukmė labai išauga):

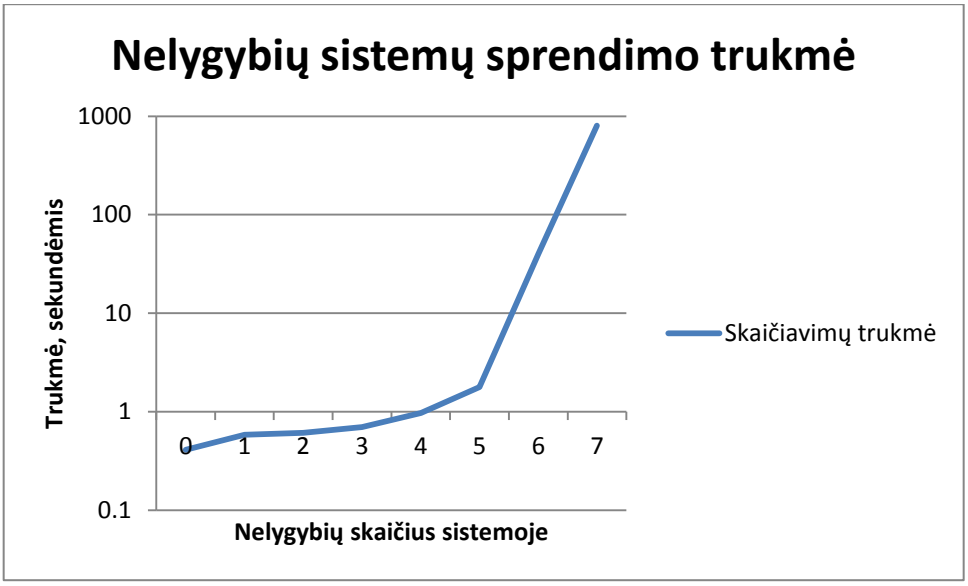
  +220*a+387*b+956*c+820*d+598*e+671*f+554*g-98*h<0
  -991*a+743*b-702*c+136*d+678*e-776*f-245*g-121*h<=0
  -308*a+508*b+838*c+477*d-619*e+894*f+312*g-382*h>0
  -120*a+675*b-834*c-482*d-46*e-89*f-723*g+305*h=0
  -332*a-621*b-882*c+10*d-957*e+93*f-73*g-916*h>0
  -822*a-10*b+229*c+178*d-867*e-562*f-958*g-911*h<=0
  +297*a+742*b+6*c-938*d-492*e-20*f-844*g-327*h<=0
  -174*a+490*b-211*c+387*d-142*e-627*f-841*g-913*h=0
  Skaičiavimų trukmė: ~40 sekundžių

- Turime sistemą su 8 nežinomaisiais ir 7 nelygybėmis:

  +496*a+877*b+76*c-985*d-834*e-778*f+844*g-650*h<=0
  -240*a-928*b-74*c-984*d-341*e-409*f-245*g-710*h<0
  +379*a-10*b+216*c+277*d-311*e-188*f+738*g-684*h<0
  +766*a-12*b+219*c-358*d-1*e-144*f+337*g+659*h=0
  -302*a-197*b-897*c+73*d+56*e+199*f+867*g-595*h<=0
  +128*a+762*b-474*c+301*d+806*e-900*f+473*g-130*h<0
  -262*a-782*b+188*c-737*d-723*e+849*f+238*g-922*h<=0
  -748*a-882*b+72*c-930*d-321*e-1*f-787*g-944*h<=0
  Skaičiavimų trukmė: ~15 minučių

Gauti matavimų rezultatai parodo, kad nelygybių sprendimo algoritmo (jį naudoja Maxima CAS) sudėtingumas yra eksponentinis. Skaičiavimų trukmė labai išauga, kai nelygybių kiekis tampa 6-ios ar daugiau nelygybių sistemoje. Taip pat verta pastebėti, kad esant šešioms nelygybėms sprendinio ilgis yra apie dvidešimt tūkstančių simbolių arba apie 20 KB, esant septynioms rezultato dydis pasieka 66 KB. Diagrama atvaizduota lograritminiu grafiku bazėje 10, toks atvaizdavimo būdas leidžia matyti vizualesnį reikšmių vaizdą, turint eksponentiškai augančias reikšmes.

Palyginimui, paskutinėje sistemoje nelygybių ženklus pakeitus lygybėmis ir dešinėje pusėje nulius pakeitus į kitus skaičius nelygius nuliui (kad nežinomieji nebūtų lygūs - 0), gavus lygybių sistemą, jos sprendimas užtruko vos 0.712135 sekundės.



Pav. 25: Sukurtos PĮ skaičiavimų trukmės priklausomybė nuo nelygybių kiekio

Realizuotas algoritmas remiasi efektyviu darbų išskirstymu skaičiavimų procesams. Toliau bus matuojama skaičiavimų trukmės priklausomybė nuo naudojamų procesų kiekio. Kadangi kompiuteris, kuriame vykdomi skaičiavimai, turi 4 branduolius, optimalus naudojamų procesų kiekis turėtų būti 4-5 procesai.

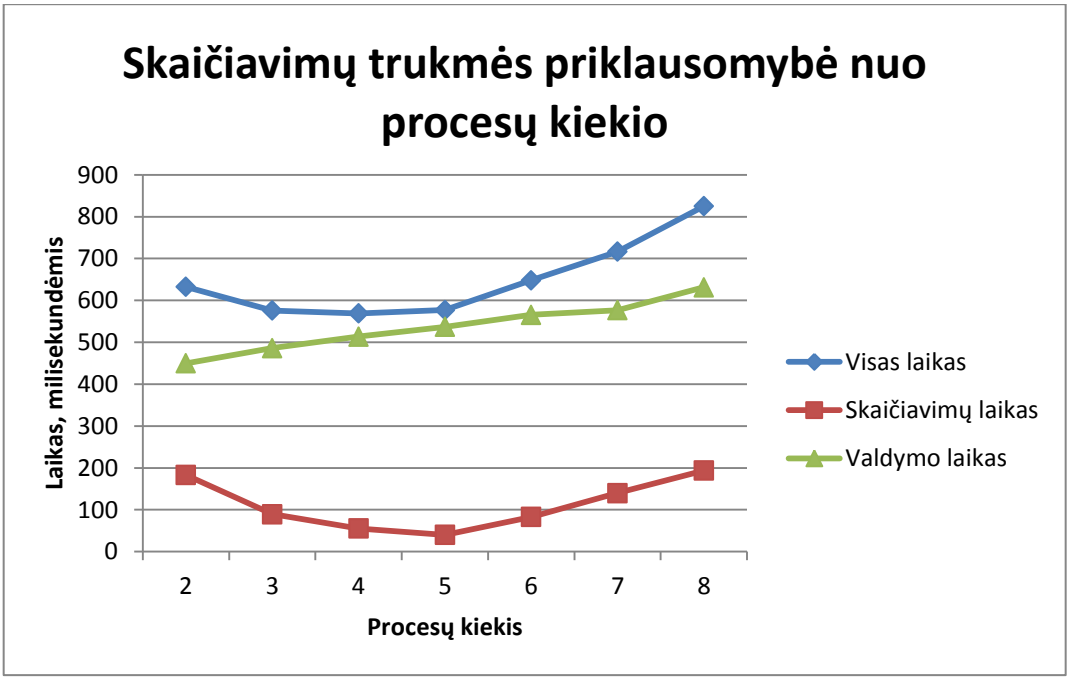
Šiam skaičiavimui bus naudojamas 60-o laipsnio polinomas, kurio šaknys yra sveikieji skaičiai nuo -29 iki 30, viso 60 šaknų. Kiekvienai iš 60 šaknų pritaikoma viena nelygybė, kuri į rezultatą įtraukia tik reikšmes didesnes už 0. Tad procesams teks atlikti 60 skaičiavimų, o rezultatai bus atrinktos šaknys nuo 1 iki 30.

Šio matavimu metu, kiekvienam procesų kiekiui, bus išmatuojamos dvi reikšmės, visas laikas, kurį sugaišo Skaičiavimų PĮ atlikdama skaičiavimus ir laikas, kuris buvo sugaištas tik atliekant skaičiavimų darbus.

Minimalus procesų skaičius yra 2, kadangi privalo būti valdantysis procesas ir bent vienas skaičiavimų procesas.

Lentelė 5: Skaičiavimų trukmės priklausomybė nuo procesų kiekio

Procesų kiekis	Visas laikas (ms)	Skaičiavimų laikas (ms)	Valdymo laikas (ms)
2	632.541	182.871	449.67
3	575.762	89.5293	486.2327
4	569.016	55.4998	513.5162
5	576.916	40.1224	536.7936
6	648.325	82.8115	565.5135
7	716.582	139.935	576.647
8	825.218	193.895	631.323



Pav. 26: Skaičiavimų trukmės priklausomybė nuo procesų kiekio

Pagal atliktus matavimus matome, kad matuojant laiką nuo užduoties pateikimo iki rezultato gavimo, efektyviausias variantas yra turėti 4 arba 5 procesus. Atsižvelgiant tik į skaičiavimų laiką, efektyviausias variantas yra turėti 5 procesus, 4 procesų variantas atsilieka nežymiai. Kitas laikas (užduoties nagrinėjimas, skaidymas, rezultatų sujungimas, Maxima serverio paleidimas ir sustabdymas) gaunamas iš viso laiko atėmus tik skaičiavimams sugaištą laiką. Kaip matome šis laikas tiesiškai auga priklausomai nuo procesų kiekio. Galima daryti išvadą, kad optimalus variantas yra turėti tiek procesų, kiek centrinis procesorius turi branduolių, šiuo atveju 4.

Kitas išmatuojamas dydis yra skaičiavimų trukmės augimo dydžio charakteristika. Šis dydis išmatuojamas tiesiškai didinant skaičiavimo darbų kiekį. Šiuo atveju vėl pasinaudosime polinomine lygtimi ir ieškosime šaknų, didesnių už 0. Matavimai pradedami su 10-o laipsnio polinomu ir baigiami 60-o laipsnio polinomu, polinomo laipsnį didinant po 10 kiekviename žingsnyje. Kaip atrodo 10-o laipsnio polinomas galima pamatyti žemiau.

$$x^{10}+5*x^9-30*x^8-150*x^7+273*x^6+1365*x^5-820*x^4-4100*x^3+576*x^2+2880*x=0$$

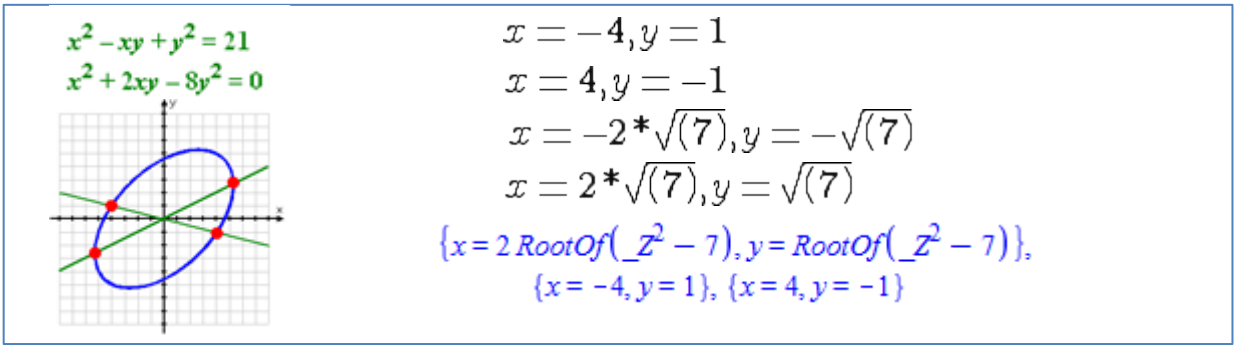
Pav. 27: 10-o laipsnio polinomas

Pav. 28: Skaičiavimų laiko priklausomybė nuo darbų kiekio

Iš gauto grafiko matoma, kad skaičiavimų trukmė yra tiesiškai priklausoma, nuo darbų kiekio. Kadangi šis matavimas buvo atliktas pasinaudojus polinominės lygties sprendimu, galima išvesti apytikslio skaičiavimų trukmės apskaičiavimo formulę formulę: ST = 0.3 + DK * 0.05, kur ST – skaičiavimų trukmė, DK – darbų kiekis, o laikas matuojamas sekundėmis.

8.3 Lygybių ir nelygybių sistemų greičio palyginimas su Maple CAS

Ši sistema (Maple [14]) buvo pasirinkta todėl, kad ji palaiko lygčių ir nelygybių sistemų sprendimą. Tačiau atidžiau panagrinėjus jos galimybes paaiškėjo, kad ji sprendžia tik tiesines nelygybių sistemas (tiriant algoritmo skaičiavimų laiką nuo nelygybių kiekio sistemoje, buvo naudojamos tik tokios sistemos). Pavyzdžiui jei turėtume paprastą kvadratinę nelygybių sistemą, sukurtoji PĮ rastų sprendinius, bet Maple CAS to nepadarytų be papildomo vartotjo įsikišimo (sveikaskaitines reikšmes suranda, tačiau turi problemų su šaknies išraiškomis):

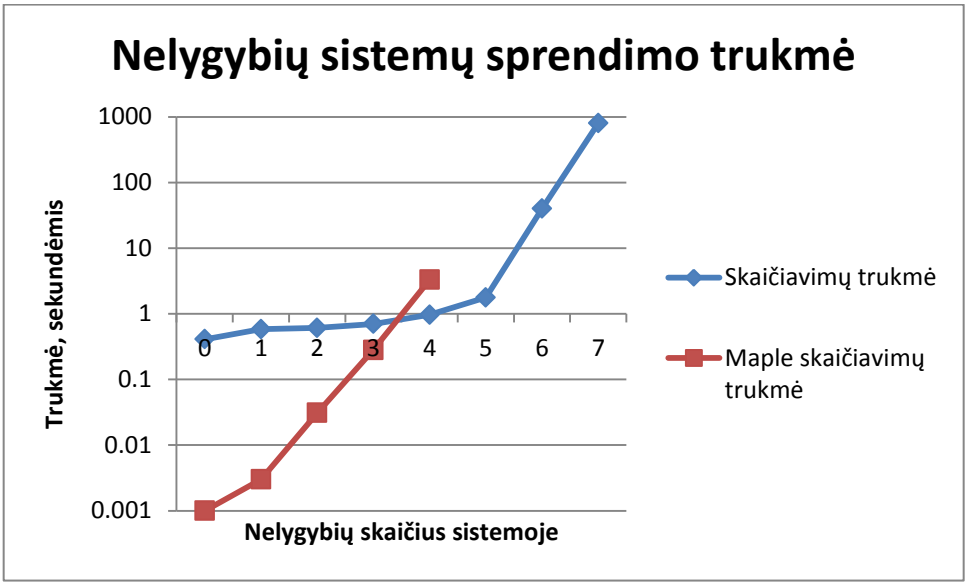


Pav. 29: Netiesinės lygčių sistemos sprendimo pavyzdys, viršuj sukurtos PĮ atsakymas, žemiau Maple atsakymas

Maple CAS lygčių ir nelygybių sistemų srendimo greitis matuojamos su tomis sistemomis, kurios buvo panaudotos matuojant sukurtos PĮ skaičiavimų greičio priklausomybę nuo nelygybių kiekio.

Lentelė 6: Sukurtos PĮ ir Maple sistemų skaičiavimų trukmės sprendžiant nelygybių sistemas

Nelygybių kiekis	Skaičiavimų trukmė (s)	Maple skaičiavimų trukmė (s)
0	0.409331	0.001
1	0.584388	0.003
2	0.60851	0.031
3	0.7	0.281
4	0.967096	3.296
5	1.77349	Nepavyko išspręsti
6	~40	Nepavyko išspręsti
7	~800	Nepavyko išspręsti



Pav. 30: Sukurtos PĮ ir Maple nelygybių sistemos sprendimų trukmės prklausomybė nuo nelygybių kiekio

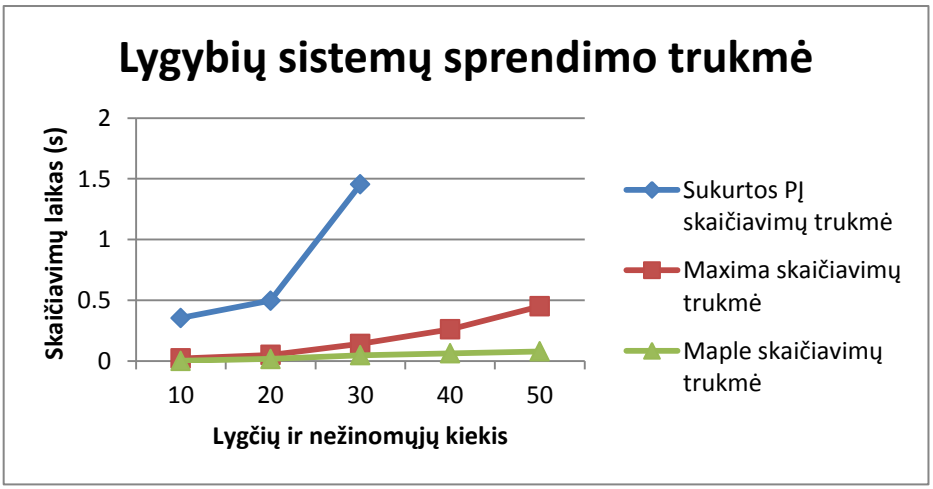
Kaip matoma iš grafiko, esant nelygybių kiekiui 3 arba mažiau Maple turi pranašumą, tai yra todėl, kad šiam paketui nereikia gaišti laiko pasiruošiant aplinką. Tačiau, kai nelygybių kiekis pasiekia 4, realizuota PĮ atsakymą gauna tris kartus greičiau. O kai nelygybių kiekis sistemoje pasiekia 5, Maple CAS nesugeba susidoroti su užduotimis, praėjus apie minutę laiko nuo sprendimo pradžios, išspausdinamas pranešimas, kad skaičiavimų atlikti nepavyko, siūloma performuluoti arba kitaip pateikti užduotį. Dėl šios priežasties grafike nėra Maple skaičiavimų trukmių pradedant reikšme 5, o kreivė numatomai kiltų daug stačiau negu kyla sukurtos PĮ kreivė.

Sukurtos programinės įrangos rezultatai šioje srityje yra geresni už Maple, dėlto, kad sukurtoji PĮ naudoja Maxima paketą, kurio nelygybių sprendimo algoritmo realizacija yra galimai daug spartesnė. Tai galėjo lemti du faktoriai, Maxima yra atviro kodo sistema ir pradėta kurti 13 metų anksčiau (1967) už Maple kompiuterinės algebros sistemą.

Matuojant lygybių sistemų sprendimo laiką buvo pastebėta silpnoji sukurtos PĮ algoritmo pusė. Sąlygos dydis labai įtakoja skaičiavimų trukmę. Atliekant šiuos matavimus buvo generuojamos lygčių sistemos turinčios nuo 10 iki 50 lygčių ir nežinomųjų.

Lentelė 7: Lygybių sistemų skaičiavimų trukmių priklausomybė nuo lygčių ir nežinomųjų kiekio

Lygčių ir nežinomųjų kiekis	Sukurto PĮ skaičiavimų trukmė (s)	Maxima skaičiavimų trukmė (s)	Maple skaičiavimų trukmė (s)
10	0.353303	0.02	0.001
20	0.496601	0.05	0.016
30	1.45456	0.14	0.047
40	Išmatuoti nepavyko	0.26	0.063
50	Išmatuoti nepavyko	0.45	0.079



Pav. 31: Lygybių sistemų skaičiavimų trukmių priklausomybė nuo lygčių ir nežinomųjų kiekio

Sukurtoji PĮ atlieka užduoties analizę, todėl daug laiko yra sugaištama, didėjant lygčių ir nežinomųjų kiekiui, apkrova stipriai išauga ir sistema daug laiko praleidžia dirbdama su eilutėmis, bet neatliekant skaičiavimus. Norint įsitikinti, kad tai nėra Maxima CAS problema, buvo išmatuoti ir jos veikimo greičiai. Lygčių sistemas teoriškai greičiausiai išsprendė Maple, tačiau jo pateiktos reikšmės neatitinka praktinio laiko, kadangi Maple paketas užgaišdavo nuo kelių iki kelių dešimčių sekundžių spręsdamas šias užduotis, bet skaičiavimams sugaišto laiko reikšmę pateikdavo mažesnę nei 0,1 sekundės.

9 Išvados

- Apžvelgtas MVC patern‘as, „Svogūno“ architektūra, priklausomybių injekcija (Dependency Injection) ir jų pranašumai prieš standartinę sluoksninę architektūrą
- Aprašytas panaudotas tarpprocesinis bendravimas, kuris remiasi komunikavimu per socket‘ų (prievadų, port‘ų) kanalus. Šis bendravimo modelis yra nepriklausomas nuo kalbos, kuria parašyta proceso programa
- Efektyviam lygčių ir nelygybių simbolinio sprendimo lygiagretaus metodo realizacijai buvo pasirinkta Maxima CAS ir lygiagretaus programavimo biblioteka MPI
- Atlikus sukurtos PĮ spartos tyrimus nustatytos jos vietos, kur sugaištama daug laiko: Maxima CAS procesų sukūrimas, tekstinis užduoties negrinėjimas, jos dalinimas ir siuntimas kitiems procesams
- Atlikus spartos priklausomybės tyrimus, nustatyta, kad PĮ sparčiausiai veikia tada, kai naudojama tiek procesų kiek yra procesoriaus branduolių (skaičiavimų vienetų). Sistemoje turint 6-ias ar daugiau nelygybių vykdymo laikas pradeda augti labai sparčiai
- Atlikus spartos palyginimą su Maple CAS nustatyta, kad sprendžiant nelygybių sistemas sukurtoji PĮ turi pranašumą, tačiau sprendžiant dideles (40-50 nežinomųjų ir tiek pat lygčių) lygčių sistemas Maple CAS yra pranašesnė
- Iš tyrimų rezultatų nuspręsta, jog sukurtoji PĮ (algoritmas), gali būti pagerinta modifikuojant tekstinę užduoties analizę ir jos skaidymą (darbas su string tipo kintamaisiais)

10 Literatūra

1. Christoph W. Kessler, Parallel Fourier Motzkin Elimination [Žiūrėta 2011-05-27]. Prieiga per internetą: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.54.657&rep=rep1&type=pdf
2. Comparison of Computer Algebra Systems [Žiūrėta 2011-05-27]. Prieiga per internetą: http://en.wikipedia.org/wiki/Comparison_of_computer_algebra_systems
3. Computer Algebra Systems [Žiūrėta 2011-05-27]. Prieiga per internetą: http://www.math.wpi.edu/IQP/BVCalcHist/calc5.html
4. Eq2Img, Enable Your Users to Write Math Equations in Your Web and Desktop Apps, 2005 [Žiūrėta 2011-05-27]. Prieiga per internetą: http://www.codeproject.com/KB/dotnet/Eq2Img.aspx
5. Griffin Caprio, Dependency Injection, 2005 [Žiūrėta 2011-05-27]. Prieiga per internetą: http://msdn.microsoft.com/en-us/magazine/cc163739.aspx
6. Fourier-Motzkin elimination algorithm [Žiūrėta 2011-05-27]. Prieiga per internetą: http://en.wikipedia.org/wiki/Fourier–Motzkin_elimination
7. Guy L. Steele, Common Lisp the Language [Žiūrėta 2011-05-27]. Prieiga per internetą: http://www.cs.cmu.edu/Groups/AI/html/cltl/cltl2.html
8. HPC-Grid Toolbox for Maple [Žiūrėta 2011-05-27]. Prieiga per internetą: http://www.maplesoft.com/products/toolboxes/HPCgrid/index.aspx
9. Jeffrey Palermo, Ben Scheirman, and Jimmy Bogard, ASP.NET MVC in Action, 2009
10. Jeffrey Palermo, The Onion Architecture, 2008 [Žiūrėta 2011-05-27]. Prieiga per internetą: http://jeffreypalermo.com/blog/the-onion-architecture-part-1/
11. LaTex [Žiūrėta 2011-05-27]. Prieiga per internetą: http://en.wikipedia.org/wiki/LaTeX
12. Lygčių sprendimas ir jų tipai [Žiūrėta 2011-05-27]. Prieiga per internetą: http://en.wikipedia.org/wiki/Equation_solving.
13. Macsyma [Žiūrėta 2011-05-27]. Prieiga per internetą: http://en.wikipedia.org/wiki/Macsyma
14. Maple: Computer Algebra System [Žiūrėta 2011-05-27]. Prieiga per internetą: http://www.math.toronto.edu/help/maple99.pdf
15. Mathematics and Computer Science Division Argonne National Laboratory, MPICH2 User‘s Guide, 2004 [Žiūrėta 2011-05-27]. Prieiga per internetą: http://phase.hpcc.jp/mirrors/mpi/mpich2/downloads/mpich2-doc-user.pdf
16. Richard H. Rand. Introduction To Maxima, 2010 [Žiūrėta 2011-05-27]. Prieiga per internetą: http://maxima.sourceforge.net/docs/intromax/intromax.html

17. Maxima – Linux man page [Žiūrėta 2011-05-27]. Prieiga per internetą: http://linux.die.net/man/1/maxima
18. Symbolic Computation (An Editorial), 1985 [Žiūrėta 2011-05-27]. Prieiga per internetą: http://www.risc.jku.at/about/editorial/editorial.pdf
19. System of Linear equations: Gaussian Elimination [Žiūrėta 2011-05-27]. Prieiga per internetą: http://www.sosmath.com/matrix/system1/system1.html
20. Windows Sockets [Žiūrėta 2011-05-27]. Prieiga per internetą: http://www.sockets.com/winsock.htm

11 Terminų ir santrumpų žodynas

ASP.NET	Active Server Pages for .NET, Microsoft web puslapių kūrimo technologija
API	Application Programming Interface
CAS	Computer algebra system, kompiuterinės algebros sistema
Eq2Img	Atviro kodo programa, verčianti Latex tekstą į paveikslėlį
IIS	Internet Information Services, web aplikacijų serveris Windows aplinkoje
LaTex	Dokumentų formavimo kalba, tinkama matematinių išraiškų vaizdavimui
Maple	Komercinė kompiuterinės algebros sistema
Maxima	Atviro kodo kompiuterinės algebros sistema
MPI	Message Passing Interface, C++ biblioteka skirta lygiagrečių programų kūrimui
Pattern	Design pattern, tarp programuotojų priimtas architektūrinės problemos sprendimas
Repositorius	Repository, duomenų pasiekimo prieiga (DB)
MVC	Model-View-Controller pattern (Modelis-Vaizdas-Kontroleris), web programų kūrimo modelis
PA	Panaudos atvejis
PĮ	Programinė įranga
Port	Kompiuterio prievadas, skirtas tinklinei ar tarpprocesorinei komunikacijai
POSIX	Portable Operating System Interface for Unix, unix OS šeima
Priklausomybių injekcija	Dependency injection, interfeisais paremtas programavimo stilius, kai tikroji klasė nustatoma programos veikimo metu
Socket	Programinė, bendravimo per prievadus, abstrakcija
sqrt	Šaknies traukimo operacija
„Svogūno“ architektūra	Architektūra, kuri remiasi sluoksnių tarpusavio ryšių valdymu ir priklausomybių injekcijomis