KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

EDVINAS BAZARAS

LĄSTELIŲ PLYŠINIŲ JUNGČIŲ MARKOVO MODELIO SUDARYMO SISTEMA

Magistro baigiamasis darbas

Darbo vadovas
doc. dr. H. Pranevičius

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

EDVINAS BAZARASS

LĄSTELIŲ PLYŠINIŲ JUNGČIŲ MARKOVO MODELIO SUDARYMO SISTEMA

Magistro baigiamasis darbas

Darbo vadovas
doc. dr. H. Pranevičius

Recenzentas
doc. dr. E. Valakevičius

KAUNAS, 2013

AUTORIŲ GARANTINIS RAŠTAS

DĖL PATEIKIAMO KŪRINIO

2013 - Birželio - 20 d.

Kaunas

Autorius, **EDVINAS BAZARAS**,

(vardas, pavardė)

patvirtina, kad Kauno technologijos universitetui pateiktas baigiamasis magistro darbas (toliau vadinama – Kūrinys)

LĄSTELIŲ PLYŠINIŲ JUNGČIŲ MARKOVO MODELIO SUDARYMO SISTEMA

(kūrinio pavadinimas)

pagal Lietuvos Respublikos autorių ir gretutinių teisių įstatymą yra originalus ir užtikrina, kad

1) jį sukūrė ir parašė Kūrinyje įvardyti autoriai;
2) Kūrinys nėra ir nebus įteiktas kitoms institucijoms (universitetams) (tiek lietuvių, tiek užsienio kalba);
3) Kūrinyje nėra teiginių, neatitinkančių tikrovės, ar medžiagos, kuri galėtų pažeisti kito fizinio ar juridinio asmens intelektinės nuosavybės teises, leidėjų bei finansuotojų reikalavimus ir sąlygas;
4) visi Kūrinyje naudojami šaltiniai yra cituojami (su nuoroda į pirminį šaltinį ir autorių);
5) neprieštarauja dėl Kūrinio platinimo visomis oficialiomis sklaidos priemonėmis.
6) atlygins Kauno technologijos universitetui ir tretiesiems asmenims žalą ir nuostolius, atsiradusius dėl pažeidimų, susijusių su aukščiau išvardintų Autorių garantijų nesilaikymu;
7) Autoriai už šiame rašte pateiktos informacijos teisingumą atsako Lietuvos Respublikos įstatymų nustatyta tvarka.

Autorius

_____

_____ (vardas, pavardė)

(parašas)

Turinys

Lentelės ... 3
Paveikslai/Diagramos ... 4
Formulės ... 6
Santrauka ... 7
Summary ... 8
1 Įvadas ... 9
1.1 Dokumento paskirtis ... 9
1.2 Aktualumas ... 9
1.3 Darbo tikslas ... 9
2 Literatūros apžvalga ir techninių sprendimų analizė ... 10
2.1 Plyšinė jungtis ... 10
2.2 Analizė ... 11
2.3 Alternatyvios sistemos ... 12
3 Projektinė dalis ... 13
3.1 Sistemos paskirtis ... 13
3.2 Sistemos tikslai ... 13
3.3 Reikalavimai sistemai ... 14
3.3.1 Funkciniai ... 14
3.3.2 Nefunkciniai ... 19
3.4 Architektūros tikslai ir apribojimai ... 22
3.4.1 Skaičiavimų bibliotekos architektūra ... 22
3.4.2 Detalizuota bibliotekos klasių diagrama ... 23
3.4.3 Grafinės sąsajos apribojimai ... 23
3.4.4 Skaičiavimų bibliotekos apribojimai ... 23
3.5 Positemių diagrama ... 24
3.6 Sistemos statinis vaizdas ... 25
3.6.1 Grafinė vartotojo sąsaja ... 25
3.6.2 Modeliavimo biblioteka ... 25
3.7 Pagrindinių sistemos objektų sąsajos aprašymas ... 26
3.7.1 Bendra programinė sąsaja visiems objektams ... 26
3.7.2 Koneksinas ... 26
3.7.3 Sritis ... 27

3.7.4 Koneksonas ... 27
3.7.5 Jungtis ... 27
3.8 Panaudojimo atvejų aprašymas ... 28
3.8.1 Panaudojimo atvejų diagrama ... 28
3.8.2 Preliminarus aktorių bei jų funkcijų sąrašas ... 28
3.8.3 Panaudojimo atvejų sąrašas ... 29
4 Tyrimo dalis ... 29
4.1 Fuzzy logic tyrimas ... 30
4.1.1 Koneksino laidžio priklausomybė nuo įtampos ... 31
4.1.2 Dviejų nuosekliai sujungtų koneksinų tyrimas ... 32
4.1.3 Markovo modelio ir fuzzy modelio palyginimas ... 33
4.2 Markovo modelis ir GPU ... 35
4.3 Kokybės įvertinimas ... 36
4.3.1 Sistemos testavimas ... 36
4.3.2 Reikalavimų išpildymas ... 37
4.3.3 Kokybės įvertinimo kriterijai ... 37
4.4 Sistemos patobulinimai ... 38
4.4.1 GPGPU panaudojimas ... 38
4.4.2 Vartotojo sąsajos patobulinima ... 40
5 Eksperimentinė dalis ... 41
5.1 Sistemos spartos tyrimas ... 41
5.1.1 Parametrų įtaka algoritmams ... 43
5.2 Parametrų įtaka algoritmų veikimo spartai ... 44
5.2.1 Stochastinis algoritmas ... 44
5.2.2 Markovo algoritmas ... 45
5.2.3 Fuzzy algoritmas ... 46
5.2.4 Koneksinų skaičius ... 46
5.3 Eksperimentinių rezultatų apibendrinimas ... 47
5.4 Markovo modelio rezultatų palyginimas ... 48
6 Išvados ... 49
7 Literatūra ... 50
8 Terminų ir santrumpų žodynas ... 52

Lentelės

Lentelė 3.1 – pradinių funkcinių reikalavimų sąrašas 14
Lentelė 3.2 – formulių parametrai ir jų aprašymai 16
Lentelė 3.3 – papildomi sistemos parametrai 17
Lentelė 3.4 – plyšinės jungties laidumo formulės parametrų aprašymas 18
Lentelė 3.5 – operacinių objektų tipai ir jungimas 25
Lentelė 3.6 – bendros programinės sąsajos aprašymas 26
Lentelė 3.7 – koneksino parametrų lentelė 26
Lentelė 4.1 – koneksino laidžio formulės parametrai 30
Lentelė 4.2 – grafikų ir koneksinų parametrų duomenys 32
Lentelė 4.3 – markovo ir fuzzy modelių parametrai prie kurių laidžio grafikai sutampa 33
Lentelė 4.4 – kokybės įvertinimo kriterijai 37
Lentelė 5.1 – ribiniai programos parametrai naudoti spartos tyrimui 41
Lentelė 5.2 – stochastinio, markovo ir fuzzy modelių generavimo laikas 42
Lentelė 5.3 – teorinė parametrų įtaka algoritmams 43
Lentelė 5.4 – eksperimentiškai nustatytas algoritmų sudėtingumas 47
Lentelė 5.5 – modelių privalumų ir trūkumų apibendrinimas 47

Paveikslai/Diagramos

Pav. 2.1 – plyšinė jungtis 10
Pav. 3.1 – dviejų būsenų tolygaus laiko markovo modelio koneksino schema 15
Pav. 3.2 – dviejų būsenų diskretaus laiko markovo modelio koneksino schema 15
Pav. 3.3 – koneksinų skaičiaus įvestis 17
Pav. 3.4 – koneksonų skaičiaus įvestis 17
Pav. 3.5 – iteracijų skaičiaus įvestis 17
Pav. 3.6 – Modelio tipo pasirinkimas 17
Pav. 3.7 – Įtampos rėžių įvestis 17
Pav. 3.8 – tiesinis įtampos kitimo protokolas 17
Pav. 3.9 – Laikrodis MAC OS 19
Pav. 3.10 – Laikrodis Linux SUSE 19
Pav. 3.11 – Laikrodis Windows 7 OS 19
Pav. 3.12 – Grafiko valdymas 19
Pav. 3.13 – Jungties parametrų valdymas 19
Pav. 3.14 – sričių poliškumo valdymas 19
Pav. 3.15 – Veiksmų iniciavimo mygtukai 19
Pav. 3.16 – Spin box (Žr. sk. 8 - 1) 20
Pav. 3.17 – Dockable widget (Žr. sk. 8 - 2) 20
Pav. 3.18 – bibliotekos objektų struktūros diagrama 22
Pav. 3.19 – bibliotekos pagrindinių objektų dalinė klasių diagrama 22
Pav. 3.20 – detalizuota klasių diagrama 23
Pav. 3.21 – posistemių diagrama 24
Pav. 3.22 – statinis sistemos vaizdas 25
Pav. 3.23 – koneksino elektrinė schema 25
Pav. 3.24 – koneksono uždara būsena 26
Pav. 3.25 – Koneksino, koneksono, bei srities grafinis pavaizdavimas 26
Pav. 3.26 – koneksono atvira būsena 26
Pav. 3.27 – panaudojimo atvejų diagrama 28
Pav. 4.1 – realizuotos programinės įrangos vaizdas 29
Pav. 4.2 – koneksino laidžio priklausomybė nuo įtampos 31
Pav. 4.3 – dviejų nuosekliai sujungtų koneksinų schema 32
Pav. 4.4 – dviejų nuosekliai sujungtų koneksinų su priešingu poiškumu grafikas 32
Pav. 4.5 – Markovo ir Fuzzy modelių palyginimas 33
Pav. 4.6 – skirtumas tarp markov ir fuzzy modelių 34
Pav. 4.7 – markov ir fuzzy modelių grafikai 34
Pav. 4.8 – GPU skaičiavimų netikslumai 35
Pav. 4.9 – GPU lygčių sprendimo bibliotekos architektūra 38
Pav. 4.10 – lygčių sprendimo pagreitėjimas 39
Pav. 4.11 – GPU tiesinių lygčių sprendimo laiko grafikas 39
Pav. 5.1 – stochastinio, markovo ir fuzzy modelių grafikai su ribinėmis reikšmėmis 41
Pav. 5.2 – GPU prieš CPU pagreitėjimas [16..64] nežinomųjų ribose 42
Pav. 5.3 – koneksinų skaičiaus įtaka stochastinio algoritmo vykdymo laikui 44
Pav. 5.4 – koneksonų skaičiaus įtaka stochastinio algoritmo vykdymo laikui 44

Pav. 5.5 – iteracijų skaičiaus įtaka stochastinio algoritmo vykdymo laikui 45
Pav. 5.6 – koneksinų skaičiaus įtakos grafikas markovo algoritmo vykdymo laikui 45
Pav. 5.7 – koneksinų skaičiaus įtaka fuzzy algoritmo vykdymo laikui 46
Pav. 5.8 – markovo modelio grafinis rezultatų palyginimas 48

Formulės

Formulė 1 – apskaičiuoti koeficientui k 15
Formulė 2 – apskaičiuoti perėjimo tikimybei iš atviros būsenos į uždarą 15
Formulė 3 – apskaičiuoti perėjimo tikimybei iš uždaros būsenos į atvirą..... 15
Formulė 4 – bendra formulė gauti perėjimo intensyvumui iš tikimybės..... 15
Formulė 5 – apskaičiuoti atviros būsenos koneksino laidį..... 15
Formulė 6 – apskaičiuoti uždaros būsenos koneksino laidį 16
Formulė 7 – skirta apskaičiuoti plyšinės jungties laidumą prie nurodytos įtampos..... 18
Formulė 8 – apskaičiuoti koneksino laidžiui prie nurodytos įtampos..... 30
Formulė 9 – konkretizuota formulė dviejų būsenų koneksino laidžiui apskaičiuot..... 30

Santrauka

Šiame darbe tiriamos markovo modelio vykdymo charakteristikos. Pagrindinis tikslas yra įrodyti arba paneigti, jog tolydaus laiko markovo modelis [1] yra geresnis[1] metodas modeliuoti ląstelių plyšinių jungčių [2] laidžio priklausomybės nuo įtampos grafikus. Markovo modelis ir jo charakteristikos lyginamos su kitais šiame darbe sugalvotais metodais, bei jau esančiais duomenimis iš kitų autorių darbų [3] [4] [5]. Sukurta programinė įranga modeliuoti betkokios konfigūracijos dviejų tipų modelių plyšinėms jungtims. Taip pat vienas iš tikslų kuriant programinę įrangą buvo išvengti PLA modeliavimo. Tam panaudotas modeliuojamo objekto suskaidymas į funkcinius objektus, kurie gali būti pilnai aprašyti naudojant objektinio programavimo metodiką. Sukurti modeliai buvo eksperimentiškai patikrinti ir nuodugniai ištirti.

Pirmame skyriuje apžvelgsime kas yra plyšinė jungtis, kaip ji veikia ir kodėl sudaryti jos kompiuterinį modelį yra aktualu. Antrasis skyrius skirtas įvairios literatūros analizei susijusiai su plyšinės jungties modeliavimu, bei jos charakteristikų aprašymu. Taip pat literatūroje apžvelgsime keletą modelių kurie naudojami tirti plyšinėms jungtims. Trečiasis skyrius skirtas detalizuoti sistemos architektūrai, pabrėžti pagrindinius ir svarbiausius aspektus, kurie liečia modelių generavimą, bei plyšinės jungties veikimo analizę. Paskutiniuose skyriuose pabandysime nustatyti koks modelis ir kodėl yra efektyviausias modeliuojant plyšinės jungties laidumo grafiką.

[1] Pagrindinis vertinimo kriterijus yra vykdymo greitis

Summary

The major goal of this study was to prove that continuous time Markov models of voltage gating of gap junction (GJ) channels is faster and better way to make digital models of connexin protein. This goal was achieved by using created software that models connexin gating voltage graphs in real time. Created software is capable of modeling two types of voltage graphs: stochastic and deterministic. To avoid using piece linear aggregates (PLA) formalism, these models were implemented strictly on object oriented programming paradigm by dividing system into object that can be fully described by classes in any programming language. Developed Markov models were used to investigate the evolution of gap junctional conductance depending on trans-junctional voltage.

The first section of this study is dedicated for describing cell gap junction, how it is functioning and why having it's digital model is important for nowadays bio-science. In the second section a lot of different literature are overviewed related to cell gap junctions modeling and it's charcteristics investigation. Some concrete models in the literature sections are outlined as well. The third sections is for describing the process of system architecture development and realization of some important aspects related with generating digital CGJ models. The last sections is for created software investigation and finding the best way to emulate CGJ behaviour. By analysing some experimental models trend graph it is going to be proved what and why some particular model is better than others for doing this task.

1 Įvadas

Šiame darbe bus yra ištirtos markovo, fuzzy, bei stochastinio modelio vykdymo charakteristikos. Sukurta bei aprašyta programinė įranga modeliuoti įvairių konfigūracijų plyšinėms jungtims. Pagrindinis tikslas yra įrodyti arba paneigti, jog tolydaus laiko markovo modelis [1] yra geresnis[2] metodas modeliuoti ląstelių plyšinių jungčių [2] laidžio priklausomybės nuo įtampos grafikus.

1.1 Dokumento paskirtis

Šiame darbe atliekama trijų tipų modelių analizė ir jų efektyvumas modeliuojant plyšinės jungties laidžio kitimo charakteristikas. Taip pat aprašoma autoriaus[3] sukurta programinė įranga skirta modeliuoti plyšinės jungties charakteristikoms naudojant markovo, stochastinį, bei fuzzy[4] modelius. Kiekvieno šių modelių charakteristikos yra nuodugniai išanalizuojamos, kad būtų galima aiškiai nustatyti kuris iš jų yra efektyviausias atlikti šią užduot. Nepriklausomai nuo tyrimo rezultatų, programinės įrangos galutinė versija realizuoti dviejų tipų modeliai: stochastinis ir markovo.

1.2 Aktualumas

Plyšinių jungčių modeliavimas vykdomos įvairiomis matematinėmis priemonėmis, naudojant sudėtingus matematinius modelius, kurie tiksliai imituoja plyšinių jungčių veiklą. Deja, šiuos modelius pritaikyti praktikoje yra sudėtinga, dėl didelių resursų sąnaudų, todėl tai verčia mokslininkus griebtis paprastesnių, efektyvesnių ir greitesnių priemonių, paieškos, šių gyvybiškai svarbių jungčių, kompiuteriniam modeliavimui atlikti.

Vienas iš įrankių galinčių paspartinti jungčių modeliavimą, tai markovo modelis, kuris leistų padidinti skaičiavimų spartą tūkstančius kartų. Todėl šiame darbe bus skiriamas dėmesys būtent jam.

1.3 Darbo tikslas

Šio darbo tikslas yra pagrįsti arba paneigti teiginį, jog naudojant markovo modelį galima efektyviau imituoti plyšinės jungties veikimą, negu naudojant kitus šiame darbe minimus modelius.

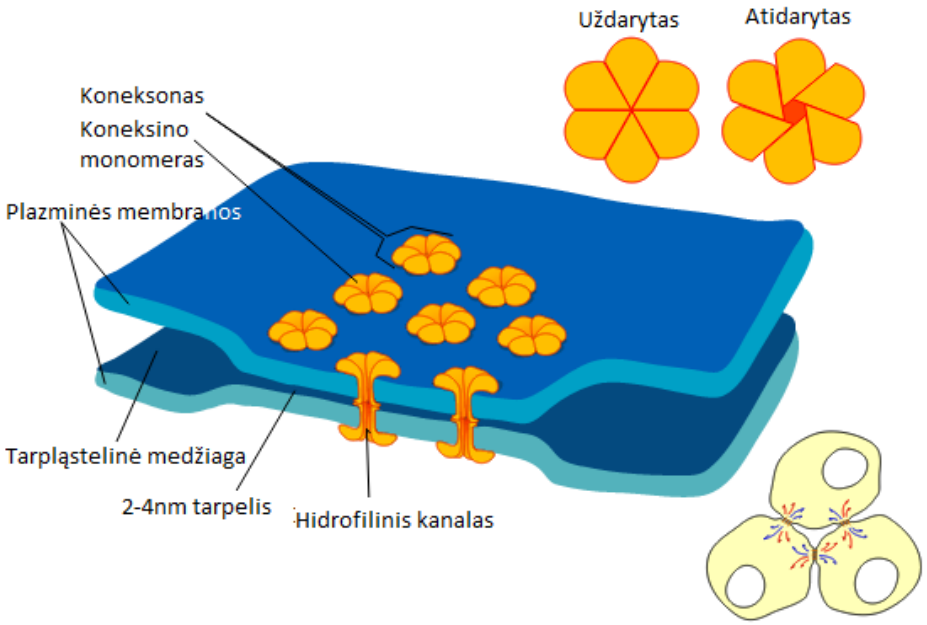
[2] Pagrindinis vertinimo kriterijus yra vykdymo greitis
[3] Edvinas Bazaras
[4] Atsitiktinai sugalvotas modelis bekuriant programinę įrangą

2 Literatūros apžvalga ir techninių sprendimų analizė

2.1 Plyšinė jungtis

Plyšinė jungtis arba dar kitaip vadinama „kontaktu“ (Iš lotynų kalbos.: nexus), tai tarpląstelinis kanalas tarp įvairių rūšių ląstelių, žmogaus ar gyvūno organizme. Šis kanalas tiesiogiai sujungia dviejų ląstelių citoplazmą, kuris leidžia laisvai vykti įvairių molekulių ir jonų kaitai.

Vienas plyšinės jungties kanalas susideda iš koneksonų (arba puskanalių) kurie sudaro jungtį per tarpląstelinę medžiagą. Plyšinės jungtys yra augalų ląstelių plazmodezmos analogas. Pastebėtina, kad plyšinių jungčių yra elektrinio laidžio tipo sinapsėse esančiose kai kuriose neuronuose. [2]



Pav. 2.1 – plyšinė jungtis

2.2 Analizė

Daugelyje audinių jonų ir mažų molekulių pasikeitimą tarp gretimų ląstelių valdo plyšinės jungtys (lot. Nexus (Žr. Sk. 8 - 14)) [6], iš kurių vienos gerai žinomos jungtys tarp neuronų – sinapsės [7] [8], tokiu būdu šios jungtys koordinuoja ląstelių veiklą. PJ (plyšinės jungtys) tiekia išteklius (jonus, mažas molekules), kad valdytų ląstelių veiklą audiniuose. Dėl to PJ dalyvauja daugelyje biologinių procesų, tokių kaip:

- vystymasis;
- raumenų susitraukimas;
- augimas;
- sekrecija (išskyrimas);
- impulsyvus dauginimasis.

Ląstelės turi kelių rūšių jungtis glaudžiąsias, desmosomas, sąaugas, bei šiame darbe nagrinėjamas plyšines jungtis [9]. Plyšinės jungtys [7] yra vienas iš sudedamųjų odontoplastų [6] sluoksnio elementų, kuris skiria dentiną ir danties minkštimą. Šio sluoksnio vientisumas ir ribotas pralaidumas palaikomas būtent šių jungčių sistemos dėka. Ruožuotojo širdies raumens ląstelių – kardiomicitų [6] įterptiniuose diskuose taip pat yra plyšinių jungčių, kurios garantuoja jonų pernašą iš vieno kardiomicito į kitą.

Struktūriniu požiūriu PJ (Žr. Sk. 8 - 6) yra tarpląstelinių kanalų (plyšinių jungčių kanalų) agregatai. Dvejopo įtampos-varžos metodo taikymas ląstelių porų mėginiuose leidžia išsiaiškinti elektrines plyšinių jungčių bei jų kanalų savybes [10] [11] [12]. Laidžio ir kinetiniai duomenys, gauti daugiakanaliame ir vienkanaliame lygmenyje, nuvedė prie apibendrintos plyšinių jungčių kanalų veikimo koncepcijos. Šiame darbe tiriamas tarpląstelinių plyšinių jungčių veikimas, bei siekiama sudaryti kompiuterinį MM (markovo modelį (Žr. Sk. 8 - 5)) šių jungčių efektyviam modeliavimui. Sakalauskaitės atliktame magistriniame darbe [4] buvo ištirti 6 skirtingi modeliai: 4 diskretaus laiko Markovo grandinių (2 būsenų 6 koneksinų, 12 koneksinų ir 3 būsenų 6 koneksinų, 12 koneksinų) ir 2 tolydaus laiko Markovo grandinių (2 būsenų 6 koneksinų, 12 koneksinų). Sakalauskaitės darbe pateikiama ląstelių PJ MM sudarymo metodika, apimanti perėjimo tikimybių skaičiavimą panaudojant nepriklausomų J. Bernulio bandymų schemą, stacionariųjų tikimybių skaičiavimą ir plyšinės jungties laidumo priklausomybės nuo įtampos skaičiavimus.

Tariama, kad plyšinė jungtis sudaryta iš daugybės lygiagrečiai sujungtų kanalų (pvz., 1000) [11]. Kiekvienas kanalas sudarytas iš 2 nuosekliai sujungtų puskanalių (koneksonų), o kiekvienas koneksonas sudarytas iš 6 lygiagrečiai sujungtų vienetų (koneksinų). Kiekvienas koneksinas gali būti atviroje arba uždaroje būsenoje, kuri priklauso nuo kanalo įtampos [10] [13]. Modelių, sukurtų naudojant šią metodiką, adekvatumas patikrintas lyginant plyšinės jungties modeliavimo rezultatus su imitacinio modeliavimo (programų, kurias atliko Nerijus Paulauskas ir Saulius Vaičeliūnas (KTU Informatikos fakulteto magistrantai)) rezultatais, kurie patikrinti su eksperimentų rezultatais. Sukurta Markovo modelių metodika panaudota kuriant plyšinės jungties modelius, kai koneksinai aprašomi 3 būsenomis: uždara, atvira ir visiškai uždara. [4] [14] [15]

2.3 Alternatyvios sistemos

Sistemų, kurios skirtos generuoti ląstelių plyšinių jungčių markovo modelius nepavyko rasti. Dauguma jungčių modeliavimų atliekama sudėtingesnių matematinių modelių pagalba naudojant matlab (Žr. Sk. 8 - 18) programą. Vienas iš pavyzdžių LR[5] modelis [16], kuris naudojamas dinaminių jungčių modeliavimui specialiomis matematinėmis išraiškomis . Taip pat bandoma šių jungčių tyrimus atlikti 3D modeliavimo priemonių pagalba (Connexin laboratory in CVRTI[6] at the Univ. of Utah). Kiekvienas iš šių modeliavimo būdų turi savų pliusų bei minusų, iš kurių didžiausias, tai milžiniškų skaičiavimo resursų poreikis, norint modeliuoti didelius ląstelių tinklus.

Nors yra nemažai skirtingų šių jungčių modeliavimo realizacijų, tačiau jų tikslai yra panašūs. Skiriasi tik realizacijos būdai ir metodikos. Pagrindiniai jungčių modeliavimo tikslai:

- Kuo tiksliau atkartoti realių jungčių veikimą
- Panaudoti kuo mažiau skaičiavimų
- Ir tai atlikti kiek įmanoma greičiau

Tokios programinės įrangos naudojimas yra kritinis taškas, kuriant bet kokią, didesnės apimties ląstelių tinklo kompiuterinį modelį, nes tai žymiai paspartintų sistemos darbo spartą ir galbūt net gi leistų atlikti realaus laiko sistemos modeliavimą.

Dažniausiai tokios sistemos naudoja PLA aprašymą. Tam tikslui modeliuojamą objektą reikia aprašyti šia kalba, o programinė įranga skirta interpretuoti PLA aprašui atliks visą darbą. Tačiau tai nėra efektyvu, nes PLA modelio interpretavimas reikalauja daug kompiuterinių resursų ir niekaip nepavyktų sukurti sparčiai veikiančios sistemos. Todėl šiame darbe ir naudojamas paprastesnis markovo modelis, kuris iš esmės yra PLA poaibis. Kitaip tariant markovo procesas yra atskiras PLA atvejis. Turint markovo modelį galima gana efektyviai realizuoti kompiuterinį šio modelio interpretatorių, nes iš esmės tereikia išspręsti tiesinių lygčių sistemą, norint gauti rezultatus.

[5] Luo and Rudy
[6] Žiūrėti paaiškinimą santrumpų žodyne, Nr. 19.

3 Projektinė dalis

3.1 Sistemos paskirtis

Sistema skirta modeliuoti ląstelių plyšinių jungčių laidžio priklausomybės nuo įtampos grafiką. Naudojant markovo ir stochastinį modelį.

3.2 Sistemos tikslai

Padėti ištirti kuris modelis padeda efektyviau imituoti plyšinės jungties veikimą.

3.3 Reikalavimai sistemai

3.3.1 Funkciniai

Šioje dalyje pateikiamas funkcinių reikalavimų sąrašas. Pirmąjame sistemos projektavimo etape buvo pateikti funkciniai reikalavimai aprašyti (Lentelė 3.1).

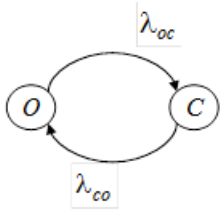
Lentelė 3.1 – pradinių funkcinių reikalavimų sąrašas

Panaudos atvejis(-ai)	**Reikalavimas**	**Pagrindimas**
1. Eksperimentinių duomenų įvedimas 2. Markovo modelio parametrų nustatymas 3. Modelio tipo parinkimas	Duomenų įvestis	Reikia įvesti kažkokius duomenis norint gauti rezultatą
1. Įvestų duomenų validavimas	Validavimas	Įvesties teisingumo patikrinimas.
1. MM palyginimas su eksperimentiniais rezultatais 2. MM validavimas	Testavimas	Reikia patikrinti ar modelis atitinka realybę.
1. MM eksportavimas į išorinę sistemą	Markovo modelio eksportavimas	Kadangi šis modelis gali būti panaudotas kitoje sistemoje reikėtų jį eksportuoti tam tikru formatu.
1. Klaidų registro peržiūra	Klaidų peržiūra	Reikia matyti kokios įvyko klaidos ir informaciją apie jas.
1. MM optimizavimas	Markovo modelio optimizavimas	Reikia optimizuoti markovo modelį, kad jis kuo greičiau veiktų.
1. Klaidų registravimas	Fiksuoti klaidas	Reikia žinoti kas ir dėl ko sugriuvo.
1. Modelio generavimas	Sugeneruoti markovo modelį	Reikia turėti sugeneruotą modelį norint atlikti tolimesnius veiksmus.

Šie reikalavimai yra pernelyg abstraktūs, kad būtų galimą sukurti sistemą šiai konkrečiai užduočiai. Todėl sekančiame etape buvo atsisakyta perteklinių reikalavimų, kurie nenaudingi sistemos funkcionalumui ir tolimesniems tyrimams. Realizuoti ir pataisyti sistemos reikalavimai pateikiami sekančiuose poskyriuose.

3.3.1.1 *Parametrų įvedimas*

Norint sugeneruoti markovo modelį sistemai reikia pateikti parametrus apie PJ[7]. Pagrindiniai parametrai siejami su mažiausia plyšinės jungties dalimi koneksinu (Pav. 3.25 – Koneksino, koneksono, bei srities grafinis pavaizdavimas) [3]. Kadangi šiame markovo modelyje koneksinas turi dvi būsenas (Pav. 3.1), atvirą bei uždarą jų laidžiai bei būsenos kaitos intensyvumas apskaičiuojami naudojant žemiau pateiktas formules.



Pav. 3.1 – dviejų būsenų tolygaus laiko markovo modelio koneksino schema

Formulė 1 – apskaičiuoti koeficientui k

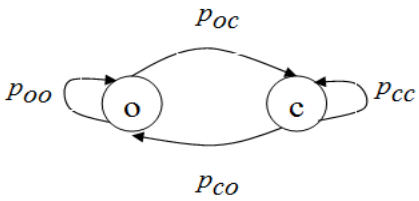
$$k = e^{(A(PV_{koneksino} - V_0))}$$

Formulė 2 – apskaičiuoti perėjimo tikimybei iš atviros būsenos į uždarą

$$p_{oc} = \frac{Kk}{1+k}$$

Formulė 3 – apskaičiuoti perėjimo tikimybei iš uždaros būsenos į atvirą

$$p_{co} = \frac{K}{1+k}$$



Pav. 3.2 – dviejų būsenų diskretaus laiko markovo modelio koneksino schema

Kadangi ši sistema skirta modeliuoti tolygaus laiko markovo modeliui, kitaip tariant MP[8], turime gauti perėjimo intensyvumus apibrėžti būsenoms iš (Pav. 3.1).

Formulė 4 – bendra formulė gauti perėjimo intensyvumui iš tikimybės

$$\lambda = \frac{p}{t}$$

Perėjimo intensyvumas gaunamas tikimybę padalinus iš laiko, pvz.: $t = 1ms$.

Tai reikštų, kad koneksinas būseną $X\{O, C\}$ laiko intervale t pakeičia p kartų į būseną $\bar{X}$.

Formulė 5 – apskaičiuoti atviros būsenos koneksino laidį

$$g_o = 2e^{\frac{PV_{koneksino}}{R_o}}$$

[7] Plyšinė jungtis
[8] Markovo procesas – žiūrėti paaiškinimą terminų ir santrumpų žodyne

Formulė 6 – apskaičiuoti uždaros būsenos koneksino laidį

$$g_c = 0.5e^{\frac{PV_{koneksino}}{R_c}}$$

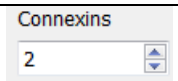
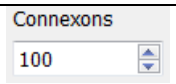
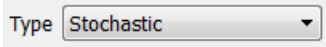
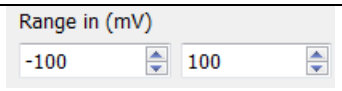
Lentelė 3.2 – formulių parametrai ir jų aprašymai

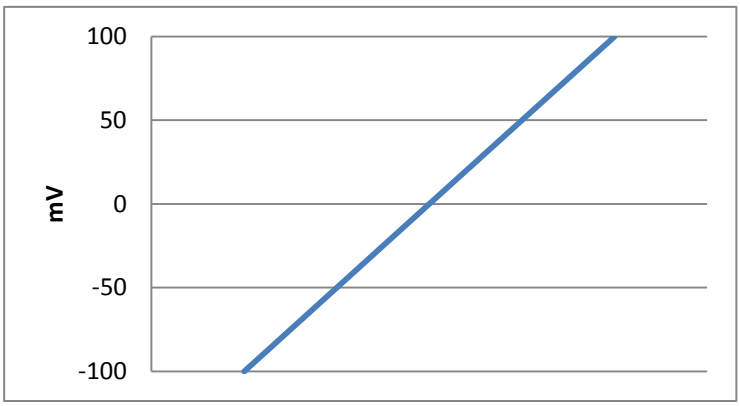
Parametras	Formulė(-s)	Iš kur gaunama	Aprašymas
k	Formulė 2 Formulė 3	Formulė 1	Koeficientas naudojamas perėjimo tikimybės skaičiavimui.
A	Formulė 1	Įvedama vartotojo	Koeficientas keičiantis perėjimo tikimybės kitimo dėsningumą.
V_0	Formulė 1	Įvedama vartotojo	Įtampa prie kurios perėjimo tikimybės į kiekvieną būseną yra vienodos.
$V_{koneksino}$	Formulė 1 Formulė 5 Formulė 6	Apskaičiuojama naudojant omo dėsnį, vartotojui įvedus visos grandinės įtampą	Įtampa krentanti ant konkretaus koneksino, bei priklausanti nuo grandinės topologijos[9].
P	Formulė 1 Formulė 5 Formulė 6	Įvedama vartotojo	Koneksino poliškumas, parametras gali turėti dvi reikšmes +1 arba -1. Jis keičia konkretaus koneksino polinį laidumą, srovės judėjimo kryptį.
K	Formulė 2 Formulė 3	Įvedama vartotojo	Koeficientas keičiantis tikimybės ribas, standartinė reikšmė $K = 1$ kadangi tikimybės intervalas paprastai būna: [0 ... 1]
p_{oc}	Formulė 4	Formulė 2	Perėjimo tikimybė iš atviros būsenos į uždarą
p_{co}	Formulė 4	Formulė 3	Perėjimo tikimybė iš uždaros būsenos į atvirą
t	Formulė 4	Įvedama vartotojo[10]	Laiko konstanta žyminti kokiame laiko intervale įvyksta perėjimų p_x
g_o		Formulė 5	Atviros būsenos koneksino laidis
g_c		Formulė 6	Uždaros būsenos koneksino laidis
R_c	Formulė 6	Įvedama vartotojo	Parametras apibrėžiantis koneksino laidį uždaroje būsenoje
R_o	Formulė 5	Įvedama vartotojo	Parametras apibrėžiantis koneksino laidį atviroje būsenoje

[9] Grandinės schemos struktūra, funkcinių dalių išsidėstymas/išdėstymas
[10] Pastaba: šioje programoje laikas t yra konstanta lygi vienetui

Iki šiol aptarėme parametrus tiesiogiai susijusius su koneksinu ir jo būsenomis, bet norint atlikti skaičiavimus reikia papildomų duomenų apie įtampą, rėžius, koneksinų, bei koneksonų kiekį ir konfigūraciją. Žemiau pateikiama papildomų parametrų lentelė.

Lentelė 3.3 – papildomi sistemos parametrai

Parametras	**Aprašymas**	**Grafinis vaizdas**
Koneksinų skaičius	Apibrėžia koks bus koneksinų skaičius puskanalyje.	 **Pav. 3.3 – koneksinų skaičiaus įvestis**
Koneksonų skaičius	Apibrėžia koks bus koneksonų skaičius plyšinėje jungtyje.	 **Pav. 3.4 – koneksonų skaičiaus įvestis**
Iteracijos	Skirtas stochastinio modelio nusistovjusio laidumo radimui.	 **Pav. 3.5 – iteracijų skaičiaus įvestis**
Modelio tipas	Parenka kokį modelį naudojant bus generuojami rezultatai. Galimi du modelių pasirinkimai: **stochastinis**[11] ir **markovo**.	 **Pav. 3.6 – Modelio tipo pasirinkimas**
Rėžiai	Įtampos rėžiai apibrėžia įtampos protokolo tipą (Pav. 3.8), šioje programoje jis yra tiesinis ir jo negalima keisti. Šie rėžiai tik apibrėžia įtampos kitimo ribas.	 **Pav. 3.7 – Įtampos rėžių įvestis**



Pav. 3.8 – tiesinis įtampos kitimo protokolas

[11] tikimybinis

3.3.1.2 *Grafinis duomenų atvaizdavimas*

Programa turi atvaizduoti laidžio priklausomybės nuo įtampos grafiką. Skaičiavimo rezultatai turi būti pateikiami grafiniu pavidalu. Jame turi būti atvaizduojama funkcija (Formulė 7).

Formulė 7 – skirta apskaičiuoti plyšinės jungties laidumą prie nurodytos įtampos

$$g_j(V_j) = \sum_{i}^{n} p(V_j)_i g(V_j)_i$$

Lentelė 3.4 – plyšinės jungties laidumo formulės parametrų aprašymas

Parametras	**Aprašymas**
V_j	Vartotojo įvedama įtampa tarp jungties galų
$p(V_j)_i$	Būsenos i stacionari tikimybė
$g(V_j)_i$	Būsenos i laidis
n	Būsenų skaičius

3.3.1.3 *Skaičiavimų nutraukimas*

Vartotojas turi galėti nutraukti skaičiavimus. Kadangi įvedus tam tikrus parametrus skaičiavimai gali ilgai užtrukti, turi būti galimybė juos nutraukt neišjungiant programos.

3.3.1.4 *Rezultatų išvalymas*

Vartotojas turi galėti išvalyti rezultatus. Pašalinti jau nubrėžtus grafikus.

3.3.1.5 *Rezultatų išvedimas*

Turi būti galimybė išvesti rezultatus į išorinę sistemą tolimesniems tyrimams.

3.3.1.6 *Markovo modelio generavimas*

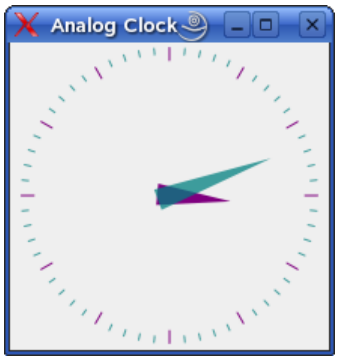
Sistema turi sugeneruoti markovo modelį ir naudojant jį apskaičiuoti laidžio priklausomybę nuo įtampos (Formulė 7).

3.3.2 Nefunkciniai

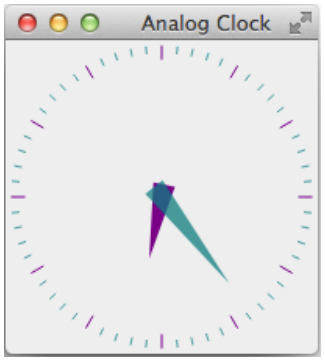
3.3.2.1 Reikalavimai išvaizdai

Konkrečių reikalavimų sistemos išvaizdai nėra. Sistemos išvaizda priklauso nuo to kurioje operacinėje sistemoje ji bus naudojama. Žemiau pateikiama Qt demonstracinės programos grafinės sąsajos pavyzdys keliose populiariausiose operacinėse sistemose:

Pav. 3.11 – Laikrodis Windows 7 OS



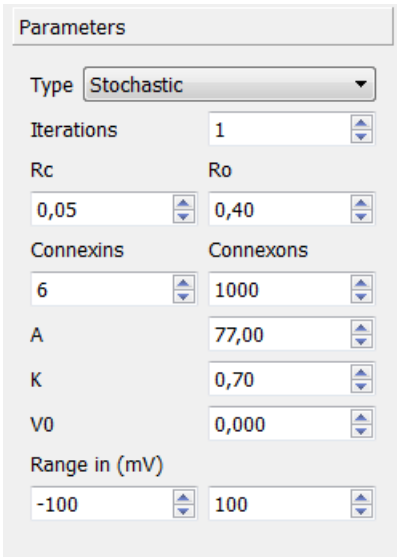
Pav. 3.10 – Laikrodis Linux SUSE



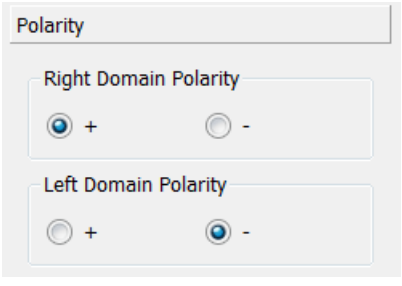
Pav. 3.9 – Laikrodis MAC OS

Programos elementų išdėstymui taip pat nenustatyta jokių reikalavimų, taigi jie buvo sugrupuoti pagal funkcines grupes. Jos buvo išskirtos keturios:

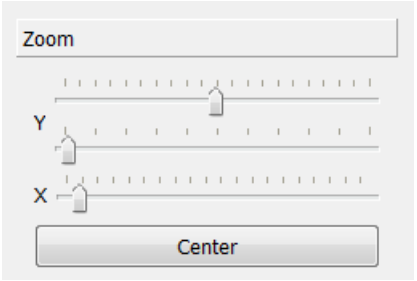
- Grafiko valdymas (Pav. 3.12)
- Poliškumo valdymas (Pav. 3.14)
- Jungties parametrai (Pav. 3.13)
- Veiksmų iniciavimas (Pav. 3.15)



Pav. 3.13 – Jungties parametrų valdymas



Pav. 3.14 – sričių poliškumo valdymas



Pav. 3.12 – Grafiko valdymas

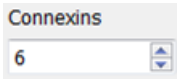
Pav. 3.15 – Veiksmų iniciavimo mygtukai

Galima išskirti bendrus reikalavimus vartotojo sąsajai atsižvelgiant į vidutinio vartotojo poreikius:

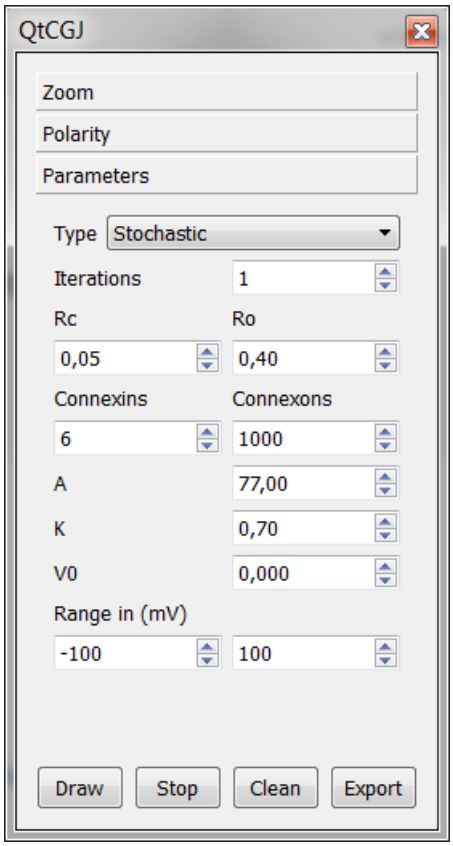
- lengvai skaitoma sąsaja;
- paprastas (nesudėtingas) panaudojimas;
- prieinamumas, kad vartotojas nesivaržytų naudodamas sistemą;
- neįkyri sąsaja (pavyzdžiui, nereikalaujanti pastoviai ką nors kelis kartus patvirtinti);
- novatoriška ir meniška išvaizda;

3.3.2.2 Reikalavimai panaudojamumui

Pagrindinės sistemos funkcijos turi būti aiškiai suprantamos be papildomos instrukcijos. Funkcionalumas neturi būti dubliuojamas ir turėtų būti vengiama funkcijų kurios nebus naudojamos. Turi būti galimybė išdidinti grafiką per visą programos langą. Taip pat leisti vartotojui nutraukti grafiko generavimą, kadangi tam tikrais atvejais tai gali užtrukti gan ilgai. Vartotojas turi matyti grafiko generavimo progresą. Parametrų įvestis turi būti apribojama rėžiais kurie nustatyti eksperimentiškai. Privaloma įvesties validacija. Skaitinės parametrų reikšmės turi būti redaguojamos klaviatūros bei pelės pagalba (Pav. 3.16). Palikti kiek įmanoma daugiau erdvės grafikui panaudojant „prišvartuojamą[12]“ valdymo skydelį (Pav. 3.17).



Pav. 3.16 – Spin box (Žr. sk. 8 - 1)



Pav. 3.17 – Dockable widget (Žr. sk. 8 - 2)

[12] Ištraukiamas, perkeliamas, keičiantis poziciją

3.3.2.3 Reikalavimai vykdymo charakteristikoms

Sistema turi kiek įmanoma greičiau įvykdyti markovo modelio generavimo algoritmą. Vykdant skaičiavimus vartotojo sąsaja neturėtų pakibti[13].

3.3.2.4 Reikalavimai veikimo sąlygoms

Sistema turi veikti įvairiose aplinkose[14], įskaitant ir pagrindinių architektūrų procesorius (t.y. x64 ir x86) tam tikslui pasiekti naudojama C++ kalba, bei vartotojo sąsajai Qt (Žr. Sk. 8 - 3) Framework[15]. Vartotojas turėtų žinoti pagrindines koneksinų bei koneksonų parametrų reikšmes.

3.3.2.5 Kultūriniai-politiniai reikalavimai

Kuriama sistema gali būti tik anglų kalba, nebūtinas daugiakalbiškumo palaikymas. Sistemos dokumentacija turėtų būti lietuvių kalba.

3.3.2.6 Teisiniai reikalavimai

Sistema turi būti kuriama nemokamais atviro kodo įrankiais, pati sistema taip pat turi būti nemokama. Sistema turi būti saugoma pagal GPL (Žr. Sk. 8 - 4) licenciją, kūrimo procesas turi atitikti visas GPL direktyvas[16].

3.3.2.7 Perspektyviniai reikalavimai

Panaudoti LU (Žr. Sk. 8 - 20) išskaidymą tiesinių lygčių sprendimo bibliotekoje, kuris pagreitintų algoritmą iki 3 kartų.

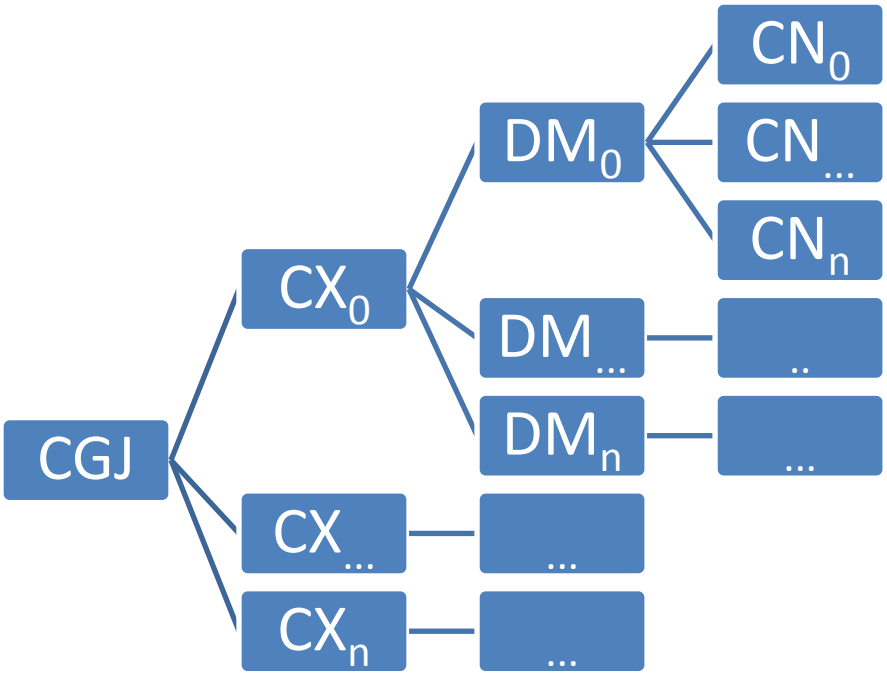
[13] Užstrigti, neduoti jokio atsako į vartotojo veiksmus
[14] Operacinėse sistemose
[15] Karkasas (Žr. Sk. 8 - 12)
[16] Nurodymai – kas kaip ir kodėl turi būti daroma

3.4 Architektūros tikslai ir apribojimai

Sistemos architektūros tikslas, pasiekti kiek įmanoma geresnį pakartotinio panaudojamumo lygį, bei suteikti programuotojui galimybę nesunkiai ją praplėsti papildomomis funkcijomis, bei naujais sudėtingesniais modeliais. Pagrindinis dėmesys turi būti skiriamas tyrimui, o ne naudojimo patogumui. Todėl grafinė sąsaja, bei biblioteka turi kelis svarbius apribojimus.

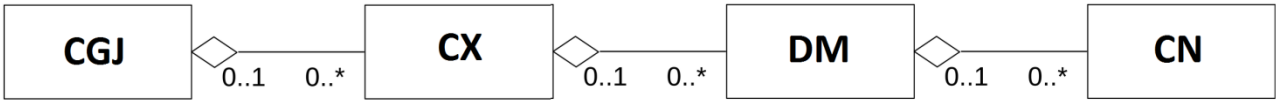
3.4.1 Skaičiavimų bibliotekos architektūra

Pasirinkta bibliotekos architektūra yra neįtikėtinai lanksi, kadangi galima sukurti betkokios konfigūracijos jungties modelį. Keturios pagrindinės klasės susietos agregaciniais [17]ryšiais.



Pav. 3.18 – bibliotekos objektų struktūros diagrama

Pavaizduotoje diagramoje galime matyti, kad plyšinė jungtis gali susidėti iš N koneksonų, koneksonas gali būti sudarytas iš N sričių, o srityje gali būti N koneksinų. Tą patį galime pavaizduoti klasių diagramoje, kaip agregacinį ryšį.

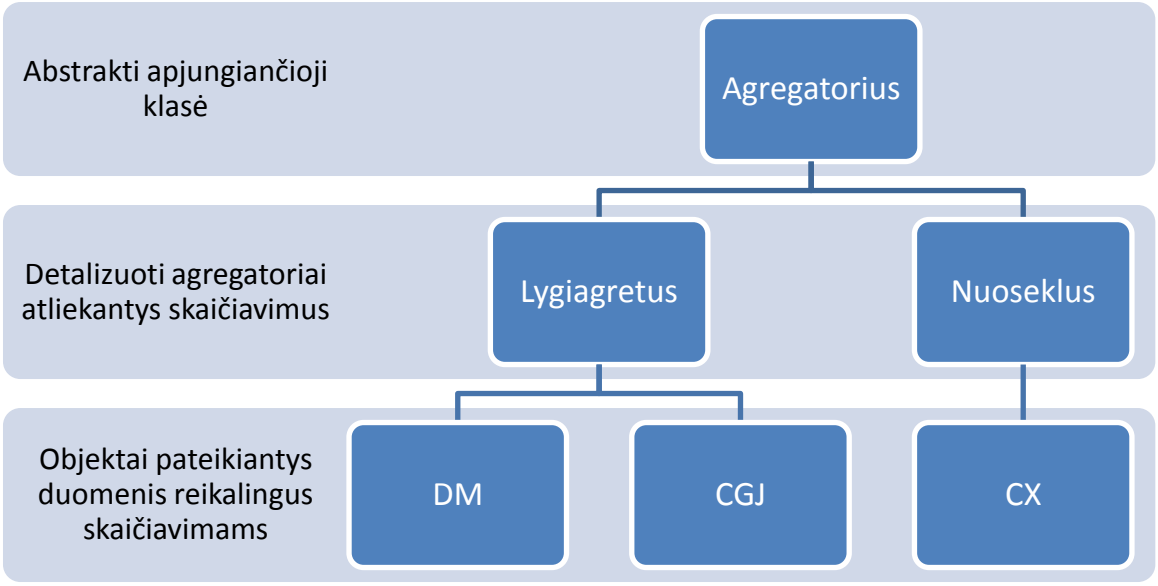


Pav. 3.19 – bibliotekos pagrindinių objektų dalinė klasių diagrama

[17] Priklausomybė, įtraukimas

3.4.2 Detalizuota bibliotekos klasių diagrama

Iš dalinės diagramos nėra visiškai aišku, kaip objektai yra sujungiami tarpusavyje, šioje diagramoje pateikiamos trys papildomos klasės, leidžiančios apjungti objektus dviejų tipų ryšiais[18]. Jungimo tipai šiuo atveju gali būti **lygiagretus**, bei **nuoseklus**.



Pav. 3.20 – detalizuota klasių diagrama

Taigi turime tris objektų sluoksnius, kuriems paskirtos konkrečios užduotis bendros paveldinčiosioms klasėms. Pirmas sluoksnis atlieka objektų talpinimo funkciją, antrasis atlieka skaičiavimus pagal objekto tipą, trečiasis sluoksnis pateikia duomenis skirtus skaičiavimams.

3.4.3 Grafinės sąsajos apribojimai

Grafinėje sąsajoje koneksinų konfigūracija nėra keičiama, naudojamas standartinis lygiagrečiai sujungtų koneksinų modelis (Pav. 3.23 – koneksino elektrinė schema). Yra galimybė keisti tik jų skaičių (Pav. 3.3 – koneksinų skaičiaus įvestis). Tas pats galioja plyšinės jungties konfigūracijai, koneksonai joje jungiami lygiagrečiai ir keičiamas tik jų skaičius (Pav. 3.4 – koneksonų skaičiaus įvestis) [5]. Lygiagretus koneksonų jungimas šiuo atveju atitinka realaus pasaulio modelį. Kitaip tariant plyšinė jungtis sudaryta iš tūkstančių lygiagrečių koneksinų kertančių ląstelės membraną [17].

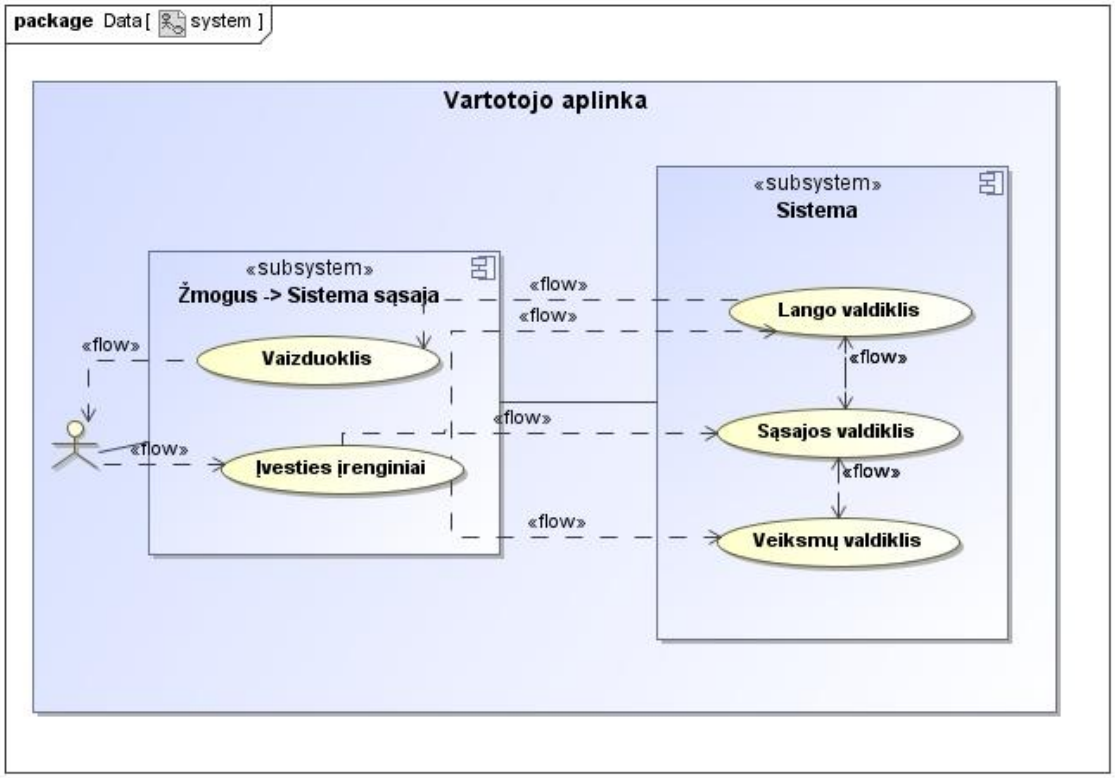
3.4.4 Skaičiavimų bibliotekos apribojimai

Turbūt vienintelis šios bibliotekos apribojimas aktualus praktiškai yra tas, kad koneksino objektas yra konkretizuotas dviejoms būsenoms. Ateityje gan neblogas patobulinimas galėtų būti, koneksino būsenų aibės dinaminis praplėtimas.

[18] Jungimo tipas

3.5 Positemių diagrama

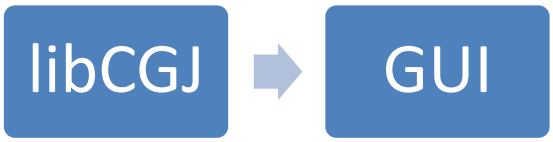
Ši diagrama skirta pavaizduoti kaip atrodys statinis sistemos vaizdas vartotojo aplinkoje. Detalizuojami įvesties, bei išvesties įrenginiai, pats vartotojas. Taip pat nurodomos duomenų srautų kryptys.



Pav. 3.21 – posistemių diagrama

3.6 Sistemos statinis vaizdas

Projektuojant sistemos statinį vaizdą svarbu išanalizuoti įvairius pasirinkimo vairiantus, kadangi tai yra vienas iš sudėtingiausių ir labiausiai laikui imlių procesų. Visų pirma aprašoma struktūrinė sistemos hierarchija, kuri šios sistemos atveju susideda iš dviejų sluoksnių (Pav. 3.22). Tada apibūdiname kaip šie paketai bendradarbiauja tarpusavyje ir suteikia didesnį architektūros funkcionalumą.



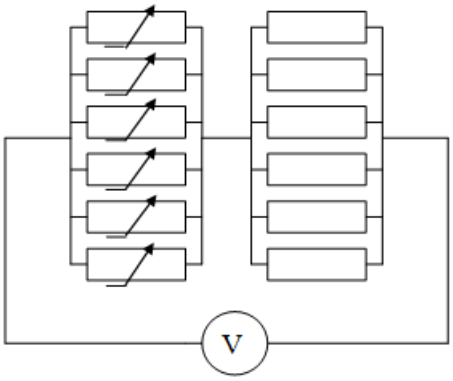
Pav. 3.22 – statinis sistemos vaizdas

3.6.1 Grafinė vartotojo sąsaja

Grafinė vartotojo sąsaja arba kitaip GUI, sukurta panaudojant Qt multiplatforminį[19] karkasą.

3.6.2 Modeliavimo biblioteka

Svarbiausia sistemos dalis vykdanti visus skaičiavimus. Joje panaudota tiesinių lygčių sistemos sprendimo biblioteka Eigen [18]. Programos kodas rašytas ir optimizuotas specialiai C++11 standartui [19]. Šis standartas leido smarkiai sumažinti programinio kodo kiekį, bet išvengti įvairių smulkių programavimo klaidų. Ši sistemos dalis realizuota DLL (Žr. Sk. 8 - 10) pavidalu todėl gali būti nesunkiai panaudojama betkurioje platformoje. Pagrindiniai bibliotekos objektai yra lygiagretaus, bei nuoseklaus jungimo klasės. Kadangi šia programine įranga galima modeliuoti koneksinus kaip parodyta (Pav. 3.23). Buvo išskirti keturi objektai atliekantis duomenų skaičiavimui paruošimą:



Pav. 3.23 – koneksino elektrinė schema

Lentelė 3.5 – operacinių objektų tipai ir jungimas

Objektas	Klasė	Jungimas
Koneksinas	Connexin	Lygiagretus
Sritis[20]	Domain	Nuoseklus
Koneksonas	Connexon	Lygiagretus
Jungtis	Junction	Lygiagretus

Lentelėje aprašyti kokie objektai atitinka kokias programinio kodo klases. Taip pat koks objektas kokį jungimo tipą atitinka. Objektai išdėstyti didėjimo tvarka, nuo pačio mažiausio koneksino iki plyšinės jungties.

[19] Veikiantį įvairiose OS (Žr. Sk. 8 - 13)
[20] puskanalis

3.7 Pagrindinių sistemos objektų sąsajos aprašymas

3.7.1 Bendra programinė sąsaja visiems objektams

Lentelė 3.6 – bendros programinės sąsajos aprašymas

Metodas	Aprašymas
`double G();`	Laidžio gavimas
`void setVoltage(const double &);`	Objekto įtampos nustatymas
`Vector getConductivity();`	Objekto laidžio vektoriaus gavimas
`Vector getStates();`	Objekto būsenų vektoriaus gavimas

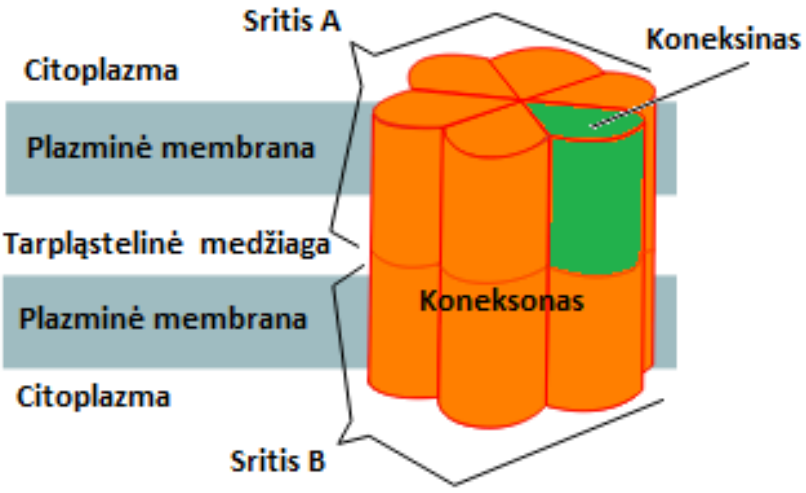
Bendrą programinę sąsaja sudaro keturi metodai kurie paveldimi iš agregacinių tėvinių klasių (Pav. 3.20 – detalizuota klasių diagrama). Vienintelis unikalus sąsajos matodas yra koneksino objekte skirtas keisti jo būsenai.

3.7.2 Koneksinas

Svarbiausias objektas, kuriam nustatomi visi pagrindiniai parametrai (Pav. 3.13). Koneksinas yra mažiausias komunikacinis vienetas jungiantis ląsteles. Šis baltymas keturis kartus kerta ląstelių membraną (po du kartus kiekvienoje iš komunikuojančiųjų ląstelių). Koneksinai ląstelių membranose yra susigrupavę po šešis vienetus kaip parodyta (Pav. 3.25), taip suformuodami puskanalius (angl. hemichannels).

Lentelė 3.7 – koneksino parametrų lentelė

Parametras	Matavimo vienetas	Standartinė reikšmė	Aprašymas
Rc	Varža (R)	300	Koneksino varža esant uždarai būsenai (Pav. 3.24)
Ro	Varža (R)	800	Koneksino varža esant atvirai būsenai (Pav. 3.26)
A		100	
K		1	
V0	Voltai (V)	0,04	Įtampa prie kurios koneksino būsenos pasikeitimo tikimybė yra lygi 50%



Pav. 3.25 – Koneksino, koneksono, bei srities grafinis pavaizdavimas

Pav. 3.24 – koneksono uždara būsena

Pav. 3.26 – koneksono atvira būsena

3.7.2.1 Koneksino programinė sąsaja

Koneksino sąsają sudaro penki metodai:

```
○ void setVoltage(const double &);
○ Vector getConductivity();
○ Vector getStates();
○ void toggle(double);
○ double G();
```

setVoltage – metodas skirtas nustatyti įtampai esančiai tarp koneksino galų, įtampa nustatoma voltais

getConductivity – gaunamas visų būsenų laidžių vektorius

getStates – perėjimo tikimybių vektorius kiekvienai būsenai

toggle – koneksino būsenos tikimybinis perjungimas, perduodamas parametras, tai būsenos pakeitimo tikimybė [0...1]. Šis metodas unikalus koneksino sąsajai. Visi kiti sąsajų metodai naudojami kituose objektuose.

G – koneksino laidis[21] (Žr. Sk. 8 - 21) dabartinėje būsenoje, matavimo vienetas piko siemensai[22]

3.7.3 Sritis

Žiūrėti lentelę (Lentelė 3.6 – bendros programinės sąsajos aprašymas).

3.7.4 Koneksonas

Žiūrėti lentelę (Lentelė 3.6 – bendros programinės sąsajos aprašymas).

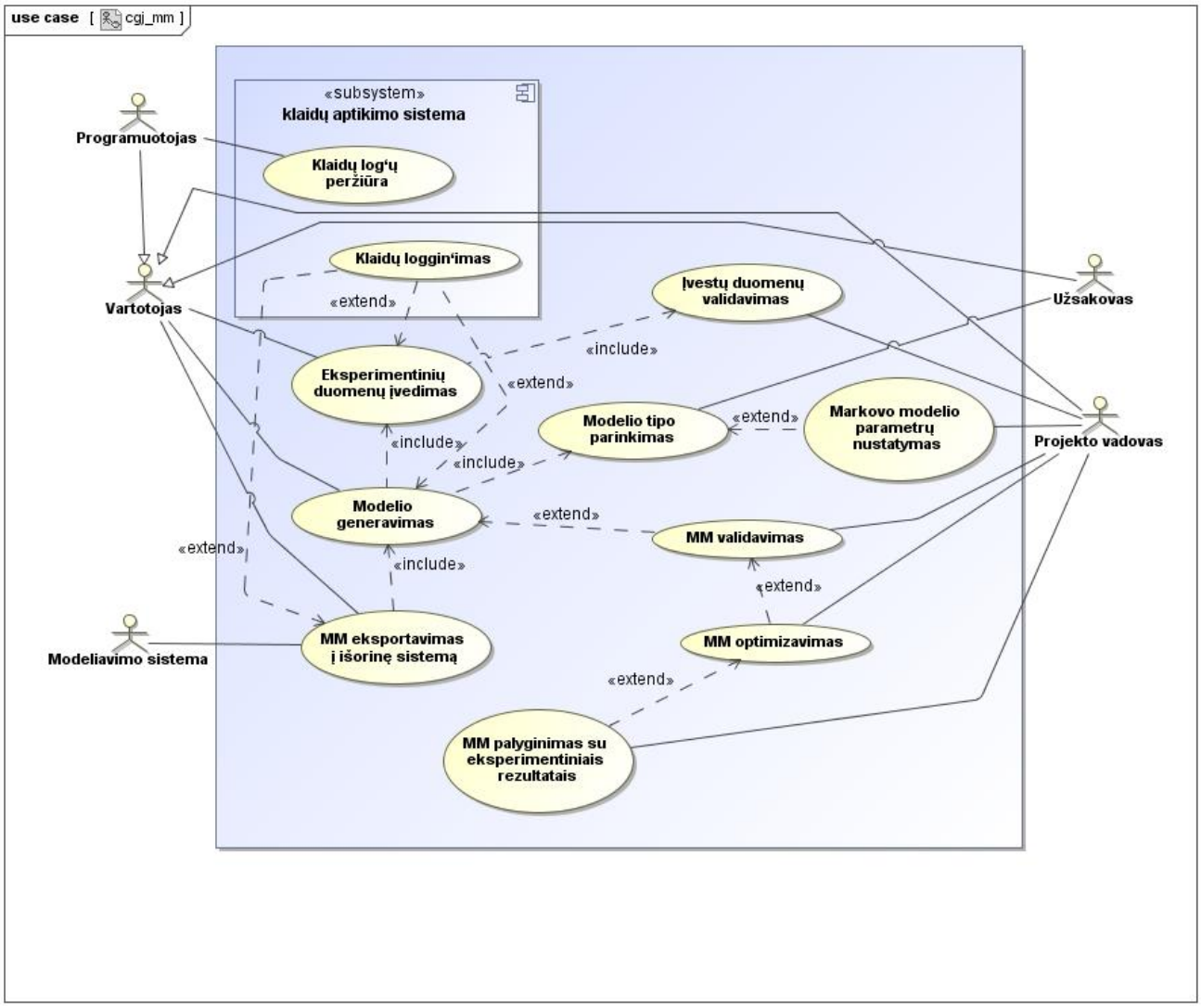
3.7.5 Jungtis

Žiūrėti lentelę (Lentelė 3.6 – bendros programinės sąsajos aprašymas).

[21] Priešingas dydis varžai
[22] Laidžio matavimo vienetai

3.8 Panaudojimo atvejų aprašymas

3.8.1 Panaudojimo atvejų diagrama



Pav. 3.27 – panaudojimo atvejų diagrama

3.8.2 Preliminarus aktorių bei jų funkcijų sąrašas

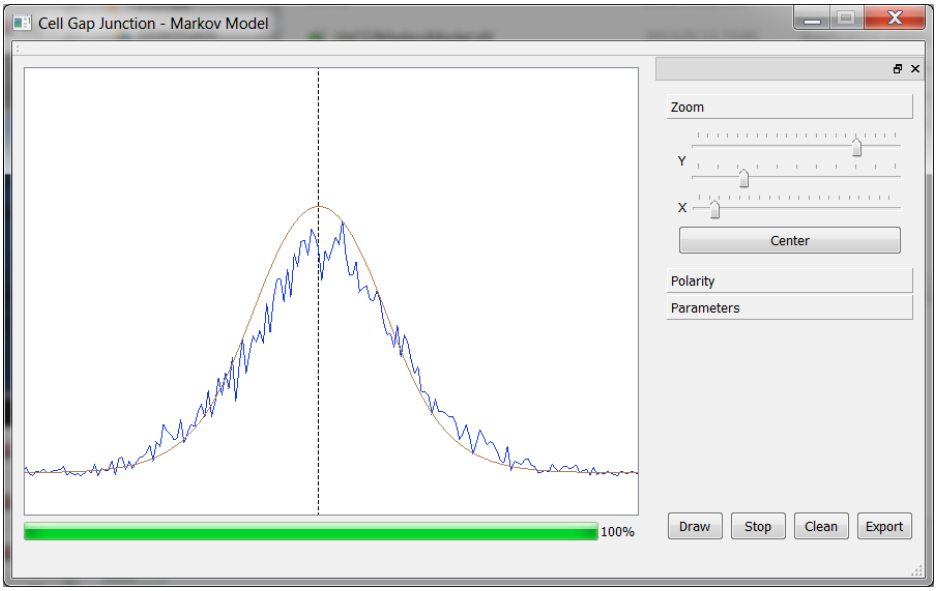
- Vartotojas – tai sistemos vartotojas kuris nori eksperimentų rezultatus paversti į markovo modelį
- Programuotojas – sistemos prižiūrėtojas ir klaidų taisytojas
- Modeliavimo sistema – išorinė sistema kuri naudoja sugeneruotą markovo modelį
- Projekto vadovas – atlieka validavimo, parametrų nustatymo ir optimizavimo funkcijas
- Užsakovas – sistemos vartotojas parenkantis kokio tipo markovo modelį naudoti

3.8.3 Panaudojimo atvejų sąrašas

- Eksperimentinių duomenų įvedimas
- Įvestų duomenų validavimas
- Markovo modelio parametrų nustatymas
- Modelio generavimas
- Modelio tipo parinkimas
- MM validavimas
- MM palyginimas su eksperimentiniais rezultatais
- MM eksportavimas į išorinę sistemą
- MM optimizavimas
- Klaidų registro peržiūra
- Klaidų registravimas

4 Tyrimo dalis

Kuriant šią sistemą, buvo realizuotos trys versijos. Pirmojoje versijoje realizuotas mano sugalvotas fuzzy modelis, kadangi tuo metu dar tiksliai nebuvo aišku, kaip realizuoti nusistovėjusių tikimybių apskaičiavimą, teko sugalvot būdą, kaip kitaip išgauti nusistovėjusias tikimybes. Todėl buvo panaudota fuzzy logic[23]. Vėlesnės versijos realizuotas naudojant markovo modelį ir stacionarių tikimybių apskaičiavimą. Šiose versijose naudojamas tiesinių lygčių sistemos sprendimas, taigi padaryti du variantai naudojant GPU ir CPU atlikti šiai užduočiai. Galutinei sistemos versijai buvo nuspręsta palikti tik CPU, nes spartos atžvilgiu šiai užduočiai lygčių sprendimas ne visiškai pasiteisino ir turėjo keliatą kitų trūkumų kuriuos aptarsime kituose skyriuose.



Pav. 4.1 – realizuotos programinės įrangos vaizdas

[23] Mišri logika

4.1 Fuzzy logic tyrimas

Šio modelio įdėja yra tame, kad pašaliname koneksino būsenos kintamąjį. Tariame, kad jis vienu metu gali būti abiejose būsenos. Tai reiškia, jis vienu metu yra ir atviras ir uždaras. Išlieka klausimas kiek atviras ir kiek uždaras jis yra, kadangi kaip jau buvo minėta, fuzzy logic priima galimybę, kad objektas gali būti tarpinėje būsenoje, tik nežinome (kolkas), kurioje tiksliai vietoje. Koneksino parametrų apskaičiavimui užtenka vienintelės įvesties, įtampos (Formulė 8).

Formulė 8 – apskaičiuoti koneksino laidžiui prie nurodytos įtampos

$$g(V) = \sum_{i}^{n} g(V)_i p(V)_i$$

Ši formulė labai panaši į jau matytą plyšinės jungties laidžio apskaičiavimo formulę (Formulė 7 – skirta apskaičiuoti plyšinės jungties laidumą prie nurodytos įtampos). Tik čia skaičiuojame konkretaus vieno vieno koneksino laidį.

Lentelė 4.1 – koneksino laidžio formulės parametrai

Parametras	**Aprašymas**
$g(V)$	Koneksino laidis prie duotos įtampos
$g(V)_i$	Koneksino būsenos i laidis
$p(V)_i$	Koneksino būsenos i tikimybė
n	Koneksino būsenų skaičius/aibė.

Kadangi duotame markovo modelyje koneksinas turi tik dvi būsenas, galime sudaryti konkretizuotą formulę:

Formulė 9 – konkretizuota formulė dviejų būsenų koneksino laidžiui apskaičiuot

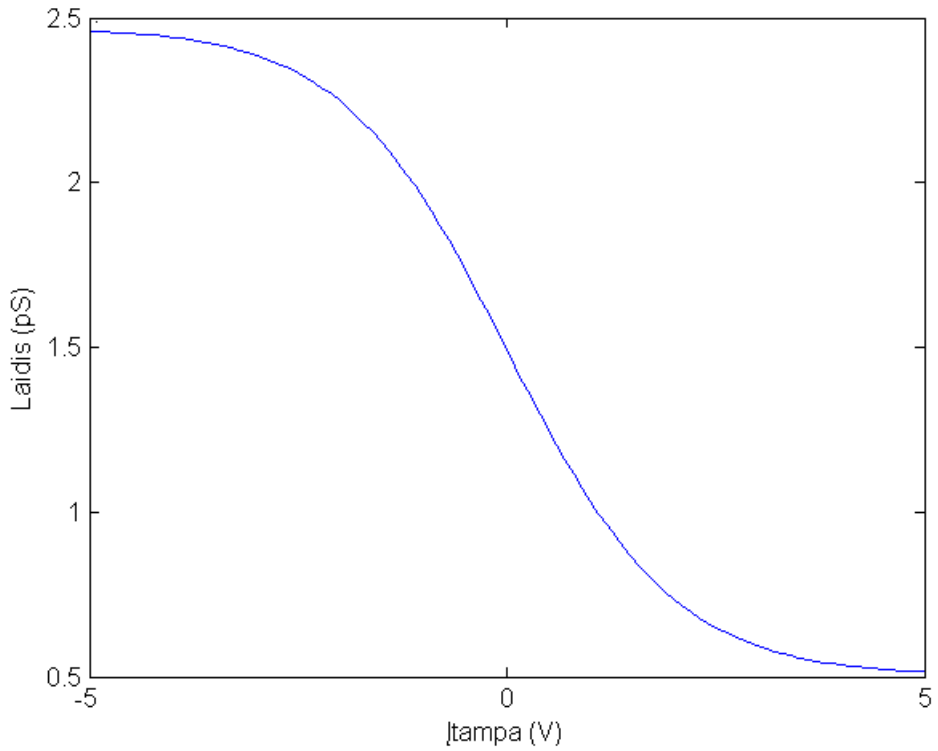
$$g(V) = g(V)_o p(V)_{co} + g(V)_c p(V)_{oc}$$

Parametras	**Aprašymas**
$g(V)$	Koneksino laidis prie duotos įtampos
$g(V)_o$	Koneksino laidis atviroje būsenoje gaunamas iš (Formulė 5).
$p(V)_{co}$	Koneksino laidis atviroje būsenoje gaunamas iš (Formulė 3).
$g(V)_c$	Koneksino laidis atviroje būsenoje gaunamas iš (Formulė 6).
$p(V)_{oc}$	Koneksino laidis atviroje būsenoje gaunamas iš (Formulė 2).

4.1.1 Koneksino laidžio priklausomybė nuo įtampos

Iš aukščiau gautos formulės (Formulė 9 – konkretizuota formulė dviejų būsenų koneksino laidžiui apskaičiuot) galime nubrėžti vieno koneksino laidžio priklausomybės nuo įtampos grafiką. Kadangi grafikui brėžti nebuvo parinkti jokie parametrai. Visos konstantos lygios vienetui arba nuliui, gautas grafikas su ne visiškai realiom įtampos reikšmėm. Grafike įtampos rėžiai kinta tarp [-5…5] Voltų. Tačiau bendram dėsningumui pamatyti to pakanka.

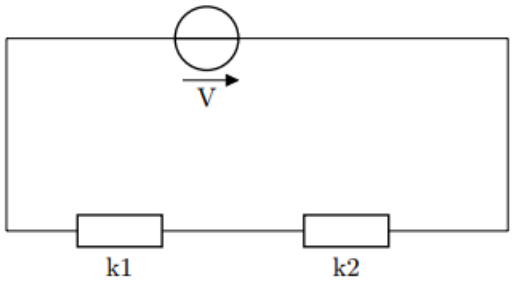
Taigi šiame grafie turime tolydų laidžio kitimą, vietoj diskretinių markovo modelio būsenų, kai laidžio reikšmė gali būti 2ps arba 0.5pS su tais pačiais parametrais kurie buvo naudoti šiam grafikui.



Pav. 4.2 – koneksino laidžio priklausomybė nuo įtampos

4.1.2 Dviejų nuosekliai sujungtų koneksinų tyrimas

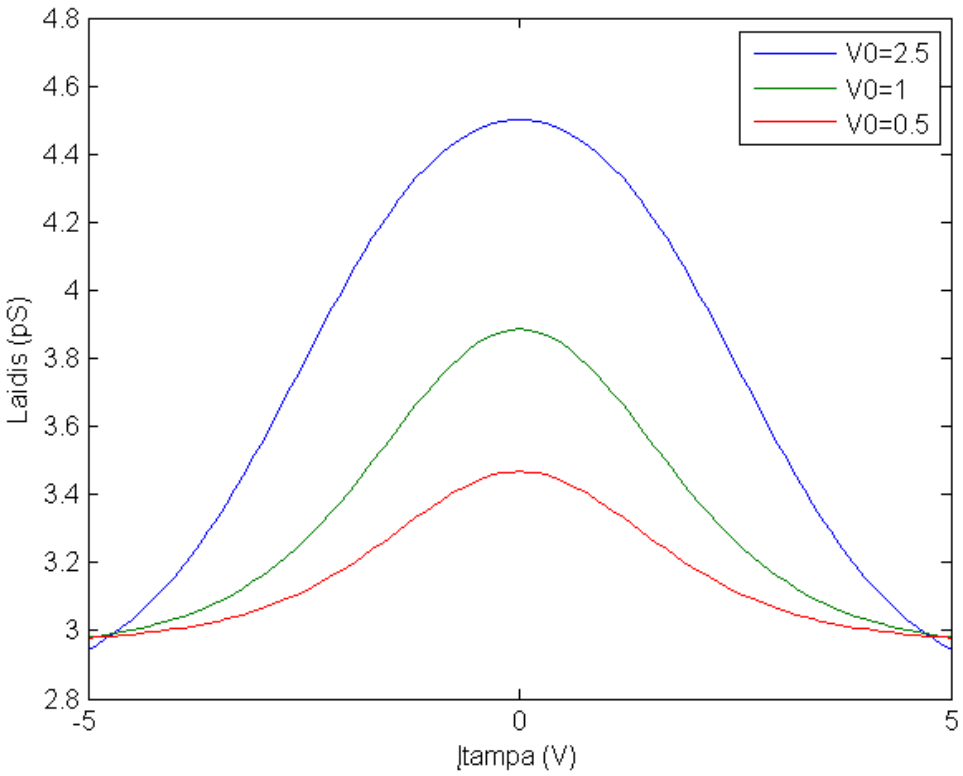
Tarkim, turime du nuosekliai sujungtus koneksinus, kaip parodyta (Pav. 4.3). Nubrėšime grafikus su parametrais pateiktais lentelėje (). Kiti parametrai nuliai arba vienetai atitinkamai, kad neįtakotų skaičiavimų reikšmių.



Pav. 4.3 – dviejų nuosekliai sujungtų koneksinų schema

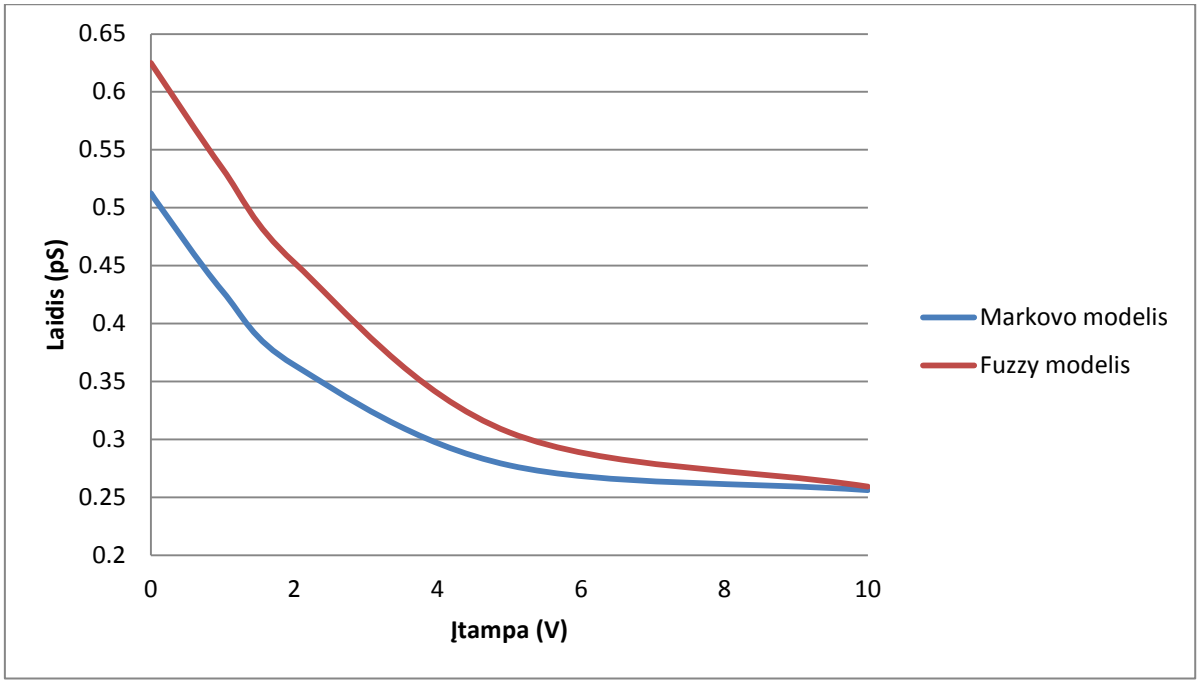
Lentelė **4.2 – grafikų ir koneksinų parametrų duomenys**

Grafikas	Koneksinas	V0	P
V0=0.5	k1	0.5	+1
	k2	0.5	-1
V0=1	k1	1	+1
	k2	1	-1
V0=2.5	k1	2.5	+1
	k2	2.5	-1



Pav. 4.4 – dviejų nuosekliai sujungtų koneksinų su priešingu poškumu grafikas

4.1.3 Markovo modelio ir fuzzy modelio palyginimas



Pav. 4.5 – Markovo ir Fuzzy modelių palyginimas

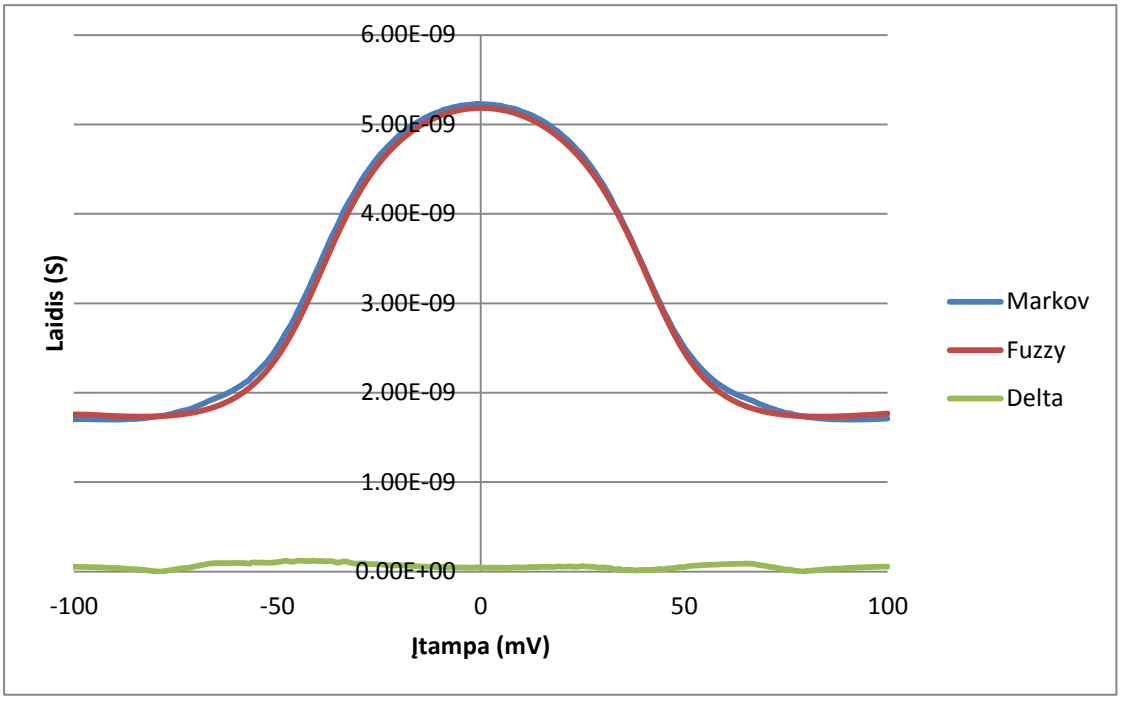
Kaip matome iš grafiko abu modeliai duoda panašius rezultatus, kuriuos būtų galima visiškai suvienodinti parinkus atitinkamus parametrus.

Pasinaudojus sukurta programa galime atlikti detalesni tyrimą ir pabandyti surasti parametrų prie kurių grafikai tampa vienodi. Rankiniu būdu parinkę parametrus, gauname gana neblogus rezultatus. Markovo modelis praktiškai idealiai atitinka Fuzzy modelį.

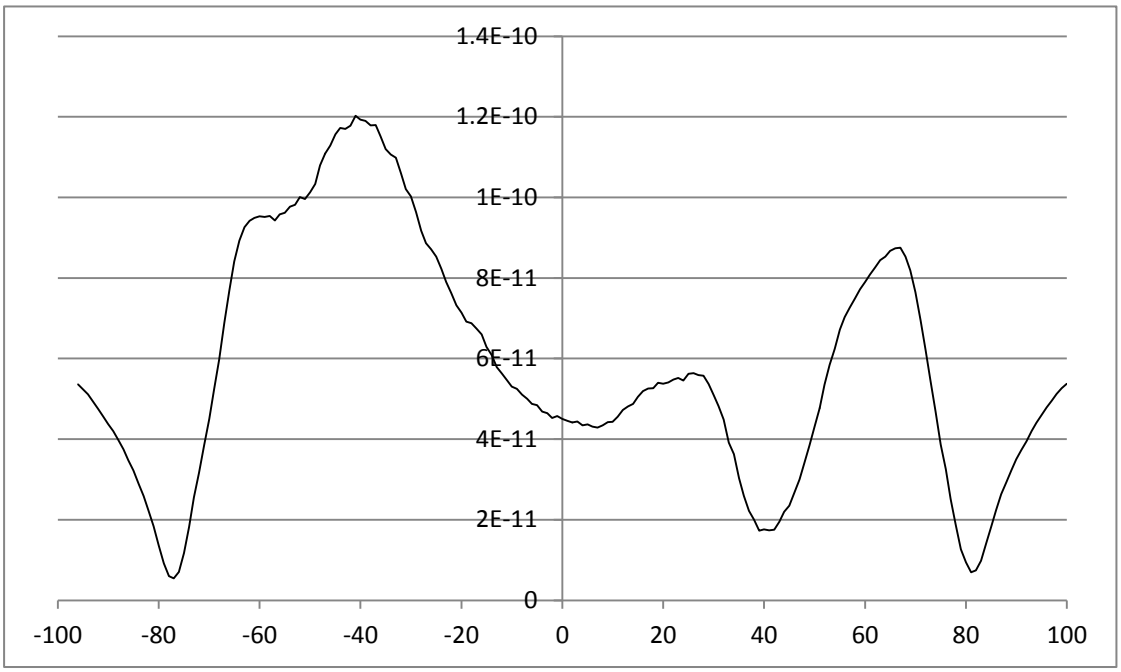
Lentelė 4.3 – markovo ir fuzzy modelių parametrai prie kurių laidžio grafikai sutampa

	Modelis	
Parametras	**Fuzzy**	**Markov**
A	94	100
V0(mV)	18	20
Rc	260	300
Ro	500	800

Su šiais parametrais nubrėžę grafikus, matome, kad jie yra praktiškai identiški (Pav. 4.7 – markov ir fuzzy modelių grafikai). O skirtumas tarp jų 10^{11} eilės. Taigi skirtumas yra 100 kartų mažesnis už reikšminę zoną kuri yra 10^{9} eilės.



Pav. 4.7 – markov ir fuzzy modelių grafikai

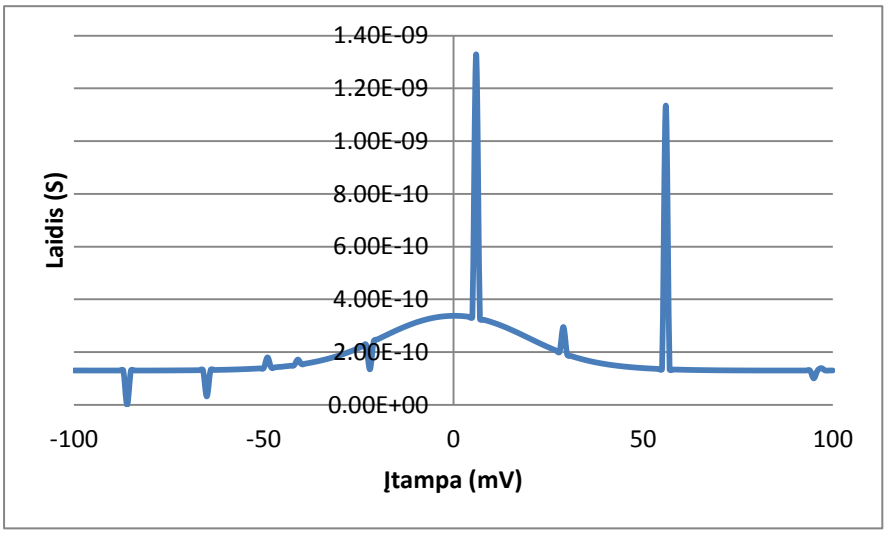


Pav. 4.6 – skirtumas tarp markov ir fuzzy modelių

4.2 Markovo modelis ir GPU

Viena iš programos versijų buvo paruošta tiesinias lygčių sistemas spręsti GPU pagalba, kadangi panaudojus vaizdo plokštės skaičiuojamąją galią būtų galima išgauti tiesišką pagreitėjimą (Pav. 4.10 – lygčių sprendimo pagreitėjimas), priklausomai nuo nežinomųjų skaičiaus. Tai reiškia su 2Gb vaizdo atminės, galime išspręsti ~10000 nežinomųjų lygčių sistemą. Didinant nežinomųjų kiekį atsiras lėtėjimas, nes vektoriai nebetilps į vaizdo plokštės atmintį.

Taip pat panaudojus GPU atsirado kita problema, skaičiavimų netikslumas (Pav. 4.8). Todėl šios programos versijos buvo atsisakyta, nes pritrūko laiko išsiaiškinti iš kur atsiranda tie netikslumai. Galimi du klaidų šaltiniai pati vaizdo plokštė, kadangi ji nėra skirta tiksliems skaičiavimams arba programinė klaida.



Pav. 4.8 – GPU skaičiavimų netikslumai

4.3 Kokybės įvertinimas

4.3.1 Sistemos testavimas

4.3.1.1 Testavimo tikslai ir objektai

Programinės įrangos testavimo tikslas, įsitikinti, kad joje neliko tam tikro tipo klaidų. Pagrindinis uždavinys programinės įrangos testavime, teisingai atsirinkti testuojamus objektus, kad testavimas nebūtų perteklinis ir padengiantis kiek įmanoma daugiau kodo. Kad užtikrinti didelės programų sistemos kokybę, kodo padengiamumas neturėtų būti mažesni nei 80%.

4.3.1.2 Testavimo apimtis ir tipai

Atliekamas automatizuotas testavimas. Naudojama „unit testing[24]“ metodika.

4.3.1.3 Reikalavimai testavimui

Testavimui reikalingi įrankiai:

- Eclipse programavimo aplinka.
- MinGW kompiliatorius palaikantis C++11 standartą.
- CUTE testavimo biblioteka įdiegta į Eclipse.

[24] Vienetų testavimas

4.3.2 Reikalavimų išpildymas

Ne visi pradiniai reikalavimai buvo pilnai išpildyti, kadangi projekto pradžioje buvo prigalvota pernelyg daug perteklinių, nenaudingų ir nepanaudojamų funkcijų. Todėl dalis reikalavimų buvo pakeista kitais, o nereikalingų ir perteklinių reikalavimų atsisakyta.

4.3.3 Kokybės įvertinimo kriterijai

Įvertinti sistemos kokybei išvesti štai tokie kokybės parametrai, aprašyti lentelėje (Lentelė 4.4).

Lentelė 4.4 – kokybės įvertinimo kriterijai

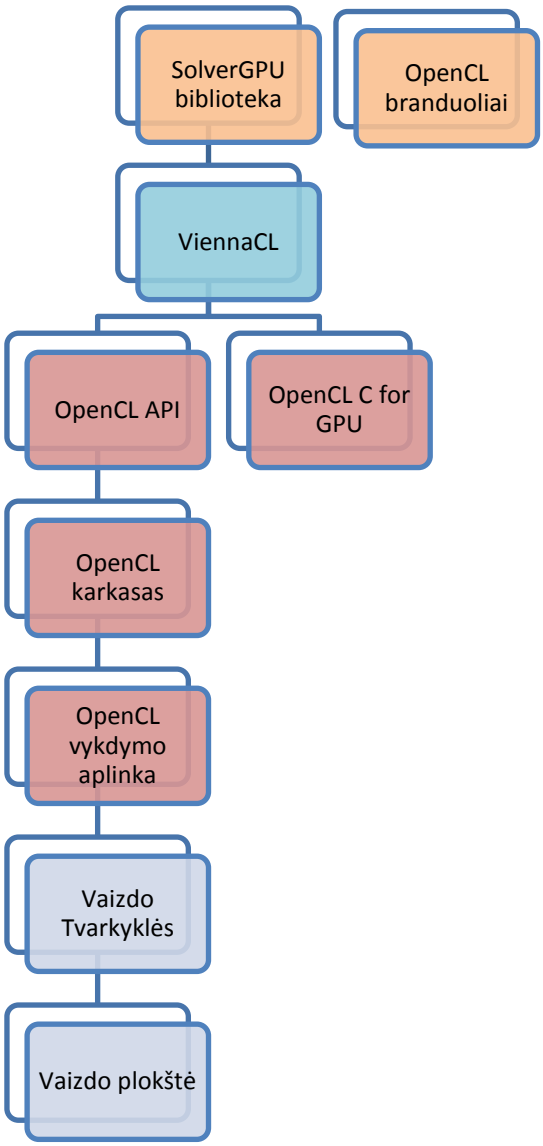
Parametras	**Aprašymas**
Išplėčiamumas	Galimybė praplėsti programinės įrangos funkcijas. Naujų modulių kūrimo galimybė.
Pernešamumas	Galimybė perkelti programinę įrangą ant kitokios techninės įrangos, operacinės sistemos.
Sąsajos galimybės	Ar gali veikti su kitomis sistemomis.
Panaudojamumas	Ar lengva išmokti dirbti su programine įranga.
Patvarumas	Kiek tolerantiška sistema vartotojo klaidoms?

4.4 Sistemos patobulinimai

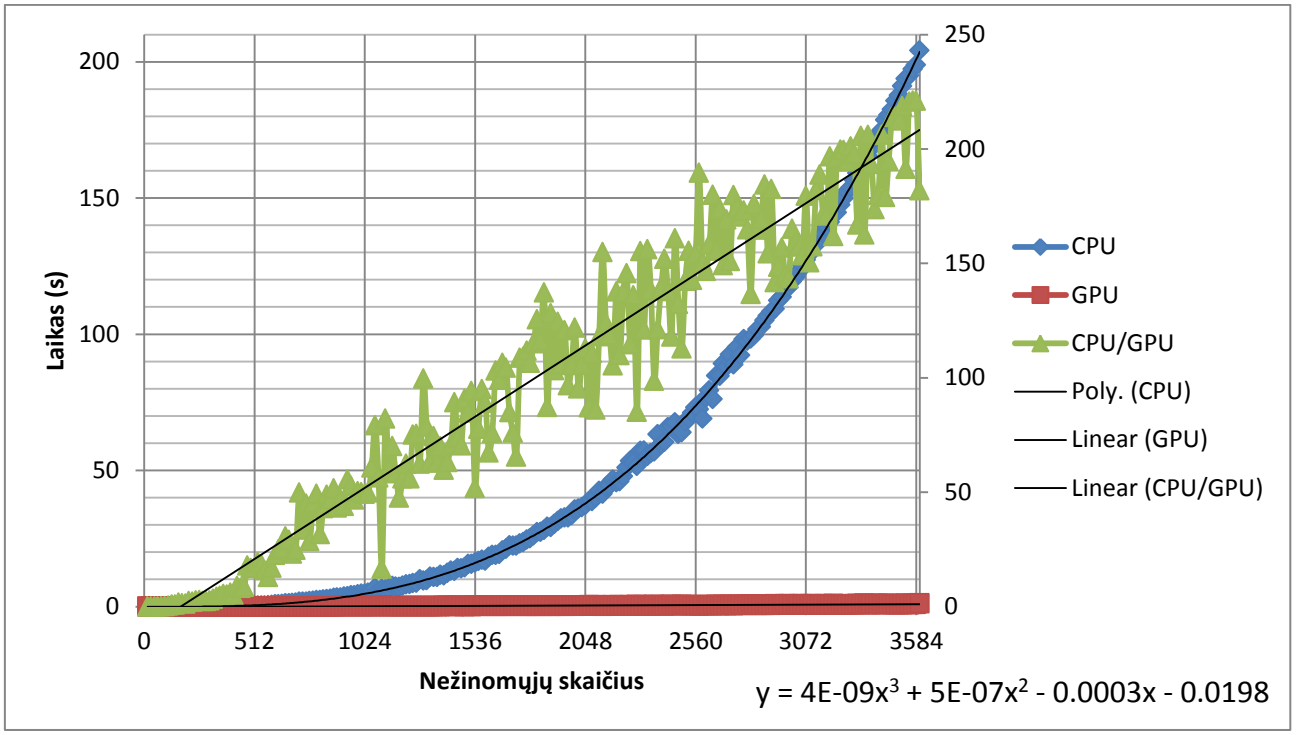
Šiame skyriuje aptarsime ką galima būtų patobulinti sukurtoje sistemoje. Keletas kosmetinių patobulinimų galėtų būt: grafiko priartinimo funkcijos pagerinimas, grafikos brėžimas skaičiavimų metu. Taip pat atskiras grafikų valdymas, trynimas, parametrų keitimas, bei galimybė surinkti betkokios konfigūracijos plyšinę jungtį. Šie atobulinimai daugiausiai skirti vartotojo sąsajai. Tačiau yra porą esminių dalykų kuriuos būtų galima realizuoti skaičiavimų bibliotekoje, t.y. fuzzy modelio palaikymas kartu su markovo, bei GPU panaudojimas skaičiavimams.

4.4.1 GPGPU panaudojimas

Galimi sistemos patobulinimai galėtų būt: GPGPU panaudojimas. Paspartinti skaičiavimus, kadangi sprendžiant lygis GPU pagalba galima išgauti pagreitėjimą prieš CPU iki ~200 kartų be papildomų optimizacijų. Sukurtos lygčių sprendimo bibliotekos architektūra:

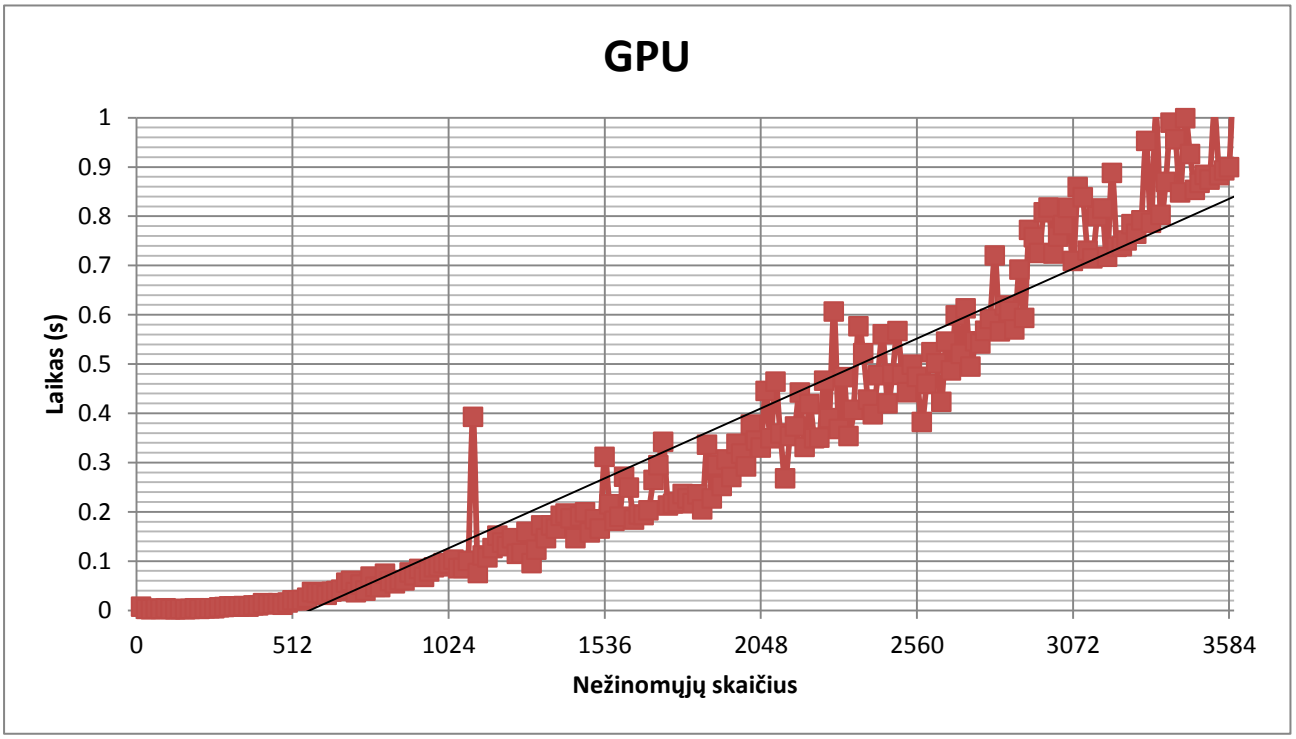


Pav. 4.9 – GPU lygčių sprendimo bibliotekos architektūra



Pav. 4.10 – lygčių sprendimo pagreitėjimas

Kaip matome iš šio grafiko didinant nežinomųjų skaičių maždaug iki 10000 išgaunamas tiesiškass pagreitėjimas prieš CPU. Tai reiškia, kad 10000 nežinomųjų lygčių sistemą GPU išspręstų ~550 kartų greičiau negu CPU.



Pav. 4.11 – GPU tiesinių lygčių sprendimo laiko grafikas

4.4.2 Vartotojo sąsajos patobulinima

...

5 Eksperimentinė dalis

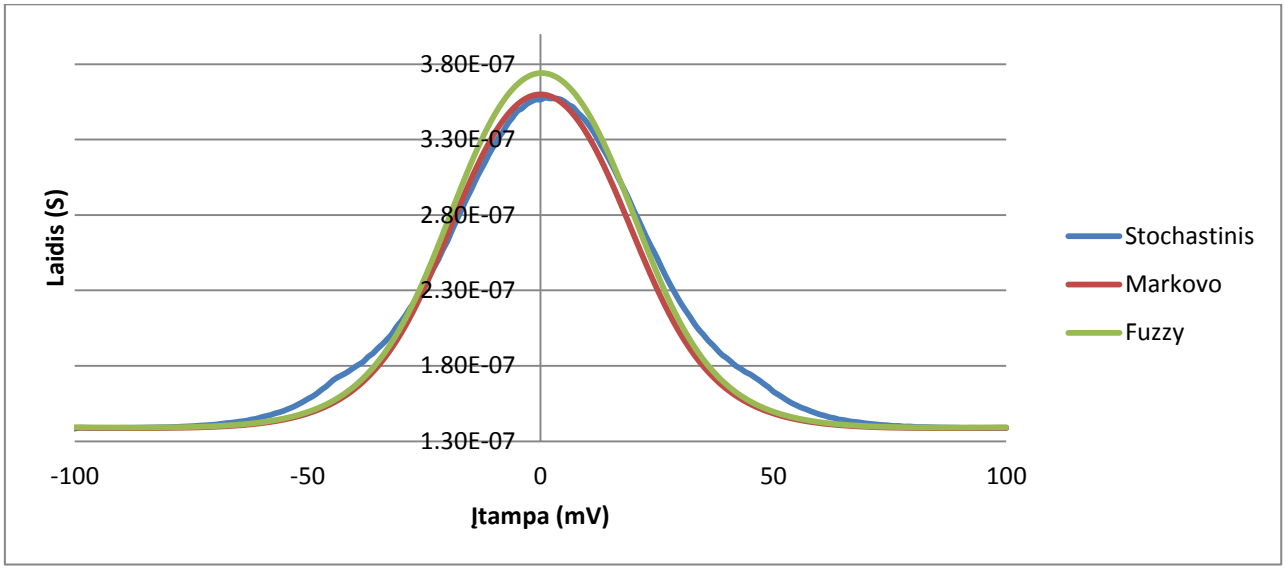
Šioje dalyje atliksime trijų algoritmų vykdymo laiko palyginimą. Taip pat palyginsime markovo modelio rezultatus su jau egzistuojančiomis sistemomis, bei sprendimais.

5.1 Sistemos spartos tyrimas

Sistemos spartos tyrimas buvo atliktas naudojant **ribinius parametrus**[25] kurie tiesiogiai įakoja algoritmų veikimo laiką:

Lentelė 5.1 – ribiniai programos parametrai naudoti spartos tyrimui

Programos parametras	**Reikšmė(-s)**
Type (modelio tipas)	$\{Stochastic, Fuzzy, Markov\}$
Iterations (iteracijos)	1
Rc	300
Ro	800
Connexins (koneksinai)	**64**
Connexons (koneksonai)	**10000**
A	100
K	1
V0	0
Range (mV) (įtampos protokolo rėžiai)	$[-100 \dots 100]$



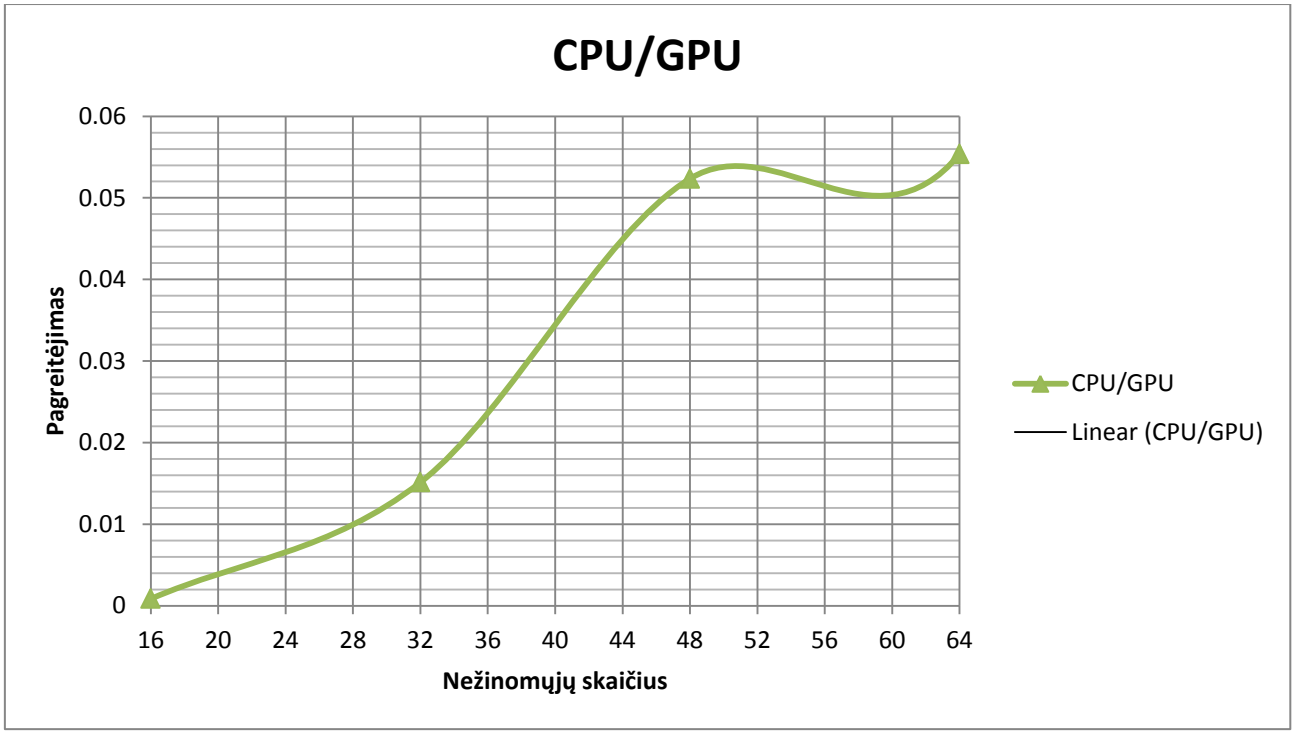
Pav. 5.1 – stochastinio, markovo ir fuzzy modelių grafikai su ribinėmis reikšmėmis

Šiai užduočiai atlikti teorinis pagreitėjimas turėtų būti 10^4 kartų. Kadangi išsimeta vienas ciklas iš algoritmo kuris iteruoja per visus koneksinus. Kitaip tariant pagreitėjimas turėtų būti proporcingas koneksinų skaičiui.

[25] Maksimalūs leidžiami parametrai – įvesti per vartotojo sąsają (lentelėje paryškinti (Lentelė 5.1 – ribiniai programos parametrai naudoti spartos tyrimui))

Lentelė 5.2 – stochastinio, markovo ir fuzzy modelių generavimo laikas

Modelis	**Laikas (s)**
Stochastinis	294.87
Markovo (CPU)	0.03036
Fuzzy	0.00210
Markovo (GPU)	0.6052



Pav. 5.2 – GPU prieš CPU pagreitėjimas [16..64] nežinomųjų ribose

Kaip matome iš šio grafiko sąlyginai mažoms lygčių sistemoms GPU naudoti neapsimoka, o kadangi realiausias variantas jog lygčių sistema turės ~7 nežinomuosius t.y. 6 koneksinų modelis, tai sprendimo sulėtėjimas bus virš 100 kartų.

5.1.1 Parametrų įtaka algoritmams

Kiekvienas parametras turi skirtingą įaka algoritmų veikimo laikui, šiame skyriuje teoriškai aprašyta, kaip, koks parametras ir kodėl įtakoja algoritmus.

Lentelė 5.3 – teorinė parametrų įtaka algoritmams

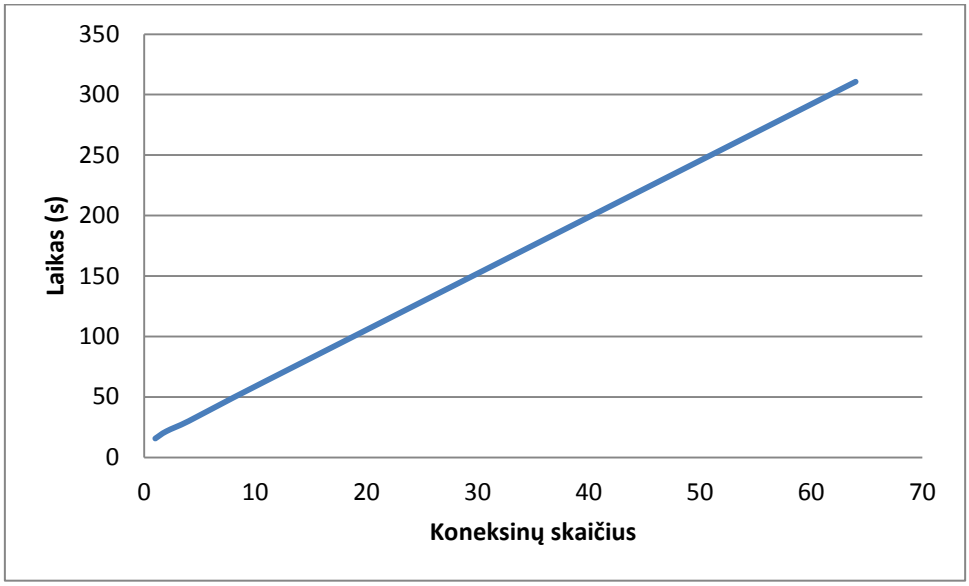
Algoritmas	Parametras	Įtaka
Stochastinis	Iterations	Padidina algoritmo sudėtingumą iki $O(n^3)$ kadangi jis apibrėžia iteracijų skaičių nusistovėjusio laidumo radimui stochastiniame modelyje.
	Connexins	Algoritmo $O(\mathrm{n})$ dedamoji. Nuo koneksinų skaičiaus vykdymo laikas didėja kvadratine priklausomybe, kadangi koneksinų būsenų aibė yra $(n+1)^2$ kur n – koneksinų skaičius.
	Connexons	Algoritmo $O(\mathrm{n})$ dedamoji. Šis parametras apibrėžia ciklą, kuris iteruoja per koneksonų aibę.
Markovo	Iterations	Neįtakoja, kadangi nusistovėjęs laidumas šiame modelyje apskaičiuojamas iš tiesinių lygčių sistemos.
	Connexins	$O(n^{\lceil 1\ldots 3 \rceil})$ algoritmo dedamoji. Tai praktiškai vienintelis parametras įtakojantis algoritmo veikimo laiką, kadangi nuo koneksinų skaičiaus priklauso būsenų aibės dydis, tuo tarpu ir tiesinių lygčių sistemos matricos dydis, o kadangi standartinei tiesinių lygčių sistemai išspręsti naudojamas Gauss-Jordan algoritmas kurio sudėtingumas yra $O(n^3)$ tai šiuo atveju ir algoritmo sudėtingumas gaunamas analogiškas.
	Connexons	Šis parametras neįtakoja algoritmo veikimo laiko, kadangi markovo modelyje jis yra tiesiog daugiklis. $g_j = ng_c$ kur n yra koneksonų skaičius, o g_c vieno koneksino laidis.
Fuzzy	Iterations	Neįtakoja, kadangi nusistovėjęs laidumas yra tiesiogiai gaunamas iš kiekvieno koneksino naudojant formulę (Formulė 9 – konkretizuota formulė dviejų būsenų koneksino laidžiui apskaičiuot).
	Connexins	$O(\mathrm{n})$ algoritmo dedamoji. Šiame algoritme koneksinų skaičius laiką įtakoja tiesiškai, kadangi neatliekamas lygčių sistemos sprendimas, laidis gaunamas iškarto naudojant formulę (Formulė 8 – apskaičiuoti koneksino laidžiui prie nurodytos įtampos).
	Connexons	Neįtakoja, kadangi daroma prielaida (kaip ir markovo modelyje), jog visi koneksinai vienodi ir šis parametras yra tiesiog daugiklis.

5.2 Parametrų įtaka algoritmų veikimo spartai

Šiame skyriuje atliksime ir aprašysime eksperimentus nustatyti koks parametras ir kaip praktiškai įtakoja algoritmus ir jų veikimo spartą.

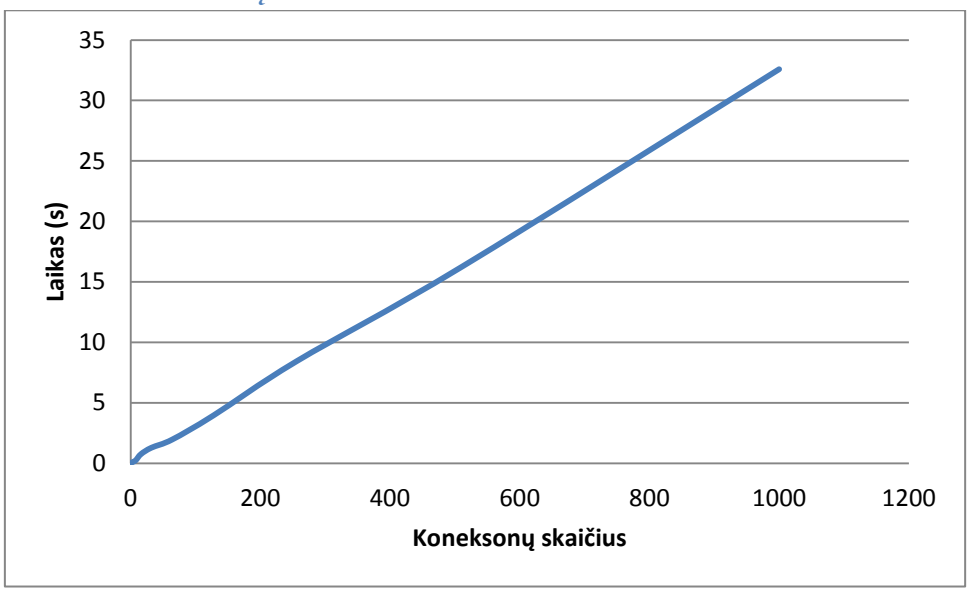
5.2.1 Stochastinis algoritmas

5.2.1.1 Koneksinų skaičius



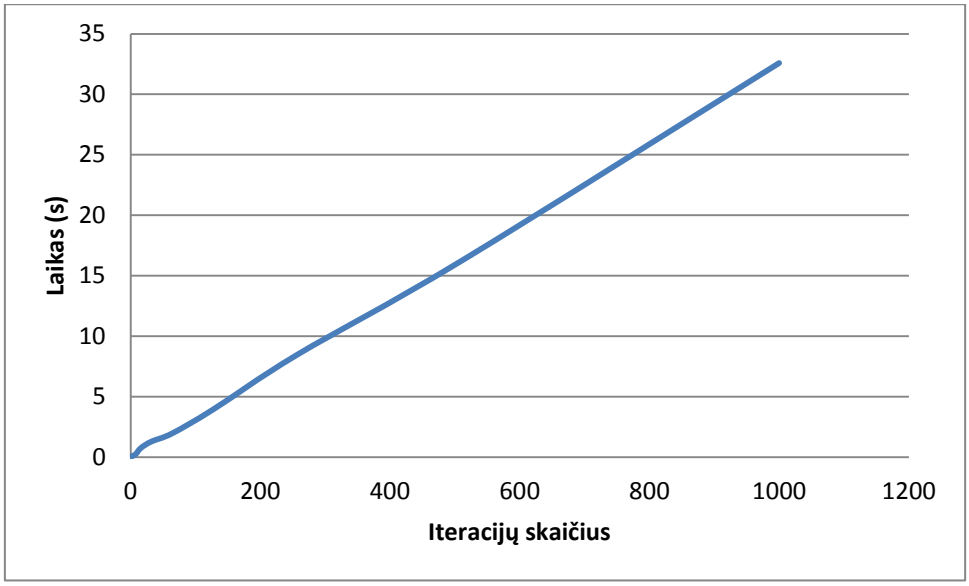
Pav. 5.3 – koneksinų skaičiaus įtaka stochastinio algoritmo vykdymo laikui

5.2.1.2 Koneksonų skaičius



Pav. 5.4 – koneksonų skaičiaus įtaka stochastinio algoritmo vykdymo laikui

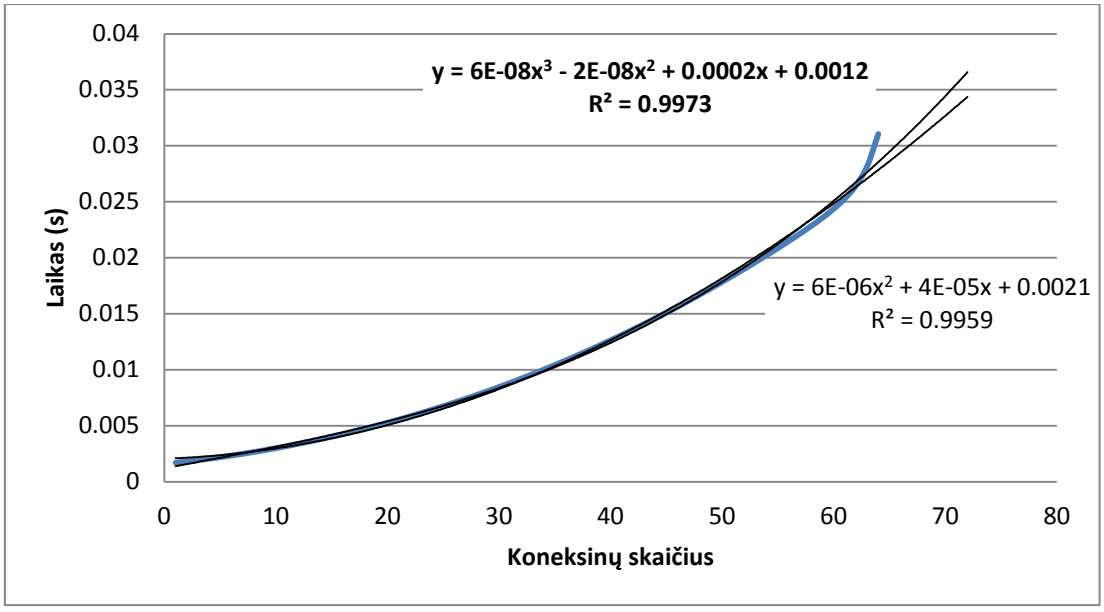
5.2.1.3 *Iteracijos*



Pav. 5.5 – iteracijų skaičiaus įtaka stochastinio algoritmo vykdymo laikui

5.2.2 Markovo algoritmas

5.2.2.1 *Koneksinų skaičius*

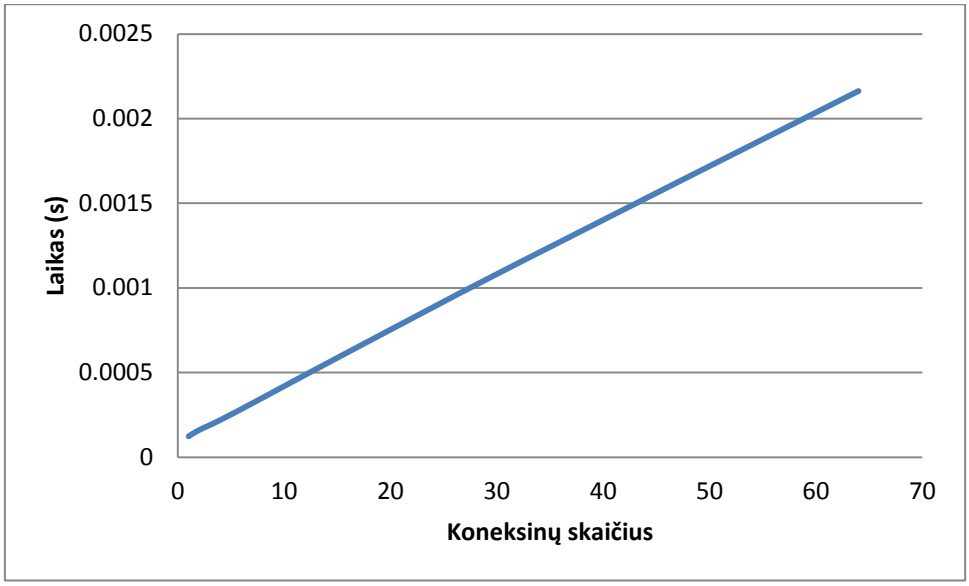


Pav. 5.6 – koneksinų skaičiaus įtakos grafikas markovo algoritmo vykdymo laikui

Kaip matome iš grafiko laiko kitimo dėsnis geriausiai atitinka trečiojo laipsnio polinomą, kadangi R^2 yra arčiau vieneto.

5.2.3 Fuzzy algoritmas

5.2.4 Koneksinų skaičius



Pav. 5.7 – koneksinų skaičiaus įtaka fuzzy algoritmo vykdymo laikui

5.3 Eksperimentinių rezultatų apibendrinimas

Nustatytas eksperimentinis skaičiavimų pagreitėjimas yra artimas teoriniam, taigi galima sakyti, jog sistema puikiai funkcionuoja ir realizuotas algoritmas pasiteisino. Eksperimentiškai nustatytas algoritmų sudėtingumas pateiktas lentelėje.

Lentelė 5.4 – eksperimentiškai nustatytas algoritmų sudėtingumas

Algoritmas	**Sudėtingumas**
Stochastinis	$O(i * cx * cn)$
Markov (CPU)	$O(cn^3)$
Markov (GPU)	$O_{CPU}(cn^1) * O_{GPU}(cn^2) \cong O(cn^1)$
Fuzzy	$O(cn^1)$

- i – iteracijų skaičius
- cx – koneksonų skaičius
- cn – koneksinų skaičius

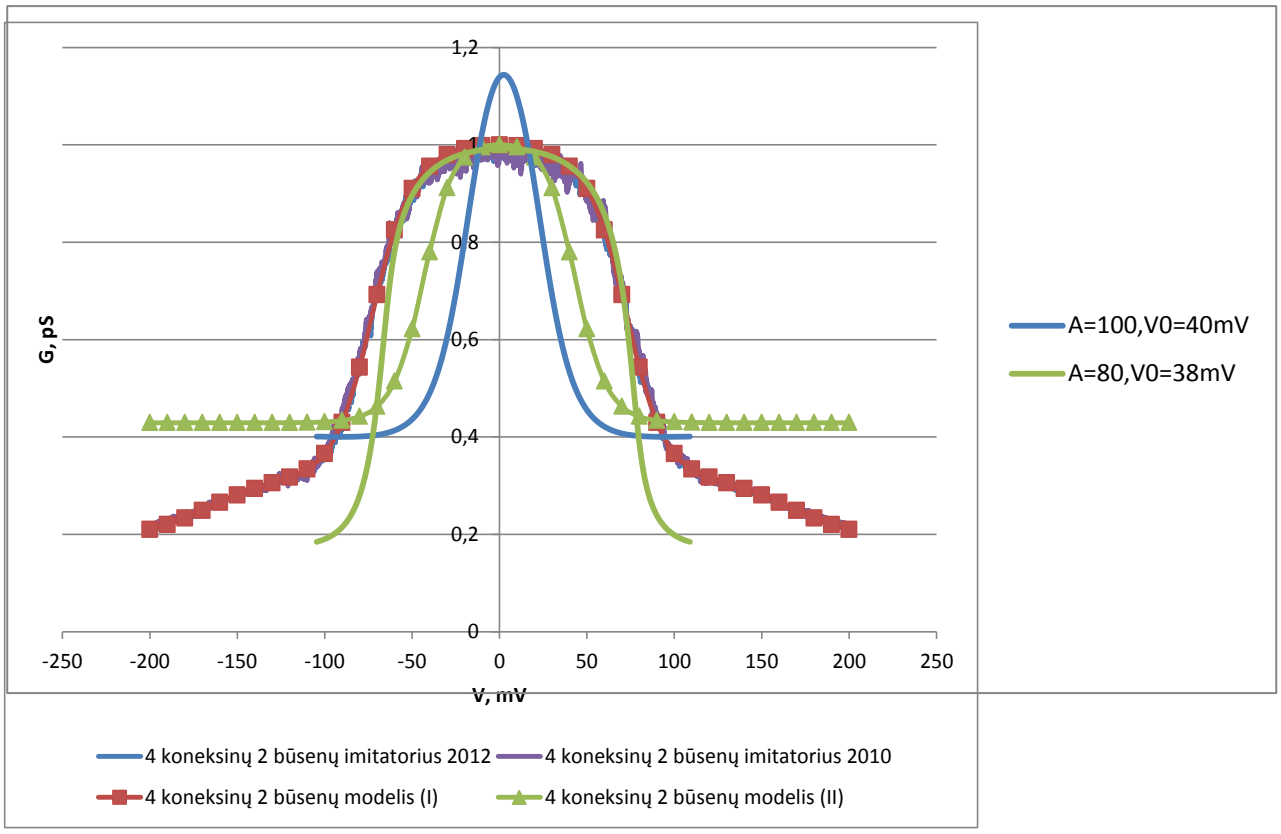
Taigi iš lentelės matome, kad du geriausi variantai yra Fuzzy, bei Markov (GPU) algoritmai. Deja, nors GPU algoritmo sudėtingumą galime sumažinti iki pirmos eilės, vykdymo laikas nebus geresnis dėl prasto optimizvimo. Geresnį vykdymo laiką būtų galima išgauti optimizavus GPU algoritmą konkrečiai užduočiai, pvz sprendžiant daug mažų tiesinių lygčių sistemų vienu metu.

Lentelė 5.5 – modelių privalumų ir trūkumų apibendrinimas

Modelis	**Privalumai**	**Trūkumai**
Stochastinis	Parodo laidžio kitimo dinamiką Nereikia spręsti lygčių sistemų	Yra pats lėčiausias iš visų modelių Sugeneruotame grafike yra triukšmas Būsenos įjunga/išjungta tipo – neatitinka realaus pasaulio
Fuzzy	Grafikas yra tolydus Skaičiavimai vykdomi greičiausiai Koneksino laidžio grafikas tolygus – atitinka realų pasaulį	Nėra laidžio kitimo dinamikos Ne visiškai sutampa su markovo modeliu
Markovo (CPU)	Grafikas yra tolydus Skaičiuoja greičiau Būsenos įjunga/išjungta tipo – neatitinka realaus pasaulio	Nėra laidžio kitimo dinamikos Reikia spręsti lygčių sistemą Būsenos įjunga/išjungta tipo – neatitinka realaus pasaulio
Markovo (GPU)	Grafikas yra tolydus Skaičiuoja greitai Yra galimybė optimizuoti GPU architektūrai ir išgauti agreitėjimą iki 1000 kartų	Nėra laidžio kitimo dinamikos Reikia spręsti lygčių sistemą Skaičiavimų rezultatai ne tikslūs Mažoms būsenų aibėms neefektyvu Būsenos įjunga/išjungta tipo – neatitinka realaus pasaulio

5.4 Markovo modelio rezultatų palyginimas

Palyginimui panaudosime „Markov models of voltage gating of gap junction channels“ [3] straipsnyje pateiktą medžiagą. Deja, ten nebuvo nurodyta su kokiais parametrais atlikti eksperimentai, taigi teko atlikti bandymus rankiniu būdu ir parinkti reikšmes kurios geriausiai grafiškai atitinka pateiktus rezultatus. Grafiko duomenų lentelė taip pat nėra pateikta, taigi sulipdysime paveiksliuką su grafiku iš excel programos.



Pav. 5.8 – markovo modelio grafinis rezultatų palyginimas

6 Išvados

Pasirinkta tema yra perspektyvi ir aktuali tiek mokslininkam tiek eiliniam vartotojui, todėl tokio tipo programinė įranga ateityje turėtų būti plačiai naudojama, žmogaus organizmo veiklos modeliavimui, bei jo reakcijai į įvairius dirgiklius tirti.

Atlikus modelių eksperimentų rezultatų analizę paaiškėjo, kad markovo modelio pagreitėjimas lyginant su stochastiniu yra tiesinis, priklausomai nuo koneksinų skaičiaus. Tai reiškia, kad markovo modelis bus tiek kartų greitesnis koks yra koneksinų skaičius plyšinės jungties konfigūracijoje.

Tačiau rezultatai nėra palankus markovo modeliui, kadangi atsitiktinai sugalvotas fuzzy modelis yra dar bent ~10 kartų efektyvesnis už markovo. Šio modlio privalumas, jog išvengiama tiesinių lygčių sistemos sprendimo, o tai reiškia, kad didėjant koneksinų skaičiui juntamas dar didesnis pagreitėjimas, nes fuzzy algoritmo sudėtingumas blogiausiu atveju yra $O(n)$, kai tuo tarpu markovo $O(n^3)$.

Išbandytas markovo modelio variantas perkeliant skaičiavimus ant GPU nepasiteisino, nes sistemos būesnų matrica yra tiesiog permaža, kad būtų juntamas realus pagreitėjimas. Priešingai, buvo stebimas iki 20 kartų lėtesnis veikimas. Tačiau teoriškai optimizavus uždavinį specialiai GPU architektūrai, galima būtų išgauti pagreitėjima iki 1000 kartų.

7 Literatūra

[1] W. Community, „Markov model,“ Wikipedia, 3 Lapkričio 2011. [Tinkle]. Available: http://en.wikipedia.org/wiki/Markov_model. [Kreiptasi 11 Lapkričio 2011].

[2] W. Community, „Gap junction,“ Wikipedia, 1 Spalio 2011. [Tinkle]. Available: http://en.wikipedia.org/wiki/Gap_junction. [Kreiptasi 4 Spalio 2011].

[3] H. Pranevicius, N. Paulauskas, M. Pranevicius, O. Pranevicius, M. Snipas ir F. Bukauskas, „Markov models of voltage gating of gap junction channels,“ 2013.

[4] A. Sakalauskaitė, „LĄSTELIŲ PLYŠINĖS JUNGTIES MODELIAVIMAS NAUDOJANT MARKOVO GRANDINES,“ Kaunas University of Technology, Kaunas, 2011.

[5] A. Sakalauskaite, H. Pranevicius, M. Pranevicius and F. Bukauskas, "Markovian Model of the Voltage Gating of Connexin-based Gap Junction Channels," *ELECTRONICS AND ELECTRICAL ENGINEERING,* vol. 111, no. 5, pp. 103-106, 2011.

[6] A. Valančiūtė, K. Baltrušaitis, A. Vitkus, I. Balnytė ir R. Kubilius, Burnos ertmės organų HISTOLOGIJA IR EMBRIOLOGIJA, Kaunas: Kauno Medicinos Universiteto Leidykla, 2008.

[7] A. Padaiga ir A. Vitkus, Bendroji histologija, Kaunas: Lietuvos veterinarijos akademija. Leidykla "Naujasis LANKAS", 2002.

[8] M. H. Hennig, "Modelling Synaptic Transmission," ANC, Informatics, University of Edinburgh, Edinburgh, 2008.

[9] V. Mildažienė, S. Jarmalaitė ir R. Daugelavičius, Ląstelės biologija, Kaunas: Vytauto Didžiojo Universiteto leidykla, 2004.

[10] N. Palacios-Prado, S. Sonntag, V. A. Skeberdis, K. Willecke and F. Bukauskas, "Gating, permselectivity and pH-dependent modulation of channels formed by connexin57, a major connexin of horizontal cells in the mouse retina," *Physiology,* vol. 587, no. 13, p. 3251–3269, 2009.

[11] N. Palacios-Prado and F. Bukauskas, "Heterotypic gap junction channels as voltage-sensitive valves for intercellular signaling," *Biphysics and computational biology,* vol. 106, no. 35, p. 14855–14860, 2009.

[12] N. Palacios-Pradoa, S. W. Briggsa, V. A. Skeberdis, M. Pranevičius, M. V. L. Bennetta and F. Bukauskasa, "pH-dependent modulation of voltage gating in connexin45 homotypic and connexin45/connexin43 heterotypic gap junctions," *Physiology,* vol. 107, no. 21, p. 9897–9902, 2010.

[13] A. P. Moreno, M. B. Rook, G. Fishman and D. C. Spray, "Gap Junction Channels: Distinct Voltage-Sensitive and Insensitive Conductance States," *Biophysical Journal,* vol. 67, pp. 113-119, 1994.

[14] "Permeability of homotypic and heterotypic gap junction channels formed of cardiac connexins mCx30.2, Cx40, Cx43, and Cx45," *American Journal of Physiology - Heart and Circulatory Physiology,* no. 293, p. H1729–H1736, 2007.

[15] N. Paulauskas, M. Pranevicius, H. Pranevicius and F. Bukauskas, "A stochastic four-state model of contingent gating of gap junction channels containing two ‘fast’ gates sensitive to transjunctional voltage," *Biophysical Journal,* vol. 96, p. 3936–3948, 2009.

[16] R. Wilders, R. Kumar, R. W. Joyner, H. J. Jongsma, E. E. Verheijck, G. Golod, A. C. van Ginneken ir W. N. Goolsby, „Action potential conduction between a ventricular cell model and an isolated ventricular cell,“ *Biophysical Journal,* t. 81, p. 2112–2121, 2001.

[17] J. Tigerholm, "A-type Potassium Channels in Dendritic Integration Role in Epileptogenesis," KTH Computer Science and Communication, Stockholm, 2009.

[18] OpenSource, „Eigen,“ Tuxfamily, 2012. [Tinkle]. Available: http://eigen.tuxfamily.org/dox/. [Kreiptasi 10 Gegužės 2013].

[19] P. Becker, „Working Draft, Standard for Programming Language C++,“ 28 Vasario 2011. [Tinkle]. Available: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2011/n3242.pdf. [Kreiptasi 10 Gegužės 2013].

8 Terminų ir santrumpų žodynas

1. Spin Box – skaičiams skirtas įvesties laukas kuriame reikšmes galima keisti pelės pagalba
2. Dockable Widget – programos lango dalis kurią galima perkelti iš vienos vietos į kitą
3. Qt – multiplatforminis C++ kalbos karkasas skirtas kurti grafinėms programoms
4. GPL – general public license, licencija apibrėžianti atviri kodo programinės įrangos teises
5. MM – markovo modelis
6. PJ – plyšinė jungtis
7. CGJ – cell gap junction, ląstelių plyšinė jungtis
8. libCGJ – ląstelių plyšinių jungčių modeliavimo biblioteka
9. lib – library (biblioteka)
10. DLL – dynamic link library (dinamiškai pajungiama biblioteka)
11. GUI – graphical user interface (grafinė vartotojo sąsaja)
12. Karkasas – programavimo kalbos bibliotekų rinkinys skirtas palengvinti tam tikrus veiksmus
13. OS – operacinė sistema
14. Nexus – lotyniškas plyšinės jungties pavadinimas
15. PLA – piece-linear aggregates (tiesiniai agregatai)
16. X86 – 32 bitų procesorių architektūra
17. X64 – 64 bitų procesorių architektūra
18. Matlab – programa skirta matematiniams skaičiavimams atlikti ir manipuliacijai su duomenimis
19. CVRTI – The Cardiovascular Research and Training Institute
20. LU – „LU decomposition“ algoritmas. Matricos išskaidymas į žemesniąją trikampę matricą ir aukštesniąją trikampę matricą
21. Laidis – siemensai
22. Markovo procesas – įvykiu seka, kai per tam tikrą laiką įvyksta N įvykių
23. Topologija – tam tikrų objektų išsidėstymas. Pvz.: elektrinės schemos elementų išdėstymas
24. Omo dėsnis – trumpai tariant U=IR. Apibrėžia grandinės elementų srovės ir įtampos kitimą priklausomai nuo varžos.
25. Unit testing – vienetų testavimas
26. Agregacinis ryšys – ryšys reiškiantis vieno objekto nuosavybę kitam
27. CX – koneksono trumpinys
28. DM – srities trumpinys
29. CN – koneksino trumpinys
30. Lygiagretus – schemos elementų jungimo tipas, kai varža mažėja. Kitaip tariant schemos varža yra lygi sujungtų elementų **laidžio** aritmetiniai sumai.
31. Nuoseklus – schemos elementų jungimo tipas, leidžiantis padidinti varžą. Kitaip tariant schemos varža yra lygi sujungtų elementų aritemetinei sumai
32. Įtampos protokolas – taisyklės kaip turi kisti įtampa bėgant laikui
33. Fuzzy logic – apytislė logika. Matematinės logikos forma kurioje remiamasi, kad reikšmės nėra tikslios ir dterministinės, o gali įgyti tarpines reikšmes pvz.: tarp nulio ir vieneto
34. GPGPU – bendros paskirties skaičiavimai atliekami grafinio procesoriaus pagalba
35. Algoritmas – veiksmų seka leidžianti pasiekti tam tikrą rezultatą