VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS

Lygiagretūs algoritmai difuzijos lygtims spręsti

Parallel Algorithms for Solving Diffusion Equations

Bakalauro baigiamasis darbas

Atliko: 4 kurso 3 grupės studentas
Matas Savickis (parašas)

Darbo vadovas: Rokas Astrauskas, J. Asist. (parašas)

Recenzentas: Algirdas Lančinskas, Doc., Dr. (parašas)

Vilnius – 2020

Padėka

- Noriu padėkoti savo šeimai už pasitikėjimą ir moralinį palaikymą per visus metus praleistus studijuojant.
- Noriu padėkoti savo darbo vadovui Rokui Astrauskui už skirtą laiką ir pagalbą rašant šį darbą. Geresnio vadovo negalėčiau prašyti.

Santrauka lietuvių kalba

Skenuojančio elektrocheminio mikroskopo veikimas aprašomas naudojantis dalinėmis diferencialinėmis lygtimis difuzijos procesams modeliuoti. Kadangi šioms lygtims nėra analitinio sprendimo, jas aproksimuojame skaitinių metodų lygtimis [BiJ05]. Siekiant gauti tikslų rezultatą reikia atlikti daug skaičiavimų, kurie gali užtrukti ilgai, todėl yra tikslinga ieškoti būdų pagreitinti skaičiavimus. Šiame darbe bus ieškoma būdų kaip naudojantis lygiagrečiais algoritmais pagreitinti modelio skaičiavimą. Lygiagretūs algoritmai buvo rašomi naudojant MPI realizaciją: mpi4py Python programavimo kalba. Pagrindinis darbo tikslas yra rasti optimalų lygiagretų algoritmą modeliuoti deguonies koncentraciją susidarančią atliekant eksperimentus su skenuojančiu elektrocheminiu mikroskopu (angl. SECM).

Atliekant eksperimentus, buvo naudojamasi VU MIF Paskirstytų skaičiavimų tinklu ir lokaliais autoriaus resursais. Buvo įgyvendinti du lygiagretūs algoritmai: vienmatis ir dvimatis. Lokalioje aplinkoje skaičiavimams buvo naudojamasi iki 5 branduolių, o MIF klasteryje iki 50 branduolių. Darbe buvo sėkmingai pagreitinti skaičiavimai taikant literatūroje aprašytus algoritmus[Čie01]. Nustatyta, kad vienmatis algoritmas yra optimaliausias būdas lygiagretinti skaičiavimams. Vienmatis algoritmas buvo greičiausias tiek lokalioje tiek ir MIF klasterio aplinkoje.

Raktiniai žodžiai: Lygiagretūs skaičiavimai, dalinės diferencialinės lygtys, Laplaso lygtys, difuzijos lygtis, šilumos perdavimo lygtis.

Santrauka anglų kalba (Summary)

The operation of a sequential electrochemical microscope is described using partial differential equations to model diffusion processes. Since there is no analytical solution to these equations, we approximate them to the equations of numerical methods. Achieving an accurate result requires a lot of calculations that can take a long time, so it makes sense to look for ways to speed up the calculations. This paper will look at ways to speed up the computation of the model using parallel algorithms. Parallel algorithms were written using the MPI implementation: mpi4py in the Python programming language. The main goal of the work is to find an optimal parallel algorithm to model the oxygen concentration generated by experiments with a scanning electrochemical microscope (SECM).

The experiments used the VU MIF Distributed Computing Network and the author's local resources. Two parallel algorithms were implemented: one-dimentional and two-dimentional Up to 5 cores were used for computations in the local environment, and up to 50 cores in the MIF cluster. The calculations were successfully accelerated using the algorithms described in the literature cite Lygeg. The one-dimensional algorithm has been found to be the most optimal way to parallelize the calculations. The one-dimensiona algorithm was the fastest in both the local and MIF cluster environments.

Keywords: Parallel calculations, partial differential equations, Laplace equations, diffusion equation, heat transfer equation.

TURINYS

Įvadas

Šiame darbe bus ieškomas optimalus lygiagretusis algoritmas norint modeliuoti skenuojantį elektroninį mikroskopą [Ram16]. Darbe bus atliekami eksperimentai su skirtingais lygiagrečiaisiais algoritmais [Čie01] naudojant MIF VU paskirstytų skaičiavimų centro [uni20] resursus bei atliekant skaičiavimus lokalioje aplinkoje. Norint atlikti tikslius difuzijos lygčių skaičiavimus tenka laukti nemažai laiko [NHS12] kol skaičiavimai baigiasi, todėl sunku greitai įvertinti modelio tikslumą ir atlikti pakeitimus. Teorinėje literatūroje yra aprašyti keli algoritmai [Čie01], kuriais naudojantis galima lygiagretinti difuzijos procesų skaičiavimus. Šis darbas siekia įvertinti lygiagrečių algoritmų spartą ir jų savybes. Tikimasi, kad darbas bus naudingas Vilniaus universiteto (VU) Matematikos ir informatikos fakulteto (MIF) mokslininkų bendradarbiavimui su Vilniaus universiteto Chemijos ir geomokslų fakultetu tolimesniems Skenuojančio elektrocheminio mikroskopo modeliavimo tyrimams [Ram16].

Darbe yra išsikeliamos keturios pagrindinės užduotys:

1. Matematinių lygčių ir Skaitinių metodų algoritmų, reikalingų tyrimui, apibrėžimas.
2. Realaus SECM modelio sudarymas.
3. Lygiagrečių algoritmų apibrėžimas.
4. Optimalaus lygiagretaus algoritmo suradimas ir realizacija.
5. Lygiagrečių skaičiavimų eksperimentai lokalioje aplinkoje.
6. Lygiagrečių skaičiavimų naudojant MIF paskirstyto skaičiavimo centro resursus.

Darbas yra suskirstytas į tris dėstymo skyrius.

1. Dalinės diferencialinės lygtys difuzijos lygtims spręsti ir skaitinių metodų algoritmai
2. SECM modelio sudarymas
3. Lygiagretūs algoritmai difuzijos lygtims spręsti

Pirmame dėstymo skyriuje (Dalinės diferencialinės lygtys difuzijos lygtims spręsti ir skaitinių metodų algoritmai) bus įvedamos matematinės uždavinio formuluotės. Bus apibrėžiamas trimatis (3D) Puasono dalinės diferencialinės lygties atvejis, lygtis bus pertvarkoma į dvimatę (2D) Laplaso lygtį. Laplaso lygčiai bus apibrėžta skaičiavimo sritis. Apibrėžus matematines lygtis bus pereita prie skaitinių metodų algoritmų. Darbe bus pasirinkta analizuoti Baigtinio skirtumo metodą (angl. Finite difference method). Bus užrašoma matematinė Baigtinio skirtumo formuluotė. Šis formuluotė pradžioje bus pateikta bet kokio dydžio dvimačiam (2D) kūnui skaičiuoti ir paskui bus pertvarkoma į vienetinio kūno lygtį paprastesniems skaičiavimams atlikti. Matematinė ir skaitinė dalis bus paremta knyga „Numerical Analyis Eighth Edition" kurią parašė Richard L. Burden ir J. Douglas Faires. [BiJ05]. Toliau bus realizuojamas skaitinis algoritmas. Realizacijai bus naudojama Python 2 programavimo kalba [Pyt20] ir matematinė biblioteka NumPy [Num20]. Užrašysime

konkretų algoritmą lygčiai skaičiuoti, pateiksime grafinę skaičiavimų reprezentaciją. Skyriaus gale skaičiavimus vienetiniam kūnui laisvai pasirinksime kraštines sąlygas tam kad vizualiai įsitikintume ar algoritmas veikia korektiškai.

Antrajame dėstymo skyriuje (SECM modelio sudarymas) bus aprašomas Skenuojančio elektrocheminio mikroskopo veikimo principas, kokios dalys sudaro SECM, kokie procesai vyksta dirbant su SECM ir kuo tai susiję su pirmame skyriuje apibrėžtomis difuzijos lygtimis. Skyriuje bus diskutuojama kuo yra naudingi tyrimai su SECM, kokiais tyrimais bus remiamasi šiame darbe ir kuo šis darbas gali būti naudingas tolimesniam VU Matematikos ir informatikos fakulteto bendradarbiavimui su VU Chemijos ir geomokslų fakultetu. Šiame skyriuje bus apibrėžiamas SECM modelis, kraštinės modelio reikšmės tiek konstantų tiek funkcijų pavidalu, kraštinių funkcijos sąlygų aproksimavimo metodas bei tikslumo reikšmės. Skyriuje bus remiamasi Felikso Ivanausko, Ingos Morkvenaitės–Vilkoncienė, Roko Astrausko bei Arūno Ramanavičiaus darbu „Modelling of Scanning Electrochemical Microscopy at Redox Competition Mode Using Diffusion and Reaction Equations", Roko Astrausko pranešimo „Difuzijos ir reakcijos procesų elektrocheminėje mikroskopijoje matematinis modeliavimas" užrašais, bei prieš tai paminėta knyga „Numerical analysis Eighth Edition" [BiJ05] Skyriaus pabaigoje pateiksime skaičiavimų grafikus gautus naudojant skirtingas, skyriuje apibrėžtas, kraštines sąlygas. Bus pasirinktas vienas modelis, kuris bus naudojamas skaičiavimams paskutiniame dėstymo skyriuje.

Trečiajame dėstymo skyriuje (Lygiagretūs algoritmai difuzijos lygtims spręsti) bus nagrinėjami lygiagretūs algoritmai. Skyriaus pradžioje apibrėšime Amdahlo dėsnį, kokių rezultatų tikimasi pasiekti realizavus lygiagrečius algoritmus ir kas yra žinučių apsikeitimo protokolas MPI. Toliau bus apibrėžiama pati lygiagrečių skaičiavimų architektūra su pagrindiniu branduoliu ir skaičiavimo branduoliais. Aprašysime du lygiagrečius algoritmus, vienmatį (1D) ir dvimatį (2D), kuriuos realizuosime naudodamiesi Python programavimo kalbos MPI realizacija mpi4py. Toliau bus atliekami eksperimentai naudojant autoriaus lokalią sistemą bei MIF paskirstytų skaičiavimų centro resursais (MIF klasteris). Trumpai pristatysime tiek lokalios aplinkos tiek ir MIF klasterio specifikaciją. Pateiksime eksperimentų, atliktų skirtingose aplinkose, rezultatus ir padiskutuosime apie gautus rezultatus. Šiame skyriuje bus remiamasi Raimondo Čiegio knyga „Lygiagretieji skaičiavimai" [Čie01], Barry Wilkinson ir Michael Allen knyga „Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers Second Edition" [iMic05], bei mpi4py dokumentacija [mpi20]

Darbo pabaigoje reziumuosime kas buvo nuveikta šiame darbe, gauti rezultatai bei išvados ir kokia linkme būtų galima tęsti šį darbą.

1. Puasono dalinės diferencialinės lygtys ir skaitiniai algoritmai

1.1. Puasono dalinės diferencialinės lygtys

Poskyrį pradėsime apibrėždami pagrindinę matematinę lygtį, kuri bus šio darbo pagrindas. Literatūroje [BiJ05] yra aprašoma Dalinė diferencialinė Puasono (angl. Poisson) lygtis (1) skirta skaičiuoti trijų dimensijų (3D) difuzijos procesus.

$$\frac{\partial^2 u}{\partial x^2}(x,y,z) + \frac{\partial^2 u}{\partial y^2}(x,y,z) + \frac{\partial^2 u}{\partial z^2}(x,y,z) = f(x,y,z) \tag{1}$$

Šiame darbe modeliuosime dvejose dimensijose, todėl iš formulės galime panaikinti z ašį ir gauti dviejų dimensijų (2D) lygtį (2).

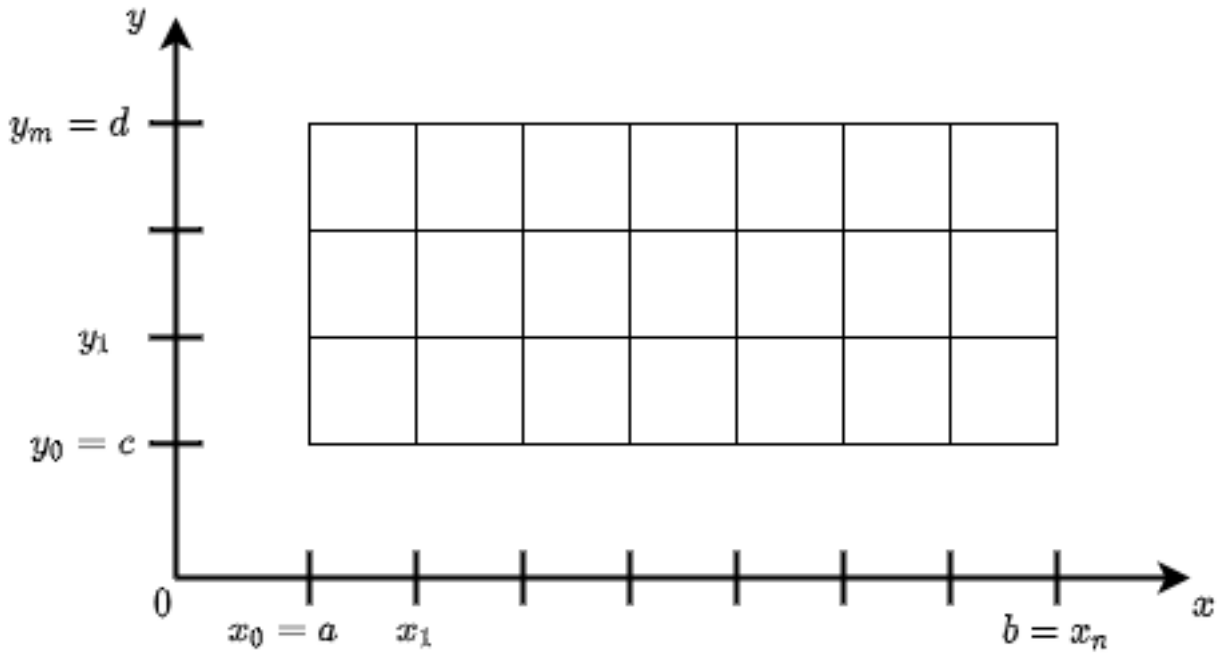
$$\frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = f(x,y) \tag{2}$$

Dešinėje lygties (2) pusėje $f(x,y)$ rodo, kad sistemoje egzistuoja vidinis difuzijos šaltinis. Šiame darbe bus modeliuojamos tik kraštinės sąlygos ir vidinių difuzijos šaltinių neturėsime, todėl $f(x,y) = 0$. Gauname dviejų dimensijų(2D) difuzijos lygtį(3).

$$\frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = 0 \tag{3}$$

Difuzijos lygtyje (3) $u(x,y)$ žymi tinklo taško reikšmę, kuri priklauso racionaliųjų skaičių aibei. Lygtyje (3) x^2 žymi tinklo ilgį, o y^2 tinklo plotį.

Apibrėžkime tinklą. Tinklas yra koordinačių sistema padalinta į įsivaizduojamus stačiakampius gretasienius, kurių viduje yra taškas turintis savo reikšmę.



1 pav. Tinklas koordinačių sistemoje

Apibrėžkime tinklo ilgį h ir plotį k.

$$h = \frac{(b-a)}{n} \tag{4}$$

$$k = \frac{(d-c)}{m} \tag{5}$$

1.2. Skaitinių metodai difuzijos lygtims spręsti

Šiame poskyryje kalbėsime apie būdus, kaip galima aproksimuoti praeitame poskyryje apibrėžtą difuzijos lygtį. Skaitinių metodų literatūroje [BiJ05] aprašomas baigtinio skirtumų (angl. Finite difference) metodas, kurį taikant, galime aproksimuoti difuzijos lygtį (3). Pagrindinė baigtinio skirtumų metodo formulė dviejų dimensijų lygčiai skaičiuoti yra:

$$\frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{h^2} + \frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{k^2} = 0 \tag{6}$$

Šioje formulėje $u(x_i,y_j)$ žymį tinklo taško reikšmę. h^2 ir k^2 žymi tinklo ilgį ir plotį kaip ir buvo apibrėžtą praeitame poskyryje (4, 5). Šią lygtį (6) galime pertvarkyti, kad vienoje lygybės pusėje gautume taško esančio koordinatėje $u(x_i, y_j)$ reikšmę.

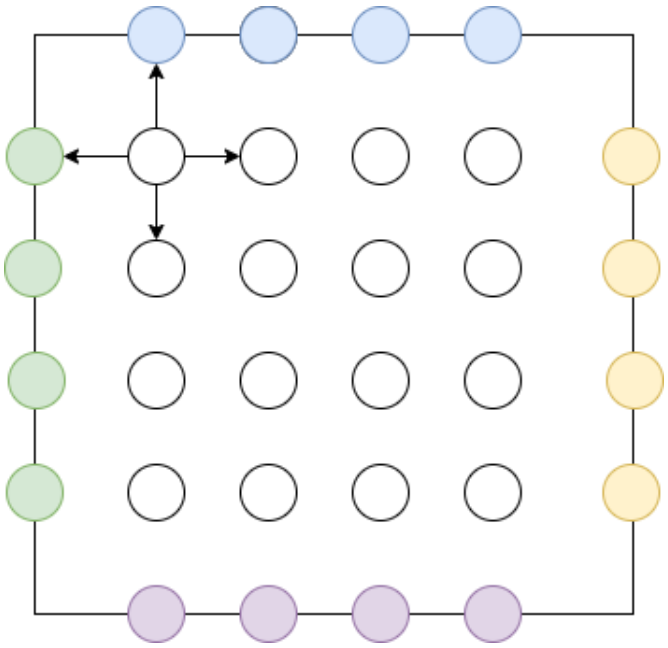
$$\frac{k^2\Big(u(x_{i+1}, y_j) + u(x_{i-1}, y_j)\Big) + h^2\Big(u(x_i, y_{j+1}) + u(x_i, y_{j-1})\Big)}{2(h^2 + k^2)} = u(x_i, y_j) \tag{7}$$

Jeigu skaičiavimuose mums nesvarbus tinklo dydis galime tinklo ilgio ir pločio reikšmes

priskirti vienetui $h = 1$ ir $k = 1$ taip gaudami lygtį

$$\frac{u(x_{i+1}, y_j) + u(x_{i-1}, y_j) + u(x_i, y_{j+1}) + u(x_i, y_{j-1})}{4} = u(x_i, y_j) \tag{8}$$

Iš formulės matome, kad norėdami rasti konkretaus taško reikšmę tinkle mums reikia sudėti kaimyninių taškų reikšmes ir padalinti iš keturių.



2 pav. Tinklo taško reikšmės skaičiavimas. Rodyklės rodo kurias reikšmes reikia sudėti, spalvoti apskritimai rodo skirtingas kraštinių sąlygų reikšmes.

Atlikdami difuzijos skaičiavimus mes įsivaizduojame, kad į skaičiuojamą sritį skirtingomis koncentracijomis patenka kažkokia medžiaga. Medžiagos koncentraciją galime apibrėžti konstantine reikšme arba funkcija. Konstantinės reikšmės atveju krašte esančius tinklo taškus nustatome tik vieną kartą ir jie nekinta. Funkcinės reikšmės atveju kraštinė sąlyga gali keistis skaičiavimo metu. Diagramoje (2 pav.) vaizduojamas tinklas, kur mėlyna, žalia, geltona ir purpurine spalvomis parodytos skirtingos kraštinės sąlygos, o norint gauti vieno tašo reikšmę rodyklėmis pavaizduotos reikmės, kurias reikia sudėti ir padalinti iš keturių.

1.3. Baigtinio skirtumų metodo algoritmas

Šiame poskyryje užrašysime algoritmą kuriuo naudodamiesi atliksime skaičiavimus. Skaičiavimus vykdysime iteratyviai iki tol, kol bus pasiektas norimas skaičiavimų tikslumas. Tikslumą arba norimą paklaidą apibrėšime kaip naujų tinklo reikšmių skirtumą nuo senų tinklo reikšmių (8).

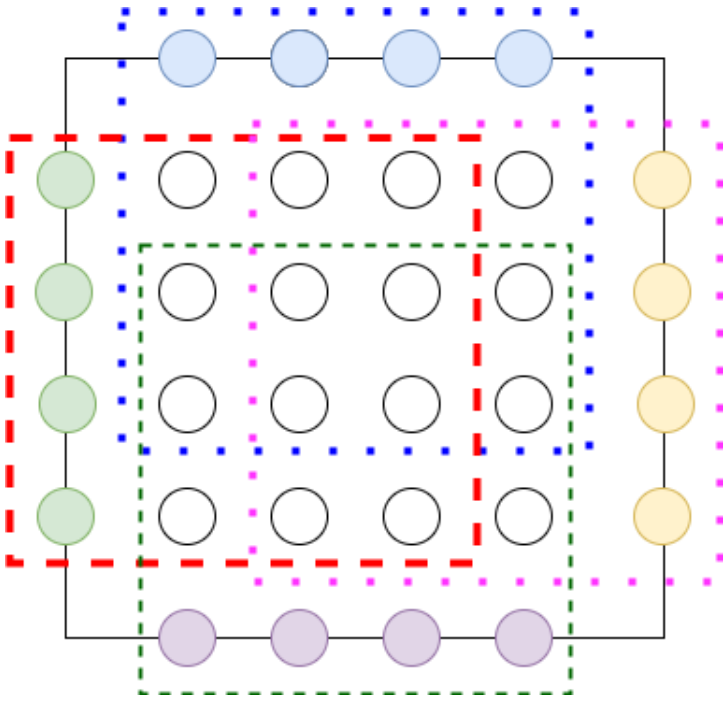
$$maksimali\_paklaida(\frac{naujas\_tinklas - senas\_tinklas}{naujas\_tinklas} * 100) \tag{9}$$

Senas tinklo reikšmes vadinsime tas reikšmes kurios buvo prieš atliekant iteraciją, o naujas tinklo reikšmes laikysime tas kurias gausime atlikę iteraciją. Skaičiavimai bus atliekami taip:

1. Nustatome tinklo kraštines sąlygas.
2. Kiekvienam vidiniam tinklo taškui pritaikome baigtinių skirtumų formulę (7).
3. Palyginame senas tinklo reikšmes su naujomis, kad gautume paklaidą.
4. Jeigu norima paklaida pasiekta, skaičiavimus nutraukiame.
5. Jeigu reikia pakeičiame kraštines sąlygas (funkcija kaip kraštinė sąlyga).
6. Grįžtame į pirmą žingsnį.

1.4. Baigtinio skirtumų metodo algoritmo realizacija

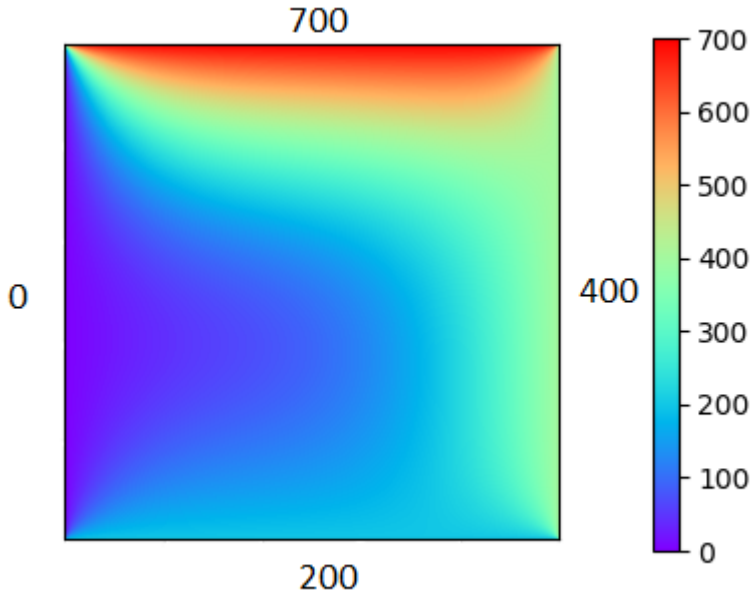
Šiame poskyryje pakalbėsime apie baigtinio skirtumų metodo realizaciją naudojant Python 2 ir NumPy biblioteką. Algoritmas aprašytas praeitame skyriuje sako, kad norint rasti vidinio tinklo taško reikšmę turime sudėti kaimyninius jo taškus (2 pav.). Vietoj to, kad skaičiuotume kiekvieno taško reikšmę atskirai, mes pasinaudosime NumPy biblioteka ir iškarto imsime keturias matricas atitinkančias keturias kaimyninius tinklo taškus (3 pav.) ir visas reikiamas aritmetines operacijas vykdysime matricai, o ne atskiriems tinklo taškams, taip smarkiai padidindami skaičiavimų greitį palyginus su algoritmu įgyvendinti naudojant tik vidinius Python funkcionalumus. Grafike spalvotais punktyriniais kvadratais pavaizduotos keturios kaimyninių reikšmių matricos.



3 pav. Skaičiavimo algoritmo vizualizacija taikant NumPy biblioteką. Punktyrais žymimos atskirtos matricos.

Norėdami pasitikrinti ar gauname tinkamas reikšmes įvykdykime skaičiavimą. Kraštinėms sąlygoms nustatykime keturias konstantines reikšmes $C_1 = 700$, $C_2 = 200$, $C_3 = 0$ $C_4 = 400$, o

tikslumo koeficientą nustatykime $0{,}001\%$. Kraštinės sąlygos nustatytos laisvai pasirenkant reikšmes, tikslumo koeficientas paimtas iš mokslinių darbų [Ram16] [NHS12] ir lygiagrečių skaičiavimų literatūros [Čie01] Atlikus skaičiavimus su šiomis reikšmėmis gauname rezultatą

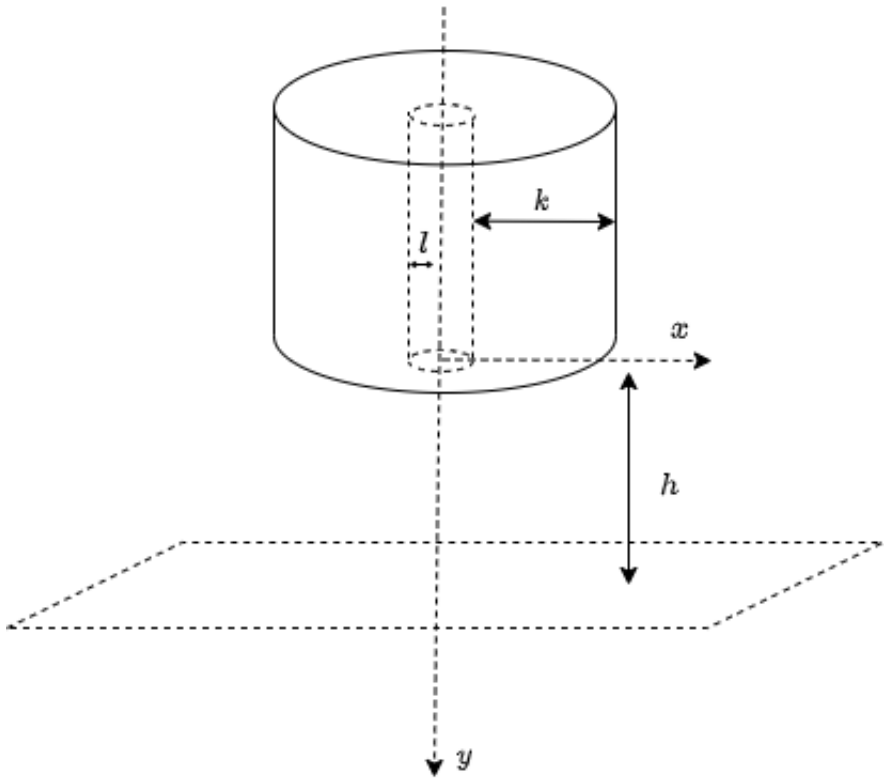


4 pav. Difuzijos modeliavimas su laisvai pasirinktom kraštinėm sąlygom. Vienetinioo dydžio kūnas.

Iš rezultato vizualiai galime matyti, kad mūsų algoritmas veikia korektiškai, sritys kuriose difuzijos kraštinės reikšmės buvo didesnės artėjant į centrą po truputį mažėjo, vidury diagramos susidarė vidutinės reikšmės. Tolesniuose skyriuose naudosime tą patį $0{,}001\%$ paklaidos koeficientą.

2. SECM modelio sudarymas

2.1. Skenuojantis elektrocheminis mikroskopas



5 pav. SECM schema

Skenuojanti elektrocheminė mikroskopija yra metodas, kai naudojant mažos apimties elektrodą, kuriuo teka elektros srovė, yra matuojamos mėginio patalpinto į tirpalą paviršiaus charakteristikos. Elektrodas gali judėti aukštyn, žemyn arba palaikyti vienodą aukšti ir judėti vienoje plokštumoje. Skenuojančiu elektrocheminiu mikroskopu galima matuoti sąveikas tarp skysčio ir kieto kūno, skysčio ir dujų bei skysčio ir skysčio. SECM gali būti naudojamas tirti kieto kūno paviršiaus topografiją ir reaktyvumą. Tyrimai atlikti naudojant SECM padėjo vystyti medžiagų mokslą ir daryti įvairius atradimus nanotechnologijų šakose. Šiuo tyrimu bus siekiama papildyti jau vykdytus Vilniaus universiteto mokslininkų darbus SECM modeliavime.

2.2. SECM modelio sudarymas

Šiame skyriuje sudarysime sudėtingesnį difuzijos modelį, kuris atitiktų VU mokslininkų jau atliktus modeliavimo tyrimus [Ram16]. Apibrėšime konstantines ir funkcines kraštines sąlygas siekdami modeliuoti deguonies difuzijos procesus naudojantis SECM. Modelis susidarys iš keturių pagrindinių elementų:

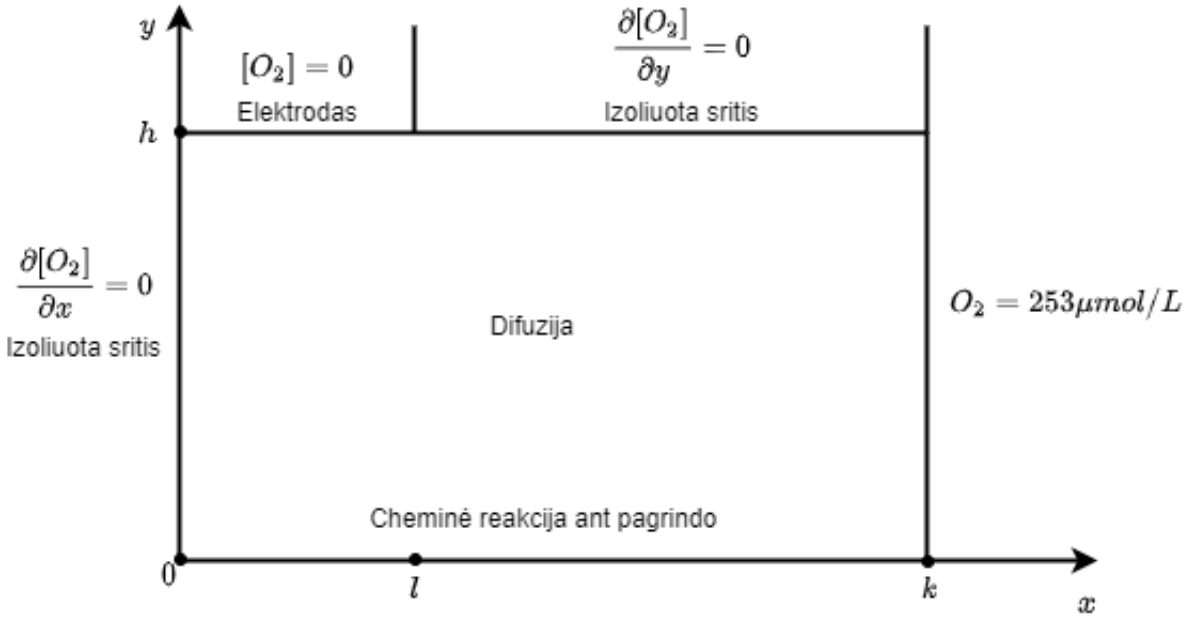
1. Konstanta – tinklo sritis kurioje pastoviu būdu bus pus įleidžiamas deguonis.

2. Izoliuota – tinklo sritis kurioje nevyksta difuzijos procesai.

3. Elektrodas – tinklo sritis, kuri kontaktuoja su elektrodu ir įgauną pastovią konstantinę reikšmę.

4. Reakcijos sritis – tinklo pagrindo sritis kurioje deguonis sąveikauja su eksperimento objektu. Šioje srityje įvyksta cheminės reakcijos todėl reikšmė yra kintama.

Grafiškai modelį galime pavaizduoti taip



6 pav. SECM modelis su kraštinėmis sąlygomis

Modelio ilgiai iš reikšmės yra paimtos iš VU mokslininkų darbo [Ram16]

1. Ilgis – $k = 80\mu m$

2. Plotis – $h = 40\mu m$

3. Elektrodo ilgis – $l = 5\mu m$

4. Deguonies padavimas – $O_2 = 253\mu mol/L$

Sudarytame modelyje elektrodas yra priartėjęs prie pagrindo, kairys modelio šonas ir dalis viršaus yra izoliuota ir kairėje pusėje yra įleidžiamas deguonis. Grafiko apačioje vyksta cheminės reakcijos. Izoliuotoje srityje difuzija nevyksta, šią sritį modeliuosime funkcijos išvestinę prilyginę nuliui (9).

$$\frac{\partial C}{\partial y}(x, y = 0) = 0 \tag{10}$$

Izoliuotos srities aproksimacijai naudosime patį paprasčiausią aproksimacijos būdą, kuris yra aprašytas literatūroje [BiJ05] (10).

$$f'(x) = \frac{f(x + h) - f(x)}{h} \tag{11}$$

Pagrindui modeliuoti naudosime dvi funkcijas:

$$f(x) = k\left|\sin\frac{\pi x}{a}\right| \tag{12}$$

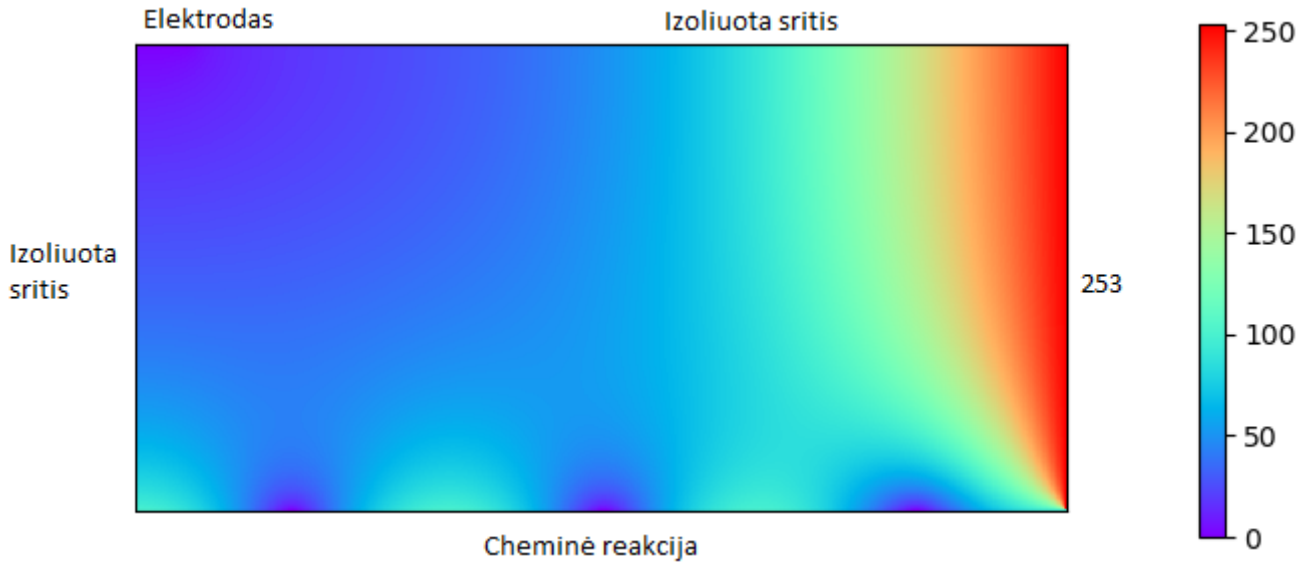
Šioje lygtyje kintamasis k yra greičio konstanta, skaičiavimuose laikysime, kad $k = 100$.

$$f(x) = \begin{cases} k, & \text{x } \varepsilon \text{ ląstelė.} \\ 0, & \text{x } \varepsilon \text{ tarpas.} \end{cases} \tag{13}$$

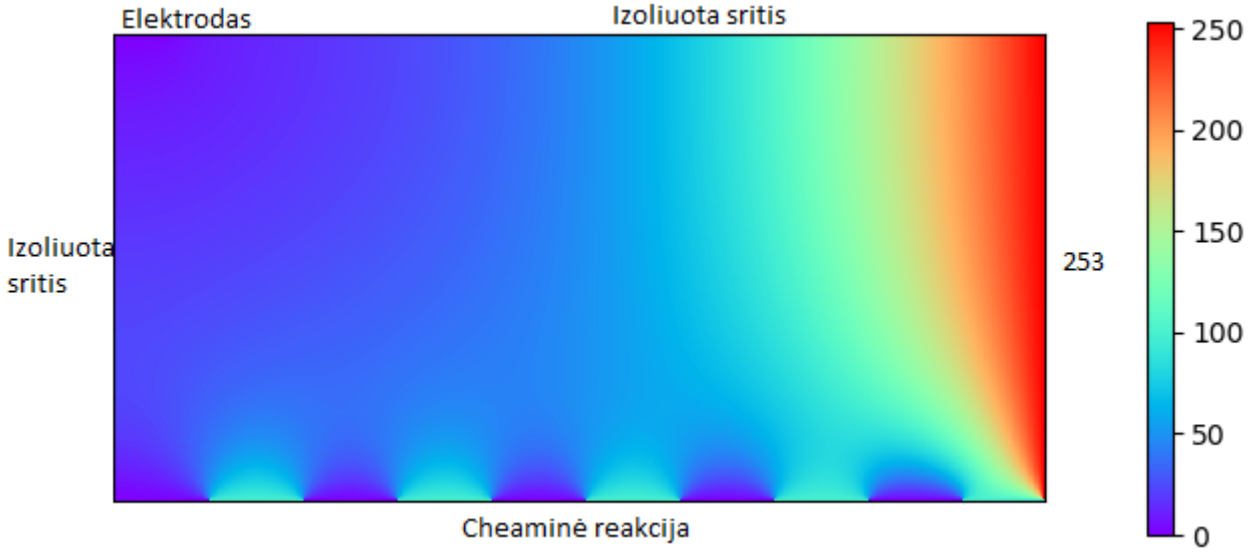
Antroji lygtis (12) yra laiptinė, kuri pagrindą padalina į sritis, kurių reikšmės bus 0 arba k. Taikant šią funkciją prilyginsime $k = 100$. Atliekant šiuos skaičiavimus taikysime anksčiau aprašytą skaitinių metodų lygtį, kuri atsižvelgia į modelio dydį (7). Skaičiavimai bus vykdomi tol kol bus pasiekta 0.001% paklaidos riba. Tinklui nustatysime 320000 taškų, arba 400 taškų pločio ir 800 taškų ilgį.

2.3. Modeliavimo rezultatai

Atlikus skaičiavimus gauname dvi diagramas:



7 pav. Sinusoidinė funkcija



8 pav. Laiptinė funkcija

Vizualiai galime matyti, kad skaičiavimai įvyko sėkmingai. Grafikų viršuje ir kairėje pusėje matome izoliuotą sritį kurioje nevyko difuzijos procesas, todėl tos dalys nedarė įtakos deguonies pasiskirstymui. Kairėje pusėje viršuje matome modeliuotą elektrodo sritį kurioje nebuvo deguonies, tačiau vykstant difuzijai deguonies sritis priartėjo prie elektrodo srities. Apatinėje dalyje matome, kaip vyko deguonies pasiskirstymas taikant sinusoidinę (7 pav.) ir laiptinę (8 pav.) funkcijas. Funkcijos modeliuoti pagrindo sritį buvo pasirinktos laisvai siekiant parodyti, kad modelis veiktų korektiškai nurodant tikslesnes, labiau realius procesus atitinkančias funkcijas. Lokalioje aplinkoje skaičiavimai truko 49 minutes 15 sekundžių, o MIF klasteryje skaičiavimai truko 1 valandą 35 minutes ir 26 sekundes. Kuriant sudėtingesnius modelius, kuriuose būtų daugiau tinklo taško ir sudėtingesnės kraštinės sąlygos skaičiavimai truktų dar ilgiau, todėl yra prasminga ieškoti būdų kaip pagreitinti skaičiavimus. Trečiame dėstymo skyriuje naudosime modelį su sinusoidine funkcija realizuodami lygiagrečius algoritmus ir vykdydami eksperimentus siekdami pagreitinti skaičiavimus.

3. Lygiagretūs algoritmai difuzijos lygtims spręsti

3.1. Lygiagrečių skaičiavimų teorija

Šiame skyriuje realizuosime lygiagrečius algoritmus siekdami pagreitinti modelio skaičiavimą. Prieš pradėdami algoritmų realizavimą turime žinoti, kokio skaičiavimų pagreitėjimo galime tikėtis. Lygiagrečių skaičiavimų teorijoje yra formulė (14) pasakanti galimą skaičiavimų pagreitėjimą. Ši formulė vadinasi Amdahlo dėsniu.

$$S_p = \frac{1}{(1-r) + \frac{r}{p}} \tag{14}$$

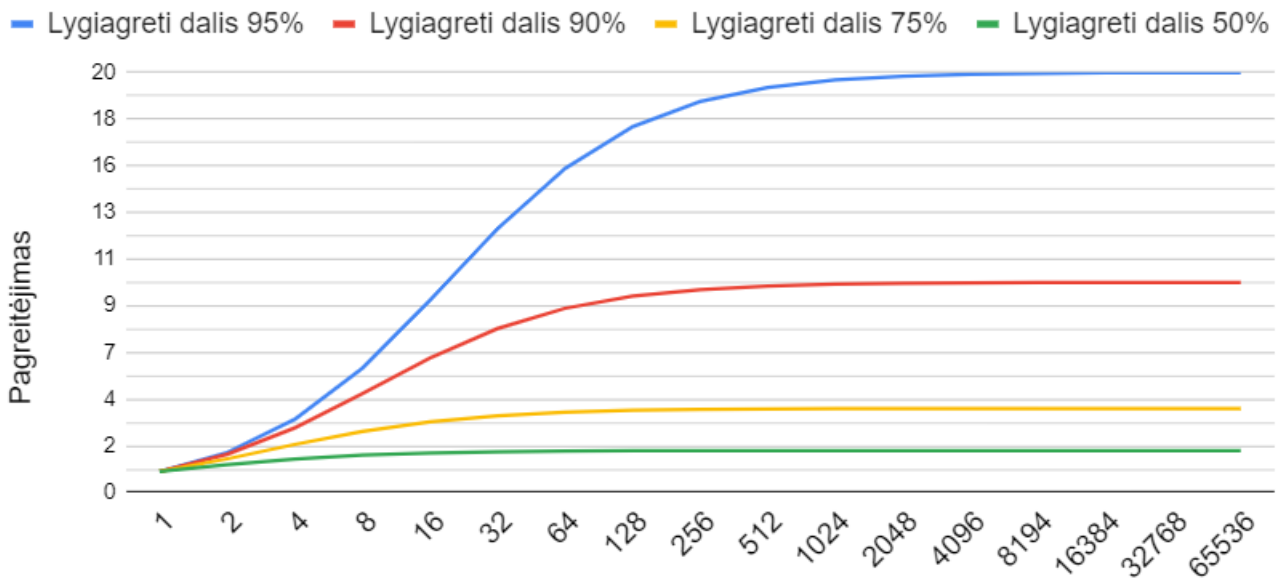
Šioje formulėje:

- S_p – Teorinis visos užduoties vykdymo pagreitėjimas.
- r – užduoties dalis, kurią galima apskaičiuoti lygiagrečiai.
- $1 - r$ – užduoties dalis, kurios negalima skaičiuoti lygiagrečiai.
- p – branduolių skaičius.

Naudojantis Amdahlo dėsniu ir žinant kuri programos dalis gali būti skaičiuojama lygiagrečiai galime pasakyti koks yra maksimalus teorinis pagreitėjimas su tam tikru branduolių skaičiumi. Amdahlo dėsnis taip pat teigia, kad didžiausias įmanomas pagreitėjimas yra 20 kartų greitesnis veikimas negu vykdant užduotį naudojant vieną branduolį:

$$\frac{1}{1-r} = 20 \tag{15}$$

Iš Amdahlo dėsnio gauname grafiką; pagreitėjimas priklauso nuo užduoties dalies kurią galimą vykdyti lygiagrečiai (9 pav.).



9 pav. Amdahlo dėsnis

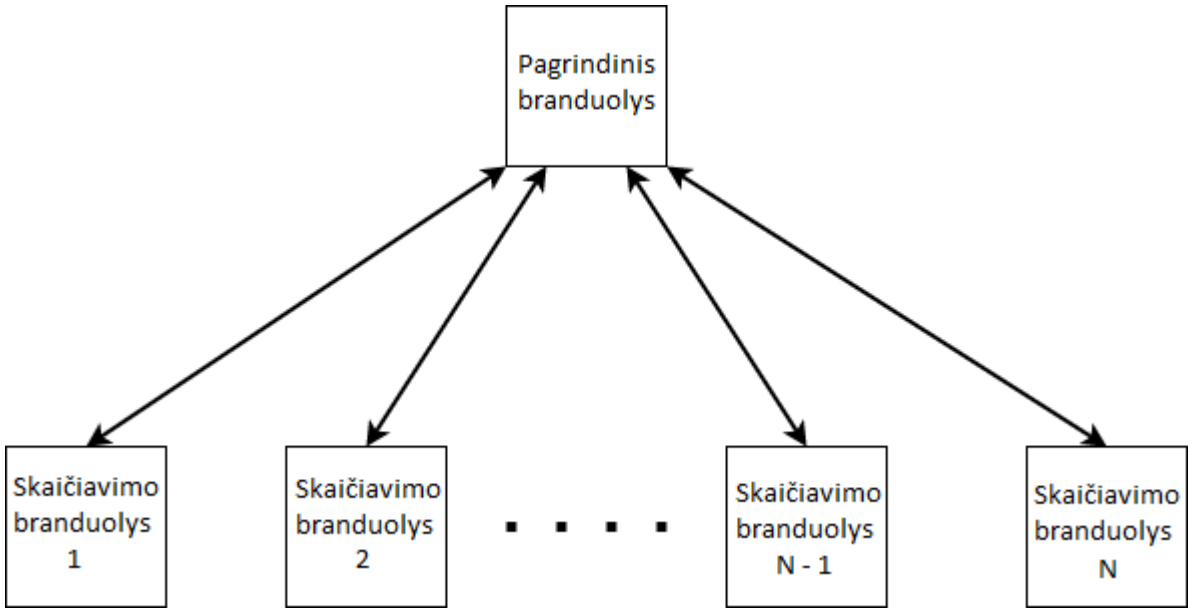
3.2. Lygiagrečių algoritmų realizacijos

Šiame skyriuje aptarsime lygiagrečius algoritmus, kuriuos naudosime paspartinti difuzijos procesų skaičiavimams ir architektūrą, kurią naudosime žinutėms siųsti. Darbe naudosime du lygiagrečius algoritmus aprašytus literatūroje [Čie01]. Algoritmai bus realizuojami Python 2 programavimo kalbą ir MPI realizacija mpi4py.

Skaičiavimams vykdyti naudosime paprastą architektūrą sistemos branduolius padalinę į dvi esmines grupes:

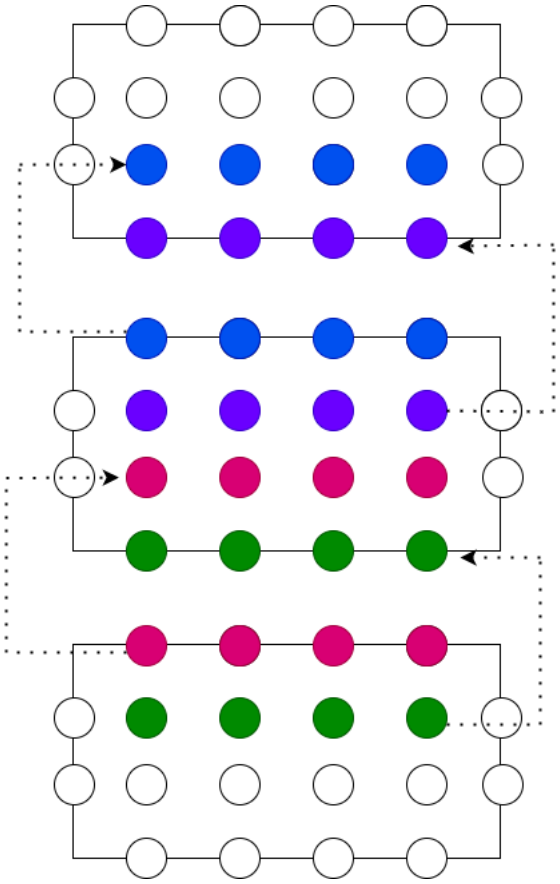
1. Pagrindinis branduolys – jo paskirtis yra išsiųsti atitinkamus tinklo taškus skaičiavimo branduoliams ir sugrupuoti rezultatus gautus skaičiavimo branduoliams baigus darbą.

2. Skaičiavimo branduoliai – jų paskirtis yra gauti pradinę informaciją iš pagrindinio branduolio, atlikti skaičiavimus iki kol bus pasiekta reikiama paklaida, išsiųsti tarpinius duomenis kaimyniniams skaičiavimo branduoliams ir grąžinti suskaičiuotus rezultatus.



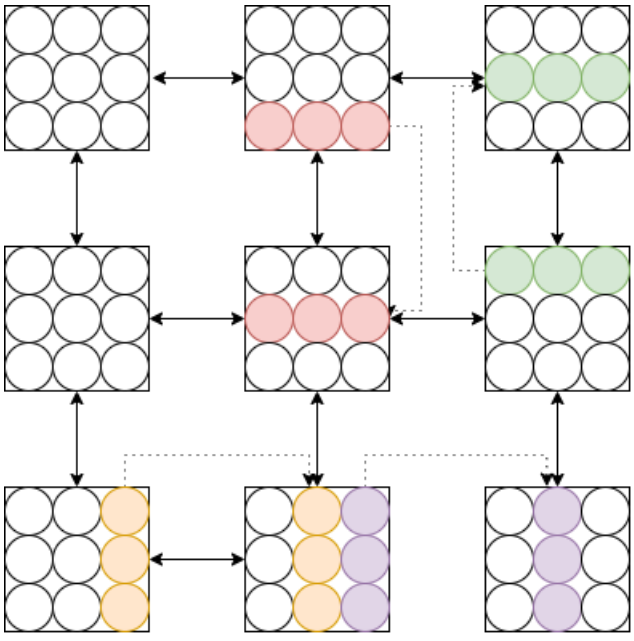
10 pav. Žinučių siuntimo architektūra

Pirmas algoritmas, kurį realizuosime yra „vienmatis diskrečiojo tinklo padalijimas“ [Čie01]. Tinklą padalijame į N lygių dalių, kur N yra kiek naudosime skaičiavimo branduolių. Pagrindinis branduolys perduoda pradinę informaciją skaičiavimo branduoliams. Kiekvienas skaičiuojantis branduolys išsiunčia kaimyniniams branduoliams pirmą ir paskutinę savo tinklo eilutę. Tokiu būdu kiekvienas branduolys gauna papildomų fiktyviųjų duomenų, kuriuos naudoją skaičiavimams kaip kraštinę sąlygą. Po kiekvienos iteracijos vyksta duomenų apsikeitimas ir naujų fiktyvių kraštinių sąlygų sudarymas. Skaičiavimai ir apsikeitimai vyksta tol, kol visi skaičiavimo branduoliai pasiekia reikiamą tikslumą. Vienam branduoliui pasiekus reikiamą tikslumą jis išsiunčia rezultatus į pagrindinį branduolį. Jeigu kaimyniniai branduoliai dar nebaigė darbo jis ir toliau siunčia jiems tą pačią informaciją iki kol kaimyniniai branduoliai taip pat pasieks reikiamą paklaidą ir baigs darbą.



11 pav. Vienmatis lygiagretusis algoritmas.

Antras algoritmas, kurį įgyvendinsime yra „dvimatis tinklo padalijimo algoritmas“. Algoritmas skirsis nuo vienmačio tuo, kaip padalinsime tinklą į lygias dalis. Tinklas bus dalinamas NxM lygių stačiakampių. Kaip ir vienmatyje algoritme po kiekvienos iteracijos skaičiuojantys branduoliai apsikeis ir sudarys naujas fiktyvias kraštines sąlygas.



12 pav. Dvimatis lygiagretusis algoritmas.

3.3. Skaičiavimų aplinkos

Skaičiavimai bus vykdomi dviejuose aplinkose: lokalioje ir MIF paskirstytų skaičiavimų tinklo aplinkoje.

Lokali aplinka:

- Procesorius – ADM Ryzen 5 1600X, 6 branduoliai
- Operatyvioji atmintis – 32GB DDRM4 1330Hz
- Operacinė sistema – Linux Ubuntu 20.04 LTS, naudojant VirtualBox 6.1.6
- Kietasis diskas – HDD 500GB

MIF paskirstytų skaičiavimų centras tinklas:

- Procesorius – 2x Intel Xeon X5650 2.66GHz = 12 cores
- Operatyvioji atmintis – 24 GB
- Operacinė sistema – Linux Debian
- Kietasis diskas – 160GB

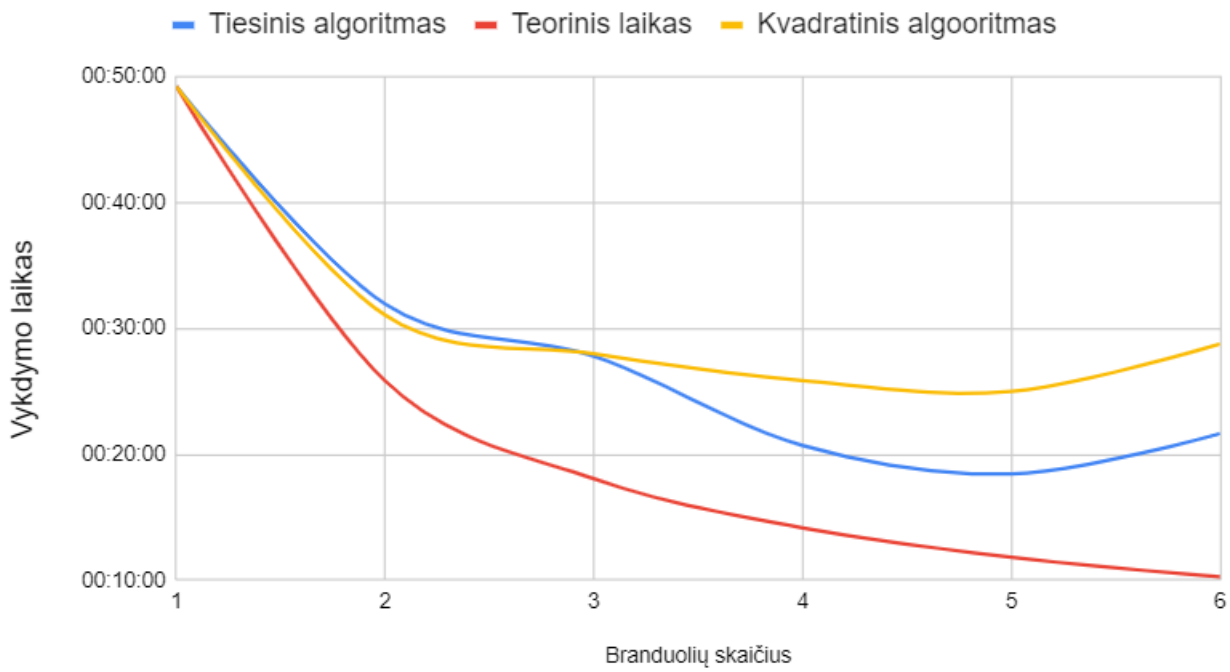
3.4. Skaičiavimų rezultatai

Pirma skaičiavimus atlikome lokalioje aplinkoje. Skaičiavimai buvo atlikti taikant sinusoidinį pagrindo modelį. Skaičiavimų parametrai buvo pasirinkti tokie kaip antrajame skyriuje:

- Tinklo ilgis – 800 taškų.
- Tinklo plotis – 400 taškų.
- Tikslumas – 0,001%
- Vykdymo laikas – trijų skaičiavimų laikų aritmetinis vidurkis.

Atlikę skaičiavimus gauname (13 pav.), kad nenaudojant lygiagrečių algoritmų skaičiavimai trunka 49 minutes 15 sekundžių. Pradėjus taikyti lygiagrečius algoritmus skaičiavimo laikas sėkmingai pradėjo mažėti. Grafike raudona linija žymi teoriškai greičiausią įmanomą laiką laikant, kad lygiagreti uždavinio dalis yra 95 procentai. Ta pati teorinė reikšmė bus ir kitose šio skyriaus diagramose. Taikant vienmatį algoritmą pavyko skaičiavimo greitį sumažinti iki 18 minučių ir 26 sekundžių naudojant 5 skaičiavimo branduolius, o taikant dvimatį algoritmą skaičiavimo greitį pavyko sumažinti iki 25 minučių ir 52 sekundžių naudojant 4 skaičiavimo branduolius. Taigi gavome, kad vienmatis algoritmas yra greitesnis negu dvimatis. Tai atitinka literatūroje aprašomą algoritmų įgyvendinimą, nes skaičiavimų pagreitėjimas naudojant dvimatį algoritmą prasidėtų tik nuo 9 skaičiavimo branduolių. Grafike matome, kad nuo 6 branduolių ribos skaičiavimai pradeda lėtėti.

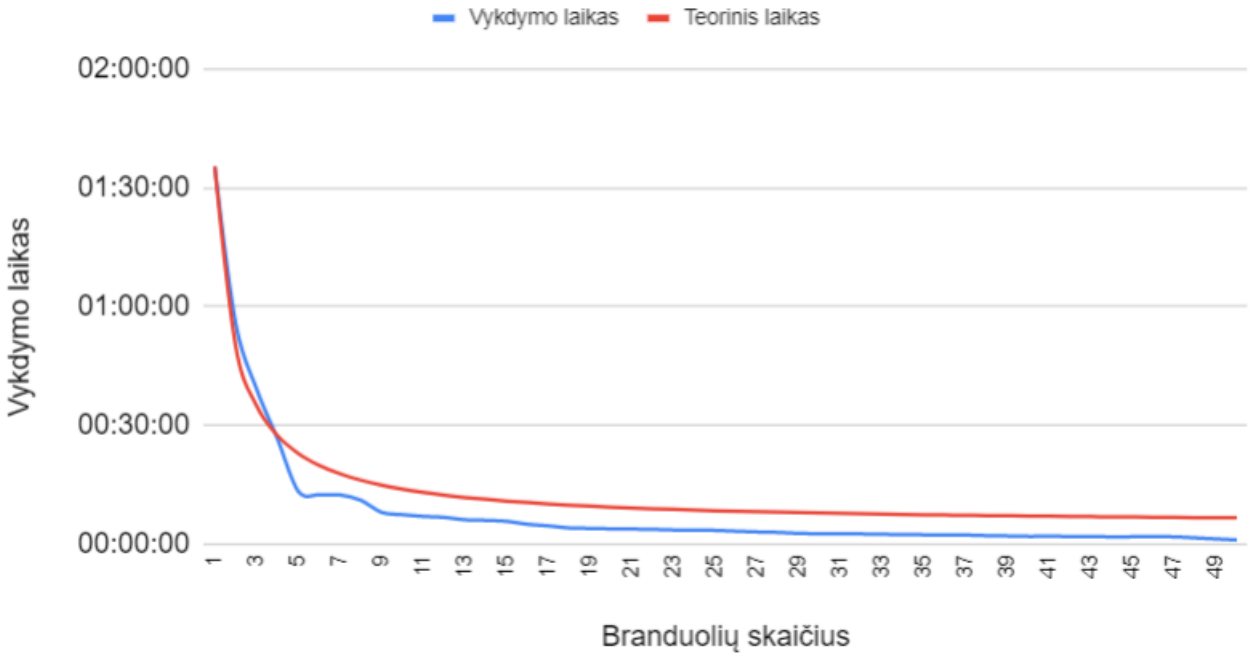
Taip nutinka dėl to, nes sistemoje nepakanka laisvų branduolių, todėl priskyrus daugiau branduolių negu yra sistemoje vienas branduolys pradeda atlikti dvi užduotis, todėl skaičiavimai sulėtėja.



13 pav. Skaičiavimo laikas lokalioje aplinkoje

Vykdant skaičiavimus MIF klasteryje naudosimės tais pačiais skaičiavimo modeliais ir parametrais. Atliekant skaičiavimus naudojant tik vieną branduolį skaičiavimo laikas yra 1 valanda 35 minutės 26 sekundės. Toks skaičiavimo greitis yra tikslus, nes lokalios aplinkos vieno branduolio pajėgumas yra didesnis negu MIF klasterio. Klasterio privalumai yra galimybė naudoti didelį branduolių skaičių.

Atliekant skaičiavimus taikant vienmatį algoritmą gauname smarkų sulėtėjimą iki 7 minučių 23 sekundžių naudojant 10 klasterio branduolių, toliau stebime tolygų lėtėjimą iki 1 minutės 30 sekundžių naudojant 40 klasterio branduolių ir paskui skaičiavimo greitis praktiškai nebekinta.

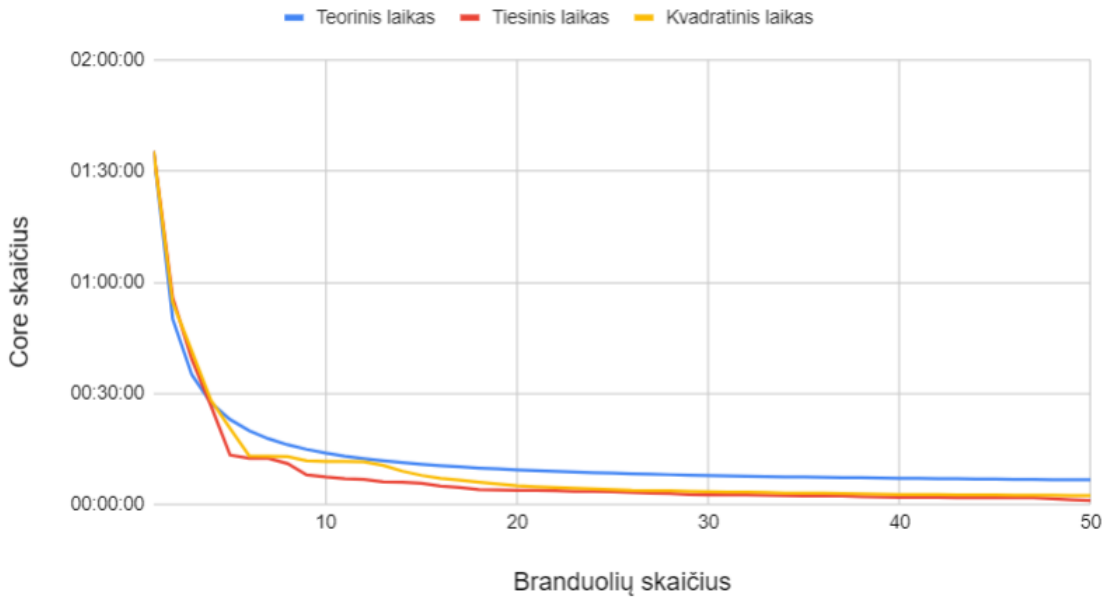


14 pav. Skaičiavimo greitis taikant vienmatis algoritmą MIF klasteryje

Panašią pagreitėjimo kreivę matome ir taikydami dvimatį algoritmą, skaičiavimai smarkiai sulėtėja iki 11 minučių 47 sekundžių naudojant branduolių, toliau lėtėja tolygiai iki 2 minučių 20 sekundžių naudojant 45 branduolius, toliau skaičiavimai praktiškai nebegreitėja.

15 pav. Skaičiavimo greitis taikant dvimatį algoritmą MIF klasteryje

Taigi matome, kad vienmatis algoritmas buvo greitesnis tiek vykdant skaičiavimus lokalioje aplinkoje tiek ir klasteryje.



16 pav. Bendras MIF klasterio skaičiavimų grafikas

Eksperimentiniai rezultatai neatitinka teorijoje aprašytų ir tikėtųsi rezultatų, taip galėjo įvyktį dėl NumPy bibliotekos.

Rezultatai

Darbe buvo pasiekti šie rezultatai:

1. Apibrėžtos ir paaiškintos matematinės lygtys ir skaitinių metodų algoritmai skirti difuzijos lygčių skaičiavimui.
2. Sudarytas SECM naudojanti Vilniaus universiteto mokslininkų darbu [Ram16]
3. Apibrėžti ir paaiškinti lygiagretūs algoritmai skirti difuzijos lygčių skaičiavimo pagreitinimui.
4. Surastas ir realizuotas optimalus vienmatis lygiagretus algoritmas.
5. Atlikti eksperimentai lokalioje aplinkoje ir nustatyta, kad optimaliam skaičiavimui negalima naudoti daugiau branduolių negu yra sistemoje ir surastas optimalus lygiagretus algoritmas – vienmatis.
6. Atlikti eksperimentai MIF klasteryje ir nustatymas optimalus vienmatis algoritmas skaičiavimams MIF klasteryje

Išvados

Darbe pavyko sėkmingai identifikuoti matematines lygtis ir skaitinius metodus skirtus difuzijos procesui modeliuoti. Darbe atkartotas VU mokslininkų darbe[Ram16] taikytas SECM modelis. Pavyko sudaryti lankstų modelį, kuriame būtų galima skaičiuoti ir paralelizuoti kitus modelius su skirtingomis kraštinėmis sąlygomis.

Naudojantis literatūroje aprašytais algoritmais[Čie01] buvo sėkmingai sutrumpintas skaičiavimų laikas. Vykdant skaičiavimus lokalioje aplinkoje modelio vykdymo laikas naudojant 1 branduolį buvo sutrumpinas nuo 49 minučių ir 15 sekundžių iki 10 minučių ir 16 sekundžių naudojant 5 branduolius. Mif klasteryje pavyko skaičiavimus pagreitinti nuo 1 valandos 35 minučių ir 26 sekundžių iki 1 minutės 30 sekundžių. Abiem skaičiavimo atvėjais gavome, kad optimalus lygiagretus algoritmas yra vienmatis algoritmas, kuriame tinklas yra dalijamas juostomis.

Ateityje būtų galima analizuoti NumPy optimizacijų įtaką skaičiavimams ir skirtingus skaitinius metodus bei lygiagrečius algoritmus.

Literatūra

[BiJ05] Richard L. Burden ir ir J. Douglas Faires. Numerical analysis eighth edition, 2005.

[Čie01] Raimondas Čiegis. Lygiagretieji algoritmai, 2001.

[iMic05] Barry Wilkinson ir Michael Allen. Parallel programming techniques and applications using networked workstations and parallel computing second edition, 2005.

[mpi20] mpi4py. Https://mpi4py.readthedocs.io/en/stable/tutorial.html, 2020.

[NHS12] Sameh Ahmed N.H.Sweilam H.M.Moharram. On the parallel iterative finite difference algorithm for 2-d poisson's equation with mpi cluster, 2012.

[Num20] NumPy. Https://numpy.org/doc/, 2020.

[Pyt20] Python. Https://docs.python.org/2/index.html, 2020.

[Ram16] Feliksas Ivanauskas Inga Morkvenaite-Vilkonciene Rokas Astrauskas Arunas Ramanavicius. Modelling of scanning electrochemical microscopy at redox competition mode using diffusion and reaction equations, 2016.

[uni20] Vilniaus universitetas. Skaitmeninių tyrimų ir skaičiavimų centras, 2020.