



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
MATEMATINIO MODELIAVIMO KATEDRA

Audrius Grikielis

N KŪNŲ PROBLEMOS TAIKYMAI IR JOS SPRENDIMO
ALGORITMŲ ANALIZĖ

APPLICATIONS OF N-BODY PROBLEM AND ITS SOLUTION
ALGORITHMS ANALYSIS

Baigiamasis magistro darbas

Technomatematikos studijų programa, valstybinis kodas 621G16001

Matematikos studijų kryptis

Vilnius, 2017

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
MATEMATINIO MODELIAVIMO KATEDRA

TVIRTINU
Katedros vedėjas

(Parašas)

(Vardas, pavardė)

(Data)

Audrius Grikielis

N KŪNŲ PROBLEMOS TAIKYMAI IR JOS SPRENDIMO
ALGORITMŲ ANALIZĖ

APPLICATIONS OF N-BODY PROBLEM AND ITS SOLUTION
ALGORITHMS ANALYSIS

Baigiamasis magistro darbas

Technomatematikos studijų programa, valstybinis kodas 621G16001

Matematikos studijų kryptis

Vadovas

(Pedag. vardas, pavardė)

(Parašas)

(Data)

Lietuvių kalbos konsultantas

(Pedag. vardas, pavardė)

(Parašas)

(Data)

Vilnius, 2017

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

MATEMATINIO MODELIAVIMO KATEDRA

Matematikos studijų kryptis

Technomatematikos studijų programa, valstybinis kodas 621G16001

TVIRTINU

Katedros vedėjas

(Parašas)

(Vardas, pavardė)

(Data)

BAIGIAMOJO MAGISTRO DARBO

UŽDUOTIS

..... Nr.

Vilnius

Studentui Audriui Griekiečiui,

baigiamojo darbo tema: „N kūnų problemos taikymai ir jos sprendimo algoritmų analizė“, patvirtinta 2017 m. kovo 17 d. dekanų potvarkiu Nr. 160fm. Baigiamojo darbo užbaigimo terminas 2017 m. birželio 12 d.

BAIGIAMOJO DARBO UŽDUOTIS:

Susipažinti su N kūnų problema, jos taikymo sritimis ir panaudojimo galimybėmis. Sudaryti N kūnų sistemos dinamiką aprašantį matematinį modelį. Ištirti tiesioginį ir Barnes'o-Hut'o N kūnų problemos sprendimo algoritmus. Pasirinkta programavimo kalba realizuoti tiriamųjų algoritmų lygiagrečiąsias versijas. Palyginti lygiagrečiųjų algoritmų spartumą bei efektyvumą.

Vadovas

.....
(Parašas)

.....
(Pedag. vardas, pavardė)

Užduotį gavau

.....
(Parašas)

.....
(Vardas, pavardė)

.....
(Data)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Audrius Grikielis, Nr. 20114326

Fundamentinių mokslų fakultetas

Technomatematika, TMfm-15

BAIGIAMOJO DARBO SAŽININGUMO DEKLARACIJA

2017 m. Birželio 5 d.

Patvirtinu, kad mano baigiamasis darbas tema „N kūnų problemos taikymai ir jos sprendimo algoritmų analizė“ patvirtintas 2017 m. kovo 17 d. dekanų potvarkiu Nr. 160fm, yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Mano darbo vadovas prof. habil. dr. Raimondas Čiegis.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

(Parašas)

(Vardas ir pavardė)

Vilniaus Gedimino technikos universitetas

Fundamentinių mokslų fakultetas

Matematinio modeliavimo katedra

ISBNB ISSN

Egz. sk.

Data ... - ... - ...

Technomatematikos studijų programos baigiamasis magistro darbas

Pavadinimas **N kūnų problemos taikymai ir jos sprendimo algoritmų analizė**

Autorius **Audrius Grikietis**

Vadovas prof. habil. dr. **Raimondas Čiegis**

Kalba



Lietuvių



Anglų

Anotacija

Baigiamajame magistro darbe tyrinėjama N kūnų problema, bei jos praktiniai taikymai molekuliniame modeliavime ir astrofizikoje. Sudarytas matematinis modelis, kuriuo aprašoma fundamentaliosiomis jėgomis tarpusavyje sąveikaujančių kūnų sistemos dinamika. Studijuojamas žvaigždžių spiečių ir elipsinių galaktikų simuliacijų, pradinių sąlygų generavimui taikomas Plummer'io modelis. Pateikti du N kūnų problemos sprendimo algoritmai: tiesioginis ir Barnes'o-Hut'o. Aprašyti VGTU klasteryje „Vilkas“ atliktų skaičiuojamųjų eksperimentų rezultatai. Eksperimentai įgyvendinti naudojant šio darbo autoriaus sukurtą modeliavimo įrankį „Butterfly“. Darbe apžvelgti istoriniai aspektai nulėmę N kūnų problemos atsiradimą.

Darbą sudaro 9 dalys: įvadas, istorinė apžvalga, matematinis modelis, N kūnų problemos taikymai, sprendimo algoritmai, N kūnų sistemos modeliavimo įrankis Butterfly, skaičiuojamieji eksperimentai, išvados, literatūra.

Darbo apimtis – 43 p. teksto be priedų, 29 iliustr., 7 lent., 30 bibliografinių šaltinių.

Atskirai pridedami darbo priedai.

Prasminiai žodžiai: N kūnų problema, empirinis jėgų lauko modelis, Plummer'io modelis, Barnes'o-Hut'o algoritmas, hierarchinės duomenų struktūros, algoritmų sudėtingumas, lygiagretieji algoritmai, uždavinio stabilumas.

Vilnius Gediminas Technical University
Fundamental Sciences Faculty
Mathematical Modelling Department

ISBN ISSN
Copies No.
Date ... - ... - ...

Technomathematics study programme master thesis.

Title: **Applications of N-body Problem and Its Solution Algorithms Analysis**

Author **Audrius Grikielis** Academic supervisor prof. hab. dr. **Raimondas Čiegis**

Thesis language

☒

Lithuanian

☐

Foreign (English)

Annotation

The master thesis investigates N-body problem and it's practical applications in molecular modeling and astrophysics. Derivation of the mathematical model, which describes dynamics of the system of bodies, that interacts among each other with fundamental forces is introduced. Plummer model used in star clusters and galaxy simulations for generating initial system's conditions is presented. Two N-body problem solution algorithms are provided: brute-force and Barnes-Hut. Efficiency and acceleration coefficients of latter algorithm's parallel versions are determined. All experiments are performed in Vilnius Gediminas Technical University computers cluster "Vilkas", while using "Butterfly" application created by the thesis author. Review of the historical aspects that predetermined emergence of the N-body problem is accomplished.

Structure: introduction, historical review, mathematical model, applications of N-body problem, solution algorithms, "Butterfly" modeling tool for the N-bodies system, computational experiments, conclusion, references.

Thesis consists of: 43 p. text without appendixes, 29 pictures, 7 tables, 30 bibliographical entries.

Appendixes included.

Keywords: N-body problem, empirical force field model, Plummer model, Barnes-Hut algorithm, hierarchical data structures, complexity of algorithms, parallel algorithms, numerical stability.

Turinys

1	Įvadas	10
2	Istorinė apžvalga	11
3	Matematinis modelis	14
4	N kūnų problemos taikymai	17
4.1	Molekulinis modeliavimas	17
4.1.1	Empirinis jėgų lauko modelis	18
4.1.2	Vidinių koordinatų sistema	21
4.2	Astrofizika	23
4.2.1	Plummer'io modelis	24
5	Sprendimo algoritmai	28
5.1	Tiesioginis algoritmas	28
5.2	Barnes'o-Hut'o algoritmas	30
6	N kūnų sistemos modeliavimo įrankis Butterfly	36
6.1	Programos nustatymai	38
7	Skaiciuojamieji eksperimentai	40
7.1	Empiriniai sudėtingumo įverčiai	40
7.2	Lygiagretieji skaičiavimai	42
7.3	Uždavinio stabilumas	47
7.4	Barnes'o-Hut'o ribinė sąlyga	48
8	Išvados	50
	Literatūra	51
	Priedai	53

Iliustracijų sąrašas

1	Isaac'as Newton'as (1643–1727 m.) [2]	11
---	---	----

2	Eksperimentai su N kūnų sistemomis [5]	13
3	Elementarusis poslinkis	14
4	Judėjimas vienmatėje erdvėje	16
5	Ryšių tempimo potencialas	19
6	Lenkimo kampo potencialas	19
7	Torsinio kampo potencialas (1-asis atvejis)	20
8	Torsinio kampo potencialas (2-asis atvejis)	20
9	Lenardo-Džonso potencialas	21
10	Hablo galaktikų klasifikavimo schema [16]	24
11	Kūnų radialiniai atstumai	27
12	Tiesioginis N kūnų problemos sprendimo algoritmas	30
13	Medžio duomenų struktūra [21]	31
14	Agile metodologijos etapų schema	37
15	Funkcijos $f(N)$ konvergavimas	41
16	Tiesioginio lygiagrečiojo algoritmo vykdymo laikas	43
17	Tiesioginio lygiagrečiojo algoritmo spartinimo koeficientas	43
18	Tiesioginio lygiagrečiojo algoritmo efektyvumo koeficientas	44
19	Tiesioginio lygiagrečiojo algoritmo laiko sąnaudų modelis	44
20	Barnes'o-Hut'o lygiagrečiojo algoritmo laiko sąnaudų modelis	45
21	Barnes'o-Hut'o lygiagrečiojo algoritmo vykdymo laiko sąnaudos	46
22	Barnes'o-Hut'o lygiagrečiojo algoritmo spartinimo koeficientas	46
23	Barnes'o-Hut'o lygiagrečiojo algoritmo efektyvumo koeficientas	46
24	Pilnutinė sistemos energija	47
25	Pilnutinė sistemos energija.	48
26	Paklaidos priklausomybė nuo parametro θ	49
27	Skaiciavimų laiko priklausomybė nuo parametro θ	49
28	Programos „Butterfly“ objektinis modelis	53
29	Programos „Butterfly“ paketų diagrama	53

Lentelių sąrašas

1	Tiesioginio nuosekliojo algoritmo vykdymo laiko sąnaudos	40
2	Barnes'o-Hut'o nuosekliojo algoritmo vykdymo laiko sąnaudos	41
3	Tiesioginio lygiagrečiojo algoritmo vykdymo laiko sąnaudos	42

4	Tiesioginio lygiagrečiojo algoritmo spartinimo ir efektyvumo koeficientai	44
5	Barnes'o-Hut'o lygiagrečiojo algoritmo vykdymo laiko sąnaudos	45
6	Barnes'o-Hut'o lygiagrečiojo algoritmo spartinimo ir efektyvumo koeficientai . .	46
7	Barnes'o-Hut'o algoritmo paklaidos ir vykdymo laiko priklausomybė nuo θ	49

1. Įvadas

Moksliniai eksperimentai yra vienas iš būdų tirti tam tikrą reiškinį. Jais remiantis gali būti tikrinamos hipotezės, formuluojami dėsniai ir teorijos. Kadangi eksperimentas yra atliekamas pagal kontroliuojamas ir atkuriamas sąlygas, stebėtojas gali gauti tikrą jį dominančio proceso reprezentaciją. Taigi, eksperimentą būtų galima apibrėžti taip – tai mokslinė procedūra, atliekama su tikslu padaryti atradimą, patikrinti hipotezę ar pademonstruoti jau žinomą faktą.

Tobulėjant kompiuteriams, augant jų skaičiuojamajai galiai, atsirado galimybė reiškinius tyrinėti naudojant kompiuterinę simuliaciją, dar vadinamą virtualiuoju eksperimentu. Simuliacijos tikslas yra kiek įmanoma tiksliau imituoti realaus reiškinio sistemą. Kitais žodžiais tariant, virtualaus eksperimento metu realybė yra modeliuojama reprezentatyvia sistema, kuria aprašomi tyrinėjamų reiškinų dėsningumai. Kompiuterinės simuliacijos yra naudojamos ne tik kaip mokslinio tyrimo įrankiai, jos taikomos ir kitose srityse: mašinoms, lėktuvams, ar kitiems mechaniniams įrenginiams modeliuoti, prognozuoti akcijų kainas biržose ir kt. Reikėtų pabrėžti, kad virtualūs eksperimentai taikomi dar ir tada, kai realių eksperimentų atlikti neįmanoma.

Šiame darbe tyrinėjama, kaip kompiuterinę simuliaciją galima pritaikyti tarpusavyje fundamentaliosiomis jėgomis sąveikaujančių dalelių sistemai modeliuoti. Toks uždavinys yra vadinamas N kūnų problema, kur N reprezentuoja sistemoje esančių kūnų skaičių. Pirmajame skyriuje apžvelgti istoriniai aspektai, nulėmę N kūnų problemos atsiradimą. Išvardyti autoriai, atlikę pirmuosius bandymus naudojant kompiuterines technologijas. Antrasis skyrius yra skirtas N kūnų sistemos dinamiką aprašančio matematinio modelio sudarymui. Modelio išvedimas pagrindžiamas baziniais klasikinės mechanikos judėjimo dėsniais, taip pat gravitacijos teorija ir energijos tvermės dėsniu. Trečiajame skyriuje plačiau aptarti N kūnų problemos taikymai modeliuojant galaktikas (astrofizikoje) ir molekules (biochemijoje). Kaip esminis skirtumas modeliuojant mikro-makro pasaulių kūnus – pateiktos sistemos potencinės energijos skaičiavimo formulės. Ketvirtasis skyrius yra skirtas N kūnų problemos sprendimo algoritmams ir jų analizei. Skaitant šį skyrių galima susipažinti su pagrindinėmis tiesioginio ir Barnes'o-Hut'o algoritmų idėjomis bei skirtumais. Penktajame skyriuje aprašytas darbo autoriaus sukurtas N kūnų sistemos modeliavimo įrankis „*Butterfly*“. Jame realizuotos ankstesniame skyriuje pateiktų algoritmų lygiagrečiosios versijos. Šeštasis skyrius dedikuotas skaičiuojamųjų eksperimentų, atliktų VGTU kompiuterių klasteryje „Vilkas“, rezultatų analizei. Čia palygintas algoritmų skaičiavimo laikas bei tikslumas, pateikti spartumo ir efektyvumo koeficientai (skaičiuojant su skirtingu procesorių skaičiumi). Taip pat sudarytas laiko sanaudų modelis, kurio koeficientai nustatyti iš empirinių duomenų.

2. Istorinė apžvalga

Antikinės Graikijos laikotarpiu jau buvo žinoma penkių planetų – Merkurijaus, Veneros, Marso, Jupiterio ir Saturno egzistencija. Graikai tikėjo, kad Žemė yra visatos centras ir kad aplink ją sukasi Saulė bei kitos planetos. Toks modelis, aiškinantis Saulės, Mėnulio ir kitų planetų judėjimą dangaus sfera, vadinamas geocentrinio modeliu arba Ptolemajaus sistema (angl. *Ptolemaic system*). Apie 280 m. pr. m. e. graikų astronomas Aristarchus'as (310–230 m. pr. m. e.) iškėlė heliocentrinę idėją. Jis teigė, kad Žemė apsisuka aplink savo ašį vieną kartą per 24 valandas ir kartu su kitomis planetomis skrieja aplink Saulę. Požiūris, kad Žemė yra Visatos centras buvo taip giliai įsišaknijusi religiniuose įsitikinimuose, kad Aristarchus'o idėja nesulaukė akademinės bendruomenės palaikymo, jį apkaltino šventvagyste ir išvijo iš Aleksandrijos. Šis iškreiptas požiūris tesėsi apie 1800 metų, kol lenkų matematikas, astronomas ir ekonomistas N. Copernicus'as (1473 m. vasario 19 d. – 1543 m. gegužės 24 d.) atgaivino Aristarchus'o teiginį, kaip tikėtiną empirinių planetų judėjimo duomenų paaiškinimą [1].

Vėliau heliocentrinio modelio idėją palaikė ir italų astronomas, matematikas, filosofas ir fizikas G. Galilei (1564 m. vasario 15 d. – 1642 m. sausio 8 d.). Jis buvo vienas iš pirmųjų astronomų efektyviai pritaikiusių teleskopą dangaus kūnų stebėjimams. Vokiečių astronomas ir matematikas J. Kepler'is (1571 m. gruodžio 27 d. – 1630 m. lapkričio 15 d.) pratesė N. Copernicus'o darbą. Remdamasis danų astronomo T. Brahe'o (1546 m. gruodžio 14 d. – 1601 m. spalio 24 d.) eksperimentų duomenimis, jis suformulavo tris dėsnius, kurie tapo moderniosios astronomijos pagrindo dalimi ir yra vadinami Kepler'io dėsniais (angl. *Kepler laws of planetary motion*). Pirmasis Kepler'io dėsnis teigia, kad planetos aplink Saulę skrieja elipsinėmis, o ne apskritomis orbitomis. Šį dėsnį vėliau teoriškai įrodė anglų fizikas, matematikas, astronomas ir filosofas I. Newton'as (1643 m. sausio 4 d. – 1727 m. kovo 31 d.). Tačiau jis siekė ne tik išsiaiškinti, kodėl tos orbitos yra elipsinės, bet ir kitą fundamentaliai svarbų dalyką: kokia jėga išlaiko planetas orbitoje aplink Saulę. Newton'as apibendrino savo pirmtakų ir savo paties darbus veikale „Matematiniai gamtos filosofijos pagrindai“, kuriame pateikė tris begalo svarbius judėjimo dėsnius, bei išdėstė universaliosios gravitacijos teoriją (angl. *theory of universal gravitation*). Trimis Newton'o dėsniais paremta visa klasikinė mechanika, kuri kartais dar yra vadinama Newton'o mechanika [1,3].



1 pav. Isaac'as Newton'as
(1643–1727 m.) [2]

Modernioji dangaus kūnų mechanika, kuri apjungia mechanines ir fizikines priežastis su stebimais dangaus kūnų judėjimo reiškiniiais, prasidėjo kartu su trijų judėjimo dėsnių, bei Newton'o gravitacijos dėsnio atradimais. Žinodamas tris skirtingas planetos koordinates ir greičius orbitoje, naudodamasis analitine geometrija, Newton'as parašė lygtį, kuri numato planetos judėjimą, t. y. apibrėžia jos orbitos savybes: poziciją, spindulį, periodą ir greitį. Praėjus keletui metų, mokslininkai įsitikino, kad judėjimo lygtys nenumato planetų orbitų tiksliai, o kai kuriais atvejais neduoda net apytikslio rezultato. To priežastys yra kitų planetų gravitacijos jėgos. Taigi, norint nustatyti tikrąją orbitą, neužtenka nurodyti trijų planetos orbitos pozicijų ir greičių (planetai esant tose pozicijose). Kitų kūnų gravitacinės jėgos veikiančios planetą taip pat turi būti žinomos. Šie pastebėjimai privedė prie taip vadinamos N kūnų problemos (angl. *N-body problem*), kuri formuluojama taip:

„Duota taškinių kūnų sistema, kurioje kūnai turi masę ir veikia vienas kitą pagal Newton'o gravitacijos dėsnį. Darant prielaidą, kad jie niekada nesusidurs, kiekvienam šios sistemos kūnui reikia surasti jo koordinates bet kuriuo laiko momentu.”

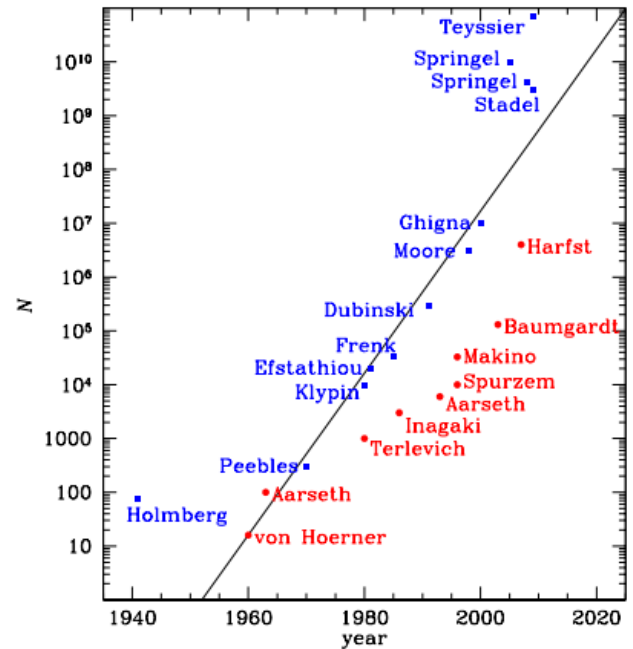
XIX a. Švedijos karalius Oscar'as II netgi pažadėjo apdovanojimą tam, kuris suras šio uždavinio sprendinį. Problemos sprendime buvo pateikta daug svarbių idėjų, kurios vėliau panaudotos kuriant chaoso teoriją (angl. *chaos theory*).

Kompiuterinės technologijos N kūnų problemos sprendimui pradėtos taikyti 1960 m. Pirmųjų simuliacijų pionieriumi laikomas vokiečių astrofizikas S. Hoerner'is (1919 m. balandžio 15 d. – 2003 m. sausio 7 d.). Tuo laikotarpiu skaičiavimo įrenginiai buvo primityvūs, todėl simuliuojamų sistemų kūnų skaičius buvo itin mažas, palyginus su eksperimentais, kuriuos šių dienų mokslininkai įgyvendina skaičiavimų centruose. S. Aarseth'o „*Gravitational N-Body Simulations*“ ir W. Dehnen'o bei J. Read'o „*N-body simulations of gravitational dynamics*“ darbuose minima, kad pirmieji eksperimentai buvo atlikti tik su $N = 16$ kūnų [4,5].

Kartu su ankstyvaisiais eksperimentais pradėti kurti problemos sprendimui taikomi specialieji integravimo algoritmai. Pastarųjų paskirtis – kiek įmanoma sumažinti aritmetinių veiksmų skaičių, taip sukuriant galimybę studijuoti didesnes N kūnų sistemas. A. Schluter'is 1961 m. pasiūlė individualių laiko žingsnių metodą (angl. *individual time-steps method*) [4, p. 21]. Šiame metode kiekvienam kūnui parenkamas individualus integravimo žingsnis remiantis kūnų pasiskirstymu erdvėje. Erdvės srityje, kurioje atstumai tarp kūnų yra didesni, t. y. pastarieji labiau išsibarstę, integravimo žingsnis parenkamas didesnis, o tankiau apkrautoje srityje – mažesnis. Tikslas yra minimizuoti skaičiavimų apimtį, tuo pačiu užtikrinant sprendinio konvergavimą. Dar vienas gerai žinomas eksperimentas įgyvendintas 1963 m., kai S. Aarseth'as realizavo

simuliatorių, su kuriuo studijavo $N = 100$ kūnų dinamiką. Toliau uždavinio dydis apytiksliai padvigubėdavo kas dvejus metus. Ši tendencija atitinka Moore'o dėsnį (angl. *Moore's law*) (2 pav.).

W. Dehnen'as bei J. Read'as sudarė ir savo darbe [5] pateikė grafiką (2 pav.), kuriame atvaizduotas uždavinio dydžio augimas pradedant pirmuoju eksperimentu. Tiesė nubrėžta grafike vaizduoja Moore'o dėsnį: $N = N_0 2^{(year - y_0)/2}$. Raudonais taškais pažymėti eksperimentai su sistemomis, kuriose dalelių susidūrimas yra leistinas (angl. *collisional systems*), o mėlynais – eksperimentai su sistemomis, kuriose dalelių susidūrimas yra neapibrėžtas (angl. *collisionless systems*). Atkreipkite dėmesį, kad čia pateiktas 1941 m. E. Holmberg'o atliktas eksperimentas. Šis eksperimentas įgyvendintas ne su kompiuteriu, o su įrenginiu, kurį sukonstravo pats Holmberg'as: dvimatės N kūnų sistemos simuliacijai jis panaudojo lemputes ir fotorezistorius [6]. Sistemos elementus reprezentavo lemputės, kur šaltinio šviesos stipris yra proporcingas elemento masei (moduliuojant šviesos stiprius Holmberg'ui pavyko atkurti duotąjį masės tankio profilį). Elementų pagreičių skaičiavimai atlikti integruojant šviesos skleidžiamą stiprumą (pastarasis išmatuotas fotorezistoriais).



2 pav. Eksperimentai su N kūnų sistemomis [5]

3. Matematinis modelis

Tarkime, kad inercinėje sistemoje yra N kūnų, kuriuos veikia vienintelė jėga, kylanti dėl tarp jų veikiančios gravitacinės sąveikos. Sistemos kūnų masės yra teigiamieji dydžiai $m_i > 0$, o koordinatės nusakomos vektoriais $\vec{x}_i = (x_{i1}, x_{i2}, x_{i3})^T$, $i = 1, \dots, N$. Žinoma, kad pagal antrąjį Newton'o dėsnį, kūno masės ir pagreičio sandauga yra lygi jį veikiančių jėgų sumai:

$$m_i \frac{d^2 \vec{x}_i}{dt^2} = \vec{F}_i = \sum_{j=1, j \neq i}^N \vec{F}_{i,j} \quad (1)$$

Newton'o gravitacijos dėsnis teigia, kad traukos jėgos tarp i -ojo ir j -ojo kūnų dydis yra proporcingas jų masių sandaugai ir atvirkščiai proporcingas atstumo tarp tų kūnų kvadratui:

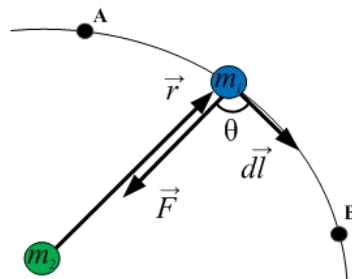
$$F_{i,j} = G \frac{m_i m_j}{r_{i,j}^2}, \quad (2)$$

čia $r_{i,j} = \|\vec{x}_i - \vec{x}_j\|$ – atstumas tarp i -ojo ir j -ojo kūnų; $G = 6.67408 \times 10^{-11} \text{ m}^3 \text{s}^{-2} \text{kg}^{-1}$ – gravitacijos konstanta; jėga $F_{i,j}$ veikia vienetinio vektoriaus $\vec{k}_{i,j} = \frac{\vec{x}_j - \vec{x}_i}{\|\vec{x}_i - \vec{x}_j\|}$ kryptimi [7, p. 26].

Kai dėl gravitacinės sąveikos kylanti jėga sistemoje išjudina kūną, perduodama energija, o fizikoje energijos perdavimas išreiškiamas atliktu darbu:

$$W = -\vec{F} \cdot \vec{l},$$

čia $\vec{l}$ – poslinkis, kurį įgavo kūnas veikiamas jėgos $\vec{F}$. Kitais žodžiais tariant, darbu išmatuojamas energijos perdavimas esant pokyčiams sistemoje. Uždaroje sistemoje kūno energija gali padidėti tik kito kūno energijos sąskaita. Taigi, jeigu sistemos neveikia jokios išorinės jėgos, joje visuomet yra toks pat energijos kiekis. Ši savybė yra vadinama energijos tvermės dėsniu (angl. *law of conservation of energy*). Modeliuojant uždaras N kūnų sistemas kompiuteriu, tvermės dėsnį galima naudoti kaip kriterijų patikrinti ar skaičiavimai buvo atlikti teisingai, ir nepakito sistemoje esančios energijos kiekis.



3 pav. Elementarusis poslinkis

Jeigu laikui bėgant jėgos didumas ir kryptis kinta, tai poslinkis skirtingais laiko momentais taip pat yra nevienodas. Tokiu atveju, reikia naudoti poslinkio diferencialą $d\vec{l}$, o perduotą energiją W skaičiuoti, kaip darbo diferencialų sumą [8, 9]:

$$dW = -\vec{F} \cdot d\vec{l} \implies W = -\int \vec{F} \cdot d\vec{l}.$$

Kampas tarp jėgos $\vec{F}$ ir poslinkio diferencialo $d\vec{l}$ vektorių yra θ (3 pav.), todėl pasinaudoję skaliarinės sandaugos apibrėžimu galime užrašyti:

$$dW = -F \cdot dl \cdot \cos(\theta).$$

Radialinis atstumas, kurį nuėjo kūnas yra $dr = dl \cdot \cos(\theta)$, tuomet į lygybę įstatę (2) gauname tokią sąryšį:

$$dW = -G \frac{m_i m_j}{r^2} dr.$$

Darbas atliekamas kūnui judant iš taško A į tašką B (3 pav.). Pažymime atstumą tarp kūnų pirmojoje pozicijoje r_A , o antrojoje pozicijoje r_B ir skaičiuojame integralą:

$$W = -G \int_{r_A}^{r_B} \frac{m_i m_j}{r^2} dr,$$

$$W = -G m_i m_j \left(\frac{1}{r_A} - \frac{1}{r_B} \right).$$

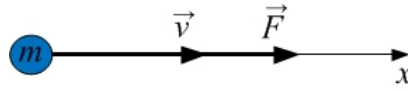
Sakykime, kad i -asis kūnas begalo nutolęs nuo koordinačių pradžios taško, įgavo poslinkį, dėl sąveikos su j -uoju kūnu ir dabar yra pozicijoje $\vec{x}_i$, kai $r_{i,j} = \|\vec{x}_i - \vec{x}_j\|$. Atliktas darbas yra vadinamas potencine energija $U_{i,j}$, kuri atsiranda dėl gravitacinės kūnų sąveikos [10, p. 18–22]:

$$U_{i,j} = -G \int_{\infty}^{r_{i,j}} \frac{m_i m_j}{r^2} dr = -G m_i m_j \left(\lim_{a \rightarrow \infty} \frac{1}{a} - \frac{1}{r_{i,j}} \right) = G \frac{m_i m_j}{r_{i,j}}. \quad (3)$$

Remdamiesi pavyzdžiu pateiktu vadovėlyje „*Molecular Modelling for Beginners*“ [10, p. 33–34], apskaičiuosime sistemos, kurią sudaro N kūnų, potencinę energiją. Perkeldami 1-ąjį kūną iš begalybės į tašką, kurį nurodo vektorius $\vec{x}_1$ nesunaudojame jokios energijos, todėl, kad sistemoje dar nėra jokių kūnų. 2-ojo kūno perkėlimas į vektoriaus $\vec{x}_2$ poziciją įtakoja energijos pokytį lygų: $U_{1,2} = G \frac{m_1 m_2}{r_{1,2}}$. Toliau iš begalybės į vektoriaus $\vec{x}_3$ poziciją perkeliame 3-įjį kūną. Pastarasis yra veikiamas 1-ojo ir 2-ojo kūnų sukeliama jėga, todėl energijos kiekis reikalingas kūnui perkelti yra: $U_{1,3} + U_{2,3} = G \left(\frac{m_1 m_3}{r_{1,3}} + \frac{m_2 m_3}{r_{2,3}} \right)$. Analogiškai tęsdami procesą, kol visi N kūnai bus perkelti, matysime, kad sistemos potencinė energija V apskaičiuojama formule:

$$V = \sum_{i=1}^{N-1} \sum_{j=i+1}^N U_{i,j}. \quad (4)$$

Nustatykime ryšį tarp jėgos ir potencinės energijos [10, p. 22–23]. Tarkime, kad masės m kūnas juda x ašies kryptimi veikiamas pastovios jėgos F (4 pav.). Tegul, kuriuo nors metu kūno momentinis greitis yra v , tuomet jo kinetinė energija $T = \frac{1}{2}mv^2$. Pasinaudoję



4 pav. Judėjimas vienmatėje erdvėje

energijos tvermės dėsniu gauname, kad kinetinės energijos T ir potencinės energijos V suma yra pastovus dydis:

$$E = T + V = \text{const.} \quad (5)$$

Lygybę (6) diferencijuojame laiko t atžvilgiu $\frac{dE}{dt} = mv\frac{dv}{dt} + \frac{dV}{dt} \implies 0 = mv\frac{dv}{dt} + \frac{dV}{dt} \frac{dx}{dt} \implies 0 = \left(m\frac{d^2x}{dt^2} + \frac{dV}{dx}\right) \frac{dx}{dt}$. Kadangi $\frac{dx}{dt} \neq 0$, tai $m\frac{d^2x}{dt^2} + \frac{dV}{dx} = 0 \implies m\frac{d^2x}{dt^2} = -\frac{dV}{dx}$ ir pagal antrąjį Newton'o dėsni (1) gauname:

$$F = -\frac{dV}{dx}.$$

Jeigu kūnas juda trimatėje erdvėje, kurioje jo pozicija nusakoma vektoriumi $\vec{x} = (x_1, x_2, x_3)^T$, tai jėgos ir potencinės energijos sąryšis nustatomas naudojant gradientą:

$$\vec{F} = -\frac{\partial V}{\partial \vec{x}} = -\nabla V = -\left(\frac{\partial V}{\partial x_1}, \frac{\partial V}{\partial x_2}, \frac{\partial V}{\partial x_3}\right)^T. \quad (6)$$

Sudarome matematinį modelį, kuriuo aprašoma N kūnų sistemos dinamika [7, p. 28]:

$$\mathbf{M} \frac{d^2 \vec{x}}{dt^2} + \nabla V(\vec{x}_1, \dots, \vec{x}_N) = 0, \quad (7)$$

čia $\mathbf{M} = \text{diag}(m_1, m_1, m_1, \dots, m_N, m_N, m_N)$ – diagonalinė matrica, kurios įstrižainėje įrašytos kūnų masės; $\vec{x} = (\vec{x}_1, \dots, \vec{x}_N) \in \mathbb{R}^{3N}$ – sistemos kūnų koordinačių vektorius ($\vec{x}_i \in \mathbb{R}^3$). Bendruoju atveju

$$\mathbf{M}_i = \text{diag}(m_i, m_i, m_i), \quad \frac{d^2 \vec{x}_i}{dt^2} = \left(\frac{d^2 x_{i_1}}{dt^2}, \frac{d^2 x_{i_2}}{dt^2}, \frac{d^2 x_{i_3}}{dt^2}\right)^T, \quad \nabla_i V = \left(\frac{\partial V}{\partial x_{i_1}}, \frac{\partial V}{\partial x_{i_2}}, \frac{\partial V}{\partial x_{i_3}}\right)^T$$

ir įstatę į (7) gauname:

$$\mathbf{M}_i \frac{\partial^2 \vec{x}_i}{\partial t^2} + \nabla_i V = 0 \implies \vec{F}_i = -\nabla_i V, \\ \text{arba} \\ F_{i_1} = -\frac{\partial V}{\partial x_{i_1}}, F_{i_2} = -\frac{\partial V}{\partial x_{i_2}}, F_{i_3} = -\frac{\partial V}{\partial x_{i_3}}.$$

4. N kūnų problemos taikymai

4.1. Molekulinis modeliavimas

Oksfordo anglų kalbos žodyne [11] sąvoka modelis apibrėžiama taip: tai idealizuotas (supaprastintas) sistemos ar proceso aprašymas, dažniausiai išreikštas matematiniais terminais, sukurtas siekiant išgauti informaciją apie sistemą ir prognozuoti procesų rezultatus. Taigi molekulinio modeliavimo pagrindinė užduotis yra kiek įmanoma tiksliau atkurti molekulinės sistemos, ar pavienės molekulės, suteikiant galimybę tyrinėti jų savybes, o molekulinio modelio – reprezentuoti molekules ir jų struktūras.

Skaiciuojamuosius eksperimentus su nedidelėmis molekulių sistemomis įmanoma atlikti rankiniu skaičiavimu, tačiau molekulinis modeliavimas neišvengiamai siejamas su kompiuteriniu modeliavimu. Skaitiniai metodai revoliucionalizavo molekulinį modeliavimą iki tokios apimties, kad daugeliu atvejų skaičiavimų neįmanoma atlikti nepanaudojus kompiuterio. Šiuo teiginiu nėra siekiama įtikinti, kad sudėtingesnis modelis yra geresnis už paprastesnį, bet kompiuteriai neabejotinai išplėtė sistemų ribas, kuriose modeliai gali būti taikomi. Naudojant molekulinį modeliavimą galima studijuoti platų spektrą sistemų: nuo atskirų molekulių iki skysčių (angl. *liquids*), ar kietųjų medžiagų (angl. *solids*), biologinių makromolekulių (angl. *biological macromolecules*), tokių kaip baltymai ir DNR. Taigi, molekulinis modeliavimas yra greitai besivystanti disciplina, kurios raidą įtakojo kompiuterinės ir programinės įrangos dramatiškas tobulėjimas.

Molekulinio modeliavimo procesas susideda iš trijų pagrindinių etapų:

- Pirmajame etape parenkamas modelis, kuriuo aprašomos tarpusavio sąveikos tarp dalelių. Naudojantis modeliu galima apskaičiuoti dalelių sistemos energiją ir nustatyti kaip ji varijuoja, keičiantis atomų ar molekulių pozicijoms.
- Antrajame etape atliekami skaičiavimai, pvz., minimizuojama sistemos energija (angl. *energy minimisation*), tyrinėjama molekulių dinamika (angl. *molecular dynamics*), vykdoma Monte Carlo simuliacija, ar konformacijos paieška (angl. *conformational search*).
- Paskutiniajame etape atliekama analizė, nustatomos sistemos savybės, bei tikrinama, ar skaičiavimai buvo atlikti teisingai.

Molekulių sistemos modeliuojamos dviem skirtingais būdais: naudojantis klasikinės mechanikos (angl. *classical mechanics*), arba kvantinės mechanikos (angl. *quantum mechanics*) dėsniais. Klasikinės mechanikos dėsniais paremtuose modeliuose atomai idealizuojami kaip masę

turintys taškiniai krūviai, t. y. atomo branduolys ir elektronai traktuojami kaip vienas objektas. Taigi čia nepaisomas elektronų judėjimas ir sistemos energija skaičiuojama atsižvelgiant tik į branduolio koordinatės erdvėje. Tokie modeliai vadinami empiriniais jėgų lauko modeliais (angl. *empirical force field models*). Kvantinė mechanika išsiskiria tuo, jog į skaičiavimus įtraukia elektronų dinamiką. Tai suteikia galimybę nustatyti sistemos savybes, kurios priklauso nuo atomo elektrinio lauko pasiskirstymo, o tiksliau tyrinėti chemines reakcijas, kuriose formuojami ir išardomi ryšiai tarp atomų [12, p. 26–28].

Dauguma molekulių sistemų, kurioms studijuoti pasitelkiami molekulinio modeliavimo metodai, yra per didelės, kad joms būtų taikomi kvantinės mechanikos principai. Kvantinė mechanika atsižvelgia į elektronus sistemoje, todėl net ir tuo atveju, jei dalis elektronų yra ignoruojami (uždavinys supaprastinamas), turi būti apdorojamas didelis skaičius dalelių, o tai šių dienų kompiuteriams yra per sudėtinga užduotis. Tuo tarpu klasikinės mechanikos dėsniai yra naudojami, kai reikia atlikti skaičiavimus su didelėmis molekulių sistemomis. A. Leach'o vadovėlyje „*Molecular Modelling. Principles and Applications*“ [12, p. 165–167] teigiama, kad empirinių modelių tikslumas kai kuriais atvejais gali prilygti kvantinės mechanikos tikslumui, tuo pačiu skaičiavimams sunaudojant mažiau laiko.

4.1.1. Empirinis jėgų lauko modelis

Dauguma molekuliniam modeliavimui naudojamų empirinių jėgų lauko modelių yra užrašomi šešių narių suma [13]:

$$V = U_{BS} + U_{AB} + U_{PT} + U_{IT} + U_{LJ} + U_{EL}, \quad (8)$$

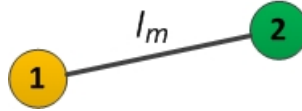
čia molekulių sistemos energija V skaičiuojama atsižvelgiant į atomų ryšių ilgių ir kampų nukrypimus nuo pusiausvyros padėties, bei atomų nerišančiąsias sąveikas (angl. *non-bonded interactions*) – elektromagnetines ir Van der Waals'o. Sudėtingesni jėgų laukai (taip trumpiau vadinami empiriniai jėgų lauko modeliai) gali turėti daugiau dedamųjų.

□ Pirmasis (8) lygties narys U_{BS} yra ryšių tempimo potencialas (angl. *bond stretching potential*). Juo modeliuojama sąveika tarp atomų porų tarpusavyje turinčių cheminius ryšius (5 pav.):

$$U_{BS} = \frac{1}{2} \sum_B k_i^l (l_i - l_i^0)^2,$$

čia B – visų ryšių tarp atomų porų aibė; l_i – ryšio ilgis; l_i^0 – charakteristinis ryšio ilgis (angl. *reference bond length*); k_i^l – ryšio stiprumo koeficientas. Parametras l_i^0 literatūroje dar dažnai

vadinamas pusiausvyruoju ryšio ilgiu (angl. *equilibrium bond length*), tačiau toks pavadinimas gali būti klaidinantis. Charakteristinis ryšio ilgis yra reikšmė, kuri gaunama, kai visi kiti jėgos lauko nariai yra eliminuojami. O pusiausvyrojo ryšio ilgio reikšmė gaunama, kai visa sistema yra mažiausioje energijos būsenoje (angl. *minimum energy state*) [12, p. 171].

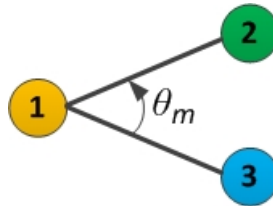


5 pav. Ryšių tempimo potencialas

□ Lenkimo kampo potencialas U_{AB} (angl. *angle-bending potential*) nurodo sąveikos stiprumą tarp trijų kovalentiniais ryšiais susietų atomų, kurie formuoja lenkimo kampą:

$$U_{AB} = \frac{1}{2} \sum_{m \in \text{angle}} k_m^\theta (\theta_m - \theta_m^0)^2,$$

čia θ_m – m -asis lenkimo kampas tarp dviejų gretimų ryšių, kurie dalijasi vienu atomu; θ_m^0 – lenkimo kampas, kai ryšiai yra pusiausvyros būsenoje; k_m^θ – lenkimo kampo stiprumo koeficientas.



6 pav. Lenkimo kampo potencialas

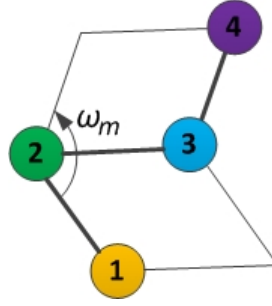
Torsinio (sukimo) kampo potencialas, kuris nusako keturių ryšiais sujungtų atomų sąveiką, aprašomas dviem skirtingais atvejais:

1. Kai visi atomai sujungti iš eilės (U_{PT});
2. Atomai sujungti ne iš eilės (U_{IT}).

□ Pirmuoju atveju potencinė energija skaičiuojama pagal šią formulę:

$$U_{PT} = \frac{1}{2} \sum_{m \in \text{torsions}} A_m (1 + \cos(n_m \omega_m - \gamma_m)),$$

čia ω_m – m -asis torsinis kampas tarp dviejų gretimų lenkimo kampų, kurie dalijasi vienu ryšiu; n_m – periodiškumo faktorius, kuris nusako pusiausvyros (angl. *equilibrium*) torsinių kampų skaičių; γ_m – fazė; A_m – amplitudė.

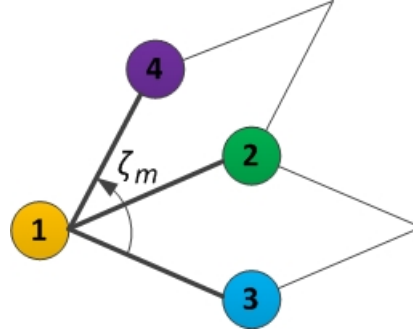


7 pav. Torsinio kampo potencialas (1-asis atvejis)

□ Antruoju atveju potencinės energijos išraiška yra tokia:

$$U_{IT} = \frac{1}{2} \sum_{m \in \text{torsions}} k_m^\zeta (\zeta_m - \zeta_m^0)^2,$$

čia ζ_m – m -asis sukimo kampas tarp keturių netvarkingai sujungtų atomų; ζ_m^0 – pusiausvyrisis sukimo kampas; k_m^ζ – sukimo kampo standumo koeficientas.

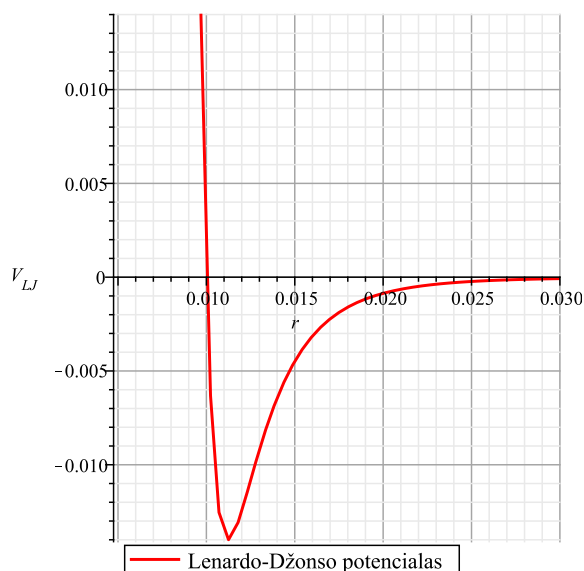


8 pav. Torsinio kampo potencialas (2-asis atvejis)

□ Penktasis narys matematiniam modelyje (8) yra Lennard'o-Jones'o potencialas:

$$U_{LJ} = \sum_{i=1}^{N-1} \sum_{j=i+1}^N 4\epsilon_{i,j} \left[\left(\frac{\sigma_{i,j}}{r_{i,j}} \right)^{12} - \left(\frac{\sigma_{i,j}}{r_{i,j}} \right)^6 \right],$$

čia $r_{i,j}$ – atstumas tarp i -ojo ir j -ojo atomų; $\epsilon_{i,j}$ – Van der Waals'o disociacijos energija; $\sigma_{i,j}$ – susidūrimo diametras. Disociacijos energija (angl. *bond dissociation energy*) nusako kokio energijos kiekio reikia, kad nutrūktų cheminis ryšys tarp atomų. Susidūrimo diametras (angl. *collision diameter*) yra mažiausias atstumas tarp atomų, kai šiems priartėjus vienas prie kito, pradeda veikti stūmos jėga (angl. *repulsion force*). Ši jėga atsiranda dėl to, jog elektronai supantys abiejų atomų branduolius yra vienodo - neigiamo krūvio. Atomams priartėjus atstumu mažesniu už susidūrimo diametrą, dėl stūmos stipriai išauga potencinė energija, taigi atsiranda nestabilumas. Jeigu atomų pora gali sureaguoti taip, kad suformuotų kovalentinį ryšį, pastarojo ilgis gali būti mažesnis už susidūrimo diametrą. Tačiau empirinis jėgų lauko modelis nenumato kovalentinių ryšių atsiradimo ar iširimo.



9 pav. Lenardo-Džonso potencialas

□ Remiantis Coulomb'o dėsnio ir potencinės energijos sąvoka sudaroma sistemos elektrostatinio potencialo U_{EL} skaičiavimo formulė:

$$U_{EL} = k \sum_{i=1}^{N-1} \sum_{j=i+1}^N \frac{q_i q_j}{r}.$$

čia k – proporcingumo koeficientas; q_i, q_j – taškiniai krūviai; $r_{i,j}$ – atstumas tarp taškinių krūvių. Proporcingumo koeficiento išraiška yra tokia: $k = \frac{1}{4\pi\epsilon\epsilon_0}$, kur ϵ – terpės dielektrinė skvarba, kuri parodo, kiek kartų terpė sumažina krūvių sąveikos jėgą; $\epsilon_0 = 8,854 \cdot 10^{-12}$ F/m – elektrinė konstanta. Elektrostatinis potencialas palyginus su Lennard'o-Jones'o potencialu yra stipresnis ir veikia, kai atomai vienas nuo kito yra nutolę dideliu atstumu.

4.1.2. Vidinių koordinatų sistema

Kuriant programinę įrangą, kurios paskirtis yra modeliuoti molekulių sistemas, labai svarbu tiksliai apibrėžti atomų pozicijas. Tai padaryti galime dviem keliais [12]. Pats paprasčiausias būdas yra naudoti Descartes'o koordinatų sistemą, t. y. nurodyti visų atomų (x, y, z) koordinatas. Alternatyvus variantas yra naudoti vidines koordinatas (angl. *internal coordinates*), kuriose kiekvieno atomo pozicija aprašoma priklausomai nuo kitų atomų pozicijų sistemoje, nurodant cheminių elementų simbolius (angl. *chemical element symbol*), ryšių ilgius (angl. *bond length*), ryšių kampus (angl. *bond angle*) ir torsinius kampus (angl. *dihedral angle*). Vidinės koordinatės užrašomos taip vadinama Z-matrica (angl. *Z-matrix*), pvz.:

$$Z = \begin{pmatrix} 1 & C & & & & & & \\ 2 & C & 1 & 1.54 & & & & \\ 3 & H & 1 & 1.0 & 2 & 109.5 & & \\ 4 & H & 2 & 1.0 & 1 & 109.5 & 3 & -180.0 \\ 5 & H & 1 & 1.0 & 2 & 109.5 & 4 & 60.0 \\ 6 & H & 2 & 1.0 & 1 & 109.5 & 5 & -60.0 \\ 7 & H & 1 & 1.0 & 2 & 109.5 & 6 & 180.0 \\ 8 & H & 2 & 1.0 & 1 & 109.5 & 7 & 60.0 \end{pmatrix}.$$

Pirmoje Z-matricos eilutėje apibrėžėme anglies atomą (C). Antroje eilutėje apibrėžtas taip pat anglies atomas, kurio ryšio su pirmuoju atomu ilgis lygus 1.54Å. Trečioje eilutėje yra vandenilio atomas (H), kurio ryšio su pirmuoju atomu ilgis lygus 1.0Å, o kampas, kurį formuoja atomai 2-3-1 yra 109.5°. Ketvirtasis atomas taip pat yra vandenilis, kurio ryšio su antruoju atomu ilgis lygus 1.0Å ir kampas, kurį formuoja atomai 4-2-1 yra 109.5°, o torsinis kampas formuojamas atomų 4-2-1-3 yra 180°. Likusieji atomai Z-matricoje apibrėžiami tokiu algoritmu:

1. Užrašome atomo, kurį norime apibrėžti (dėl patogumo algoritmo apibrėžime jį vadinsime *X*), indeksą sistemoje ir cheminio elemento simbolį;
2. Nurodome atomo indeksą (Z-matricoje jis jau turi būti apibrėžtas) su kuriuo atomas *X* turi ryšį ir užrašome to ryšio ilgį;
3. Nurodome atomo indeksą, kuris turi ryšį su *X*, arba atomo indeksą su kuriuo ryšį turi pats *X* ir užrašome kampą tarp šių trijų atomų;
4. Nurodome atomo indeksą, kuris turi ryšį bent su vienu atomu grandinėje ir užrašome šių keturių atomų formuojamą torsinį kampą.

Mažiau vidinių koordinatų reikia tik pirmiesiems trims atomams. Pirmasis atomas gali būti padėtas bet kur erdvėje, todėl neturi jokių vidinių koordinatų. Antrajam atomui reikia nurodyti tik ryšio ilgį su pirmuoju atomu, o trečiajam atomui – ryšio ilgį su vienu iš pirmųjų atomų ir kampą tarp šių trijų atomų. Visuomet galima vidines koordinates transformuoti į Descartes'o koordinates ir atvirkščiai. Vidines koordinates patogiau naudoti, kai reikia apibrėžti vienos molekulės atomų struktūrą, tuo tarpu Descartes'o koordinatės labiau tinka, kai reikia apibrėžti aibę diskrečiųjų molekulių.

4.2. Astrofizika

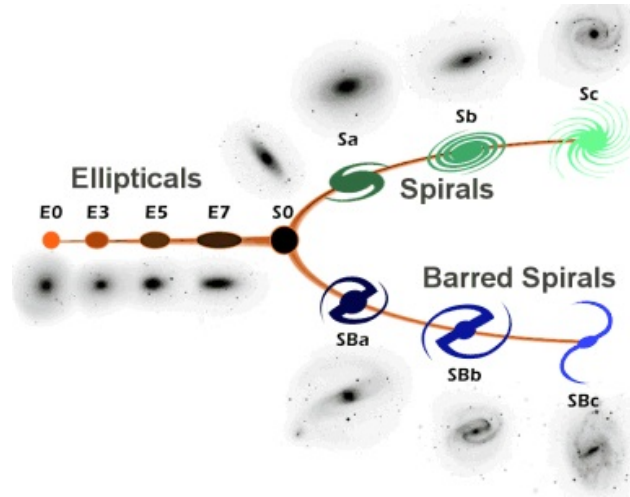
Kartu su teorine ir eksperimentine fizika, skaičiuojamoji fizika tapo pagrindiniu įrankiu, naudojamu atsakyti į astrofizikinius klausimus. Kompiuterinės galios didinimas, bei specialios įrangos ir taikomųjų programų kūrimas atvėrė galimybes tiesiogiai modeliuoti galaktikų evoliuciją. Šiuo metu duomenys gauti iš virtualių eksperimentų yra apdorojami tokiu pačiu būdu, kaip ir duomenys gauti iš realių stebėjimų teleskopu. N kūnų modeliavimo technika tapo vienu iš galingiausių įrankių studijuojant astronomines sistemas, kuriose kūnai tarpusavyje sąveikauja gravitacinėmis jėgomis, t. y. Saulės sistemas, žvaigždžių spiečius, galaktikų klasterius ir didelio mastelio Visatos struktūras [14].

Dauguma matomų dalelių Visatoje yra susitelkusios galaktikose, kurios yra pagrindinės astronomijos ekosistemos. Jose žvaigždės gimsta, vystosi ir miršta. Bendros galaktikų struktūrinės savybės ir jų pasiskirstymas erdvėje yra nustatomos pagal galaktikų formavimosi procesus. Tuo tarpu kitos savybės, tokios kaip žvaigždžių ir dujų kiekis galaktikose – pagal žvaigždžių formavimosi, jų evoliucijos procesus. Jeigu fizikai neklysta, tuomet galaktikose ir visoje Visatoje yra daug daugiau masės nei yra apskaičiuojama pagal matomas žvaigždes ir dujas. Manoma, jog Visatoje dominuoja tamsioji materija, išskyrus galaktikų vidines dalis, kur esti paprastų dalelių koncentracija, apsupta daug masyvesnių tamsiosios materijos halų (angl. *dark matter halo*) [15].

Galaktikos yra labai svarbūs „statybiniai“ Visatos blokai. Kai kurios iš jų yra labai paprastos, o kitų struktūros yra itin kompleksiškos. Amerikiečių astronomas E. Hubble'as 1926 m. sukūrė galaktikų klasifikavimo schemą. Jis stebimas galaktikas suskirstė į du pagrindinius tipus: elipsines ir spiralines (jos atitinkamai yra žymimos E ir S raidėmis). Elipsinės galaktikos yra ovalios formos ir žvaigždės jose sukasi netvarkingai, įvairiomis kryptimis. Tuo tarpu spiralinės galaktikos yra plokščios diskinės žvaigždžių sistemos, kurios aplink galaktikos centrą sukasi pagal laikrodžio rodyklę arba prieš laikrodžio rodyklę (priklausomai nuo stebėtojo pozicijos). E. Hubble'as elipsines galaktikas sunumeravo skaičiais nuo 0 iki 7, kurie charakterizuoja galaktikų elipsiškumą. $E0$ nurodo, jog galaktika yra beveik apvali, o $E7$ – jog galaktika yra labai elipsiška. Spiralinėms galaktikoms priskyrė raides nuo a iki c . Pastarosios charakterizuoja galaktikų spiralinių vijų (angl. *spiral arms*) kompaktiškumą. Pavyzdžiui, Sa spiralinių galaktikų vijos yra standžiai apsivijusios centrą, o Sc – plačiai. Taip pat reikėtų paminėti, kad astronomai pastebėjo, jog spiralinių galaktikų apvalių centrinių regionų dydis priklauso nuo to, kaip standžiai jų vijos yra apsivijusios centrą. Pavyzdžiui, Sa centriniai regionai yra didesni palyginus su Sc galaktikų centriniais regionais [17].

Spiralines galaktikas Hubble'as išskyrė dar į du subtipus: normaliosios spiralinės (dažnai

tiesiog spiralinės) ir skersinės spiralinės galaktikos. Skersinės spiralinės galaktikos yra atskiriamos prie S papildomai pridėdant raidę B . Taigi, priklausomai nuo vijų standumo jos žymimos taip: SBa , SBb , SBc . Hubble'o sudarytą galaktikų klasifikavimo schemą galite patyrinti Vakarų Vašingtono universiteto internetiniame puslapyje [16] pateiktame paveikslėlyje (10 pav.).



10 pav. Hablo galaktikų klasifikavimo schema [16]

4.2.1. Plummer'io modelis

Norint atlikti gravitacinę simuliaciją, reikia žinoti pradines kūnų koordinates ir greičius. Paprasčiausias variantas yra generuoti pradines sąlygas atsitiktinai nurodant regiono ribas, taip pat mažiausius ir didžiausius kūnų greičius, bei mases. Tačiau eksperimentas su tokiais pradinėmis sąlygomis gali baigtis tuo, jog didžioji dalis dalelių įgiję pabėgimo greitį (angl. *escape velocity*) v_{esc} atitrūks nuo klasterio gravitacinio lauko, o orbitose aplink klasterio centrą liks skrieti per mažai kūnų, jog būtų galima tyrinėti sistemos dinamiką. Dėl šios priežasties mokslininkai kuria galaktikų masės pasiskirstymo modelius (angl. *mass distribution model*). Naudojant šiuos modelius galima sugeneruoti tokias pradines sąlygas, su kuriomis jau simuliacijos pradžioje sistemoje vyrautų gravitacinių jėgų pusiausvyra, išlaikanti kūnus orbitoje.

Vienas iš anksčiausiai sukurtų galaktikų masės pasiskirstymo modelių yra Plummer'io modelis. Reikėtų paminėti, kad A. Schuster'is 1883 m. savo publikacijoje pateikė šio modelio matematinį aprašymą. Tačiau vėliau 1911 m. būtent Plummer'is parodė, jog šis modelis gerai atspindi realių stebėjimų metu gautus empirinius duomenis. Plummer'io modelis yra pritaikytas modeliuoti žvaigždžių spiečius arba elipsines galaktikas $E0$, kurių elipsiškumas yra itin mažas. Modelio matematinis aprašymas sukurtas darant prielaidą, jog modeliuojamame klasteryje egzistuoja sferinė simetrija. Esant šiai simetrijai potencialas $\Phi(r)$ ir masės tankio pasiskirstymas $\rho(r)$ priklauso tik nuo atstumo iki sferos centro r . Taip pat, generuojant pradines sąlygas su

Plummer'io modeliu laikomasi sąlygos, kad kūnų greičių pasiskirstymas yra izotropinis [18, p. 11–12].

Mažiausią pabėgimo greitį, kuriam esant kūnas išskrieja iš klasterio, pastarasis įgyja tada, kai jo kinetinė energija

$$T = \frac{1}{2}mv^2$$

tampa lygi potencinei energijai U (viso klasterio atžvilgiu). Masės m kūno potencinė energija

$$U = -m\Phi(r)$$

yra apskaičiuojama naudojantis Plummer'io potencialu [18, p. 13]:

$$\Phi(r) = -\frac{GM}{\sqrt{r^2 + a^2}}, \quad (9)$$

čia r – atstumas tarp kūno ir sferos centro; G – gravitacinė konstanta; M – klasterio masė, kuri yra lygi visų jį sudarančių kūnų masių sumai; a – generuojamo klasterio branduolio spindulys, trumpiau vadinamas Plummer'io spinduliu. Užrašę lygybę tarp kinetinės ir potencinės energijos išraiškų gauname:

$$\frac{1}{2}mv_{esc}^2 + m\Phi(r) = 0 \implies v_{esc}(r) = \sqrt{-2\Phi(r)}. \quad (10)$$

Norint apskaičiuoti masės tankį $\rho(r)$, kurio reikia sugeneruoti potencialui $\Phi(r)$, naudojama diferencialinė Gauss'o dėsnio forma:

$$\nabla \vec{g}(r) = -4\pi G\rho(r), \quad (11)$$

kur $\vec{g}(r)$ – masyvaus objekto kuriamas gravitacinis laukas (N/kg). Šis laukas reprezentuoja kūno sukeltą gravitacinę jėgą kitam 1 kg masės kūnui, kuris nuo pirmojo yra nutolęs atstumu r .

Trečiajame darbo skyriuje nustatėme, jog kūną veikianti jėga yra lygi neigiamam potencialo gradientui (6). Todėl galime užrašyti:

$$\vec{g}(r) = -\nabla\Phi(r). \quad (12)$$

Įstatę gautąjį sąryšį (12) į diferencialinę Gauss'o dėsnio formą (11) gauname:

$$\nabla^2\Phi(r) = 4\pi G\rho(r). \quad (13)$$

Atidžiau išsižiūrėjus matyti, jog tai yra Poisson'o lygties (angl. *Poisson's equation*) išraiška. Ka-

dangi $\Phi(r)$ priklauso tik nuo vieno kintamojo, tai lygtį (13) perrašome taip:

$$\frac{1}{r^2} \frac{d}{dr} \left(r^2 \frac{d}{dr} \Phi(r) \right) = 4\pi G \rho(r) \implies \frac{d^2}{dr^2} \Phi(r) + \frac{2}{r} \frac{d}{dr} \Phi(r) = 4\pi G \rho(r). \quad (14)$$

Surandame Plummer'io potencialo (9) pirmąją ir antrąją išvestines:

$$\begin{aligned} \frac{d}{dr} \Phi(r) &= GM \frac{r}{\sqrt{(r^2 + a^2)^3}}, \\ \frac{d^2}{dr^2} \Phi(r) &= GM \frac{-2r^2 + a^2}{\sqrt{(r^2 + a^2)^5}}. \end{aligned}$$

Įstatome apskaičiuotas išvestines į (14) ir išvedame masės tankio formulę:

$$\rho(r) = \frac{M}{4\pi} \frac{3a^2}{\sqrt{(r^2 + a^2)^5}}. \quad (15)$$

Radialinė masės pasiskirstymo funkcija (angl. *radial mass distribution function*) sferos tūryje bendruoju atveju yra tokia [18, 19]:

$$m(r) = \int_0^r 4\pi r^2 \rho(r) dr. \quad (16)$$

Integruojamas reiškiny yra labai panašus į dešiniąją Poisson'o lygties pusę. Padalijus (14) iš gravitacijos konstantos G ir įstatę į (16) gauname:

$$\begin{aligned} m(r) &= \frac{1}{G} \int_0^r \frac{d}{dr} \left(r^2 \frac{d}{dr} \rho(r) \right) = \frac{1}{G} \frac{d}{dr} \left(r^2 \frac{d}{dr} \rho(r) \right) \Big|_0^r = \\ &= \frac{1}{G} r^2 \frac{d}{dr} \Phi(r) = M \frac{r^3}{\sqrt{(r^2 + a^2)^3}} = M \left(1 + \frac{a^2}{r^2} \right)^{-\frac{3}{2}}. \end{aligned} \quad (17)$$

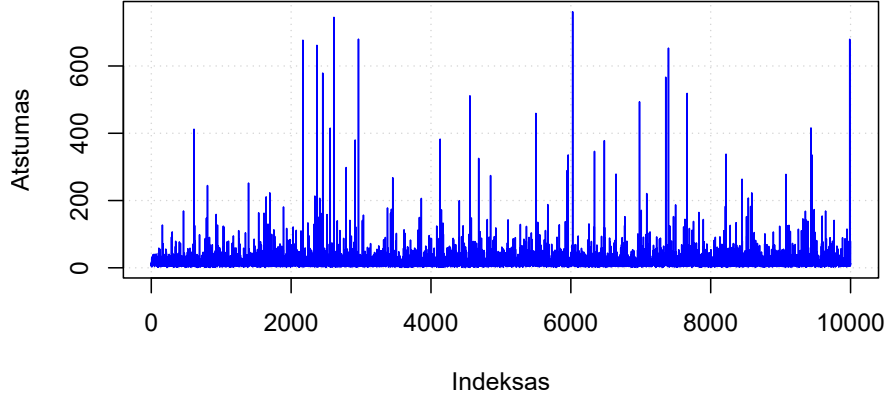
Iš (17) išreiškiame atstumą r iki klasterio branduolio centro:

$$m(r) = M \left(1 + \frac{a^2}{r^2} \right)^{-\frac{3}{2}} \implies r(m) = a \left[\left(\frac{M}{m} \right)^{\frac{2}{3}} - 1 \right]^{-\frac{1}{2}}. \quad (18)$$

Taikant Plummer'io modelį eksperimento pradinių sąlygų generavimui, funkcija (18) yra naudojama kūnų atstumų iki branduolio centro nustatymui. Pavyzdžiui, skaičiuojant i -ojo kūno atstumą, pagal tolygųjį skirstinį yra generuojamas atsitiktinis dydis $m_i \in [0, M)$ (tariama kūno masė). Kuo m_i yra arčiau M , tuo atstumas r tarp kūno ir branduolio centro yra didesnis. Jeigu generuojant atsitiktinius dydžius nebūtų tenkinama sąlyga $m_i < M$, tuomet egzistotų labai nedidelė tikimybė, jog kūnui bus priskirtas atstumas lygus begalybei. Tikrosios masės Plummer'io modelyje visiems kūnams yra priskiriamos vienodos:

$$m_i = \frac{M}{N}, \quad i = 1, \dots, N.$$

Pabandėme sugeneruoti radialinius atstumus r su šiais parametrais: $N = 10000$, $M = 1.0$, $a = 10.0$. Iš gautų rezultatų matyti, kad didžioji dalis kūnų yra branduolio dalis, arba susitelkę apie branduolį ir tik pavieniai kūnai yra nutolę didesniu atstumu (11 pav.).



11 pav. Kūnų radialiniai atstumai

Apskaičiavus radialinį atstumą, toliau atsitiktinai generuojamos kitos sferinės koordinatės: azimutinis ir poliarinis kampai. Azimutinis kampas yra priskiriamas intervale $\phi \in [0, 2\pi]$, tuo tarpu poliarinis kampas – $\theta \in [0, \pi]$. Sferinės koordinatės nesunkiai gali būti transformuojamos į Descartes'o koordinatinių sistemą naudojant šias formules:

$$\begin{aligned} x &= r \cdot \sin(\theta) \cos(\phi), \\ y &= r \cdot \sin(\theta) \sin(\phi), \\ z &= r \cdot \cos(\theta). \end{aligned} \tag{19}$$

Generuojant sistemos kūnų greičius atsižvelgiama į minimalų pabėgimo greitį (10). Kūno greitis v negali būti lygus arba didesnis, nei pabėgimo greitis, t. y. turi galioti sąlyga $v < v_{esc}$. Todėl pagal tolygųjų skirstinį generuojamas atsitiktinis dydis $\gamma \in [0, 1)$, iš kurio padauginus v_{esc} gaunamas pradinis kūno greitis:

$$v = \gamma \cdot v_{esc}(r) = \gamma \cdot \sqrt{-2\Phi(r)}.$$

Toliau generuojami azimutinis, bei poliarinis kampai ir pagal (19) sferinės vektoriaus komponentės transformuojamos į Descartes'o koordinates.

5. Sprendimo algoritmai

Kiek daugiau, nei prieš 300 metų Newton'as sudarė formules taikomas kūnų dinamikos skaičiavimuose (žr. (1), (2) formules). Tuo laikotarpiu suformuluota ir N kūnų problema. Be skaitmeninių kompiuterių eksperimentai su didelėmis N kūnų sistemomis buvo neįmanomi, tačiau pirmieji eksperimentai naudojant kompiuterius (XX a. septintasis dešimtmetis) taip pat nebuvo didelės apimties. Tik tobulėjant technologijoms atsirado galimybė tyrinėti vis didesnes sistemas, kuriose kūnų skaičius per penkiasdešimtmetį išaugo nuo $N = 16$ iki $N = 10^9$. Mokslinių tyrimų atžvilgiu, tai dramatiškas postūmis, suteikiantis galimybes modeliuoti molekulių sistemas, skysčių dinamiką, žvaigždžių spiečių ir galaktikų evoliuciją. Tačiau netgi dabar tyrėjai siekia didesnio skaičiavimų spartumo – galimybės simuliuoti dar sudėtingesnius reiškinius. Tai ypač aktualu kvantinėje chemijoje, kur modeliuojant atomų tarpusavio sąveiką atsižvelgiama ir į aplink atomą orbitoje skriejančių elektronų konfigūraciją. Taip pat ir kosmologijoje, kur vienas didžiausių eksperimentų buvo atliktas 2005 m. bandant atkurti Visatos evoliuciją. Todėl yra kuriami ir tobulinami algoritmai, kurie leistų sumažinti bazinių operacijų skaičių, bei pagreitintų skaičiavimus. Šiame skyriuje aptarsime du N kūnų problemos sprendimo metodus: tiesioginį ir Barnes'o-Hut'o algoritmus.

5.1. Tiesioginis algoritmas

Taikomojoje programoje paprasčiausia realizuoti tiesioginį N kūnų problemos sprendimo algoritmą (angl. *brute-force method*). Kita vertus, šis algoritmas yra pats sudėtingiausias bazinių veiksmų skaičiaus atžvilgiu. Asimptotinis tiesioginio algoritmo sudėtingumo įvertinimas yra $\mathcal{O}(N^2)$, kur baziniu veiksmu yra laikomas vienos gravitacinės jėgos, kylančios iš dviejų kūnų sąveikos, skaičiavimas. Čia antrosios eilės sudėtingumas gaunamas todėl, kad norint nustatyti vieno kūno astojamąją jėgą, reikia apskaičiuoti gravitacines jėgas, kuriomis jį veikia visi likusieji sistemos kūnai. Tokių jėgų skaičius yra lygus $N - 1$. Kadangi sistemoje yra N kūnų, tai iš viso reikia apskaičiuoti $N(N - 1) = N^2 - N$ jėgas. Atmetus antrąjį narį, kuris santykyje su visų bazinių veiksmų skaičiumi yra mažas dydis, kai N yra labai didelis, gaunamas algoritmo sudėtingumo įvertinimas. Netgi atsižvelgus į trečiąjį Newton'o dėsnį, kuris teigia, kad du kūnai veikia vienas kitą vienodo didumo tačiau priešingų kryptų jėgomis, gaunamas toks pat įvertinimas. Nors bazinių veiksmų skaičius ir sumažėja du kartus:

$$\frac{1}{2}N(N - 1) = \frac{1}{2}(N^2 - N) = \mathcal{O}(N^2).$$

Tarkime, kad problemos sprendinys yra ieškomas laiko intervale $0 \leq t \leq T$. Tuomet įvedamas diskretusis tinklas pagal laiką t , kuriame tiesioginiu algoritmu sprendžiamas (7) uždavinys:

$$w = \{t_k : t_k = k\tau, k = 0, 1, \dots, K, t_K = T\}, \quad (20)$$

čia τ – žingsnis tarp dviejų laiko momentų, t_k – diskrečiojo tinklo mazgas (laiko momentas), K – simuliacijos iteracijų skaičius. Puikiai žinoma, kad klasikinėje mechanikoje bazinę jėgos skaičiavimo formulę aprašo antrasis Newton'o dėsnis (1). Pagreitis yra greičio pokytį nusakantis dydis ir gali būti nustatomas apskaičiuojant greičio išvestinę $v' = a$. N kūnų simuliacijose greičio išvestinė aproksimuojama baigtinių skirtumų schema:

$$a^k = \frac{v^{k+1} - v^k}{\tau}, \quad (21)$$

čia v^k , a^k – kūno greitis ir pagreitis laiko momentu t_k . Įstatę (21) į (1) išvedame greičio skaičiavimo formulę:

$$F = m \left(\frac{v^{k+1} - v^k}{\tau} \right) \implies v^{k+1} = v^k + \frac{F}{m} \tau. \quad (22)$$

Analogiškai pritaikę baigtinių skirtumų schemą gauname kūno padėties erdvėje skaičiavimo formulę:

$$v^k = \frac{x^{k+1} - x^k}{\tau} \implies x^{k+1} = x^k + v^k \tau. \quad (23)$$

čia x^k – kūno koordinatė laiko momentu t_k .

Užrašykime tokią lygybę $v^{k+1} := v(t_k + \tau)$, tuomet išskleidę funkciją v^{k+1} Taylor'o eilute galime nustatyti baigtinių skirtumų formulės paklaidą [20]:

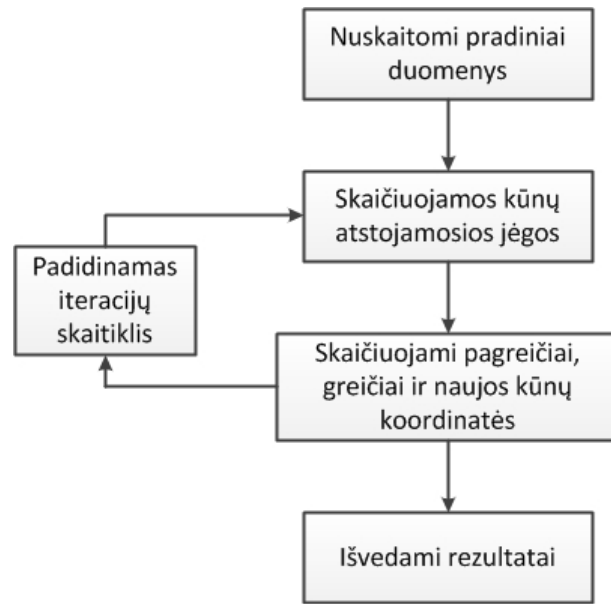
$$v^{k+1} = v^k + \tau v'^k + \frac{\tau^2}{2} v''^k \implies \frac{v^{k+1} - v^k}{\tau} = v'^k + \frac{\tau}{2} v''^k.$$

Matome, kad baigtinių skirtumų formulę aproksimuoja išvestinė v'^k tikslumu $\mathcal{O}(\tau)$. Alternatyvia siekiant mažesnių paklaidų yra centrinių skirtumų formulė. Jos tikslumas įvertinamas $\mathcal{O}(\tau^2)$.

Naudodamiesi (2), (22) ir (23) formulėmis, bei taikydami iteracinę procedūrą galime apskaičiuoti kūnų padėtis diskrečiojo tinklo (20) mazguose:

$$\begin{aligned} x^0, v^0 &- \text{pradiniai duomenys;} \\ x^1 &= x^0 + v^0 \tau, \quad v^1 = v^0 + \frac{F^0}{m} \tau, \\ x^2 &= x^1 + v^1 \tau, \quad v^2 = v^1 + \frac{F^1}{m} \tau, \\ &\dots \\ x^{k+1} &= x^k + v^k \tau, \quad v^{k+1} = v^k + \frac{F^k}{m} \tau. \end{aligned}$$

Pateikiame tiesioginio N kūnų problemos sprendimo algoritmo struktūrinę schemą (12 pav.).



12 pav. Tiesioginis N kūnų problemos sprendimo algoritmas

5.2. Barnes'o-Hut'o algoritmas

Siekiant sumažinti bazinių operacijų skaičių Barnes'o-Hut'o algoritme įvedama aproksimacija, kurios pagrindinė idėja yra grupuoti vienas nuo kito nedideliu atstumu nutolusius kūnus. Tokia aproksimacija, taikoma vietoje tiesioginio visų jėgų skaičiavimo, leidžia uždavinio sudėtingumą sumažinti nuo $\mathcal{O}(N^2)$ iki $\mathcal{O}(N \log N)$. Šį algoritmą 1986 m. savo straipsnyje publikavo J. Barnes'as ir P. Hut'as.

Skaiciavimuose kūnų grupė laikoma vienu objektu jeigu jos masės centras yra pakankamai nutolęs nuo stebėjimo taško. Grupės masė yra lygi visų ją sudarančių kūnų masių sumai:

$$M = \sum_{i \in C} m_i.$$

Masės centru erdvėje laikomas visų grupės kūnų koordinatčių svertinis vidurkis:

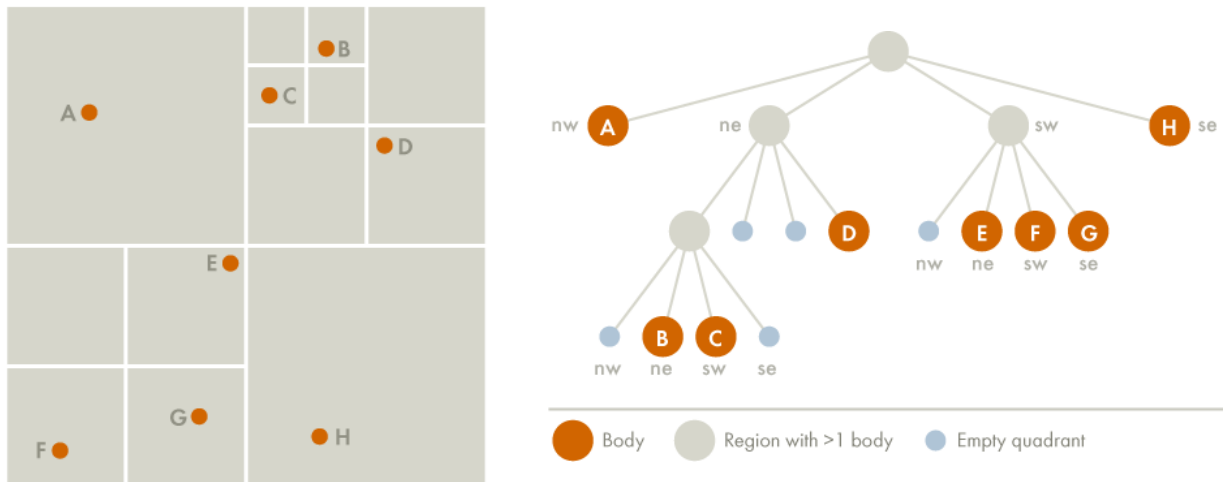
$$x^* = \frac{1}{M} \sum_{i \in C} x_i m_i,$$

čia C – grupės kūnų indeksų aibė, m_i – i -ojo kūno masė.

Barnes'o-Hut'o algoritmo realizacijai naudojamas duomenų medis, kurio vidinių viršūnių vaikų skaičius priklauso nuo simuliuojamos kūnų sistemos erdvės dimensijų skaičiaus. Jeigu simuliacija vykdoma dvimatėje erdvėje, tuomet medžio vidinės viršūnės turi keturis vaikus (angl.

quad-tree), o trimatės erdvės atveju – aštuonis vaikus (angl. *octree*). Toks vaikų skaičius pasirinktas neatsitiktinai, kiekviena medžio vidinė viršūnė reprezentuoja dvimatės (trimatės) erdvės regioną.

Tarkime, kad simuliacija yra vykdoma dvimatėje erdvėje, tuomet duomenų medžio šaknis reprezentuoja visą sistemos erdvę, o jos keturi vaikai – tos erdvės kvadrantus. Duomenų medžio konstravimas vykdomas rekursyviai dalijant regionus į kvadrantus tol, kol kiekviename regione lieka tik po vieną kūną. Tokius regionus reprezentuoja išorinės medžio viršūnės. Reikėtų pabrėžti, kad kai kurios išorinės viršūnės gali ir neturėti joms priskirto kūno, t. y. regionai gali likti tušti (13 pav.).



13 pav. Medžio duomenų struktūra [21]

Atstojamoji jėga veikianti sistemos kūną yra skaičiuojama iteruojant per medžio viršūnes, pradedant šaknimi. Jeigu vidinės viršūnės masės centras yra pakankamai toli nuo kūno, tuomet taikoma aproksimacija. Visi kūnai patenkantys į tos viršūnės regioną laikomi vienu objektu, kurio pozicija yra grupės masės centras, o masė – lygi grupės kūnų masių sumai. Jeigu atstumas tarp kūno ir medžio viršūnės masės centro yra nepakankamai didelis, toliau rekursyviai iteruojama per tos viršūnės vaikus. Tokia aproksimacija veiksminga todėl, kad gravitacijos jėga yra atvirkščiai proporcinga atstumo tarp kūnų kvadratui, t. y. kuo kūnai toliau vienas nuo kito, tuo tarpusavio traukos jėga yra mažesnė. Taigi esant didesniai atstumui, aproksimacijos paklaida taip pat yra mažesnė.

Kad skaičiuojant Barnes'o-Hut'o algoritmu būtų galima patikrinti, ar atstumas r tarp vidinės viršūnės masės centro ir kūno yra pakankamai didelis, įvedama ribinė vertė θ . Šis parametras yra lyginimas su regiono pločio s ir atstumo r santykiu:

$$\frac{s}{r} < \theta. \quad (24)$$

Jeigu (24) sąlyga tenkinama, tuomet jėga skaičiuojama tarp kūno ir kūnų grupės, priešingu atveju iteruojama per subregionus (viršūnės vaikus). Reguluojant parametą θ kinta skaičiavimų tikslumas ir apimtis (tuo pačiu ir bendras simuliacijai atlikti reikalingas laikas). Parinkus $\theta = 0$ jokia vidinė viršūnė negali būti laikoma vienu kūnu ir algoritmas degeneruoja į tiesioginį algoritmą. Plačiau apžvelkime du pagrindinius Barnes'o-Hut'o algoritmo etapus.

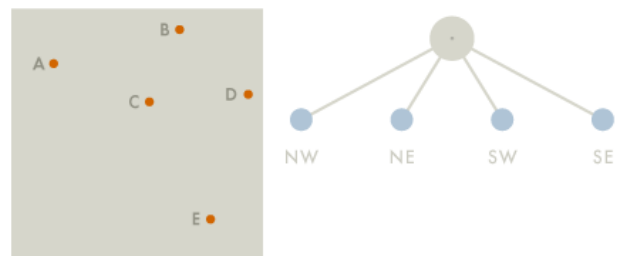
Medžio duomenų struktūros konstravimas. Kaip jau minėjome anksčiau, konstruojant duomenų medį kūnai įterpiami rekursyviai. Norint įterpti kūną i į medžio viršūnę w , vykdoma tokia rekursyvi procedūra [21]:

1. Jeigu viršūnė w yra vidinė, perskaičiuoti jos masę, bei masės centrą ir subregione kuris riboja kūną i pradėti rekursyvią procedūrą nuo 1-ojo žingsnio.
2. Jeigu viršūnė w yra išorinė ir neturi kūno, joje išsaugoti kūną i .
3. Jeigu viršūnė w yra išorinė, tačiau jau turi joje išsaugotą kūną, padalinti regioną į keturis kvadrantus (į aštuonis oktantus, jeigu sistemos erdvė trimatė) ir patalpinti abu kūnus atitinkamose erdvės srityse taikant tą pačią rekursyvią procedūrą.

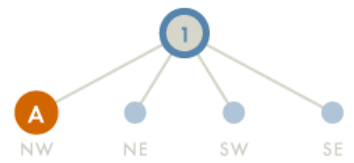
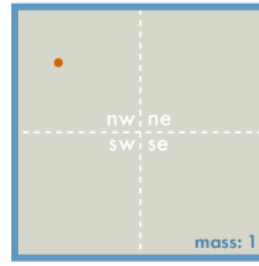
Kiekviena išorinė viršūnė gali turėti vieną, arba nė vieno kūno. Ši sąlyga yra būtina konstruojant medžio duomenų struktūrą.

□ Pateikiame medžio duomenų struktūros konstravimo pavyzdį, kai dvimatėje sistemoje yra penki kūnai (12 pav. pažymėti raidėmis A , B , C , D , E). Pavyzdys plačiau išanalizuotas internetiniame puslapyje patalpintame straipsnyje [22]. Norime atkreipti dėmesį, kad straipsnio autorius vidinių viršūnių vaikų reprezentuojamus subregionus pažymėjo atitinkamai: NW – šiaurės vakarų; NE – šiaurės rytų; SW – pietvakarių; SE – pietryčių.

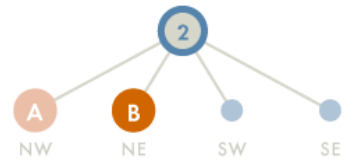
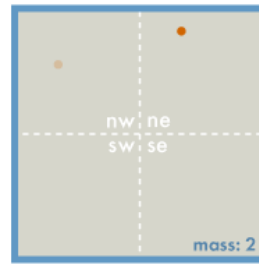
1.) Duomenų struktūros konstravimo etapo pradžioje nustatomos erdvės ribos į kurias patenka visi sistemos kūnai. Tuomet sukuriamas medžio šaknis atstovaujanti visą sritį ir keturi jos vaikai (kiekvienam iš keturių regionų).



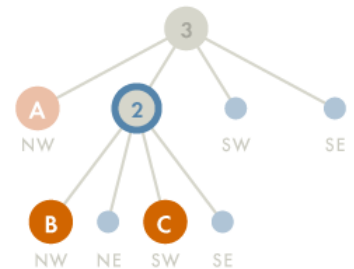
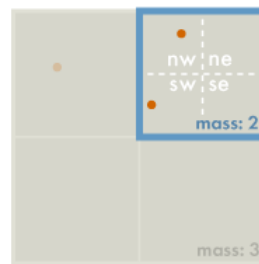
2.) Pradedamas sistemos kūnų įterpimas į duomenų struktūrą. Kūno *A* koordinatės riboja šiaurės vakarų *NW* regionas. Todėl kūnas išsaugomas tą regioną atstovaujančioje išorinėje viršūnėje. Šiame pavyzdyje visų kūnų masės yra lygios vienetui.



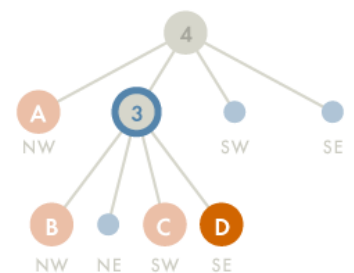
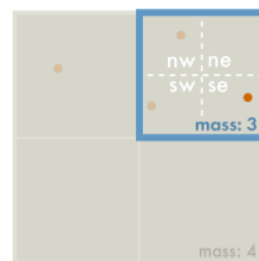
3.) Kūnas *B* patenka į regioną *NE*, todėl yra išsaugomas šį regioną reprezentuojančioje medžio viršūnėje. Išsaugojus kūną perskaičiuojama tėvinės medžio viršūnės masė ir masės centras.



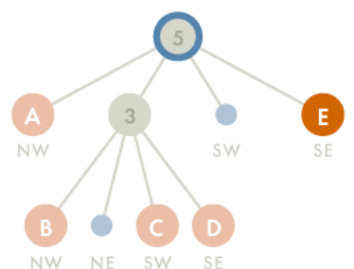
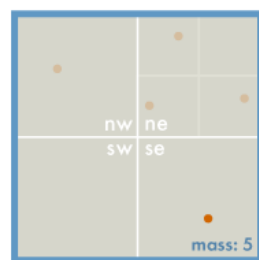
4.) Kūnas *C* taip pat priklauso *NE* regionui, tačiau jame jau yra išsaugotos kūnas. Todėl *NE* dalijamas į keturis kvadrantus ir tuomet kūnai *B* bei *C* patalpinami į atitinkamus subregionus.



5.) Kūnas *D* patenka į regioną, kuriame išsaugoti du kūnai. Iteruojama per viršūnės vaikus, kol surandamas neužimtas subregionas, kuris riboja kūno koordinatės.



6.) Paskutinis kūnas išsaugomas pietrytiniame regione *SE*. Atkreipkite dėmesį į vidinių viršūnių mases. Medžio šaknis reprezentuoja visą sritį, todėl jos masė yra lygi visų sistemos kūnų masei.

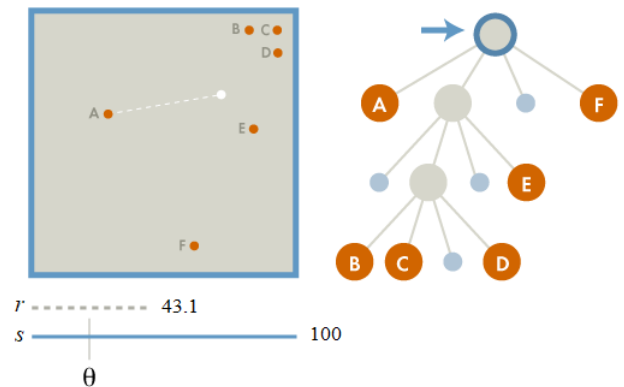


Kūnų atstojamųjų jėgų skaičiavimas. Atstojamosios jėgos taip pat skaičiuojamos taikant rekursyvią procedūrą. Norint apskaičiuoti kūno A atstojamąją jėgą vykdomas toks algoritmas (pradedant duomenų medžio šaknimi) [21]:

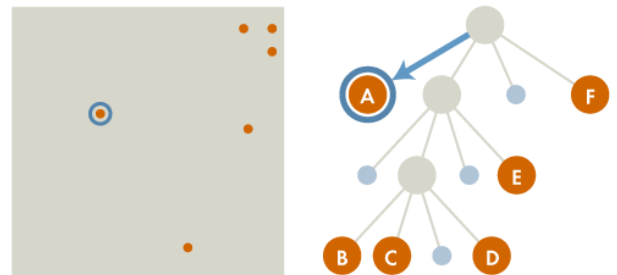
1. Jeigu viršūnė w yra išorinė ir turi joje išsaugotą kūną (jeigu neturi – pereinama prie 2-ojo punkto), be to šis kūnas nėra pats A , tai apskaičiuojama tarp jų veikianti gravitacinė jėga. Pastaroji pridedama prie atstojamosios jėgos. Priešingu atveju (jeigu viršūnė w yra vidinė) pereinama prie 3-iojo punkto.
2. Parenkama sekanti tame pačiame medžio gylyje esanti viršūnė ir algoritmas vykdomas nuo 1-ojo punkto.
3. Jeigu vidinės viršūnės reprezentuojamo regiono pločio s ir atstumo r santykis yra mažesnis už ribinę vertę $\frac{s}{r} < \theta$, tai skaičiuojama jėga veikianti tarp viršūnės w ir kūno A . Priešingu atveju rekursyviai iteruojama per viršūnės w vaikus (pradedant 1-uoju punktu).

□ Panagrinėkime internetiniame puslapyje [22] pateiktą kūno atstojamosios jėgos skaičiavimo pavyzdį, kai sistemą sudaro 6 kūnai, o ribinė sąlyga – $\theta = 0.5$. Esant tokiai parametro θ reikšmei, atstumas r tarp kūno ir viršūnės masės centro turi būti daugiau, nei dvigubai didesnis $r > 2s$, kad būtų galima aproksimuoti jėgos skaičiavimą. Regiono talpinančio visus sistemos kūnus plotis pavyzdyje yra lygus šimtui ilgio vienetų ($s = 100$).

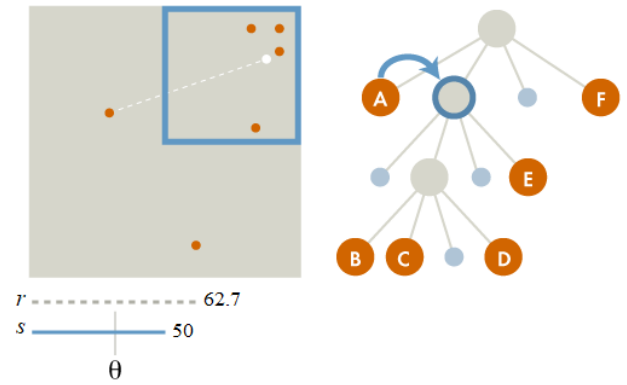
1.) Rekursyvi procedūra pradedama medžio šaknimi (pastaroji yra vidinė). Atstumas tarp kūno A ir viršūnės masės centro $r = 43.1$. Tikrinama ribinė sąlyga: $\frac{100}{43.1} = 2.32 > \theta = 0.5$. Kadangi nelygybė (24) negalioja, tai atliekama iteracija per viršūnės vaikus.



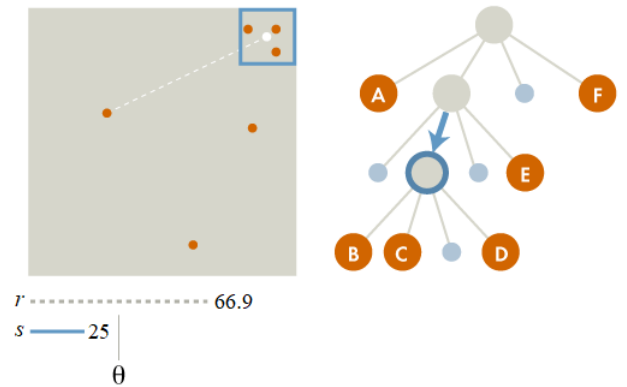
2.) Šiaurės vakarų regionas yra saugomas išorinėje medžio viršūnėje, kurioje patalpintas pats kūnas A . Todėl pereinama prie kitos, tame pačiame medžio gylyje esančios viršūnės.



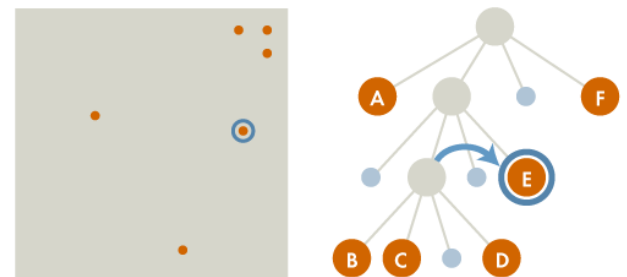
3.) Šiaurės rytų sritį reprezentuoja vidinė viršūnė. Atstumas tarp kūno A ir srities centro yra $r = 62.7$. Tikrinama aproksimacijos sąlyga: $\frac{50}{62.7} = 0.797 > 0.5$. Sąlyga (24) nėra tenkinama – toliau iteruojama per viršūnės vaikus.



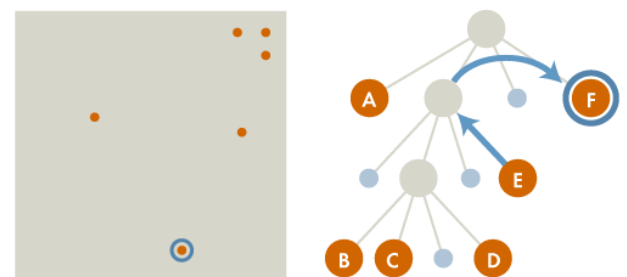
4.) Pirmoji viršūnė yra išorinė, tačiau neturi joms išsaugoto kūno. Tuo tarpu antroji viršūnė yra vidinė ir vizualiai galima matyti, kad atstumas r yra daugiau, nei dvigubai didesnis už regiono plotį s . Tuo įsitikinama apskaičiuojant santykį: $\frac{25}{66.9} = 0.374 < 0.5$. Ribinė sąlyga (24) yra tenkinama, todėl visus kūnus patenkančius į šį regioną galima laikyti vienu objektu. Skaičiuojama jėga tarp kūno A ir viršūnės.



5.) Apskaičiuota jėga pridėjama prie atstojamosios jėgos ir pereinama prie sekančios viršūnės. Patikrinus nustatoma, jog trečioji viršūnė tuščia, o ketvirtoji yra išorinė ir riboja kūno E koordinatas. Tuomet skaičiuojama jėga tarp A ir E , o rezultatas pridedamas prie atstojamosios.



6.) Kadangi viršūnių tame pačiame lygyje daugiau nėra, tai grįžtama į tėvinę viršūnę ir iteruojama toliau. Nustatoma, kad sekanti viršūnė tuščia, o paskutiniojoje saugomas kūnas F . Tuomet skaičiuojama jėga tarp A ir F , bei sumuojamas rezultatas. Tuo užbaigiama kūno A atstojamosios jėgos skaičiavimo procedūra.



6. N kūnų sistemos modeliavimo įrankis Butterfly

Šio darbo autoriaus sukurtas N kūnų sistemos modeliavimo įrankis „Butterfly“ yra pritaikytas eksperimentams su sistemomis, kurios yra studijuojamos astrofizikoje. Tokiose sistemose kūnai sąveikauja gravitacinėmis jėgomis, todėl su jomis atliekami virtualūs eksperimentai yra vadinami gravitacinėmis N kūnų simuliacijomis (angl. *gravitational N-body simulations*). Gravitacinės simuliacijos metu sąveikos tarp sistemą sudarančių dalelių yra tarpusavyje nepriklausomos. Pavyzdžiui, trečiasis kūnas neįtakoja antrojo kūno sąveikos jėgos, kuria pastarasis veikia pirmąjį kūną. Dėl šios priežasties, N kūnų sistemoms galioja superpozicijos principas ir bet kurio kūno atstojamoji jėga, bei naujos koordinatės, gali būti apskaičiuotos nepriklausomai nuo kitų dalelių. Čia turima omenyje, kad skaičiavimus bet kuriame laiko žingsnyje galima atlikti lygiagrečiai.

Lygiagrečiųjų skaičiavimų praktikoje sprendžiamos resursų panaudojimo problemos, siekiant kuo efektyviau išnaudoti kompiuterių klasterio skaičiuojamuosius pajėgumus. N kūnų problemos uždavinio dydis priklauso nuo kūnų sudarančių modeliuojamą sistemą skaičiaus. Kadangi kiekvieno kūno atstojamąją jėgą įmanoma nustatyti nepriklausomai nuo kitų kūnų, tai skaičiavimus galima atlikti skirtinguose įrengiuose. Padalinus sistemos kūnus p skaičiui procesorių, skaičiavimų apimtis atitinkamai sumažėja p kartų. Todėl bazinių operacijų laiko sąnaudos sprendžiant tiesioginiu algoritmu yra įvertinamos taip:

$$t_1 = \gamma \frac{N^2}{p}. \quad (25)$$

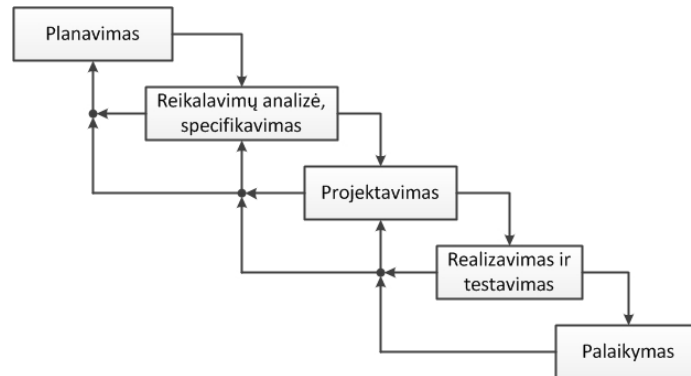
Toks sudėtingumas gaunamas naudojant bendrosios atminties (angl. *shared memory*) lygiagrečiųjų kompiuterį. Tuo tarpu skaičiuojant su paskirstytosios atminties (angl. *distributed memory*) lygiagrečiaisiais kompiuteriais reikia atsižvelgti į duomenų perdavimo sąnaudas t_2 . Pastarosios bendruoju atveju įvertinamos taip [23]:

$$t_2 = (\alpha + \beta N)f(p),$$

čia α – pranešimo išsiuntimo starto laikas (angl. *start time*); β – vieno elemento siuntimo laikas (angl. *bandwith*); p – skaičiavimams naudojamų procesorių skaičius; $f(p)$ – funkcija, kuri priklauso nuo lygiagrečiojo kompiuterio tinklo topologijos ir persiuntimo mechanizmo. Jeigu yra naudojamas *all-to-all* persiuntimas ir kompiuterių tinklas yra žvaigždinis, tai gaunamos itin didelės persiuntimo sąnaudos:

$$t_2 = (\alpha + \beta N)p^2. \quad (26)$$

Taikomojoje programoje „*Butterfly*“ įdiegti du N kūnų problemos sprendimo algoritmai: tiesioginis ir Barnes'o-Hut'o (algoritmai plačiau aprašyti penktajame darbo skyriuje). Jų funkcionalumas realizuotas *BruteForce* ir *BarnesHut* klasėse. Algoritmų lygiagrečiųjų versijų realizacijai panaudota duomenų perdavimo sąsaja MPI (angl. *message passing interface*) [30].



14 pav. Agile metodologijos etapų schema

Programinės įrangos „*Butterfly*“ kūrimui buvo taikoma Agile metodologija (14 pav.), kuri siūlo svarbiausius sistemos kūrimo etapus vykdyti iteraciniu būdu. Pateikiame Agile metodologijos etapų aprašymus [25]:

- Pirmajame etape generuojama projekto idėja, numatomas terminas iki kada turi būti pasiektas norimas rezultatas, planuojami resursai.
- Antrojo etapo metu analizuojama kokias funkcijas turi atlikti sistema, bei kokie apribojimai jai taikomi. Analizė apibendrinama parašant reikalavimų specifikacijos dokumentą.
- Trečiajame etape sistema projektuojama: sudaromas jos objektinis modelis, braižomos klasių diagramos ir kt.
- Ketvirtasis etapas yra skirtas suprojektuotos sistemos programavimui ir testavimui. Tikrinama, ar sukurti komponentai ir bendrai visa sistema, atitinka reikalavimus.
- Penktasis etapas prasideda, kai sistema yra viešai pristatoma jos vartotojams. Pastarieji naudodamiesi produktu aptinka kūrimo metu neištaisytas klaidas, taip pat siūlo patobulinimus, kelia naujus reikalavimus sistemai.

Iteratyvus etapų vykdymas yra naudojamas ne tik kuriant programinę įrangą. Šiuolaikinė matematinio modeliavimo metodologija taip pat siūlo tyrimo metu grįžti į praeitus etapus. Kiekvienoje iteracijoje tobulinamos ir modifikuojamos sudarytos schemos, modeliai ir taikomi algoritmai, kad būtų panaudotos naujausios greitai besivystančios kompiuterinės technologijos, bei moksliniai metodai ir išvengta galimų klaidų ar netikslumų. Taip siekiama geresnio rezultato – virtualaus objekto, kuris naudojamas modeliuojamo realiojo objekto pažinimui, prognozei, projektavimui ir valdymui.

6.1. Programos nustatymai

Didžiojoje dalyje taikomųjų programų su kuriomis susiduriama kasdienėje veikloje, vartotojui yra leistina pasirinkti nustatymus, pagal kuriuos yra vykdomos programoje įdiegtos procedūros. „Butterfly“ nustatymai realizuoti pasitelkus XML (angl. *Extensible Markup Language*). Ši duomenų struktūrų bei jų turinio aprošomoji kalba yra rekomenduojama W3C konsorciumo (angl. *World Wide Web Consortium*), kuris kuria standartus taikomus žiniatinklyje [26].

XML dokumentą sudaro blokai, kurie hierarchiškai saugomi medžio duomenų struktūroje. Blokai yra sudaryti iš aprašų (angl. *markup*) ir programoje naudojamos informacijos. Pateikiame „Butterfly“ nustatymų failo „Settings.xml“ struktūrą:

```
<?xml version="1.0"?>
<Settings>
  <SystemDimensions>3</SystemDimensions>
  <BodiesCount>100</BodiesCount>
  <IterationsCount>1</IterationsCount>
  <StepSize>0.1</StepSize>
  <RegularizationParameter>0.0001</RegularizationParameter>
  <GravitationalConstant>1.0</GravitationalConstant>
  <Algorithm>
    <Name>BarnesHut</Name>
    <Parameters>0.5</Parameters>
  </Algorithm>
  <InitialData random="true">
    <InputFile>InitialData.dat</InputFile>
    <RandomDataGenerator save="true">
      <OutputFile>InitialData.dat</OutputFile>
      <PlummerModel>
        <TotalMass>1.0</TotalMass>
        <CoreRadius>1.0</CoreRadius>
      </PlummerModel>
    </RandomDataGenerator>
  </InitialData>
  <Results>
    <Coordinates save="true">
      <OutputFile>Coordinates.dat</OutputFile>
    </Coordinates>
    <Energy save="true">
      <OutputFile>Energy.dat</OutputFile>
    </Energy>
    <ElapsedTime save="true" />
  </Results>
</Settings>
```

Programų moduliai naudojami perskaityti XML dokumentus ir suteikti prieigą prie juose saugomos informacijos, vadinami XML procesoriais (angl. *XML processor*). Moduluose aprašytos klasės suteikia galimybę programuotojui išnagrinėti (angl. *parse*) XML dokumentus ir įkelti informaciją į programą taikant kelis pagrindinius metodus. XML procesorių metoduose realizuoti sudėtingi mechanizmai, kurie išnagrinėja faile įrašytus simbolius pagal XML sintaksę.

„*Butterfly*“ yra naudojamas A. Kapoulkine'o sukurtas procesorius „*pugixml 1.8*“ [27]. Šis modulis yra platinamas pagal MIT licensiją - tai atvirojo kodo programinė įranga, kuri be jokių apribojimų gali būti integruojama į kitus projektus. Pateikiame dviejų pagrindinių „*pugixml 1.8*“ klasių *xml_document* ir *xml_node*, bei kelių dokumento „*Settings.xml*“ nagrinėjimui panaudotų metodų, pavyzdžius.

Klasė *xml_document* reprezentuoja visą XML dokumentą ir yra medžio duomenų struktūros šaknis. Norint į programą įkrauti failą, sukuriamas XML dokumento objektas ir iškviečiamas metodas *load_file()*:

```
xml_document* cDocument = new xml_document;  
cDocument->load_file("Settings.xml");
```

Toliau nuskaitomas pagrindinis dokumento blokas, šiuo atveju tai yra *<Settings>*. Atskirus dokumento blokus reprezentuoja *xml_node* klasė. *<Settings>* yra medžio šaknies vaikas, kuriam nuskaityti iš duomenų struktūros taikomas metodas *child()*:

```
xml_node cSettings = cDocument->child("Settings");
```

Tarkime, kad programa nori sužinoti kokį algoritmą vartotojas pasirinko *N* kūnų problemos sprendimui ir kokie yra to algoritmo parametrai. *<Algorithm>* yra bloko *<Settings>* vaikas, todėl jo nuskaitymui taip pat naudojamas metodas *child()* su atitinkama parametro reikšme:

```
xml_node cAlgorithm = cSettings->child("Algorithm");
```

Programą dominanti informacija (algoritmo pavadinimas, bei parametrai) saugoma atskiruose blokuose, kurie hierarchinėje struktūroje yra priskirti *<Algorithm>*. Čia metodais *text()*, bei *get()* iš *<Name>* ir *<Parameters>* grąžinamos simbolių eilutės su blokuose saugomais duomenimis:

```
const char* strAlgorithmName = cAlgorithm.child("Name").text().get();  
const char* strParameters = cAlgorithm.child("Parameters").text().get();
```

Programos kodą, kuris išnagrinėja visą *Settings.xml* dokumentą ir paruošia nustatymus naudojimui programoje, galite rasti *Settings.cpp*, bei *Settings.h* projekto failuose. Daugiau informacijos apie „*pugixml 1.8*“ naudojimą pateikta tinklalapyje [27].

7. Skaičiuojamieji eksperimentai

7.1. Empiriniai sudėtingumo įverčiai

Atlikome keletą eksperimentų, kurių metu matavome simuliacijų vykdymo laiką. Tikslas – iš empirinių duomenų nustatyti nuosekliųjų tiesioginio ir Barnes'o-Hut'o algoritmų sudėtingumo įverčius. Pradiniai duomenys eksperimentams sugeneruoti atsitiktinai, pagal tolygųjį skirstinį $X \sim U(a, b)$, kuriame tikimybė apskaičiuojama šia tankio funkcija:

$$p(x) = \begin{cases} 0, & x < a \\ \frac{1}{b-a}, & a \leq x \leq b \\ 0, & x > b \end{cases}$$

Kūnų masės parinktos intervale $m_i \in (0.0, 1.0]$, koordinatės – $x_{i_k} \in [-0.5, 0.5]$, greičiai – $v_{i_k} \in [-0.01, 0.01]$, čia $i = 1, \dots, N$, $k = 1, 2$, gravitacijos konstanta – $G = 1.0$.

Pirmiausiai pateikiame nuosekliojo tiesioginio algoritmo skaičiavimų laiko matavimus. Primename, kad šio algoritmo asimptotinis sudėtingumas yra $\mathcal{O}(N^2)$. Eksperimentai atlikti su skirtingų dydžių uždaviniais, pradedant $N_1 = 500$ ir kiekvienoje iteracijoje didinant jį du kartus $N_i = 2N_{i-1}$. Atlikome po penkias simuliacijas su kiekvienu N_i ir apskaičiavome laiko vidurkius, kad gautumėme geresnę statistiką. Rezultatus pateikėme 1 lentelėje ir 16 pav.

1 lentelė Tiesioginio nuosekliojo algoritmo vykdymo laiko sąnaudos

i	N_i	T_1	T_2	T_3	T_4	T_5	$\langle T \rangle_i$	$\langle T \rangle_i / \langle T \rangle_{i-1}$
1	500	7.84356	7.86384	7.8645	7.89084	7.86210	7.864968	–
2	1000	31.5271	35.4121	35.3752	35.3717	35.3983	34.61688	4.40140
3	2000	139.183	142.567	141.855	141.721	136.989	140.463	4.04956
4	4000	566.435	566.387	567.143	522.590	522.877	549.0864	3.909118
5	8000	2263.65	2268.33	2267.70	2264.94	2279.11	2268.746	4.131856
6	16000	9077.48	9084.51	9077.18	9078.58	9095.06	9082.562	4.00334
7	32000	36718.2	36704.1	36503.7	33069.8	32750.3	35149.22	3.86997

Kaip matome iš paskutiniojo duomenų lentelės stulpelio, padidinus uždavinio dydį du kartus skaičiavimų laikas padidėja apytiksliai keturis kartus. Taigi, gavome rezultatą panašų į tokį, kokio ir buvo galima tikėtis remiantis tiesioginio algoritmo asimptotiniu sudėtingumo įverčiu.

Barnes'o-Hut'o nuosekliuoju algoritmu eksperimentai atlikti naudojant tuos pačius pradinių duomenų failus, kaip ir skaičiuojant tiesioginiu algoritmu. Parametras, nuo kurio priklauso

koku atstumu nuo stebėjimo taško turi būti kūnų grupė, kad ją būtų galima laikyti vienu objektu, parinktas pastovus – $\theta = 0.5$. Matavimai su tuo pačiu uždavinio dydžiu atlikti po penkis kartus, bei apskaičiuotas laiko vidurkis. Taip pat atliktas palyginimas siekiant nustatyti kiek kartu padidėjo simuliacijos trukmė padvigubinus uždavinio dydį. Rezultatus pateikiame 2 lentelėje.

2 lentelė Barnes'o-Hut'o nuosekliojo algoritmo vykdymo laiko sąnaudos

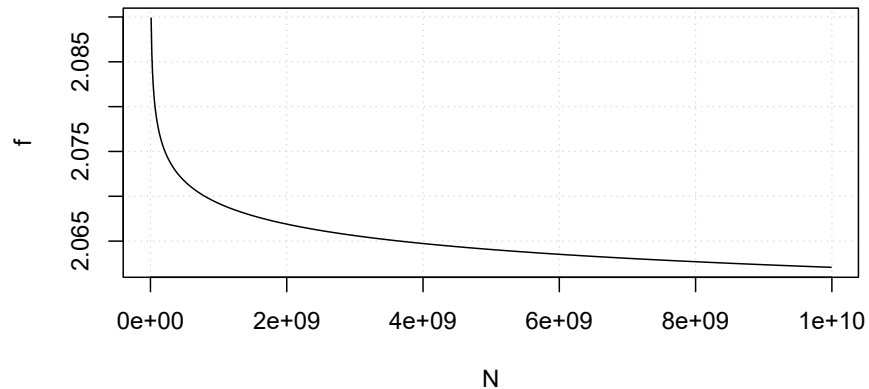
i	N_i	T_1	T_2	T_3	T_4	T_5	$\langle T \rangle_i$	$\langle T \rangle_i / \langle T \rangle_{i-1}$
1	500	3.61019	3.60899	3.61541	3.6098	3.62446	3.61377	–
2	1000	9.57608	9.59992	9.58437	9.57202	9.59059	9.584596	2.652243
3	2000	24.0672	24.0332	24.1066	24.0848	24.0885	24.07606	2.511954
4	4000	56.2463	56.1006	50.4590	50.5725	58.0876	54.2932	2.25507
5	8000	123.674	124.913	124.890	124.677	124.518	124.5344	2.293738
6	16000	261.146	262.615	262.404	263.260	262.419	262.3688	2.106798
7	32000	580.852	588.125	588.400	588.219	588.916	586.9024	2.236937

Skaičiuojant su algoritmais, kurių asimptotinis sudėtingumas yra $\mathcal{O}(N \log N)$, padvigubinus uždavinio dydį N , vykdymo laiko sąnaudos taip pat turėtų padvigubėti. Taip yra todėl, kad

$$f(N) = \frac{N \cdot \log N}{\frac{N}{2} \cdot \log \frac{N}{2}} = 2 \cdot \frac{\log N}{\log N - \log 2} = 2 \cdot \frac{\log N}{\log N - 1},$$

kai $N \rightarrow \infty$, tai $f(N) \rightarrow 2$, nes $\frac{\log N}{\log N - 1} \rightarrow 1$.

Iš duomenų lentelės matyti, jog konvergavimas sulėtėjo ketvirtajame eksperimente, kadangi santykis svyruoja apie $\langle T \rangle_i / \langle T \rangle_{i-1} \approx 2.2$, kai $i = 4, \dots, 7$. Tačiau tikėtina, jog atlikus eksperimentus su didesniais uždaviniais, būtų fiksuojamas artėjimas prie teorinio įverčio, nes funkcijos $f(N)$ konvergavimas yra lėtas (15 pav.). Pavyzdžiui, kai $N = 10^{10}$, tai $f(N) = 2.06208$.



15 pav. Funkcijos $f(N)$ konvergavimas

7.2. Lygiagretieji skaičiavimai

Eksperimentai atlikti su lygiagrečiosiomis tiesioginio ir Barnes'o-Hut'o algoritmų versijomis, naudojant skirtingą procesorių skaičių: $p = 2, 4, 8, 16$. Skaičiavimo laiko matavimams pasirinkome tokius pačius uždavinio dydžius, kaip ir skaičiuojant su nuosekliomis versijomis, kad galėtumėme nustatyti algoritmų spartinimo, bei efektyvumo koeficientus. Spartinimo koeficientas parodo pagreitėjimą, kuris yra pasiekiamas uždavinį sprendžiant lygiagrečiuoju algoritmu ir yra skaičiuojamas pagal šią formulę:

$$S_p = \frac{T_{p=1}}{T_p},$$

čia p – procesorių skaičius; $T_{p=1}$ – nuosekliojo algoritmo vykdymo laikas. Efektyvumo koeficientas parodo, kokia dalis skaičiavimo įrenginių galios naudojama uždavinio sprendimui lygiagrečiuoju algoritmu. Šis koeficientas skaičiuojamas taip:

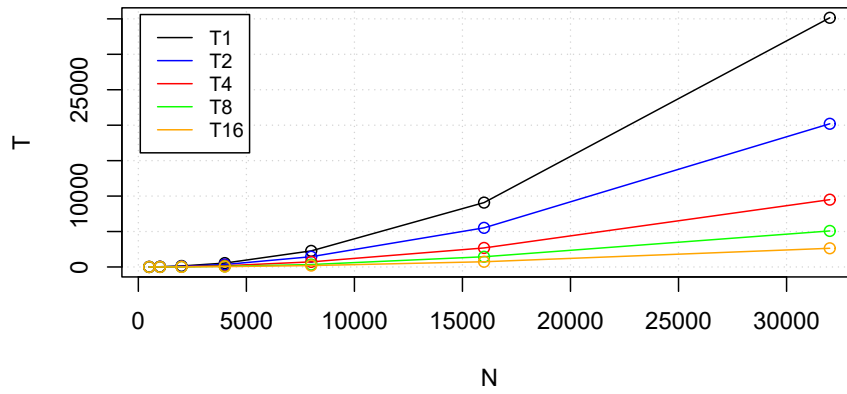
$$E_p = \frac{S_p}{p}.$$

Lygiagrečiųjų algoritmų efektyvumas mažėja, kai tokiam pačiam uždavinio dydžiui, naudojamas didesnis procesorių skaičius. To priežastis yra augančios duomenų persiuntimo tarp procesorių sąnaudos. Pasiekus tam tikrą p ribą, duomenų persiuntimo laiko prieaugis viršija bazinėms algoritmo operacijoms naudojamo laiko sumažėjimą, todėl bendras vykdymo laikas pradeda augti. Eksperimentų metu tokio reiškinių užfiksuoti nepavyko. Pateikiame laiko matavimus atliktus su tiesioginiu algoritmu.

3 lentelė Tiesioginio lygiagrečiojo algoritmo vykdymo laiko sąnaudos

N_i	$T_{p=2}$	$T_{p=4}$	$T_{p=8}$	$T_{p=16}$
500	5.96904	2.94138	1.58965	0.904286
1000	23.2311	11.6466	5.92759	3.30446
2000	96.5909	46.7074	24.7328	12.4136
4000	425.472	190.018	95.2884	49.7935
8000	1498.26	769.264	385.145	198.014
16000	6019.06	3084.47	1541.48	797.987
32000	26399.8	12494.9	6682.82	3136.76

Idealiu atveju, kai nėra duomenų persiuntimo sąnaudų, lygiagretusis algoritmas vykdyto laiką turėtų paspartinti apytiksliai tiek kartu, kiek skaičiavimams yra naudojama procesorių, t. y. $\frac{N^2}{p}$. Tačiau tokio rezultato būtų galima tikėtis nebent skaičiuojant bendrosios atminties lygiagrečiuoju kompiuteriu, kuriame perdavimo sąnaudos lyginant su paskirstytosios atminties

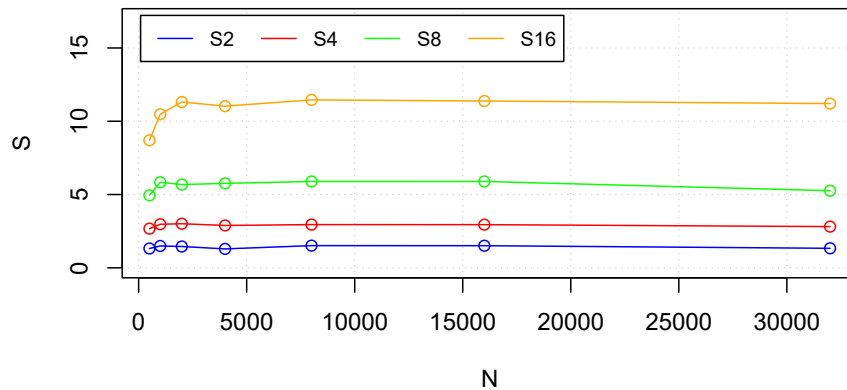


16 pav. Tiesioginio lygiagrečiojo algoritmo vykdymo laikas

kompiuteriu yra mažos. Kita vertus, esant pastoviam procesorių skaičiui ir didėjant uždavinio dydžiui, vis daugiau laiko yra sunaudojama bazinėms operacijoms. Taip yra todėl, kad pranešimo išsiuntimo starto laikas (žr. 26 formulę), kurį sunaudoja procesorius, yra gerokai didesnis už vieno elemento persiuntimo laiką. Amdahl'o dėsnis teigia, kad spartinimo koeficiento riba yra lygi procesorių skaičiui, kai uždavinio dydis artėja į begalybę:

$$\lim_{N \rightarrow \infty} S_p = p$$

Šio dėsningumo iš eksperimentinių duomenų nustatyti nepavyko. To priežastis greičiausiai yra labai neefektyvus duomenų perdavimo mechanizmas, kai visi procesoriai siunčia duomenis visiems kitiems procesoriams (angl. *all-to-all*) (17 pav.).



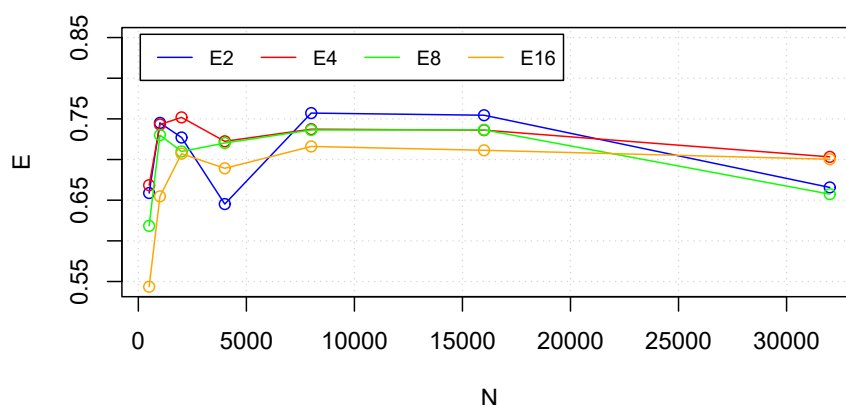
17 pav. Tiesioginio lygiagrečiojo algoritmo spartinimo koeficientas

Lygiagrečiojo algoritmo bendrąjį vykdymo laiką sudaro bazinėms operacijoms t_1 ir duomenų persiuntimui t_2 sunaudojamo laiko suma:

$$T = t_1 + t_2.$$

4 lentelė Tiesioginio lygiagrečiojo algoritmo spartinimo ir efektyvumo koeficientai

N_i	$S_{p=2}$	$E_{p=2}$	$S_{p=4}$	$E_{p=4}$	$S_{p=8}$	$E_{p=8}$	$S_{p=16}$	$E_{p=16}$
500	1.31763	0.658815	2.67390	0.668475	4.94761	0.618451	8.69742	0.543589
1000	1.49011	0.745055	2.97227	0.743068	5.83995	0.729994	10.4758	0.654738
2000	1.51703	0.758515	3.00730	0.751825	5.99429	0.749286	11.3153	0.707206
4000	1.57570	0.783265	3.13731	0.784328	6.08147	0.760184	11.7343	0.733394
8000	1.56653	0.783265	3.15426	0.788565	6.57331	0.821664	12.7448	0.796550
16000	1.64567	0.822835	3.38337	0.845843	6.30086	0.787608	12.3072	0.769200
32000	1.74008	0.870040	3.70191	0.925478	6.91530	0.864413	13.3305	0.8331563



18 pav. Tiesioginio lygiagrečiojo algoritmo efektyvumo koeficientas

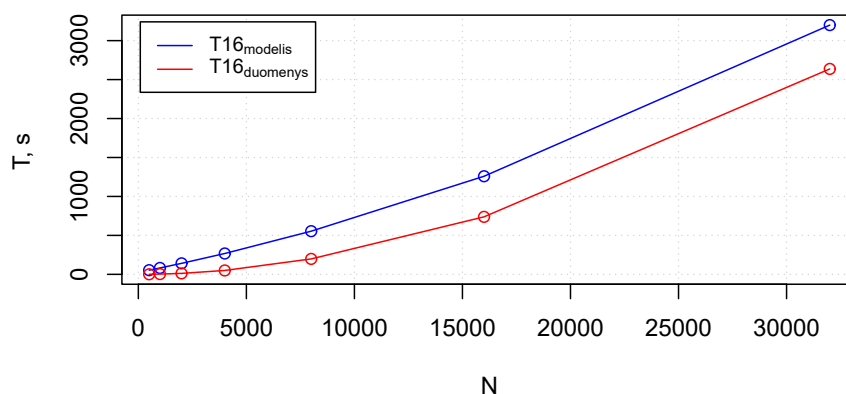
Tiesioginio algoritmo bazinių operacijų (25), bei duomenų perdavimo (26) laiko sąnaudų formules užrašėme ankstesniame skyriuje. Pasinaudodami šiomis formulėmis, galime sudaryti laiko sąnaudų modelį, kurio paskirtis yra prognozuoti simuliacijos laiką.:

$$T_p = \gamma \frac{N^2}{p} + (\alpha + \beta N) p^2.$$

Modelio koeficientus nustatėme iš empirinių duomenų (3 lentelė):

$$\gamma = 2.21 \cdot 10^{-5}, \quad \alpha = 10^{-1}, \quad \beta = 2.15 \cdot 10^{-4}.$$

Modelio ir eksperimentinių duomenų grafikus galite palyginti 19 pav.



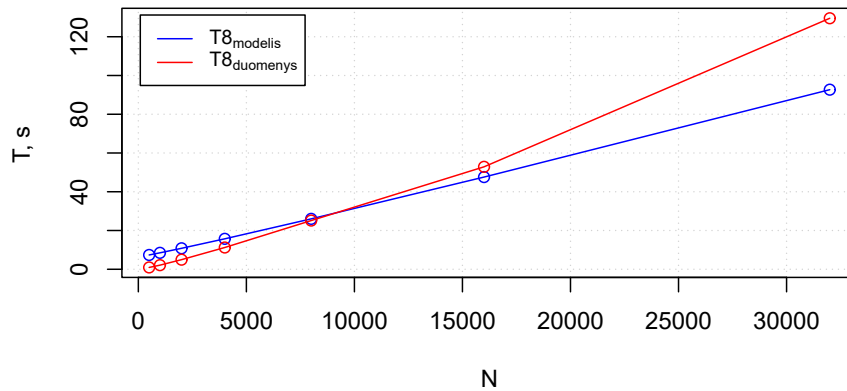
19 pav. Tiesioginio lygiagrečiojo algoritmo laiko sąnaudų modelis

Skaiciuojant su Barnes'o-Hut'o algoritmu pavyko pasiekti pagreitejimą, tačiau ne tokį, kokio norėtusi iš lygiagretaus algoritmo. Barnes'o-Hut'o algoritmo spartinimo ir efektyvumo ko-

5 lentelė Barnes'o-Hut'o lygiagrečiojo algoritmo vykdymo laiko sąnaudos

N_i	$T_{p=2}$	$T_{p=4}$	$T_{p=8}$	$T_{p=16}$
500	2.60077	1.44470	0.937235	0.69560
1000	6.56239	3.55961	2.11017	1.38355
2000	16.2399	8.77376	4.95593	3.06686
4000	38.4395	19.8339	11.2315	6.79023
8000	84.4479	45.1071	25.1323	15.0893
16000	191.768	94.0029	52.8706	32.5964
32000	487.045	221.293	129.536	90.8562

eficientų rezultatai gauti blogesni, nei skaičiuojant su tiesioginiu algoritmu. Pavyzdžiui, eksperimentuojant su šešiolika procesorių, skaičiavimo laiką pavyko paspartinti tik 8 kartus (lyginant su nuosekliuoju Barnes'o-Hut'o algoritmu). Efektyvumas buvo ypač mažas skaičiuojant su dviem procesoriais, 23 pav., bei 6 lentelėje galima matyti, jog jis svyruoja tik 20–40% režiuose. Tokie rezultatai, tik dar kartą parodo, koks neefektyvus yra *all-to-all* duomenų perdavimo mechanizmas. Šiuose literatūros šaltiniuose [28, 29] yra siūlomi efektyvesni metodai.



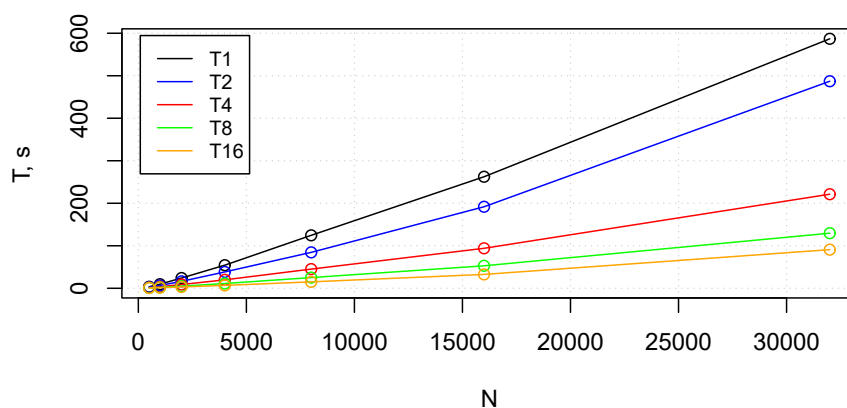
20 pav. Barnes'o-Hut'o lygiagrečiojo algoritmo laiko sąnaudų modelis

Laiko sąnaudų modelyje antrasis narys t_2 yra toks pat, kaip ir tiesioginiame algoritme. Čia skiriasi tik bazinių operacijų vykdymo laikas t_1 , kadangi algoritmo asimptotinis sudėtingumas yra $\mathcal{O}(N \log N)$:

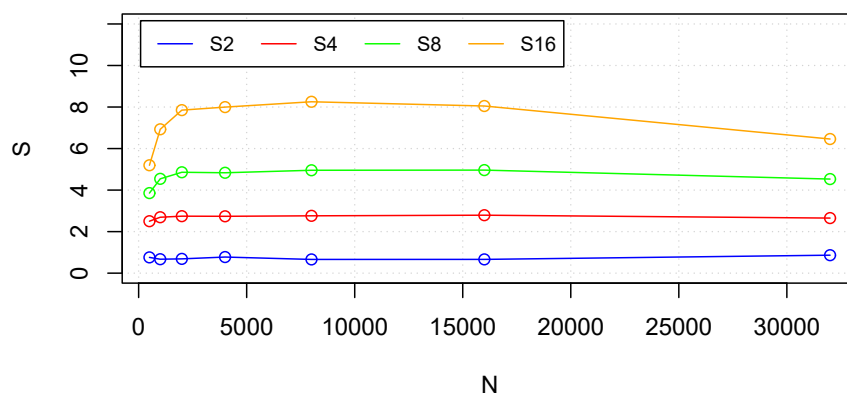
$$T_p = \gamma \frac{N \log N}{p} + (\alpha + \beta N)p^2.$$

Modelio koeficientus nustatėme iš empirinių duomenų (5 lentelė):

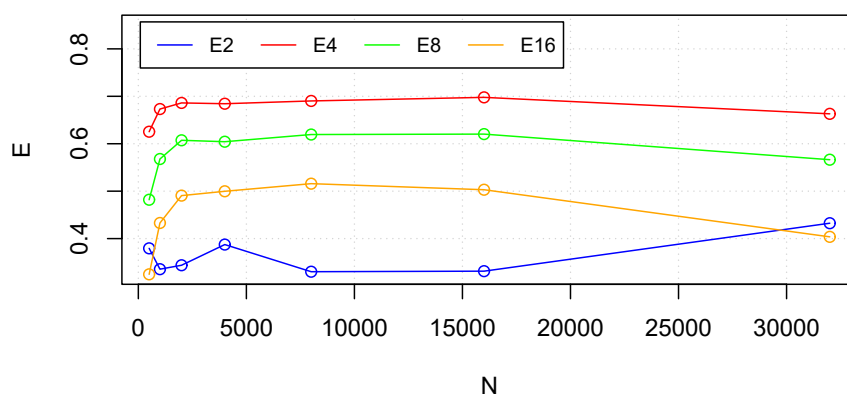
$$\gamma = 9.80 \cdot 10^{-4}, \quad \alpha = 10^{-1} \quad \beta = 1.35 \cdot 10^{-5}.$$



21 pav. Barnes'o-Hut'o lygiagrečiojo algoritmo vykdymo laiko sąnaudos



22 pav. Barnes'o-Hut'o lygiagrečiojo algoritmo spartinimo koeficientas



23 pav. Barnes'o-Hut'o lygiagrečiojo algoritmo efektyvumo koeficientas

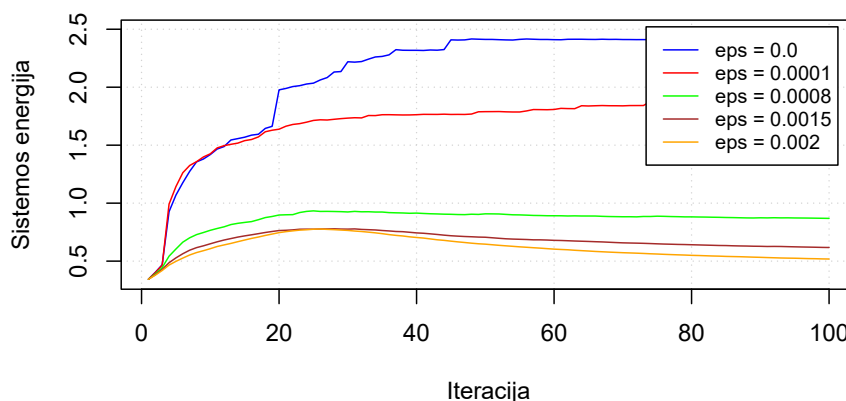
6 lentelė Barnes'o-Hut'o lygiagrečiojo algoritmo spartinimo ir efektyvumo koeficientai

N_i	$S_{p=2}$	$E_{p=2}$	$S_{p=4}$	$E_{p=4}$	$S_{p=8}$	$E_{p=8}$	$S_{p=16}$	$E_{p=16}$
500	1.38950	0.694750	2.50140	0.625350	3.85578	0.481972	5.19518	0.324699
1000	1.46053	0.730267	2.69258	0.673149	4.54210	0.567762	6.92754	0.432971
2000	1.48252	0.741263	2.74410	0.686025	4.85803	0.607254	7.85039	0.490650
4000	1.41243	0.706216	2.73739	0.684349	4.83401	0.604251	7.99578	0.499736
8000	1.47469	0.737345	2.76086	0.690215	4.95515	0.619394	8.25316	0.515823
16000	1.36816	0.684079	2.79107	0.697768	4.96247	0.620309	8.04901	0.503063
32000	1.20503	0.602514	2.65215	0.663038	4.53081	0.566351	6.45969	0.403730

7.3. Uždavinio stabilumas

Fizikoje gerai žinoma, kad uždaroje sistemoje, kurių neveikia jokios išorinės jėgos, pilnutinė mechaninė energija E yra pastovus dydis. Kitais žodžiais tariant, energija iš niekur neatsiranda ir niekur neišnyksta. Tačiau vienos rūšies energija gali virsti kitos rūšies energija. Šią uždaros sistemos savybę apibrėžia Energijos tvermės dėsnis: $E = T + V$, čia T – sistemos kinetinė energija, V – sistemos potencinė energija. Energijos tvermės dėsnis galioja ir N kūnų sistemoms, kuriose jėgos kyla tik dėl gravitacinės kūnų tarpusavio sąveikos. Šią savybę puikiai galima pritaikyti tikrinant N kūnų uždavinio stabilumą.

Eksperimentui sugeneravome $N = 2000$ kūnų sistemą dvimatėje erdvėje. Integravimo žingsnį parinkome $\Delta t = 0.1$, o gravitacijos konstantą prilyginome vienetui $G = 1.0$. Pradinių sąlygų generavimui taikėme Plummer'io modelį (žr. 4.2 poskyrį). Visos sistemos masę ir klasterio branduolio spindulį nustatėme atitinkamai $M = 1.0$ ir $a = 1.0$. Kiekvienos iteracijos metu apskaičiavome sistemos pilnutinę energiją E ir nubraižėme grafiką (24 pav.). Iš grafiko mato-



24 pav. Pilnutinė sistemos energija

me, kad gautasis sprendinys neatitinka realiame pasaulyje veikiančio dėsningumo, nes pilnutinė sistemos energija išaugo net keletą kartų.

Sistemos energijos kiekis tam tikrose iteracijose įgijo didelius šuolius, nes kūnams priartėjus vienas prie kito mažu atstumu, stipriai išaugo jėgos, kylančios dėl tarpusavio gravitacinės sąveikos. Šį reiškinį galima sušvelninti jėgos skaičiavimo formulėje įvedant papildomą parametą. Prie atstumo $r_{i,j}$ pridėjus mažą dydį ϵ apribojama galimybė kūnams įgyti be galo didelius greičius:

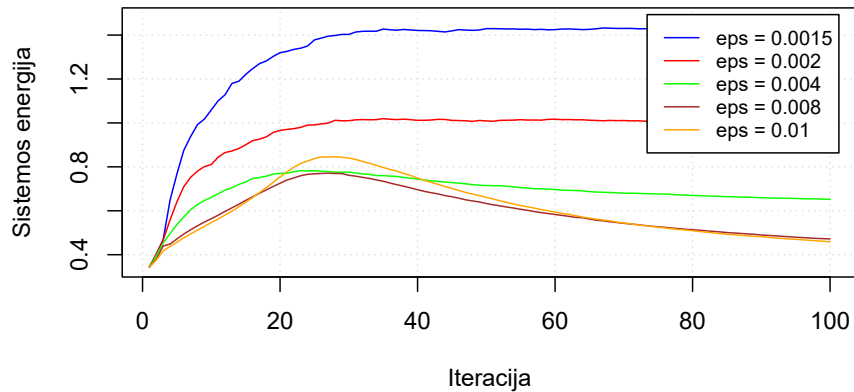
$$F_{i,j} = G \frac{m_i m_j}{r_{i,j}^2 + \epsilon}.$$

Dydis ϵ reguliarizuoja uždavinį padidindamas sistemos stabilumą ir yra vadinamas reguliarizacijos parametru. Nustatėme $\epsilon = 0.0001$ ir su tomis pačiomis pradinėmis sąlygomis pakartojome

eksperimentą. Iš gautųjų rezultatų įsitikinta, jog nežymiai padidinus atstumą tarp kūnų, galima stabilizuoti sistemą. Pakartojome simuliaciją dar keletą kartų su skirtingais ϵ , rezultatus pateikėme 24 pav.

7.4. Barnes'o-Hut'o ribinė sąlyga

Eksperimentui su Barnes'o-Huto ribine sąlyga θ pasirinkome 7.3 skyrelyje naudotas pradinės sąlygas su regularizacijos parametru $\epsilon = 0.0015$. Tačiau atlikę skaičiavimus nustatėme, jog sprendžiant su Barnes'o-Hut'o algoritmu, tokia regularizacijos parametro reikšmė yra per maža, kad uždavinys būtų stabilus. Todėl atlikome keletą iteracijų su skirtingomis reikšmėmis ir nustatėme optimalesnį variantą $\epsilon = 0.01$. Priešingu atveju, dėl dalelių, kurios įgauna didelius greičius ir gerokai nukrypsta nuo tikrųjų savo skriejimo trajektorijų, nebūtume galėję įvertinti algoritmo paklaidos.



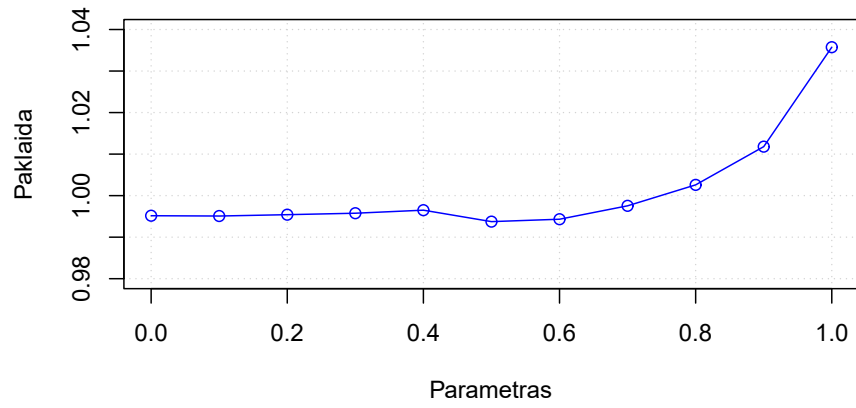
25 pav. Pilnutinė sistemos energija.

Tiesioginis algoritmas yra tiksliausias žinomas būdas spręsti N kūnų problemą, todėl jo sprendinį galima naudoti kitų algoritmų tikslumui įvertinti. Išmatavome tiesioginio algoritmo skaičiavimų laiką, kai $N = 2000$ ir simuliacijos nustatymuose nurodyta atlikti 100 iteracijų. Gautasis laikas yra $T = 12.3175$ s. Eksperimentams parinkome tokį patį integravimo žingsnį $\Delta t = 0.1$, kaip ir ankstesnių sačiavimų metu. Atlikome 11 bandymų su skirtingomis ribinės sąlygos parametro reikšmėmis. Bandymų metu matavome Barnes'o-Hut'o algoritmo vykdymo laiką ir tikslumą, lyginant su tiesioginiu algoritmu.

Iš 7 lentelės, bei 27 pav. matyti, jog paklaida nežymiai didėja parametrai artėjant į vienetą $\theta \rightarrow 1$. Čia paklaidą skaičiavome, kaip didžiausią atstumą tarp dalelių visos simuliacijos metu:

$$\max ||\vec{x}_i - \vec{x}_i^{BH}||, \quad i = 1, \dots, 2000,$$

čia $\vec{x}_i$ – i -osios dalelės koordinačių vektorius, skaičiuojant tiesioginiu algoritmu; $\vec{x}_i^{BH}$ – i -osios dalelės koordinačių vektorius, skaičiuojant Barnes'o-Hut'o algoritmu.

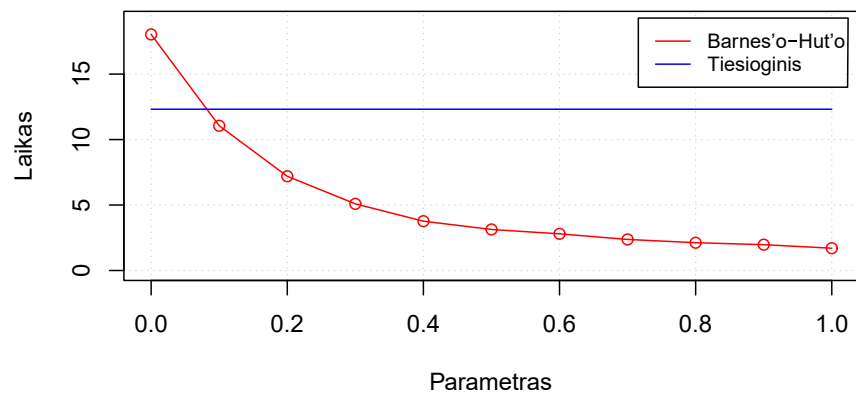


26 pav. Paklaidos priklausomybė nuo parametro θ

7 lentelė Barnes'o-Hut'o algoritmo paklaidos ir vykdymo laiko priklausomybė nuo θ

θ	δ_θ	T_θ	T/T_θ
1.0	1.035745	1.70219	7.23627
0.9	1.011798	1.97241	6.24490
0.8	1.002593	2.12186	5.80505
0.7	0.997548	2.37272	5.19130
0.6	0.994318	2.80004	4.39904
0.5	0.993742	3.13384	3.93048
0.4	0.996499	3.76622	3.27052
0.3	0.995767	5.08990	2.41999
0.2	0.995414	7.19511	1.71193
0.1	0.995090	11.0537	1.11433
0.0	0.995163	18.0299	0.68317

Laiko atžvilgiu, Barnes'o-Hut'o algoritmo pranašumas mažėja, ribinės sąlygos paramet-
rui artėjant į nulį $\theta \rightarrow 0$. Kai parametras yra lygus nuliui $\theta = 0$, Barnes'o-Hut'o algoritmas
tampa lėtesnis už tiesioginį, nes medžio duomenų struktūros konstravimas užima nemažai papild-
domo laiko.



27 pav. Skaičiavimų laiko priklausomybė nuo parametro θ

8. Išvados

N kūnų problemos taikymai fizinių reiškinių tyrinėjimams molekuliniame lygmenyje padeda vystyti tokioms mokslo šakoms kaip nanotechnologijos, reologija, ar biochemija. Naudojant molekulinės dinamikos simuliacijas tyrinėjamos skysčių, dujų, ar kietųjų medžiagų savybės. Kaip tiriamoji priemonė, ji padeda patikrinti teorines hipotezes naudojant kompiuterio skaičiuojamąją galią. Simuliacijos rezultatai patvirtinami atliekant laboratorinį eksperimentą, kuris atskirais atvejais gali būti labai brangus. Taigi ekonomiškiau pirma eksperimentuoti virtualiai ir tik vėliau persikelti į realią laboratoriją.

Kadangi N kūnų problemos neina išspręsti analitiškai, tai jos sprendimui naudojamas kompiuteris. Tiksliau, dažniausiai naudojamas daugiau, nei vienas kompiuteris, nes kūnų skaičius praktiniuose uždaviniuose yra didelis. Molekulinės dinamikos simuliacijose jis gali būti ir šeštos, ar septintos eilės, o gravitacinėse simuliacijose, net ir devintos eilės dydis. Todėl studijuojami greitesni N kūnų problemos sprendimo algoritmai, bei galimybės juos išlygiagretinti. Siekiant spartumo ir efektyvumo, naudojant lygiagrečiojo kompiuterio skaičiuojamąją galią.

Tiesioginio N kūnų problemos sprendimo algoritmo asimptotinis sudėtingumo įvertis yra $\mathcal{O}(N^2)$. Tai patvirtina ir atliktas eksperimentas, kuris plačiau aprašytas septintajame darbo skyriuje. Nustatyta, kad padvigubinus uždavinio dydį skaičiavimų laikas padidėja keturis kartus. Barnes'o-Hut'o algoritmo empirinis sudėtingumo įvertinimas taip pat atitinka teorines išvadas. Lyginant su tiesioginiu algoritmu, Barnes'o-Hut'o parodė pranašumą skaičiavimų laiko atžvilgiu. Tačiau egzistuoja algoritmai, su kuriais uždavinio sudėtingumą galima sumažinti iki pirmosios eilės, t. y. $\mathcal{O}(N)$. Tokie algoritmai yra vadinami greitaisiais daugiapoliais algoritmais (angl. *fast multipole methods*) ir yra plačiai studijuojami šių dienų mokslininkų. Kaip trūkumą būtų galima išskirti sudėtingą jų realizaciją taikomojoje programoje. Teorinė greitųjų daugiapolių algoritmų apžvalga, bei su jais atlikti skaičiuojamieji eksperimentai aprašyti šiame darbe [24].

Skaičiuojant lygiagrečiaisiais algoritmais labai neefektyvu naudoti *all-to-all* duomenų perdavimo mechanizmą. Tai patvirtina ir 7-ajame darbo skyriuje aprašyti eksperimentai. Ypač neefektyvus resursų panaudojimas gautas skaičiuojant su dviem procesoriais, Barnes'o-Hut'o algoritmu. Čia efektyvumo koeficientas esant skirtingiems uždavinio dydžiams svyravo nuo 20% iki 40% (23 pav.). Tačiau tokio mažo efektyvumo priežastimi gali būti ir netinkamas darbų paskirstymas procesoriams, kai vienas procesorius iteracijoje užbaigia skaičiavimus anksčiau už kitą ir tam tikrą laiko tarpą neatlieka jokio naudingo darbo.

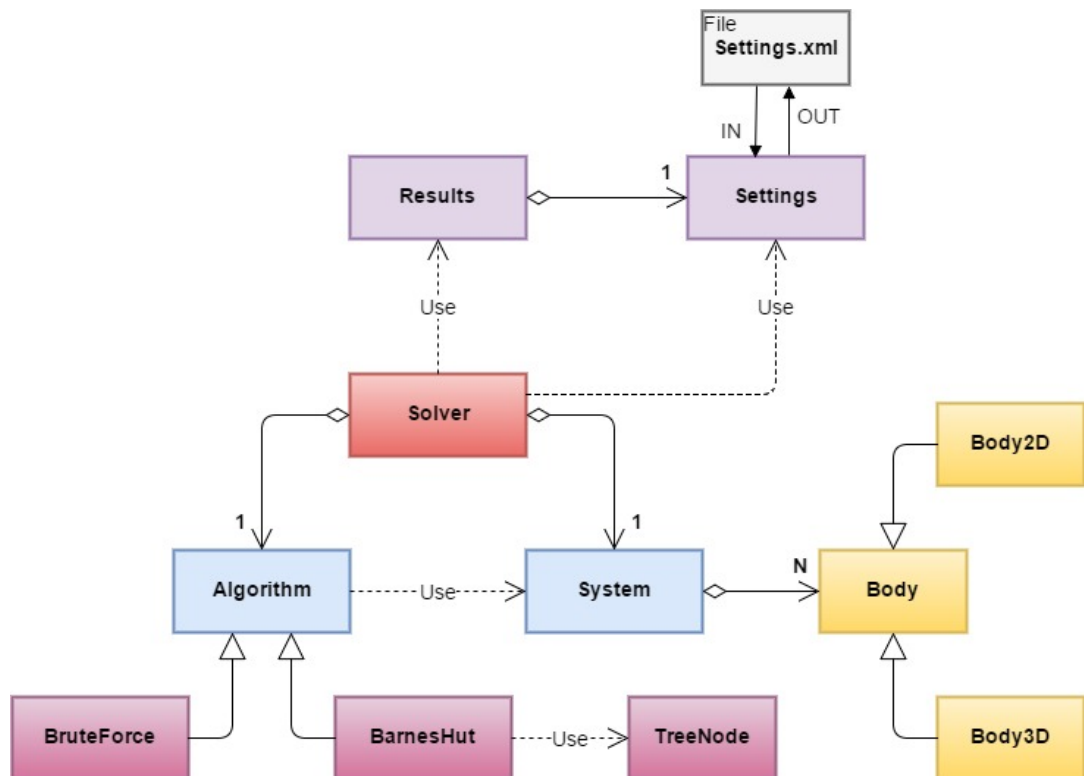
Literatūra

- [1] Papadatos, K. *The Equations of Planetary Motion and their Solution*. The General Science Journal, 2016. ISSN: 1916-5382
- [2] From Old Books [žiūrėta 2017-02-09].
Prieiga internete: <http://www.fromoldbooks.org/Aubrey-HistoryOfEngland-Vol3/pages/vol3-401-Sir-Isaac-Newton/>
- [3] Liu, S. *From Geocentric to Heliocentric Model of the Universe, and the Alternatives Perspectives*. China: The Education University of Hong Kong, 2005.
- [4] Aarseth, S. J. *Gravitational N-Body Simulations*. USA, New York: Cambridge University Press, 2003.
- [5] Dehnen, W., Read, J. *N-body simulations of gravitational dynamics*. USA, New York: Cornell University Library, 2011.
- [6] NASA/IPAC Extragalactic Database [žiūrėta 2017-04-23].
Prieiga internete: https://ned.ipac.caltech.edu/level5/Biviano2/Biviano5_1.html
- [7] Meyer, K. R., Hall, G. R., Offin, D. *Introduction to Hamiltonian Dynamical Systems and the N-Body Problem*. USA, New York: Springer Science+Business Media, 2009.
- [8] Bogdanovičius, A. *Fizikos pagrindai inžinerijoje. 1 dalis*. Vilnius: Technika, 2010.
- [9] Martinėnas, B. *Fizika*. Vilnius: Technika, 2008.
- [10] Hinchliffe, A. *Molecular Modelling for Beginners*. UK, Manchester: Wiley, 2003.
- [11] Oxford Dictionaries [žiūrėta 2016-01-19].
Prieiga internete: <https://en.oxforddictionaries.com/definition/model>
- [12] Leach, A. R. *Molecular Modelling. Principles and Applications*. USA, New Jersey: Prentice Hall, 2001.
- [13] Xie, C. *Molecular Dynamics for Everyone: A Technical Introduction to the Molecular Workbench Software*. USA: Concord Consortium.
- [14] Naab, T. *N-body Simulations and Galaxy Formation*. Germany: Ludwig Maximilian University of Munich, 2006.

- [15] Larson, R. B. *Galaxy Formation and Evolution*. USA: Yale University.
- [16] Western Washington University [žiūrėta 2017-04-22].
Prieiga internete: https://www.wwu.edu/skywise/a101_galaxies.html
- [17] Hubble Space Telescope [žiūrėta 2017-04-22].
Prieiga internete: <https://www.spacetelescope.org/images/heic99020/>
- [18] Hut, P., Makino, J. *Initial Conditions: Plummer's Model*. USA: The Art of Computational Science, 2005.
- [19] Barnes, J. E. *Gravitational Fields*. USA: University of Hawaii, 2005.
- [20] Čiegis, R. *Diferencialinių lygčių skaitiniai sprendimo metodai*. Vilnius: Technika, 2003.
- [21] Ventimiglia, T., Wayne, K. *The Barnes-Hut Algorithm*. USA: Princeton University, 2003.
- [22] arbor.js [žiūrėta 2016-10-13]. Prieiga internete: <http://arborjs.org/docs/barnes-hut>
- [23] Čiegis, R. *Lygiagrečiai algoritmai ir tinklinės technologijos*. Vilnius: Technika, 2005.
- [24] Darve, E. *The Fast Multipole Method: Numerical Implementation*. USA: Stanford University, 1999.
- [25] Shore, J., Warden, S. *The Art of Agile Development*. USA: O'Reilly, 2008.
- [26] Extensible Markup Language (XML) [žiūrėta 2017-03-17].
Prieiga internete: <https://www.w3.org/XML/>
- [27] pugixml [žiūrėta 2017-03-17]. Prieiga internete: <http://pugixml.org/>
- [28] Liu, P., Bhatt, S. N. *Experiences with Parallel N-Body Simulation*. USA: IEEE Transactions on Parallel and Distributed Systems, 2000.
- [29] Grama, A., Kumar, V., Sameh, A. *Scalable parallel formulations of the Barnes-Hut method for n-body simulations*. USA: Parallel Computing, 1997.
- [30] Open MPI [žiūrėta 2017-02-12]. Prieiga internete: <https://www.open-mpi.org/>

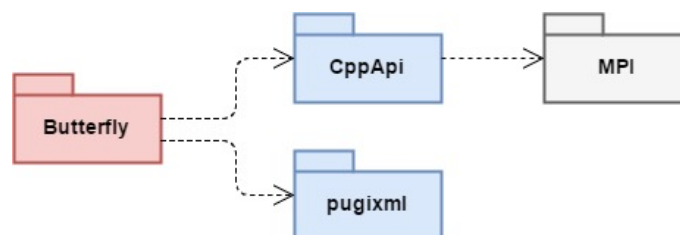
Priedai

Pridedame N kūrų sistemos modeliavimo įrankio „*Butterfly*“ kodą. Visas kodas yra parašytas C++ kalba, laikantis objektinio programavimo principų. Atskirų programos dalių aprašymai anglų kalba pateikti kode, prie klasių, metodų ar kintamųjų deklaracijų. Baigiamojo darbo failuose galite rasti programos dokumentaciją HTML formatu. Dokumentacijos generavimui panaudotas automatinis programinės įrangos dokumentacijos generavimo įrankis „*Doxygen*“.



28 pav. Programos „*Butterfly*“ objektinis modelis

Programoje yra naudojamas šio darbo autoriaus sukurtas „*CppApi*“ paketas, kuriame realizuotos klasės, skirtos: duomenų išsaugojimui ir įkėlimui iš failų (*FileStream*); sistemos laikrodžio ir kalendoriaus informacijos nuskaitymui (*Clock*); programos operacijų vykdymo laiko matavimams (*Stopwatch*); patogesniai darbui su MPI sąsaja (*MpiWrapper*). Paketo „*CppApi*“ kodas nėra pateiktas darbo prieduose, tačiau jį galima rasti „*Butterfly*“ projekto failuose.



29 pav. Programos „*Butterfly*“ paketų diagrama