KAUNO TECHNOLOGIJOS UNIVERSITETAS
ELEKTROS IR ELEKTRONIKOS FAKULTETAS

Tomas Stirblys

SAVOSIOS KETURRAČIO ROBOTO LOKALIZACIJOS ALGORITMO KŪRIMAS IR ANALIZĖ

Baigiamasis magistro projektas

Vadovas
Doc. dr. Renaldas Urniežius

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS
ELEKTROS IR ELEKTRONIKOS FAKULTETAS
AUTOMATIKOS KATEDRA

SAVOSIOS KETURRAČIO ROBOTO LOKALIZACIJOS ALGORITMO KŪRIMAS IR ANALIZĖ

Baigiamasis magistro projektas
Valdymo technologijos (kodas 621H66001)

Vadovas
(parašas) Doc. dr. Renaldas Urniežius
(data)

Recenzentas
(parašas)
(data)

Projektą atliko
(parašas) Tomas Stirblys
(data)

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Elektros ir elektronikos
(Fakultetas)

Tomas Stirblys
(Studento vardas, pavardė)

Valdymo technologijos (621H66001)
(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „Savosios keturračio roboto lokalizacijos algoritmo kūrimas ir analizė“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ____ m. _____ _____ d.
Kaunas

Patvirtinu, kad mano, **Tomo Stirblio**, baigiamasis projektas tema „Savosios keturračio roboto lokalizacijos algoritmo kūrimas ir analizė“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____
(vardą ir pavardę įrašyti ranka) *(parašas)*

Autorius – Tomas Stirblys.

Darbo pavadinimas - savosios keturračio roboto lokalizacijos algoritmo kūrimas ir analizė. Magistro baigiamasis projektas / vadovas doc. dr. Renaldas Urniežius.
Kauno technologijos universitetas, elektros ir elektronikos fakultetas.

Kalba – Lietuvių.
Puslapių skaičius - 45.
Kaunas, 2015.

SANTRAUKA

Šio magistrinio darbo pagrindinis tikslas – sukurti ir eksperimentiškai išbandyti keturračio roboto savosios lokalizacijos metodą, naudojant enkoderių, akselerometro ir giroskopo matavimus. Apžvalginėje dalyje yra išanalizuoti alternatyvūs metodai, kurie plačiai naudojami šiandieniniuose lokalizacijos uždaviniuose. Apžvelgti tiek aparatūriniai sprendimai, tiek programiniai. Metodologinėje dalyje aprašytas keturračio roboto matematinis modelis bei jo Matlab – Simulink modelis. Toliau aprašomas algoritmas, naudojamas roboto lokalizacijai sekti. Eksperimentinėje dalyje tiriamas roboto matematinis modelis bei sukurtas santykinės entropijos filtras.

Reikšminiai žodžiai:
Jutiklių sintezė, savoji lokalizacija, santykinės entropijos filtras.

Author - Tomas Stirblys.

Title of project - Development and Analysis of Dead-Reckoning Localization Algorithm for Four-Wheeled Robot.

Final project of master degree / supervisor PhD. Renaldas Urniežius;

Kaunas University of Technology, Faculty of Electrical and Electronics Engineering.

Language – Lithuanian.

Pages – 45.

Kaunas, 2015.

SUMMARY

The main objective of this final project of master degree - to develop and experimentally test the four-wheeled robot's own localization method using the encoder, an accelerometer and gyroscope measurements. Review part is to analyze alternative methods which are widely used in modern localization tasks. Both hardware and software solutions have been reviewed. Four-Wheeled robot's mathematical model and its Matlab - Simulink model was described in the methodological part. Next, algorithm, used to track the localization of the robot, is described. Robot's mathematical model and created relative entropy filter are tested in experimental part of this project.

Keywords:

Sensor fusion, own localization, relative entropy filter.

TURINYS

SANTRUPOS IR ŽYMĖJIMAI

LED (angl. Light emitting diode)	Šviesos diodas
INS (angl. Inertial navigation System)	Inercinė lokalizacijos sistema
IMU (angl. Inertial measurement unit)	Inercinio matavimo prietaisas
GPS (angl. Global positioning system)	Globali navigacinė sistema
C#	Programavimo kalba

ĮVADAS

Savosios lokalizacijos uždavinys yra vienas svarbiausių uždavinių projektuojant robotą. Kadangi savosios lokalizacijos jutikliai perduoda informaciją apie nedidelį padėties pokytį, plačiai naudojamos išorinės lokalizacijos sistemos (GPS) gali to pokyčio neužfiksuoti. Arba esant nepasiekiamam GPS signalui tai - apskritai yra neįmanoma. Dėl šios priežasties yra naudojami inerciniai savosios lokalizacijos jutikliai ir naudojami metodai, leidžiantys tuos jutiklius apjungti ir dar tiksliau nustatyti objekto lokalizaciją.

Šiame baigiamajame projekte yra nagrinėjamos alternatyvios sistemos ir metodai, leidžiantys atlikti roboto lokalizaciją, kai išoriniai lokalizacijos sprendimai yra nepanaudotini arba norima didelio pozicionavimo tikslumo. Projekto tikslas yra sukurti savosios lokalizacijos metodą keturračiui robotui, kur apjungiami enkoderių, akselerometro ir giroskopo duomenys, kadangi enkoderių matavimai yra nebetikslūs, kai roboto ratas slysta. Tam ir kuriamas filtras, kuriame aprašomas visas roboto judėjimo matematinis modelis, kuris leidžia kompensuoti tą enkoderių matavimų netikslumą. Metodas yra kuriamas UAB „Rubedo sistemos“ įmonės prašymu, kur jis bus pritaikytas realiam keturračiui robotui. Metodas taip pat išbandomas su keturračio roboto matematiniu modeliu. Šiam tikslui įgyvendinti yra matematinėmis formulėmis aprašomas keturračio roboto judėjimas bei sukuriamas Matlab – Simulink modelis. Toliau yra sudaromas santykinės entropijos filtras, naudojamas įvertinti visus matavimus. Tada simuliuojant roboto judėjimą bei pridedant Gauso triukšmą prie matavimų, galima įvertinti sukurto filtro veikimą, palyginant signalus su triukšmu, nufiltruotus bei tikruosius. Šių eksperimentų rezultatai yra pateikiami eksperimentinėje dalyje.

Rinkoje yra didelis lokalizacijos sistemų pasirinkimas, bet kol kas sistemos, kurios naudoja tik inercinių jutiklių matavimus yra nepakankamo tikslumo, norint užtikrinti lokalizaciją, kai išoriniai navigaciniai signalai yra negalimi. Šiam tikslumui pagerinti yra naudojami skirtingi algoritmai, tų jutiklių matavimams apdoroti. Šiame darbe yra pateikiamas alternatyvus algoritmas šiems duomenims įvertinti.

1 APŽVALGINĖ DALIS

1.1 Neinercinės lokalizacijos sistemos

Neinercinė lokalizacijos sistema – tai sistema, kuri, remdamasi faktiniais duomenimis ir realiu laiku gali suskaičiuoti objekto padėtį. Priešingai negu inercinėse navigacijos sistemose, ši sistema gali rasti faktinę objekto padėtį, o ne skaičiuoti objekto judėjimo trajektorijos, remdamasi jo ankstesne buvimo vieta.

Vienas iš lokalizacijos sprendimo būdų yra palydoviniu ryšiu paremtos sistemos, kurios šiuo metu yra labiausiai paplitusios pasaulyje. Sutrumpintai tai vadinama GPS (*angl. Global Positioning System).* Ši sistema buvo sukurta ir išvystyta JAV, bet alternatyvos buvo pasiūlytos ne tik Amerikoje, bet ir Rusijoje, kur sistema žinoma kaip Glonass arba Europoje sukurta sistema, vadinama Galileo. Visos šios sistemos naudoja palydovo signalus. Pradžioje, kai 2002 m. JAV leido naudoti GPS duomenis civiliniams tikslams, tai ši technologija buvo taikoma tik automobilinėms lokalizacijos sistemoms. Vėliau, vystant GPS technologiją, tai buvo perkelta į mažesnius nešiojamus įrenginius. Dabartiniai šios technologijos privalumai yra tie, kad tai yra patikimas, lengvai panaudojamas bei pigus sprendimas. Teoriškai tikslumas gali būti pasiekiamas iki 7,8 metrų [1].

Kitas praktikoje naudojamas metodas yra lokalizacija, naudojant bevielio interneto tinklo signalus. Ši technologija yra taikoma pastatuose, kur yra įrengiami bevielio interneto prieigos taškai ir pagal įrenginyje priimamų signalų lygius galima suskaičiuoti kokios yra įrenginio koordinatės. Esminis dalykas yra tai, kad prie tinklo įrenginys neprivalo prisijungti, metodui užtenka matuoti signalo stiprumą. Pasitelkus metodu rezultatus, pozicionavimo tikslumas gali būti nuo 1 iki 10 metrų [2].

Vienas iš tiksliausių metodų, lokalizacijos skaičiavimui, yra garso bangomis paremta technologija. Ši technologija yra pagrįsta atstumo matavimu, naudojant garso bangas. Pagrindinis trūkumas yra tai, kad jos veikimui užtikrinti reikia be galo daug ultragarsinių siųstuvų – imtuvų, išdėstytų pastatuose, nes sistema veikia tik tiesioginio matomumo zonose. Sistemos didžiausias privalumas yra tai, kad yra pasiekiamas aukštas pozicionavimo tikslumas, kur paklaida neviršija 1 centimetro [3].

Vienas iš ne dažnai naudojamų metodų, apsprendžiančių įrenginio lokalizaciją, yra LED apšvietimo panaudojimas. Pagrindinis šio metodo trūkumas, kad LED šviestuvai turi būti aukštos kokybės, kurie galėtų komutuoti reikiama dažno impulsus, kuriuos galėtu priimti imtuvas įrenginyje,

kurio lokalizacija yra skaičiuojama. Impulsai žmogaus akiai – nepastebimi, bet šiuolaikiniai imtuvai yra įgalūs, šiais impulsais užkoduotą, informaciją priimti. Šiuo metodu veikiantys prietaisai gali pasiekti pozicionavimo tikslumą nuo 1 iki 8 metrų [4].

Kitas, įrenginio koordinatėms skaičiuoti, metodas yra paremtas mobiliojo ryšio signalų stiprumais. Vienas iš būdų yra panašus kaip ir anksčiau minėtas, bevielio interneto taškų panaudojimas, kai įrenginys tiesiogiai matuoja bokštų siunčiamus telefoninio ryšio signalų stiprius ir taip skaičiuoja savo padėtį. Metodas yra praktiškai labai retai naudojamas dėl prasto pozicionavimo tikslumo ir nepatikimumo. Galimas pozicionavimo tikslumas yra nuo 30 iki 200 metrų [5].

Lokalizacijos uždaviniui spręsti, kartais yra pasitelkiamas žemės magnetinis laukas, bet dėl savo didelių matavimo netikslumų jis yra naudojamas tik kaip pagalbinė priemonė lokalizacijai patikslinti. Kadangi yra matuojamas realus fizikinis dydis – magnetinio lauko stipris, tai taip pat sukelia problemas susijusias su šalutiniais magnetiniais laukais[6].

Visų neinerciniais davikliais paremtų lokalizacijos sprendimų didžiausias privalumas yra tai, kad šios sistemos turi nuo laiko nepriklausomą tikslumą. Pagrindiniai trūkumai yra tai, kad šioms sistemoms reikia kažkokio papildomo techninio sprendimo ar įrangos, kuri per brangi ar sunkiai įgyvendinama. Kiti sprendimai taipogi gali būti nepritaikyti pastato viduje, kas mūsų atveju yra labai svarbu.

1.2 Inercinės lokalizacijos sistemos

Inerciniais jutikliais paremti lokalizacijos sprendimo metodai pirmiausia buvo pradėti naudoti JAV, o pirmasis algoritmas veikė vieno giroskopo pagalba, kuris buvo taikomas nepilotuojamos raketos trajektorijai keisti. Vėliau sistema buvo tobulinama ir inercinių jutiklių duomenys buvo apjungiami, naudojant duomenų apjungimo algoritmus, dėl ko lokalizacijos nustatymas tapo vis tikslesnis ir labiau pritaikomas praktikoje [7].

Tipinės, inerciniais davikliais (akselerometras, giroskopas) paremtos, lokalizacijos sistemos remiasi tuo, kad pradžioje yra žinoma objekto padėtis, o vėliau jau yra skaičiuojama kaip judėjo objektas, žinant būsenos kintamuosius (pagreitis ir kampinis greitis). Šiuo atveju pagreitį integruojant laike du kartus gaunama objekto trajektorija, o giroskopo duomenis integruojant kartą, gaunama objekto orientacija. Dėl šios priežasties šitas sprendimas nėra geras, nes integruojant jutiklių matavimus, kartu yra integruojamos ir paklaidos, kas galutiniame rezultate įneša didžiulius netikslumus. Šitą problemą sprendžiant yra kuriami įvairūs aparatiniai ir programiniai sprendimai, kaip įvairių skaitmeninių filtrų taikymas [7].

Yra patentuoti įvairūs metodai, kuriuos naudojant yra pasiekiamas pakankamai aukštas pozicionavimo tikslumas. Viena iš patentuotų technologijų yra "NavShoe", kuri naudoja tiek inercinių jutiklių duomenys, tiek GPS duomenys. Tai technologija leidžianti sekti žmogaus lokalizaciją. Įrenginys yra įtaisytas bato pade ir žmogui einant yra skaičiuojamas jo nueitas kelias. Metodo išskirtinumas yra tas, kad akselerometro matavimų paklaidos yra smarkiai sumažinamos ir naudojant dvigubą integravimą nebedidėja kvadratine priklausomybe, o auga tik tiesine priklausomybe, kas gerokai pagerina lokalizacijos uždavinį. Metodo kūrėjai pasinaudojo prielaida ta, kad žingsniuojant, kai žmogaus koja paliečia paviršių, jos greitis yra lygus nuliui. Kitas metodo privalumas yra tai, kad kojos pagreitis yra kur kas didesnis, nei pačio kūno, tai yra daug lengviau fiksuojamas žingsnis, negu pačio kūno judesys. Tai šiuo atveju paklaida didėja nepriklausomai nuo laiko, o priklausomai nuo nueito kelio. Lauko sąlygomis, metodas taip pat naudoja GPS signalo duomenis, kurie dar labiau leidžia patikslinti lokalizaciją [8].

Kitas gana populiarus lokalizacijos metodas buvo pasiūlytas kompanijos – „u-blox“. Metodas yra plačiai taikomas automobilių navigacinėse sistemose, kai dingus ar esant per silpnam GPS signalui yra skaičiuojama automobilio judėjimo trajektorija, naudojant enkoderių, greičių dėžės, akselerometro ir giroskopo duomenis. Duomenų apjungimui yra naudojamas vienas populiariausių skaitmeninių filtrų analogiškoms sistemoms - Kalmano filtras. Gamintojai teigia, kad jų sukurtas prietaisas sugeneruoja tik 2 metrų paklaidą, važinėjant 10 minučių požeminėje aikštelėje, kur GPS signalas yra nepanaudojamas [9].

Vieną iš alternatyvų, išvardintiems metodams, pasiūlė Jeilio universiteto mokslininkai, kurie lokalizacijos metode naudoja iš akselerometro ir kameros gaunamą informaciją. Šiuo metodu yra apjungiami įprasti akselerometro matavimai, kurie yra lyginami su kameromis gautais rezultatais. Apdorojant kameros vaizdus yra apskaičiuojama kokia trajektorija juda objektas. Metodo autoriai teigia, kad sistema gali pasiekti 0.1 metro pozicionavimo tikslumą [10]. Alternatyvūs šaltiniai taip pat teigia, kad lokalizacijos uždavinius galima spręsti naudojant vien tik vaizdų apdorojimo algoritmus ar tai kombinuoti su kitų šaltinių matavimaisi, kaip infragarsiniai siųstuvai – imtuvai ar ultragarsiniai siųstuvai – imtuvai.

1.3 Savosios lokalizacijos metodai

Naudojant anksčiau minėtas sistemas ar joms analogiškas, visais atvejais yra susiduriama su matavimų triukšmingumu bei poreikiu tą triukšmą eliminuoti ar kaip galima labiau sumažinti. Šiam tikslui dažnai naudojami įvairūs filtrai. Lokalizacijos uždaviniuose, kai yra naudojami inerciniai jutikliai, kaip pavyzdžiui akselerometras ar giroskopas, plačiai ir dažnai yra naudojami Kalmano ar jo modifikacijomis paremti filtrai.

Kalmano filtras yra pasikartojantis skaičiavimo algoritmas sukurtas apskaičiuoti prognozes ir prognozių variacijas laike kintantiems modeliams. Jis gali būti panaudotas bet kuriame modelyje, kuris aprašomas būsenų erdvės forma. Beveik visi standartiniai laike kintantys modeliai gali būti aprašomi šia forma. Kalmano filtras yra taikomas sukonstruoti prognozes ir prognozių variacijas kintant laikui. Kiekvienas proceso žingsnis leidžia prognozuoti sekantį stebėjimą iš prieš tai buvusios prognozės stebėjimo. Tai yra, kiekviena einanti iš eilės prognozė atnaujinama žinant prieš tai buvusią. Atnaujinimo taisyklės kiekvienai prognozei yra prieš tai buvusio stebėjimo svorinis vidurkis ir prognozės paklaida. Intriguojanti Kalmano filtro savybė, kad atnaujinimo taisyklės svoriniai koeficientai parenkami užtikrinant minimalias prognozių variacijas. Svoriniai koeficientai, dar kitaip vadinami Kalmano stiprinimo koeficientai, turi tokia pačią įtaką kaip lygiosios konstantos eksponentiniame lyginime. Kalmano filtras gali būti naudojamas realaus laiko uždaviniams įgyvendinti. Kitaip tariant, kiekviena laiko reikšmė yra stebima ir sekanti prognozė gali būti apskaičiuota. Dėl to filtras plačiai naudojamas inžinerijoje, gamtos moksluose, ekonomikoje, finansuose ir žinoma lokalizacijos uždaviniuose [11].

Kalmano filtras buvo sukurtas 1960 m., kai mokslininkui Rudolf E. Kalman buvo pavesta užduotis sukurti metodą, leidžianti patikslinti Appolo erdvėlaivio navigacinės sistemos tikslumą. Filtro vienas iš privalumų yra tai, kad šis metodas nereikalauja didelių skaičiavimo resursų, o jam realizuoti pakanka palyginti paprastų skaičiavimų įtaisų. Kitas šio filtro privalumas yra tai, kad jo filtravimo savybės kinta laikui bėgant, priklausomai nuo pačio filtruojamo signalo [11].

Jeigu kalbėtume apie pirmuosius bandymus taikyti Kalmano filtrą, tai jau minėtame Appolo orlaivio navigacijos sistemoje, tai buvo sėkmingai įgyvendinta. Į Kalmano filtrą buvo paduodami signalai iš valdymo įtaisų ir būsenos kintamieji iš inercinių matavimo įtaisų, kurie matavo pagreitį ir orlaivio orientaciją. Turint fizikinius orlaivio parametrus buvo galima prognozuoti ir tiksliai aprašyti kaip orlaivis reaguos į valdymo signalus. Todėl turint duomenis iš valdymo įrenginių ir būsenos kintamuosius, jau galima tuos matavimus palyginti ir prognozuoti, kuriais matavimais pasikliauti labiau. Todėl pagrindinis Kalmano filtro privalumas tai ir yra, kad jis sugeba pats nuspręsti kuriais

matavimais pasikliauti: ar būsenos kintamaisiais, ar apskaičiuotais iš valdymo signalų, o gal naudoti tų matavimų apjungimą, kai yra įvertinama kiekvieno matavimo triukšmingumo dispersija [11].

Gautus valdymo signalus, iš kurių galima nuspėti sistemos būseną, galime aprašyti išraiška 1.1.

$$x_t = F_t x_{t-1} + B_t u_t + w_t \tag{1.1}$$

Čia:

x_t – numatoma būsena,

x_{t-1} – buvusi būsena,

F_t – sistemos būsenos kitimą aprašanti išraiška,

B_t – sistemos reakciją, į valdymo signalą, aprašanti išraiška,

u_t – valdymo signalas,

w_t – matavimo triukšmas.

Toliau yra aprašomi būsenos kintamieji, kurie pavyzdžiui realiai matuoją sistemos pagreitį.

$$z_t = H_t x_t + v_t \tag{1.2}$$

Čia:

z_t – būsenos matavimai,

H_t – transformacija, reikalinga būseną perskaičiuoti į matavimo rezultatą,

v_t – matavimo triukšmas.

Kalmano filtras yra sudarytas iš keleto žingsnių: prognozavimo ir matavimo atnaujinimo. Prognozavimui naudojama formulė yra pateikiama 1.3 formulėje.

$$P_{t|t-1} = F_t P_{t-1|t-1} F_t^T + Q_t \tag{1.3}$$

Čia:

$P_{t|t-1}$ – prognozuojamo dydžio dispersija,

F_t – išraiška, pagal kurią yra nusprendžiama kuo labiau pasitikėti: matavimais ar skaičiavimais, pagal matavimų dispersiją,

Q_t – triukšmo išraiška.

Reikšmės atnaujinimas:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t\left(z_t - H_t\hat{x}_{t|t-1}\right) \tag{1.4}$$

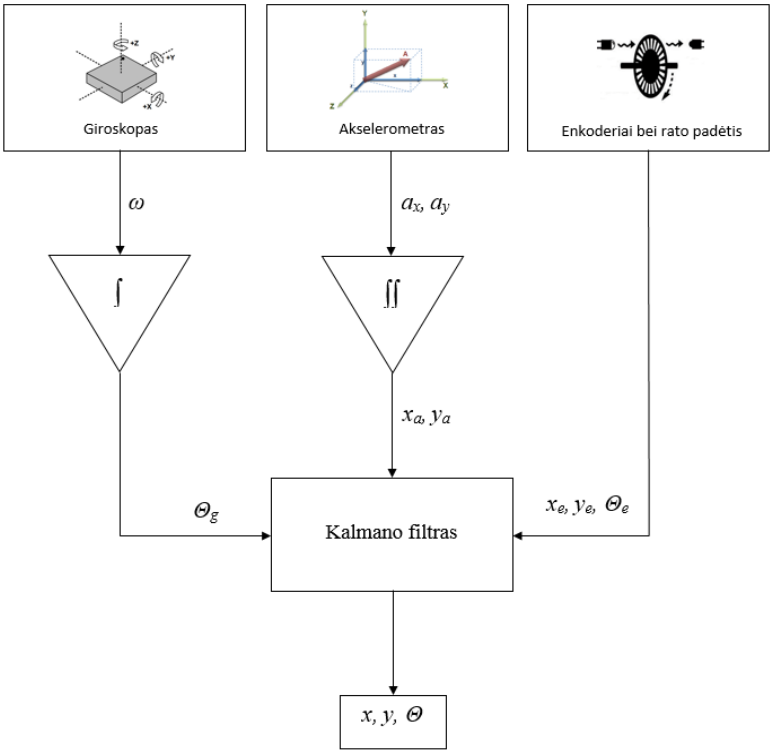
Dispersijos atnaujinimas:

$$P_{t|t} = P_{t|t-1} - K_t H_t P_{t|t-1} \tag{1.5}$$

Čia:

$$K_t = P_{t|t-1}H_t^T(H_t P_{t|t-1}H_t^T + R_t)^{-1} \tag{1.6}$$

Kalmano filtras puikiai tinka lokalizavimo uždaviniams, kai turime valdymo signalus ir tų valdymo signalų matavimo rezultatus. Pavyzdžiui turėdami keturratį robotą, jo lokalizacijai taip pat galima panaudoti Kalmano filtrą. Tada valdymo signalai būtų rato pasukimas bei enkoderių matavimai, o būsenos kintamieji – akselerometro ir giroskopo matavimai. Keturračio roboto lokalizacijai nustatyti yra jungiami duomenys iš skirtingų jutiklių. Tai vadinama jutiklių sinteze (angl. *Sensor Fusion*). Šiuo atveju yra apjungiami enkoderių, akselerometro ir giroskopo duomenys. Vaizdžiai šį duomenų apjungimą galima atvaizduoti žemiau pateikta funkcine schema taip, kaip parodyta 1.1 paveiksle.



1.1 pav. Kalmano filtro taikymo pavyzdys

Būtent šiuo filtrų besiremiant, Japonijos mokslininkai ištyrė jo panaudojimo galimybę, keturračiam robotui [12]. Metode yra kombinuojami enkoderių, akselerometro ir giroskopo matavimai. Signalai buvo naudojami pagerinti ir pataisyti keturračio roboto savosios trajektorijos matavimus. Čia pagrindinė metodo priemonė yra Kalmano filtras. Eksperimentų metu buvo susidurta su tokiomis problemomis, kaip inercinių daviklių paklaidų netiesiškumu, kai paklaidos atsiranda dėl šalutinių priežasčių, kaip temperatūros kitimas ar paviršiaus nelygumas. Kita problema buvo tai, kad realus robotas nėra idealiai pagamintas, o fizikinės lygtys, aprašančios roboto judėjimą negali tiksliai įvertinti tokių dalykų, kaip ratų netobulumas, kelio netobulumas ar netikslūs duomenys apie atstumą tarp rato ašių. Šios problemos įveda tam tikras problemas, dėl kurių lokalizacijos skaičiavimo tikslumas dar sumažėja. Buvo gauti rezultatai kai lokalizacijai yra naudojama tik enkoderių duomenys, enkoderių ir giroskopo bei visų matavimų duomenys. Nustatyta, kad prie 1,5 m/s linijinio greičio bei 10° posūkio kampo, paklaidos buvo 67,4 cm naudojant tik enkoderių duomenis, 49,4 cm naudojant enkoderių ir giroskopo matavimus bei 47,8 cm naudojant enkoderių, giroskopo ir akselerometro matavimus. Kai linijinis greitis buvo 1 m/s, o posūkio kampas 5°, paklaidos buvo 14,1 cm naudojant tik enkoderių duomenis, 8,2 cm naudojant enkoderių ir giroskopo matavimus bei 4,6 cm naudojant enkoderių, giroskopo ir akselerometro matavimus [12].

Šiame darbe nagrinėjamo savosios lokalizacijos metodo pagrindinė idėja yra santykinės entropijos filtras. Šis metodas leidžia vienu metu ne tik apdoroti inercinių jutiklių matavimus, bet ir užtikrinti, kad gauti įverčiai tenkintų iš anksto apibrėžtas ribojimo sąlygas. Filtro panaudojimas taip pat užtikrina priimtiną skaičiavimo greitaveiką ir duoda patikimus atsakymus, dinaminių trikdžių sąlygomis, parazitinių vibracijų arba smūgių aplinkoje. Svarbi išvada yra tai, kad santykinės entropijos metodas pateikia tuos pačius matematinius sprendimus kaip ir Kalmano filtras, todėl galima teigti, kad Kalmano filtro skaičiavimo metodika yra vienas iš santykinės entropijos atvejų [13].

2 METODINĖ DALIS

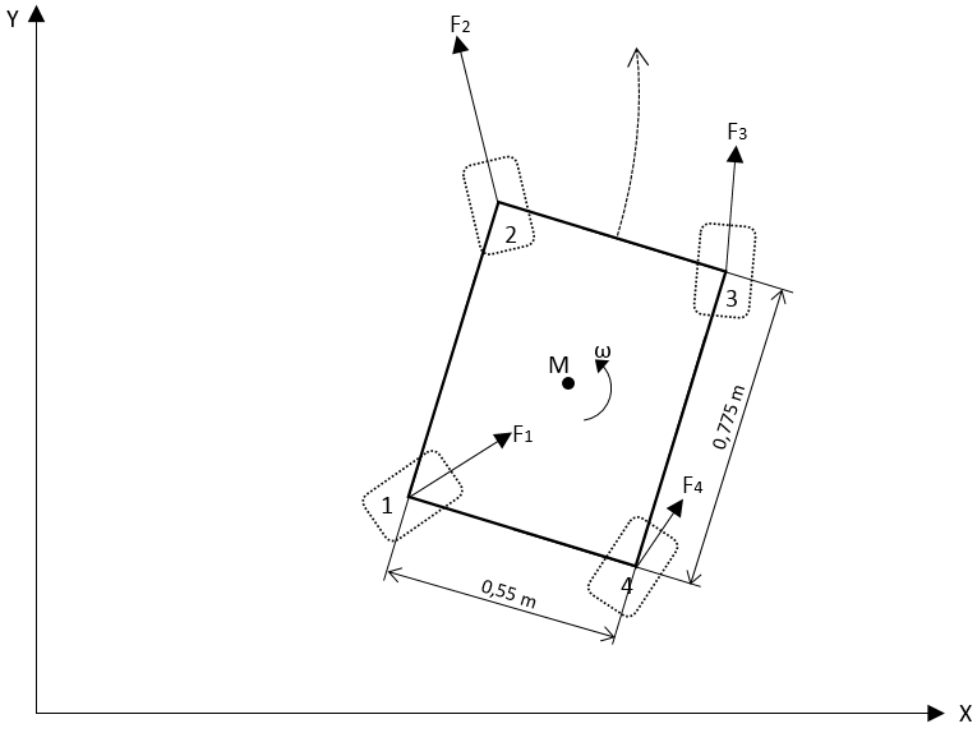
2.1 Keturračio roboto matematinis modelis

Aprašant roboto dinaminį judėjimą matematinėmis išraiškomis, pradžioje bus paaiškinama kaip tos išraiškos sudaromos ir kokius reikalavimus turi atitikti modelis. Pasinaudojus matematinėmis išraiškomis yra sudaromas šios sistemos modelis. Matematinis modelis yra įgyvendintas Matlab - Simulink programinio įrankio pagalba.

Šiame skyriuje taip pat yra kalbama atskirai apie ratų elektrinių variklių modelio kūrimą ir pačios roboto judėjimo trajektorijos matematinį modeliavimą.

2.1.1 Matematinio modelio aprašymas

Keturračio roboto ratai yra varomi keturių DC variklių, kurie tiesiogiai suka roboto ratus, kaip parodyta 2.1 paveiksle.



2.1 pav. Keturračio roboto modelis

Kiekvienas roboto ratas turi savo pasukimo kampą. Dėl rato pasukimo kampo, rato sukuriama varomoji jėga pirmiausia yra projektuojama į roboto koordinačių sistemą, o po to į stebėtojo koordinačių sistemą. Kadangi robotas yra laikomas kaip kietasis kūnas, galime taikytis superpozicijos principą ir visas, skirtingus taškus veikiančias, jėgas sudėti ir gauti bendrą kūno atstojamąją. Gavus bendrą atstojamąją ir žinant pasisukimo kampą, galima suskaičiuoti roboto centro judėjimą, stebėtojo koordinačių sistemos atžvilgiu. Roboto pasisukimo kampas yra randamas iš sukimo momentų apie roboto masės centrą.

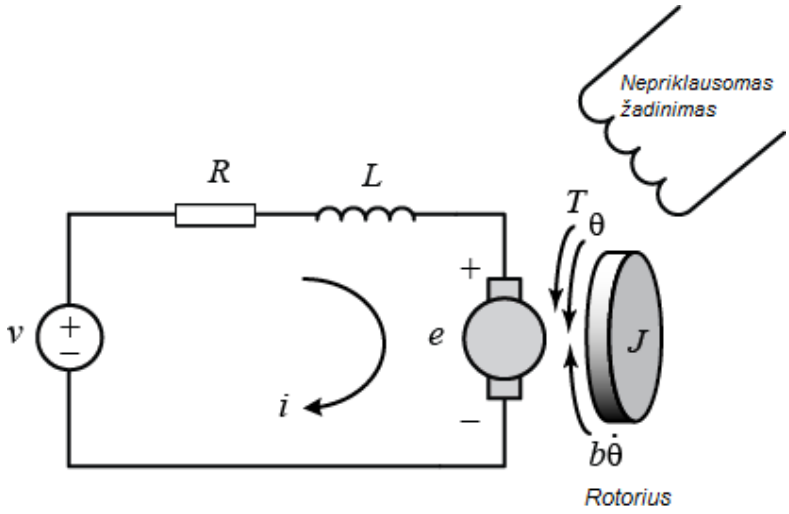
Pagrindiniai roboto parametrai pateikti 2.1 lentelėje.

2.1 lentelė. Pagrindiniai roboto parametrai

Eil. nr.	Pavadinimas	Reikšmė	Matavimo vienetai
1	Ilgis	0,775	m
2	Plotis	0,55	m
3	Masė	5	kg

2.1.2 Roboto rato DC variklio modeliavimas

Roboto rato variklio matematinis modelis yra kuriamas, naudojant Matlab (Simulink) programinę įrangą. DC variklio funkcinė schema parodyta 2.2 paveiksle. Variklio kampinis greitis – reguliuojamas įtampos šaltinio įtampa.



2.2 pav. DC variklio funkcinė schema

Priimame, kad rotorius ir ratas sukasi tuo pačiu kampiniu greičiu. DC variklio generuojamas sukimo momentas yra tiesiogiai proporcingas grandinės srovei ir magnetiniam laukui. Šiuo atveju priimame, kad magnetinis laukas yra fiksuotas ir prieiname išvados, kad variklio sukimo momentas priklauso nuo srovės, padaugintos iš variklio galios konstantos K_t. Taip gauname sukimo momento T priklausomybę – 2.1 formulėje.

$$T = K_t i \tag{2.1}$$

Atgalinė generuojama srovė (*angl. back EMF)* yra proporcinga kampiniui veleno sukimosi greičiui padaugintam iš elektrovaros konstantos K_e. Gauname išraišką – (2.2).

$$e = K_e \dot{\theta} \tag{2.2}$$

Toliau pagal Niutono ir Kirchhofo dėsnius gauname DC variklio dif. lygtis (2.3), (2.4), pagal kurias ir bus sudaromas matematinis modelis.

$$\frac{di}{dt} = \frac{(-Ri + u - c\varphi \cdot \omega)}{L} \tag{2.3}$$

$$\frac{d\omega}{dt} = \frac{(c\varphi \cdot i - k_{tr,DC} \cdot \omega)}{I} \tag{2.4}$$

Kur:

$i, [A]$ – Roboto rato variklio srovė;

$R, [\Omega]$ – Roboto rato variklio apvijų varža;

$u, [V]$ – Roboto rato variklio įtampa;

$I, [kg \cdot m^2]$ – Roboto rato variklio inercijos momentas;

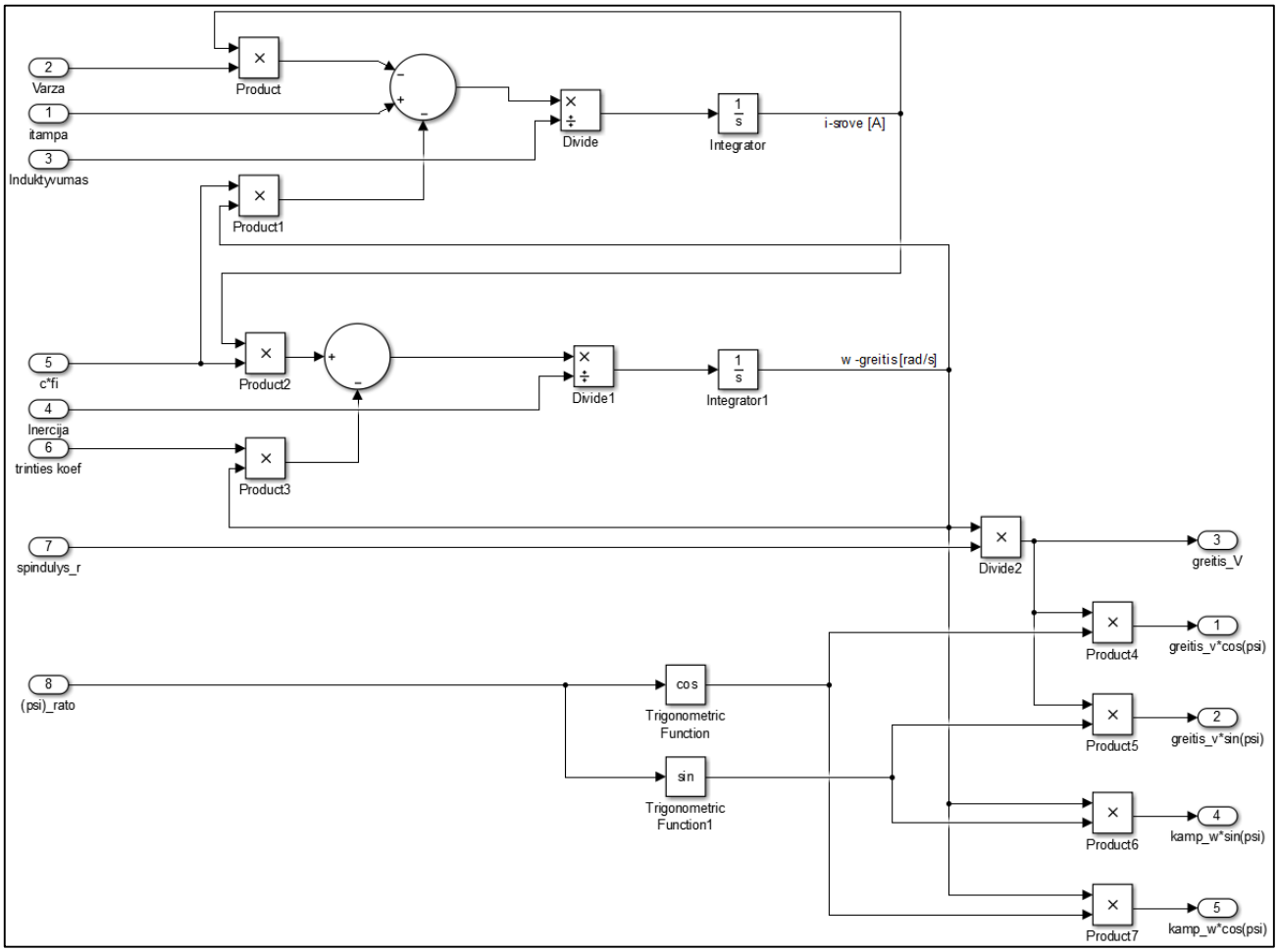
$L, [H]$ – Roboto rato variklio induktyvumas;

$k_{tr,DC}$ – Roboto rato variklio trinties koeficientas;

$c\varphi, [{}^{Nm}/_{A}]$ – Roboto rato variklio galios konstanta, kur $K_t = K_e = c\varphi$;

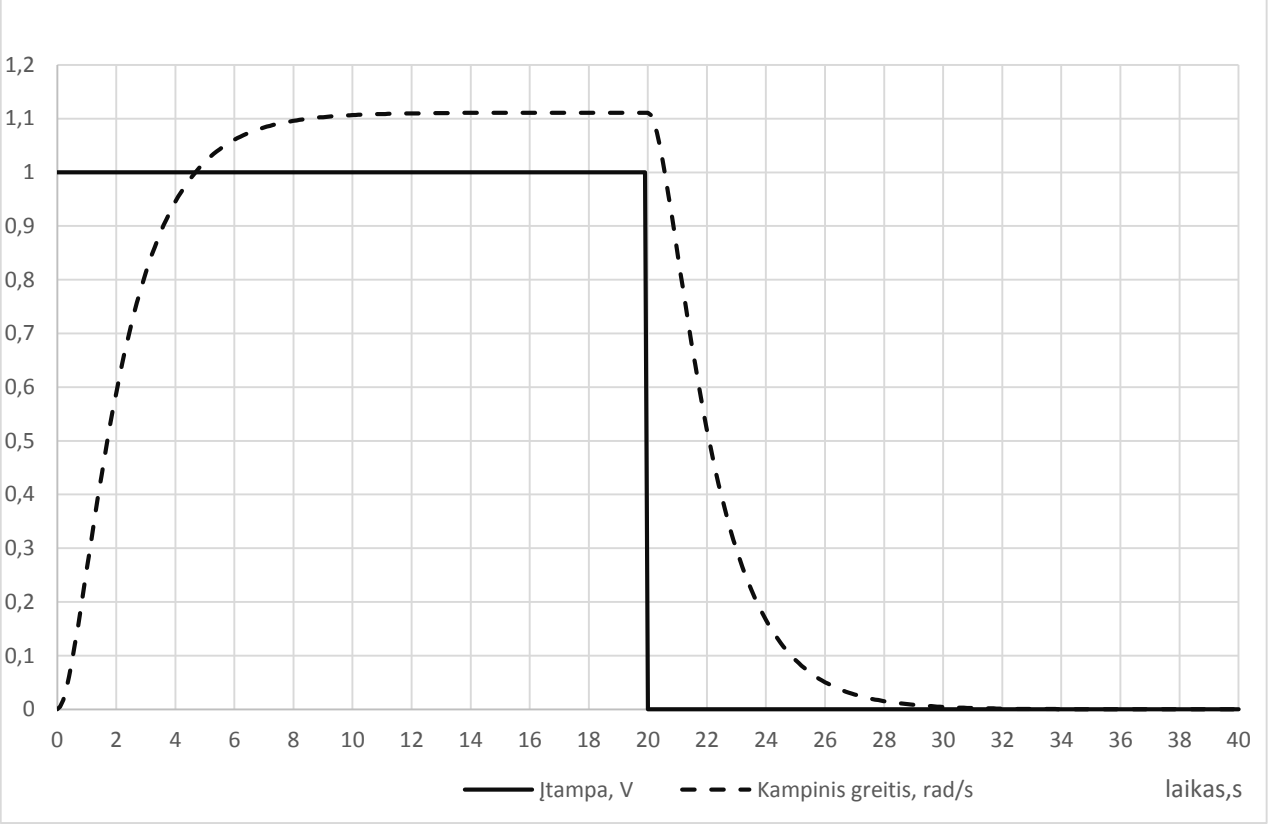
$\omega, \left[{}^{rad}/_{s}\right]$ – Kampinis rato greitis;

Turint dif. lygtis (2.3, 2.4), galima sudaryti Matlab-Simulink modelį. Šį matematinį modelį patalpinsime į Simulink (subsystem) bloką, kurį po to bus patogu naudoti kaip atskirus roboto ratus. Įgyvendintas matematinis modelis parodytas 2.3 paveiksle.



2.3 pav. Rato variklio matematinis modelis

Sudarius roboto rato matematinį modelį yra atliekamas rato sukimosi greičio eksperimentas, priklausomai nuo įėjimo įtampos. Eksperimento rezultatai pateikti 2.4 paveiksle.

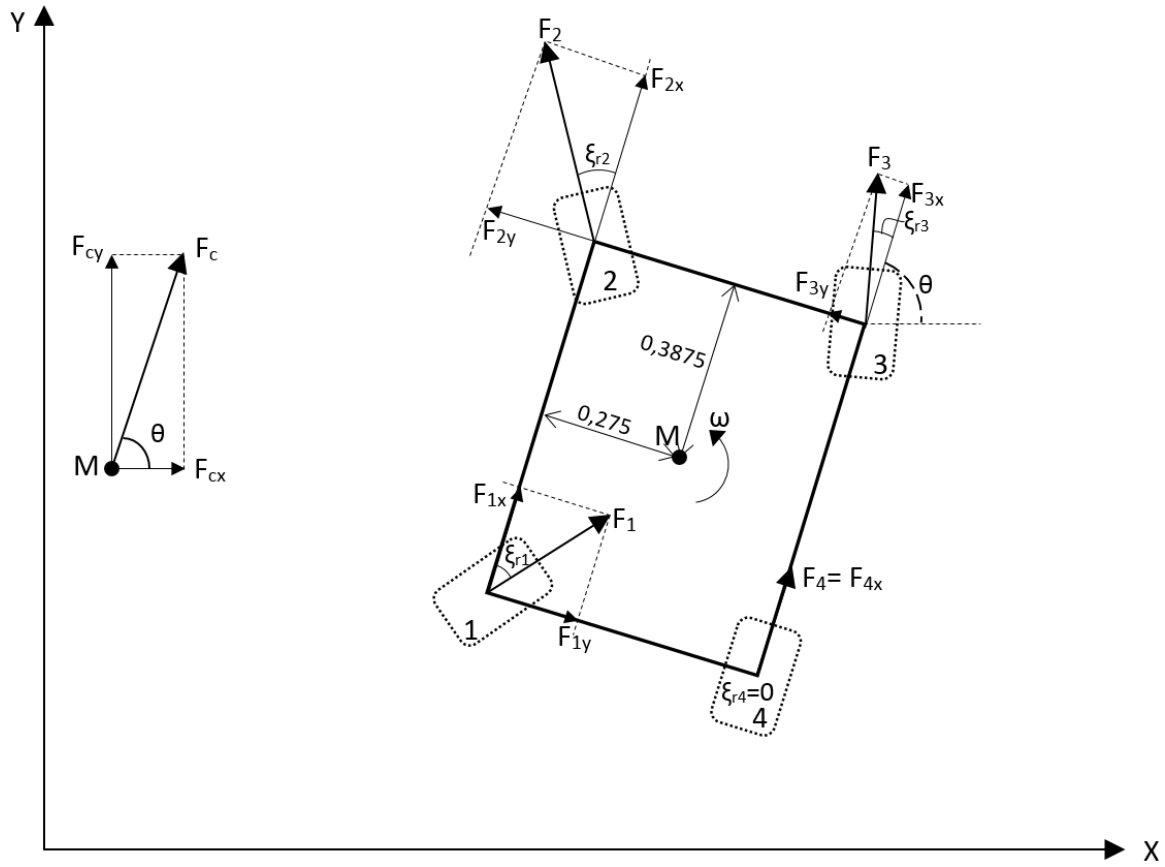


2.4 pav. Rato kampinio greičio priklausomybė nuo įėjimo įtampos

2.1.3 Roboto judėjimo modeliavimas

Prieš sudarant matematines lygtis, aprašančias roboto judėjimo trajektoriją, paaiškinama kaip projektuojamos jėgos į stebėtojo koordinačių sistemą. Turint roboto rato kuriamą varomąją jėgą ir rato pasukimo kampą, galime tą jėgos vektorių projektuoti į roboto koordinačių sistemą, kur ilgoji kraštinė atitinka x ašį, o trumpoji kraštinė y ašį. Turint šitas projekcijas ir roboto orientaciją erdvėje, galima šitas jėgas suprojektuoti į stebėtojo koordinačių sistemą. Visi šitie veiksmai toliau ir paaiškinami išsamiau.

Pirmiausia pažymimos robotą varančios jėgos (2.5 pav.).



2.5 pav. Robotą varančios jėgos ir jų projekcijos į roboto koordinačių sistemą

Čia:

M – roboto masės ir geometrinis centras;

F_1, F_2, F_3, F_4 – kiekvieno rato varančioji jėga;

F_{1x}, F_{1y}, F_{2x}, F_{2y}, F_{3x}, F_{3y}, F_{4x}, F_{4y} – kiekvieno rato varančiosios jėgos projekcijos į roboto koordinačių sistemą;

ξ_{r1}, ξ_{r2}, ξ_{r3}, ξ_{r4} – kiekvieno rato pasukimo kampas;

θ – roboto pasisukimo kampas, stebėtojo atžvilgiu.

Jėgų projekcijos į roboto koordinačių sistemą:

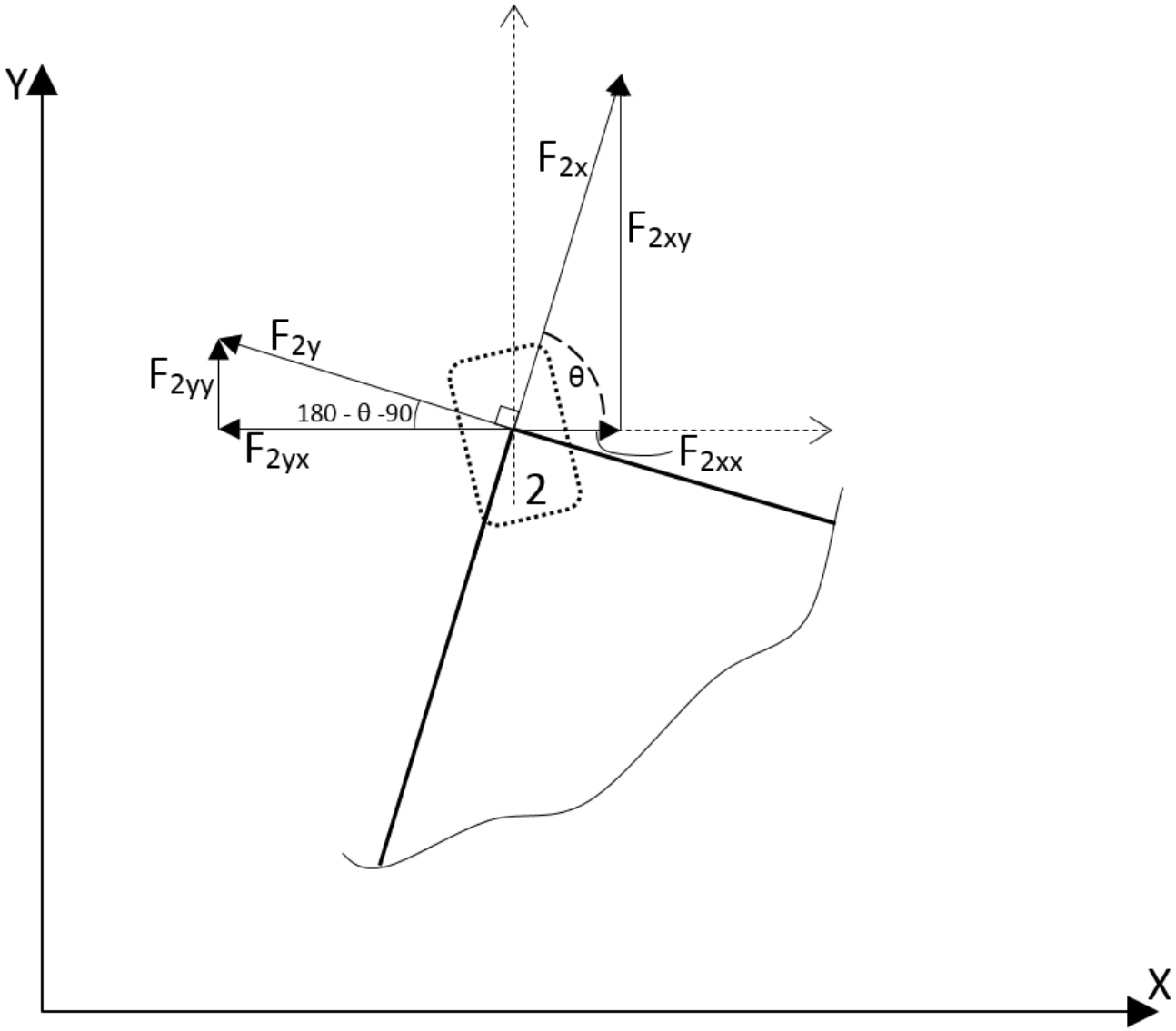
$$F_{1x} = F_1 \cos \xi_{r1} \ , \quad F_{1y} = F_1 \sin \xi_{r1} \tag{2.5}$$

$$F_{2x} = F_2 \cos \xi_{r2} \ , \quad F_{2y} = F_2 \sin \xi_{r2} \tag{2.6}$$

$$F_{3x} = F_3 \cos \xi_{r3} \ , \quad F_{3y} = F_3 \sin \xi_{r3} \tag{2.7}$$

$$F_{4x} = F_4 \cos \xi_{r4} \ , \quad F_{4y} = F_4 \sin \xi_{r4} \tag{2.8}$$

Turint projekcijas į roboto koordinačių sistemą, tas jėgas projektuojame į stebėtojo koordinačių sistemą. Projekcijos parodytos 2.6 paveiksle.



2.6 pav. Jėgų projekcijos į stebėtojo koordinačių sistemą

Čia:

F_{2x}, F_{2y} – kiekvieno rato varančiosios jėgos projekcijos į roboto koordinačių sistemą;

F_{2xx}, F_{2yx}, F_{2xy}, F_{2yy} – kiekvieno rato varančiosios jėgos projekcijos į stebėtojo koordinačių sistemą;

θ – roboto pasisukimo kampas, stebėtojo atžvilgiu.

Jėgų projekcijos į stebėtojo koordinačių sistemą:

$$F_{1xx} = F_{1x}\cos\theta\ , \quad F_{1xy} = F_{1x}\sin\theta \tag{2.9}$$

$$F_{1yx} = -F_{1y}\cos(90-\theta)\ , \quad F_{1yy} = F_{1y}\sin(90-\theta) \tag{2.10}$$

$$F_{2xx} = F_{2x}\cos\theta\ , \quad F_{2xy} = F_{2x}\sin\theta \tag{2.11}$$

$$F_{2yx} = -F_{2y}\cos(90-\theta)\ , \quad F_{2yy} = F_{2y}\sin(90-\theta) \tag{2.12}$$

$$F_{3xx} = F_{3x}\cos\theta\ , \quad F_{3xy} = F_{3x}\sin\theta \tag{2.13}$$

$$F_{3yx} = -F_{3y}\cos(90-\theta)\ , \quad F_{3yy} = F_{3y}\sin(90-\theta) \tag{2.14}$$

$$F_{4xx} = F_{4x}\cos\theta\ , \quad F_{4xy} = F_{4x}\sin\theta \tag{2.15}$$

$$F_{4yx} = -F_{4y}\cos(90-\theta)\ , \quad F_{4yy} = F_{4y}\sin(90-\theta) \tag{2.16}$$

Turint jėgų projekcijas galima sudaryti dinamines lygtis, kurias naudojant bus sudarytas matematinis modelis. Kadangi robotas yra laikomas kaip kietasis kūnas, galime taikytis superpozicijos principą ir visas, skirtingus taškus veikiančias, jėgas sudėti ir gauti bendrą kūno atstojamąją. Žemiau yra pateikiamos išraiškos, aprašančios jėgų projekcijas į stebėtojo koordinačių sistemą.

X ašis:

$$k_1\omega_1(t)rcos\xi_{r1}\cos\theta(t) + k_2\omega_2(t)rcos\xi_{r2}\cos\theta(t) + k_3\omega_3(t)rcos\xi_{r3}\cos\theta(t) + k_4\omega_4(t)rcos\xi_{r4}\cos\theta(t) - k_1\omega_1(t)rsin\xi_{r1}\cos(90^\circ-\theta(t)) - k_2\omega_2(t)rsin\xi_{r2}\cos(90^\circ-\theta(t)) - k_3\omega_3(t)rsin\xi_{r3}\cos(90^\circ-\theta(t)) - k_4\omega_4(t)rsin\xi_{r4}\cos(90^\circ-\theta(t)) = v_{cx}(t)k_{tr} + m\frac{dv_{cx}(t)}{dt}; \tag{2.17}$$

Y ašis:

$$k_1\omega_1(t)rsin\xi_{r1}\sin(90°-\theta(t))+k_2\omega_2(t)rsin\xi_{r2}\sin(90°-\theta(t))+$$
$$k_3\omega_3(t)rsin\xi_{r3}\sin(90°-\theta(t))+k_4\omega_4(t)rsin\xi_{r4}\sin(90-\theta(t))+$$
$$k_1\omega_1(t)rcos\xi_{r1}\sin\theta(t)+k_2\omega_2(t)rcos\xi_{r2}\sin\theta(t)+k_3\omega_3(t)rcos\xi_{r3}\sin\theta(t)+$$
$$k_4\omega_4(t)rcos\xi_{r4}\sin\theta(t)=v_{cy}(t)k_{tr}+m\frac{dv_{cy}(t)}{dt}; \quad (2.18)$$

Sukimo momentas apie M tašką:

$$I_{zz,roboto}\frac{d^2\theta(t)}{dt^2}=-k_{tr,sukimosi}\frac{d\theta(t)}{dt}-k_1\omega_1(t)sin\xi_{r1}\cdot a+k_2\omega_2(t)sin\xi_{r2}\cdot a+$$
$$k_3\omega_3(t)sin\xi_{r3}\cdot a-k_4\omega_4(t)sin\xi_{r4}\cdot a-k_1\omega_1(t)cos\xi_{r1}\cdot b-k_2\omega_2(t)cos\xi_{r2}\cdot b+$$
$$k_3\omega_3(t)cos\xi_{r3}\cdot b+k_4\omega_4(t)cos\xi_{r4}\cdot b; \quad (2.19)$$

Čia:

$k_1=k_2=k_3=k_4$ – Kiekvieno rato trinties koeficientai,

v_{cx}, $[^m/_s]$ – Roboto centro greičio projekcija į X ašį,

v_{cy}, $[^m/_s]$ – Roboto centro greičio projekcija į Y ašį,

m, $[kg]$ – Roboto masė,

$I_{zz,roboto}$, $[kg\cdot m^2]$ – Roboto inercija apie Z ašį,

$k_{tr,sukimosi}$ – Roboto sukimosi trinties koeficientas,

a, $[m]$ – Atstumas tarp masės centro ir rato sinuso projekcijos,

b, $[m]$ – Atstumas tarp masės centro ir rato kosinuso projekcijos,

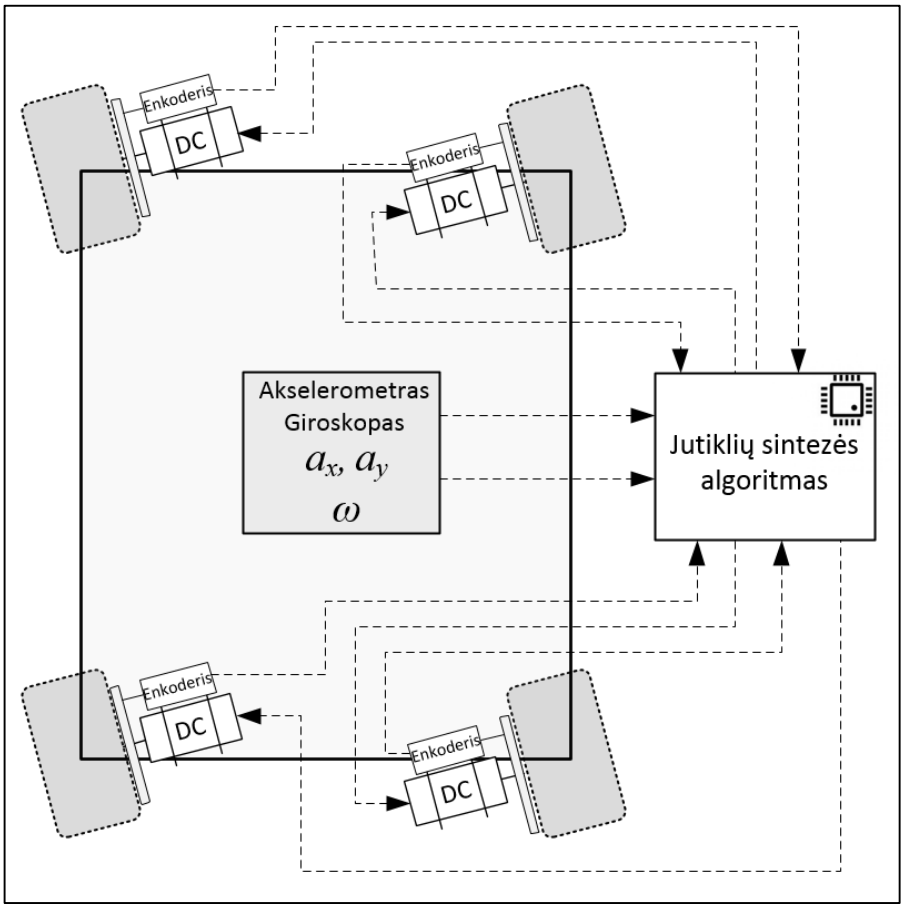
k_{tr} – Bendras roboto trinties koeficientas,

r, $[m]$ – rato spindulys;

Iš (2.17), (2.18), (2.19) lygčių galima rasti tris nežinomuosius, tai yra centro greičio projekcijas ir pasisukimo kampą, stebėtojo koordinačių sistemoje. Greičio projekcijas integruojant laike yra gaunama nueito kelio trajektorija. Remiantis (2.17), (2.18), (2.19) lygtimis yra sudaromas matematinis modelis, aprašantis roboto judėjimą. Pilnas roboto matematinis modelis yra pateiktas pirmame priede (Priedas 1. Matlab – Simulink keturračio roboto matematinis modelis).

2.2 Savosios lokalizacijos metodo kūrimas

Keturračio roboto savosios lokalizacijos metodas yra kuriamas realiai sistemai, kurioje yra apjungiami enkoderių, akselerometro ir giroskopo matavimai. Sistema atvaizduojama 2.7 paveiksle. Inerciniais matavimo įtaisais paremtas judėjimo trajektorijos tikslinimas yra naudingas tuo, kad jeigu ratas slysta enkoderių parodymai nėra teisingi ir trajektoriją reikia patikslinti, naudojant akselerometro ir giroskopo duomenis.



2.7 pav. Realaus modelio struktūra

Inercinių jutiklių navigacinės sistemos remiasi tuo, kad pradžioje yra žinoma objekto koordinatės, o vėliau yra skaičiuojama kaip judėjo objektas. Akselerometro matavimą (pagreitį) integruojant laike du kartus, gaunamas kūno poslinkis. Giroskopo pagalba yra matuojamas roboto kampinis greitis, kurį taip pat integruojant laike, gaunamas roboto pasisukimo apie z ašį matavimas. Metodo pagrindinė problema yra tai, kad yra matuojami realūs dydžiai (pagreitis, kampinis greitis), kurie yra integruojami, norint gauti padėtį ir pasisukimo apie z ašį kampą. Matavime atsiradusios

paklaidos, kurios integruojant taip pat sumuojasi ir laikui bėgant išauga iki netoleruojamų verčių. Šitai problemai išspęsti yra naudojami įvairūs metodai ir filtrai.

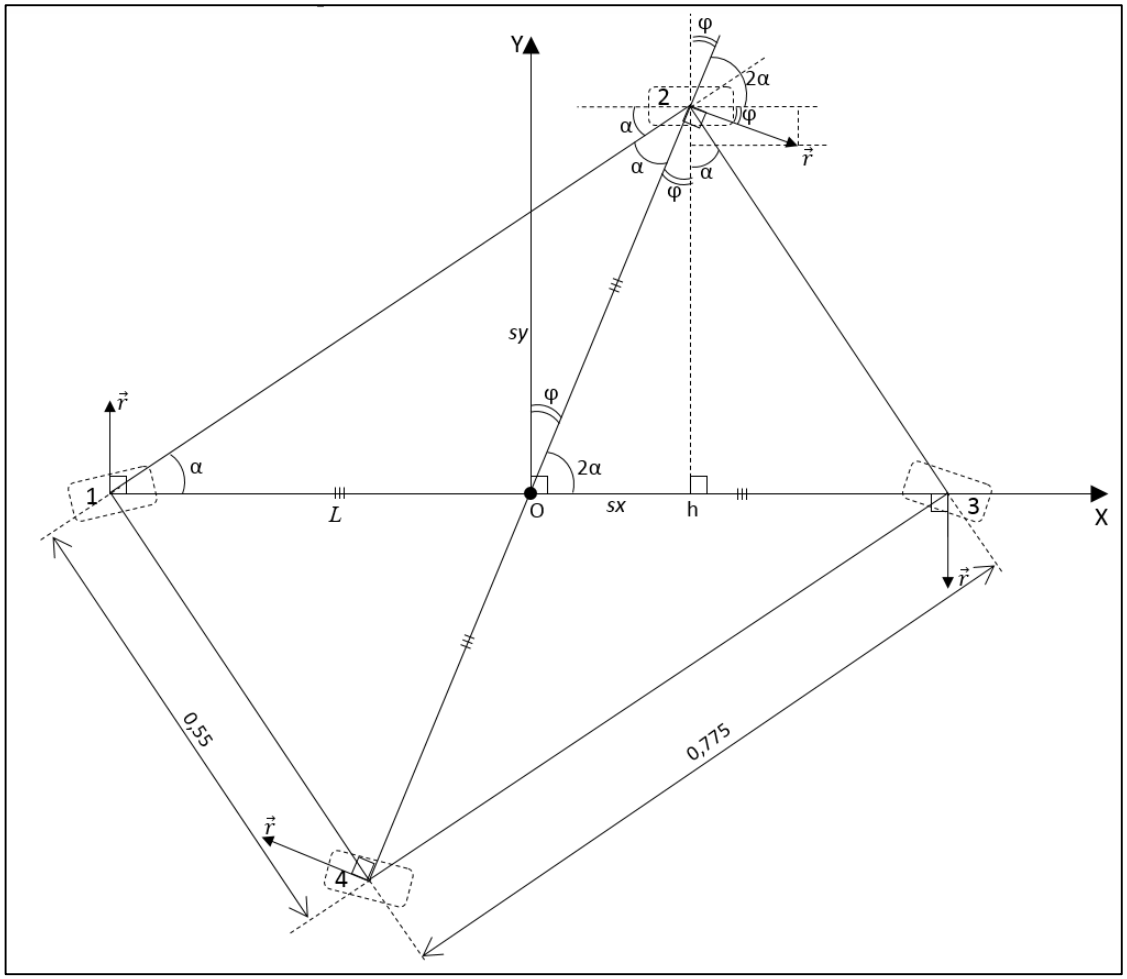
Šitame darbe kuriamas savosios lokalizacijos metodas naudoja santykinės entropijos [14] teoriją, kur yra apjungiami enkoderių, akselerometro ir giroskopo matavimai. Sekančiuose poskyriuose bus kalbama kaip kuriamas filtras ir bus pateikiamas jo veikimo algoritmas.

2.2.1 Skaitmeninis filtras

Kuriant metodą roboto savajai lokalizacijai yra naudojama standaus kūno mechanikos teorija [15], kur standaus kūno judėjimas gali būti atskiriamas į poslinkio ir posūkio dedamąsias. Tai reiškia, kad kiekvienu diskretizavimo intervalu kiekvienas roboto ratas pasislenka tokiu pat atstumu kaip roboto centras ir nueina kelią puslankiu, kurio centras yra roboto centras. Kadangi robotas yra laikomas kietuoju kūnu, o puslankio centras atitinka roboto centrą, tai visi ratai nueis tą patį puslankio absoliutinį atstumą.

Todėl galima daryti išvadą, kad tikrasis roboto rato nueitas kelias bus lygus poslinkiui lanku apie roboto centrą plius poslinkiui x ir y ašimi. Šita prielaida, kad roboto judėjimas yra suma poslinkių ir puslankių, leidžia naudoti superpozicijos principą.

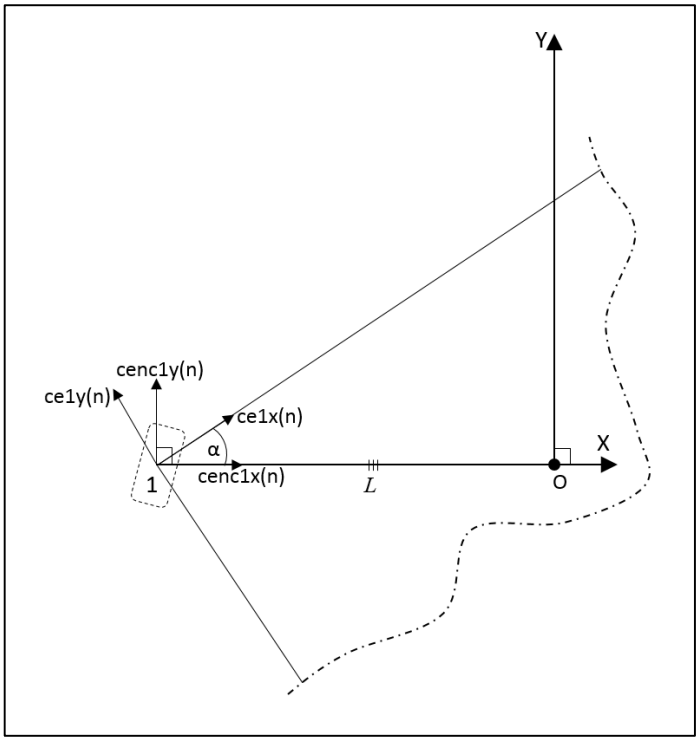
Pasinaudojus šitomis prielaidomis sudaromas modelis, kuris bus naudojamas filtro sintezei. Modelis pateikiamas 2.8 paveiksle.



2.8 pav. Modelis, naudojamas filtro sintezei

Pažymime ∠O12 – α. Trikampis ΔO12 yra lygiašonis, dėl to ∠O12 = ∠O21. Iš trikampio savybės, kad kampų suma – 180°, gauname, kad ∠1O2 = 180° – α – α. Tada kampas ∠3O2 = 2α. Pagal trikampio panašumo savybes matome, kad Δ2h3 ir Δ123 yra panašūs. Toliau pagal tas pačias geometrines taisykles randami kiti kampai. Sukimo vektoriaus $\vec{r}$ pasisukimo kampą, pažymime – φ. Iš stataus trikampio Δ123, gauname $\alpha = \arctan\frac{55}{77,5} = 0,617191\ rad = 35,36°$, tada kampas $\varphi = 90° - 2\alpha$. Iš stačiakampio lygčių gauname, kad $L = \frac{77,5}{2\cos\alpha} = 47,5164\ cm$.

Iš Simulink modelio gautas ratų jėgų projekcijas į roboto koordinačių sistemą pirmiausia reikia perskaičiuoti į kuriamo filtro roboto koordinačių sistemą, kur x ašis eina per pirmąjį ir trečiąjį ratus. Kaip tai atliekama parodyta 2.9 paveiksle.



2.9 pav. Simulink modelio projekcijos į filtro modelio koordinačių sistemą

Čia turėdami projekcijas iš Simulink modelio (ce1x(n), ce1y(n)) jas perskaičiuojame į filtro modelio projekcijas (cenc1x(n), cenc1y(n)), kur n – diskretizavimo intervalas. Perskaičiavimas vykdomas pagal sekančias formules (2.20, 2.21, 2.22, 2.23).

$$cenc1x(n) = ce1x(n)Cos[\alpha] - ce1y(n)Sin[\alpha];$$
$$cenc1y(n) = ce1y(n)Cos[\alpha] + ce1x(n)Sin[\alpha]; \quad (2.20)$$

$$cenc2x(n) = ce2x(n)Cos[\alpha] - ce2y(n)Sin[\alpha];$$
$$cenc2y(n) = ce2y(n)Cos[\alpha] + ce2x(n)Sin[\alpha]; \quad (2.21)$$

$$cenc3x(n) = ce3x(n)Cos[\alpha] - ce3y(n)Sin[\alpha];$$
$$cenc3y(n) = ce3y(n)Cos[\alpha] + ce3x(n)Sin[\alpha]; \quad (2.22)$$

$$cenc4x(n) = ce4x(n)Cos[\alpha] - ce4y(n)Sin[\alpha];$$
$$cenc4y(n) = ce4y(n)Cos[\alpha] + ce4x(n)Sin[\alpha]; \quad (2.23)$$

Gautos vertės yra realūs matavimai, kurie gaunami iš enkoderių. Šitas realias vertes ir naudosime sukurtame filtre kur kartu vertinsime ir akselerometro ir giroskopo duomenis. Filtre rato enkoderio matavimas yra lyginamas su realiu keliu, kurį nueina ratas, naudojant poslinkio ir posūkio dedamųjų atskyrimo teoriją kietajam kūnui.

Norint rasti tikrąsias kiekvieno rato poslinkio vertes, pasižymime puslankiu nueito kelio atstumą per diskretizavimo intervalą – *r(n)* ir poslinkio x ir y ašimis – *sx(n)* ir *sy(n)*. Žymėjimas parodytas 2.9 paveiksle. Dabar tikrosios ratų poslinkių išraiškos pateikiamos (2.24, 2.25, 2.26, 2.27) formulėse.

$$\begin{aligned} &Pirmam\ ratui: \\ &enc1x(n) = sx(n); \\ &enc1y(n) = sy(n) + r(n); \end{aligned} \tag{2.24}$$

$$\begin{aligned} &Antram\ ratui: \\ &enc2x(n) = sx(n) + r(n)cos\varphi; \\ &enc2y(n) = sy(n) - r(n)sin\varphi; \end{aligned} \tag{2.25}$$

$$\begin{aligned} &Trečiam\ ratui: \\ &enc3x(n) = sx(n); \\ &enc3y(n) = sy(n) - r(n); \end{aligned} \tag{2.26}$$

$$\begin{aligned} &Ketvirtam\ ratui: \\ &enc4x(n) = sx(n) - r(n)cos\varphi; \\ &enc4y(n) = sy(n) + r(n)sin\varphi; \end{aligned} \tag{2.27}$$

Toliau seka įvertinti akselerometro ir giroskopo išmatuotas ir formulėmis aprašytas išraiškas. Kampą θ galima aprašyti iš posūkio lanko išraiškos *r(n)*, kur *n* – diskretizavimo intervalas. Išraiška pateikta (2.28) formulėje.

$$\theta(n) = \frac{r(n) \cdot 180}{L\pi}; \tag{2.28}$$

Tada iš formulių (2.24 – 2.27) galima pasakyti roboto trajektoriją, prieš tai atlikus *sx(n), sy(n), r(n)* išraiškų skaičiavimus. Šios išraiškos skaičiuojamos aproksimuojant akselerometro ir giroskopo matavimus, tam tikrame diskretizavimo intervale. Čia yra naudojamas santykinės entropijos filtras [16]. Filtras yra taikomas sukonstruoti prognozes ir prognozių variacijas kintant laikui. Kiekvienas proceso žingsnis leidžia prognozuoti sekantį stebėjimą iš prieš tai buvusios prognozės stebėjimo. Tai yra, kiekviena einanti iš eilės prognozė atnaujinama žinant prieš tai buvusią. Atnaujinimo taisyklės kiekvienai prognozei yra prieš tai buvusio stebėjimo svorinis vidurkis ir prognozės paklaida. Intriguojanti filtro savybė, kad atnaujinimo taisyklės svoriniai koeficientai parenkami užtikrinant minimalias prognozių variacijas. Svoriniai koeficientai, dar kitaip vadinami stiprinimo koeficientai, randami skaičiuojant Hamiltonianą.

Filtras yra svarbus, nes gali būti panaudotas realiame laike. Kitaip tariant, kiekviena laiko reikšmė yra stebima ir sekanti prognozė gali būti apskaičiuota. Todėl skaičiuojant Hamiltoniano vertę, mes prognozuojame akselerometro ir giroskopo matavimą, įvertindami maksimalų pagreičio pokytį per diskretizavimo intervalą ir maksimalią kampinio pagreičio vertę. Šiems skaičiavimams reikalingos išraiškos yra pateiktos prieduose (priedas 2. Filtro matematinės išraiškos).

Norint rasti prognozuojamų kintamųjų įverčius yra skaičiuojama Hamiltoniano išraiška, kuri pateikta (2.29) formulėje. Žemiau pateikta išraiška aprašyta tik trims diskretizavimo intervalams, prieduose (priedas 2. Filtro matematinės išraiškos) yra pateikiama pilna skaičiavimo išraiška.

$$H = -\frac{(\mathrm{ax}_{12}-\mathrm{cax}_{12})^2+(\mathrm{ay}_{12}-\mathrm{cay}_{12})^2}{\sigma \mathrm{a}^2} - \frac{(\mathrm{ax}_{12}+\Delta\mathrm{ax}_{23}-\mathrm{cax}_{23})^2+(\mathrm{ay}_{12}+\Delta\mathrm{ay}_{23}-\mathrm{cay}_{23})^2}{\sigma \mathrm{a}^2} -$$
$$\frac{(\mathrm{ax}_{12}+\Delta\mathrm{ax}_{23}+\Delta\mathrm{ax}_{34}-\mathrm{cax}_{34})^2+(\mathrm{ay}_{12}+\Delta\mathrm{ay}_{23}+\Delta\mathrm{ay}_{34}-\mathrm{cay}_{34})^2}{\sigma \mathrm{a}^2} -$$
$$\frac{(-\mathrm{cenc1x1}+\mathrm{enc1x1})^2+(-\mathrm{cenc1y1}+\mathrm{enc1y1})^2}{\sigma \mathrm{e}^2} - \frac{(-\mathrm{cenc1x2}+\mathrm{enc1x2})^2+(-\mathrm{cenc1y2}+\mathrm{enc1y2})^2}{\sigma \mathrm{e}^2} -$$
$$\frac{(-\mathrm{cenc1x3}+\mathrm{enc1x3})^2+(-\mathrm{cenc1y3}+\mathrm{enc1y3})^2}{\sigma \mathrm{e}^2} - \frac{(-\mathrm{cenc1x4}+\mathrm{enc1x4})^2+(-\mathrm{cenc1y4}+\mathrm{enc1y4})^2}{\sigma \mathrm{e}^2} -$$
$$\frac{(-\mathrm{cenc2x1}+\mathrm{enc2x1})^2+(-\mathrm{cenc2y1}+\mathrm{enc2y1})^2}{\sigma \mathrm{e}^2} - \frac{(-\mathrm{cenc2x2}+\mathrm{enc2x2})^2+(-\mathrm{cenc2y2}+\mathrm{enc2y2})^2}{\sigma \mathrm{e}^2} -$$
$$\frac{(-\mathrm{cenc2x3}+\mathrm{enc2x3})^2+(-\mathrm{cenc2y3}+\mathrm{enc2y3})^2}{\sigma \mathrm{e}^2} - \frac{(-\mathrm{cenc2x4}+\mathrm{enc2x4})^2+(-\mathrm{cenc2y4}+\mathrm{enc2y4})^2}{\sigma \mathrm{e}^2} -$$
$$\frac{(-\mathrm{cenc3x1}+\mathrm{enc3x1})^2+(-\mathrm{cenc3y1}+\mathrm{enc3y1})^2}{\sigma \mathrm{e}^2} - \frac{(-\mathrm{cenc3x2}+\mathrm{enc3x2})^2+(-\mathrm{cenc3y2}+\mathrm{enc3y2})^2}{\sigma \mathrm{e}^2} -$$
$$\frac{(-\mathrm{cenc3x3}+\mathrm{enc3x3})^2+(-\mathrm{cenc3y3}+\mathrm{enc3y3})^2}{\sigma \mathrm{e}^2} - \frac{(-\mathrm{cenc3x4}+\mathrm{enc3x4})^2+(-\mathrm{cenc3y4}+\mathrm{enc3y4})^2}{\sigma \mathrm{e}^2} -$$

$$\frac{(-\mathrm{cenc4x1}+\mathrm{enc4x1})^2+(-\mathrm{cenc4y1}+\mathrm{enc4y1})^2}{\sigma e^2} - \frac{(-\mathrm{cenc4x2}+\mathrm{enc4x2})^2+(-\mathrm{cenc4y2}+\mathrm{enc4y2})^2}{\sigma e^2} -$$
$$\frac{(-\mathrm{cenc4x3}+\mathrm{enc4x3})^2+(-\mathrm{cenc4y3}+\mathrm{enc4y3})^2}{\sigma e^2} - \frac{(-\mathrm{cenc4x4}+\mathrm{enc4x4})^2+(-\mathrm{cenc4y4}+\mathrm{enc4y4})^2}{\sigma e^2} -$$
$$\frac{(-\mathrm{c}\theta_1+\Delta\theta_1)^2}{\sigma\theta^2} - \frac{(-\mathrm{c}\theta_2+\Delta\theta_2)^2}{\sigma\theta^2} - \frac{(-\mathrm{c}\theta_3+\Delta\theta_3)^2}{\sigma\theta^2} - \frac{(-\mathrm{c}\theta_4+\Delta\theta_4)^2}{\sigma\theta^2} \quad (2.29)$$

Čia:

$\sigma e -$ standartinis nuokrypis enkoderių matavimams;

$\sigma a -$ standartinis nuokrypis akselerometro matavimams;

$\sigma\theta -$ standartinis nuokrypis gyroskopo matavimams;

$\Delta t -$ diskretizavimo periodas;

$enc1x1$ … $enc4y4$ – apskaičiuoti enkoderių poslinkiai, po Δt;

$cenc1x1$ … $cenc4y4$ – išmatuoti enkoderių poslinkiai, po Δt;

ax_{12} … Δay_{34} – prognozuojama pagreičio vertė, Δt intervale;

cax_{12} … cay_{34} – akselerometro pagreičio vertė, Δt intervale;

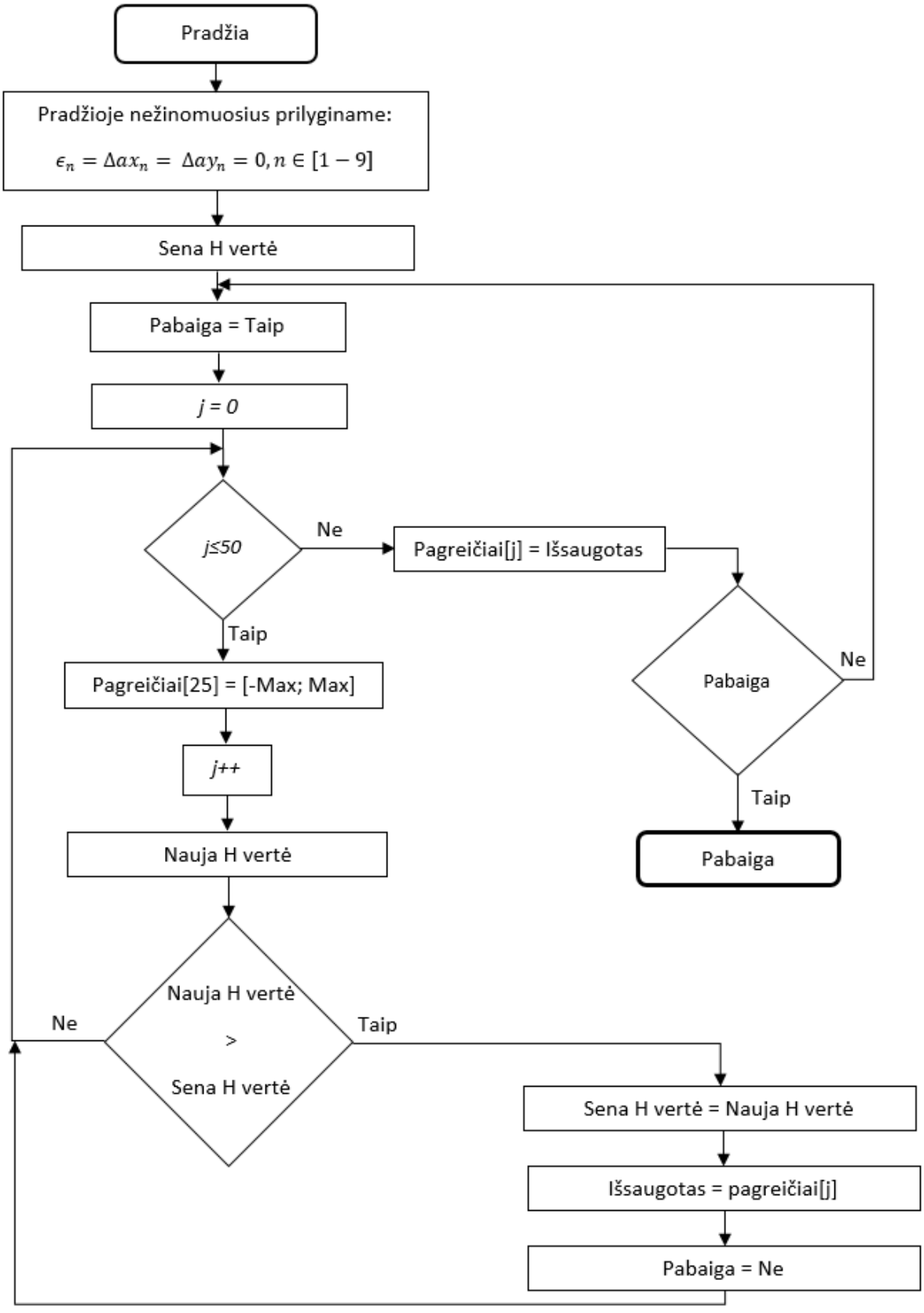
$\Delta\theta_1$ … $\Delta\theta_4$ – apskaičiuotas roboto pasisukimo kampas, po Δt;

$c\theta_1$ … $c\theta_4$ – giroskopu matuojamas pasisukimo kampas, po Δt.

2.2.2 Optimizavimo algoritmas

Kaip anksčiau minėjome filtro esmė yra prognozuoti matavimus, skaičiuojant svorinius koeficientus. Norint atrasti teisingiausius nežinomųjų įverčius yra atliekamas optimizacijos algoritmas, kiekvienu diskretizavimo metu.

Šiuo atveju turime optimizavimo uždavinį, kur tikslo funkcija galima laikyti Hamiltoniano išraišką ir algoritmo pagalba, prognozuojant nežinomuosius, bus ieškoma maksimalios tikslo funkcijos reikšmės.



2.10 pav. Optimizavimo algoritmas

Optimizavimo algoritmas naudojamas rasti geriausia nežinomųjų įverčių kombinaciją. Pradžioje ieškomi nežinomieji patalpinami į matricą - pagreičiai[25], kurios reikšmės yra atitinkamai: Δax_{23}, Δay_{23}, Δax_{34}, Δay_{34}, Δax_{45}, Δay_{45}, Δax_{56}, Δay_{56}, Δax_{67}, $\Delta ay_{67}, \Delta ax_{78}$, $\Delta ay_{78}, \Delta ax_{89}$, Δay_{89}, Δax_{910}, Δay_{910}, ϵ_{12}, ϵ_{23}, ϵ_{34}, ϵ_{45}, ϵ_{56}, ϵ_{67}, ϵ_{78}, ϵ_{89}, ϵ_{910};

Pirmiausia visos reikšmės yra prilyginamos nuliui ir skaičiuojama Hamiltoniano vertė, kuri prilyginama – sena H vertė. Prieš kiekvieną Hamiltoniano skaičiavimo žingsnį, taip pat yra skaičiuojamos reikšmės: r_1, sx_1, sy_1, ax_{12}, ay_{12}, $enc1x(n)$, $enc1y(n)$, $enc2x(n)$, $enc2y(n)$, $enc3x(n)$, $enc3y(n)$, $enc4x(n)$, $enc4y(n)$, $\Delta\theta(n)$. Gavus senąją Hamiltoniano vertę, pabaigos kintamajam yra priskiriama „Taip“ reikšmė ir toliau pradedamas ciklas, kur j – ciklų skaičius. Cikle pagreičio matricos kiekvienam kintamajam yra priskiriama arba maksimali arba minimali reikšmė, su kuria yra suskaičiuojama Hamiltoniano reikšmė ir jeigu ji didesnė už senąją Hamiltoniano reikšmė, tai ta vertė prilyginama naujo Hamiltoniano vertei. Toliau Išsaugomas tas matricos kintamojo indeksas bei jo vertė, o pabaigos bitas prilyginamas nuliui („Ne“). Jeigu palyginime Nauja Hamiltoniano vertė nėra didesnė už senąją vertę, grįžtama į ciklo pradžią. Po kiekvieno ciklo pakeistas matricos narys įgyją pirminę vertę t.y. nulį. Kai ciklas yra baigtas, rasta matricos vertė, su kuria Hamiltonianas buvo didžiausias, išsaugoma masyve. Kitame žingsnyje yra tikrinama ar pabaigos kintamojo bitas yra lygus nuliui („Ne“) ar vienetui („Taip“). Pabaigos bitas tampa lygus vienetui, kai vykdant ciklą bent kartą yra randama nežinomojo vertė, su kuria naujo Hamiltoniano vertė buvo didesnė nei seno Hamiltoniano vertė.

Algoritmo schema yra gana supaprastinta, bet pačio algoritmo esmė, kad vykdant pirmąją iteraciją, randamas to nežinomojo įvertis su kuriuo suskaičiuotas Hamiltonianas buvo didžiausias. Radus tą įvertį, vykdoma sekanti iteracija, po kurios randamas sekantis nežinomojo įvertis su kuriuo Hamiltonianas buvo didžiausias ir tai yra vykdoma iki tol, kol po iteracijos nebėra užfiksuota nė vienas nežinomojo įvertis, su kuriuo naujai suskaičiuotas Hamiltonianas yra didesnis už senąją Hamiltoniano reikšmę. Tokiu principu yra randami geriausi nežinomųjų įverčiai su kuriais skaičiuojama roboto trajektorija ir orientacija. Kadangi skaičiuojant Hamiltonianą yra vertinami tiek enkoderių matavimai, tiek giroskopo, tiek akselerometro, galima teigti, kad tai yra maksimalios entropijos filtras.

Visas algoritmas, leidžiantis apjungti jutiklių duomenis ir nustatantis roboto lokalizaciją, taip pat buvo įgyvendintas C# programavimo kalboje, naudojant Microsoft Visual Studio programavimo aplinką. Visa metodo programa yra pateikiama priede – priedas 3. Metodo C# programa. O žemiau yra pateikiamas optimizavimo programos supaprastintas atitikmuo, C# programavimo aplinkoje.

```
int i, j;
        for (i = 0; i <= 24; i++)
        {
           pagreiciai[i] = 0;
        }
        old_hamiltonian = CalculateHamiltonian( pagreiciai[0]... pagreiciai[24] );
        int candidate_j = -1;
        short candidate_verte = 0;
        int iteracijos = 0;
        bool done = false;
        while (!done)
        {
           iteracijos++;
           done = true;
           for (j = 0; j < 50; j++)
           {
              int ind = j >> 1;
              short prev_value = pagreiciai[ind];
              if ((j & 0x01) == 0x01)
              {
                 if (pagreiciai[ind] == -1)
                    pagreiciai[ind] = 0;
                 else
                    pagreiciai[ind] = -1;
              }
              else
              {
                 if (pagreiciai[ind] == 1)
                    pagreiciai[ind] = 0;
                 else
                    pagreiciai[ind] = 1;
              }
              short nauja_verte = pagreiciai[ind];

              delta_ax23 = max_a_XY * (double)pagreiciai[0];
              /*Tarpiniai nežinomieji priskiriami analogiškai*/
              epsilon910 = max_a_Z * (double)pagreiciai[24];

              new_hamiltonian = CalculateHamiltonian( delta_ax23 ... epsilon910);

              if (new_hamiltonian > old_hamiltonian)
              {
                 old_hamiltonian = new_hamiltonian;
                 candidate_j = j;
                 candidate_verte = nauja_verte;
                 done = false;
              }
              pagreiciai[ind] = prev_value;
           }
           if (!done)
              pagreiciai[candidate_j >> 1] = candidate_verte;
        }
```

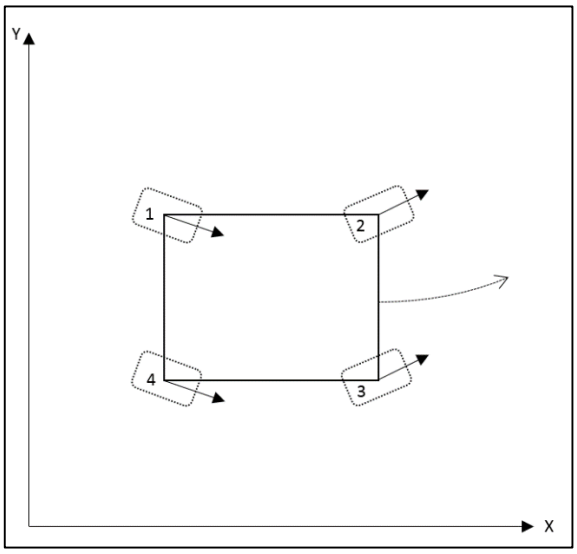
3 TYRIMO REZULTATŲ DALIS

3.1 Eksperimentinio tyrimo tikslas ir uždaviniai

Atliekamų eksperimentų tikslas – ištirti sukurto keturračio roboto savosios lokalizacijos metodo veikimą, remiantis sukurtu keturračio roboto matematiniu modeliu. Tikslui įgyvendinti yra būtina ištirti roboto matematinio modelio adekvatumą, remiantis standaus kūno dinamikos dėsniais. Priėmus išvadą, kad roboto matematinis modelis yra adekvatus realiam roboto modeliui, reikia atlikti eksperimentus su filtru, apjungiančiu enkoderių, akselerometro ir giroskopo matavimus. Šiai užduočiai įgyvendinti bus naudojami matematinio modelio enkoderių, pagreičių ir kampinio greičio matavimai, prie kurių bus pridėtas triukšmas, paremtas Gauso skirstiniu. Gautus triukšmingus matavimus naudosime filtro algoritme, po kurio triukšmingi signalai taps nufiltruoti ir bus naudojami roboto lokalizacijai patikslinti. Skirtumas tarp triukšmingų ir nufiltruotų rezultatų yra siekiamas tyrimo rezultatas.

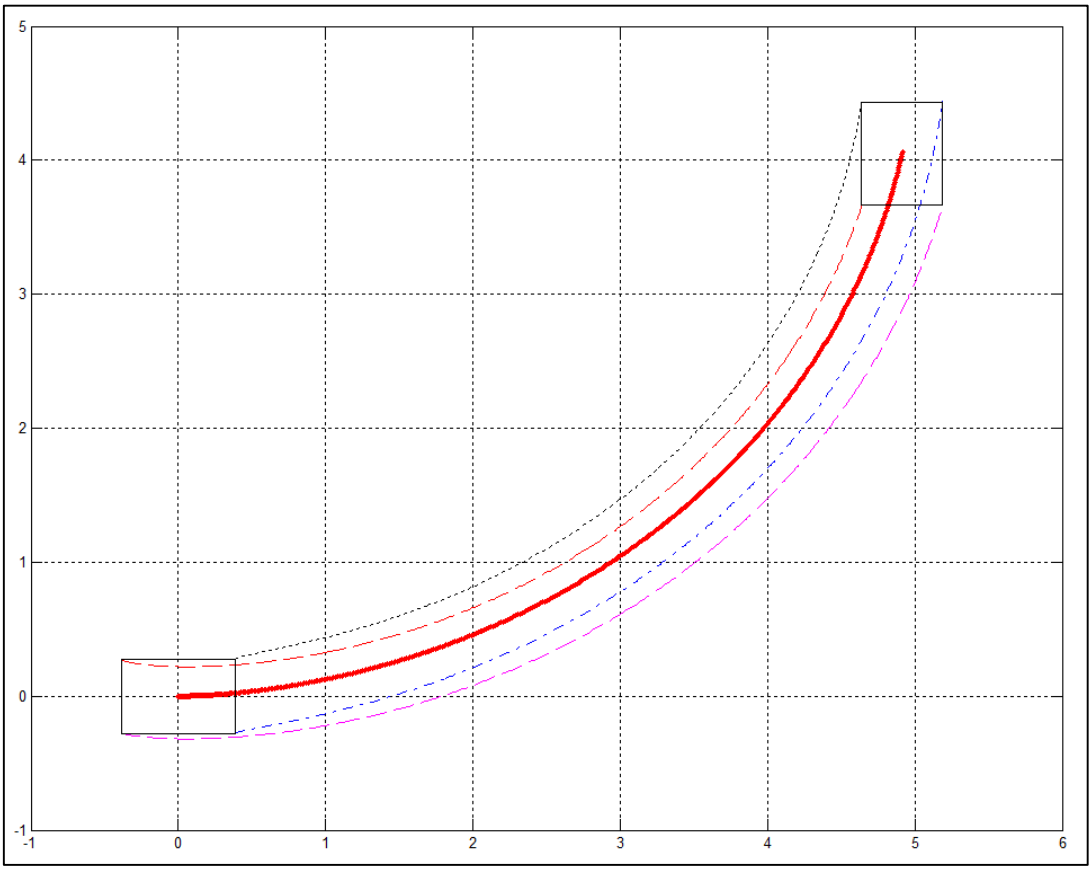
3.2 Roboto matematinio modelio tyrimas

Sudarius Matlab – Simulink matematinį modelį yra atliekama eilė eksperimentų, įsitikinant, kad roboto trajektorija atitinka realią roboto dinamikos teoriją. Pirmasis matematinio modelio eksperimentas atliekamas padavus į kiekvieną rato variklį tą pačią valdymo įtampą ir pasukus roboto ratus kaip parodyta 3.1 paveiksle.



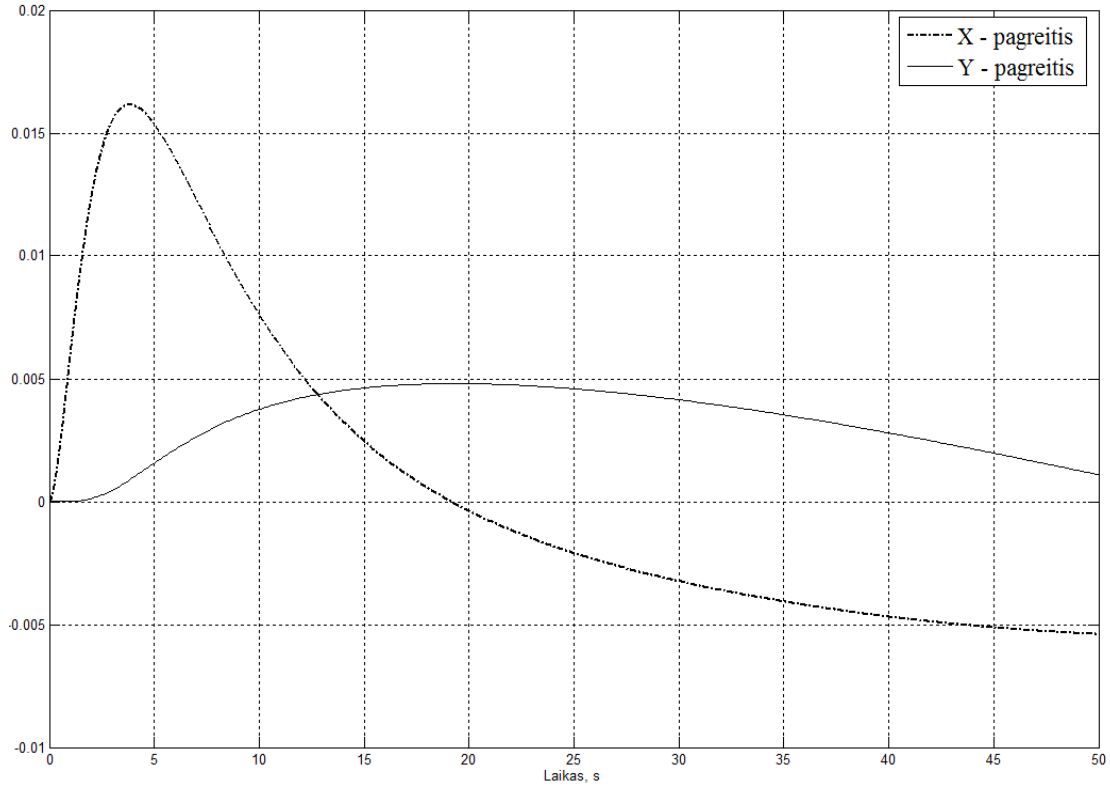
3.1. pav. Eksperimentams naudojamas roboto matematinis modelis

Čia visi ratai yra pasukti tuo pačiu kampu (π/6), tik pirmas ir ketvirtas į neigiamą pusę, o antras ir trečias į teigiamą. Simuliacijos laikas – 50s. Trajektorija yra pateikiama 3.2 paveiksle.



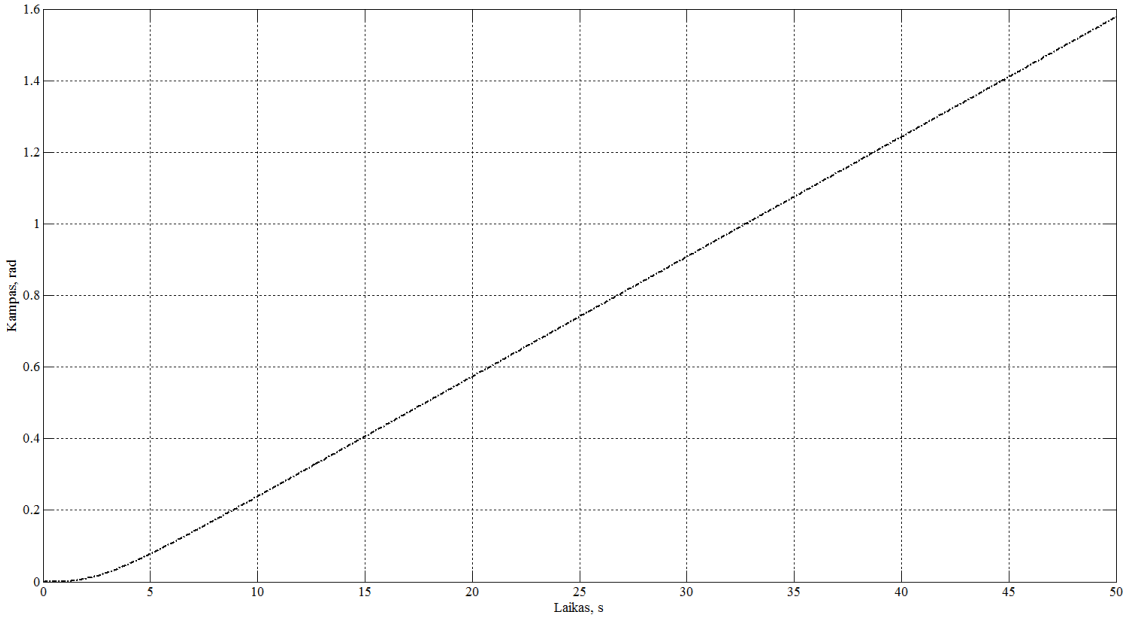
3.2. pav. Roboto trajektorijos eksperimentas

Pagal šią roboto trajektoriją 3.3 paveiksle yra pateikiamas roboto centro judėjimo pagreičio projekcijos į x ir y ašis.



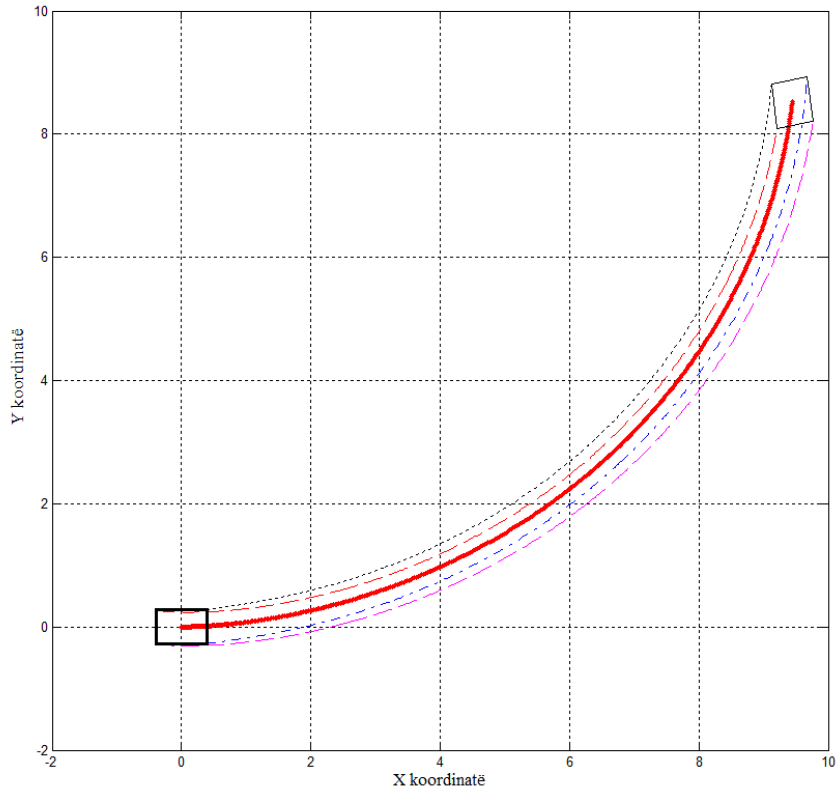
3.3. pav. Roboto centro pagreičio projekcijos

Toliau pagal aukščiau aprašytą judesį 3.4 paveiksle yra pateikiamas roboto kampo pokytis, vykdant anksčiau minėtą simuliaciją.



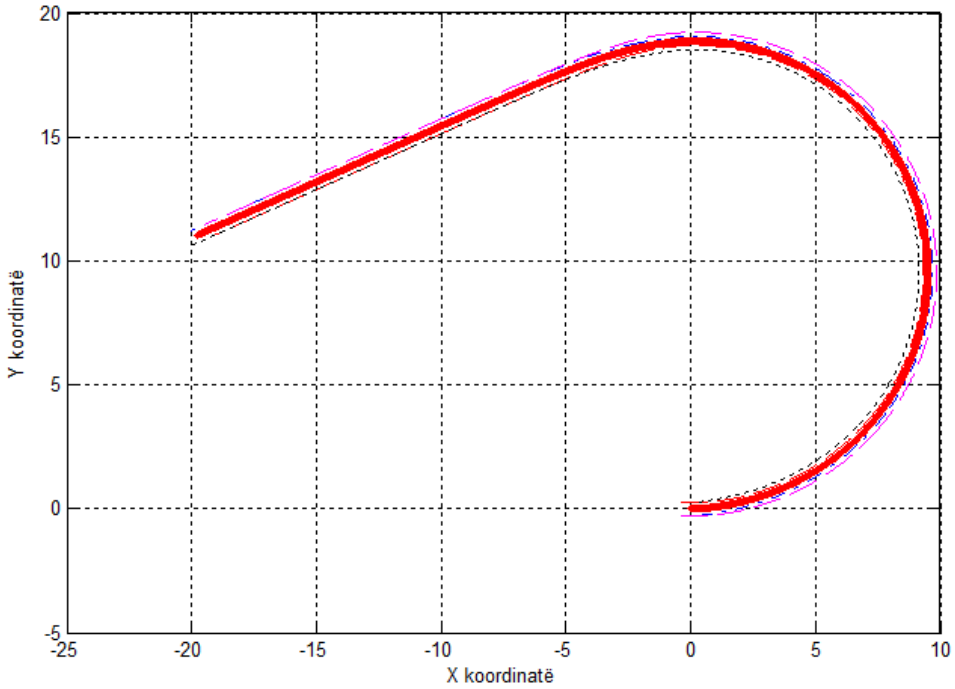
3.4. pav. Roboto orientacija

Todėl pirmojo eksperimento rezultatai iš esmės atitinka norėtą gauti rezultatą. Kitame žingsnyje atliekami dar keletas eksperimentų, gaunant skirtingas trajektorijas, norint įsitikinti modelio tinkamumu. Sekantis eksperimentas, kai visi roboto ratų pasukimo kampai yra lygūs nuliui, o įtampa paduodama į ratus – skirtinga. Į trečią ir ketvirtą ratą paduota įtampa yra didesnė, nei paduodama į pirmą ir antrą (žr. 3.1 pav.). Simuliacijos laikas – 50s, o trajektorija – pateikiama 3.5 paveiksle.



3.5. pav. Roboto trajektorija, kai ratai nepasukti, o 3 ir 4 rato įtampa – didesnė nei 1 ir 2

Kitame žingsnyje atliekame modelio eksperimentus su skirtingomis sąlygomis. Eksperimento rezultatai, kai visi ratai nepasukti, bet įtampa į vieną puse paduodama didesnė nei į kitos pusės ratus, o po kurio laiko išlyginama. Rezultatas pavaizduotas 3.6 paveiksle.



3.6. pav. Roboto trajektorija

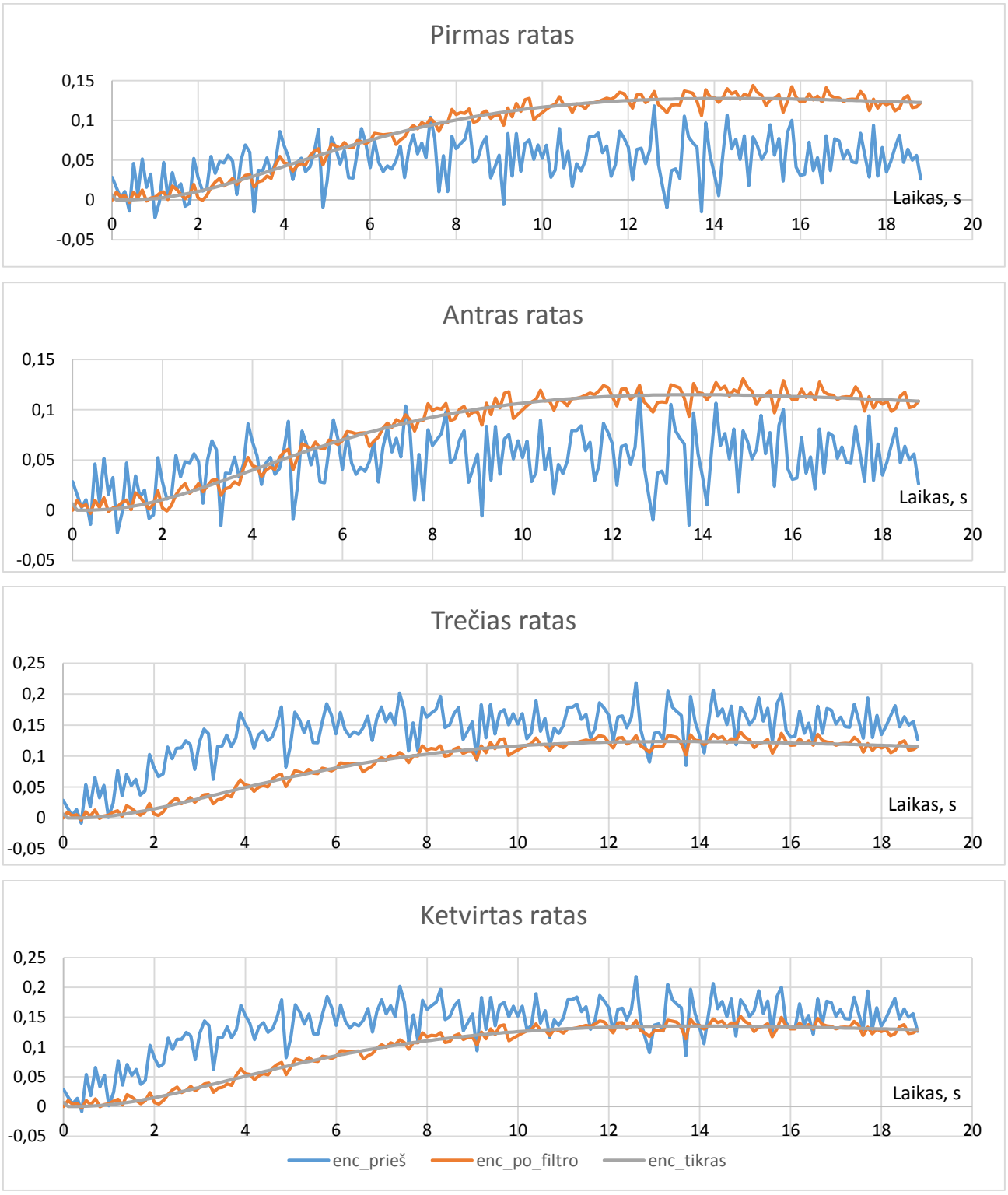
Atlikus eilę kitų eksperimentų, kur buvo stebimos įvairios trajektorijos, kurios priklausė nuo ratų orientacijos ir ratų sukimo momento, buvo įsitikinta matematinio modelio adekvatumu realiam robotui, o simuliacijos duomenys buvo naudojami sukurto maksimalios entropijos filtro tyrimui.

3.3 Savosios roboto lokalizacijos metodo tyrimas

Sukūrus filtrą, triukšmingiems duomenims apdoroti, buvo nustatytas filtro tinkamumas matematinio modelio panaudojimui. Prognozavimo filtrui yra naudojama dešimt diskretizavimo žingsnių, kur apdorojami enkoderių, akselerometro ir giroskopo matavimai. Enkoderio matavimai yra gaunami tiesiogiai iš sukurto rato variklio (žr. 2.3 pav.) matematinio modelio, kur prie matavimo yra pridedamas Gauso triukšmas, kurio vidurkis lygus nuliui, o dispersija lygi 0,0006 [m/s]. Sekantis matavimas yra x ašies ir y ašies pagreičiai, kurie gaunami taip pat iš matematinio modelio. Prie šitų matavimu taip pat pridedamas Gauso triukšmas, kurio vidurkis lygus nuliui, o dispersija lygi 0,01[m/s^2]. Toliau į filtrą paduodame posūkio kampą, prie kurio taip pat yra pridedamas Gauso triukšmas su vidurkiu lygiu nuliui ir dispersija lygia 0,2[rad].

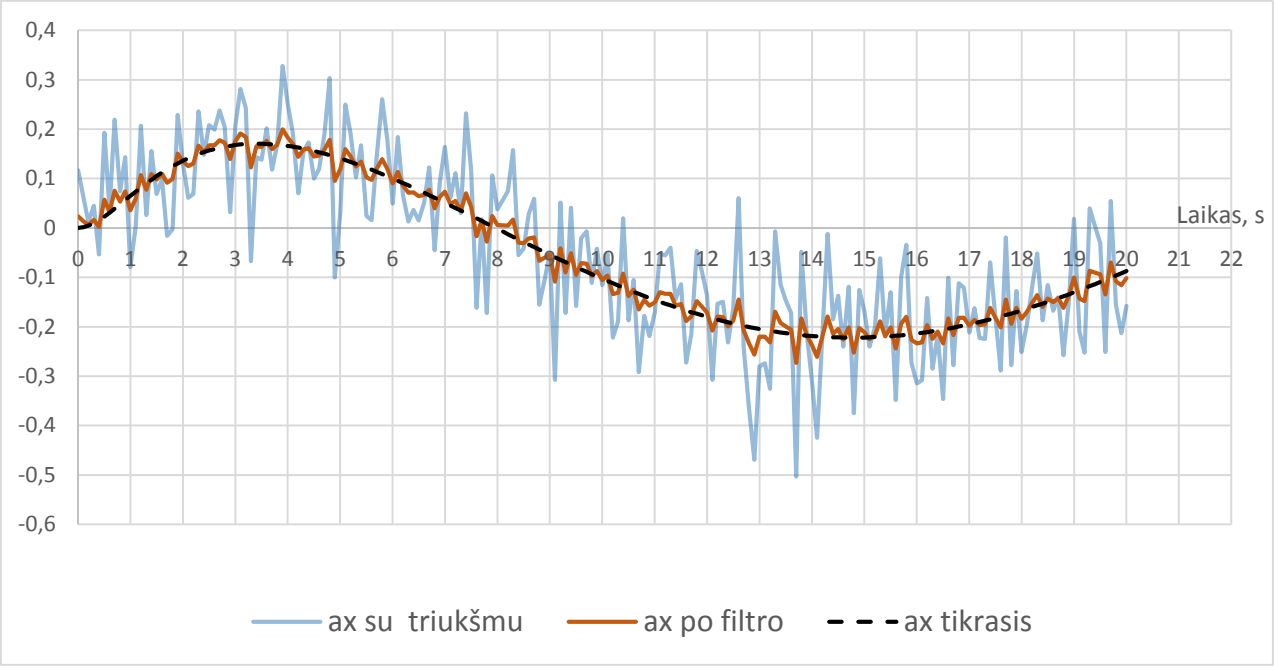
Turint triukšmingus matavimus ir naudojant mūsų aprašyta maksimalios entropijos filtrą galime gauti jau nufiltruotus matavimus. Vienas iš didžiausių sukurto metodo privalumų yra tai, kad filtras geba atskirti, kai ratas sukasi ir slysta, kadangi santykinės entropijos filtras yra paremtas

fizikiniu keturračio roboto judėjimo modeliu, kuris ir leidžia apskaičiuoti tikrąjį rato nueitą kelią. Toliau palyginame enkoderių matavimus su triukšmu, tikruosius rato poslinkius ir poslinkius po filtro (3.7 pav.).



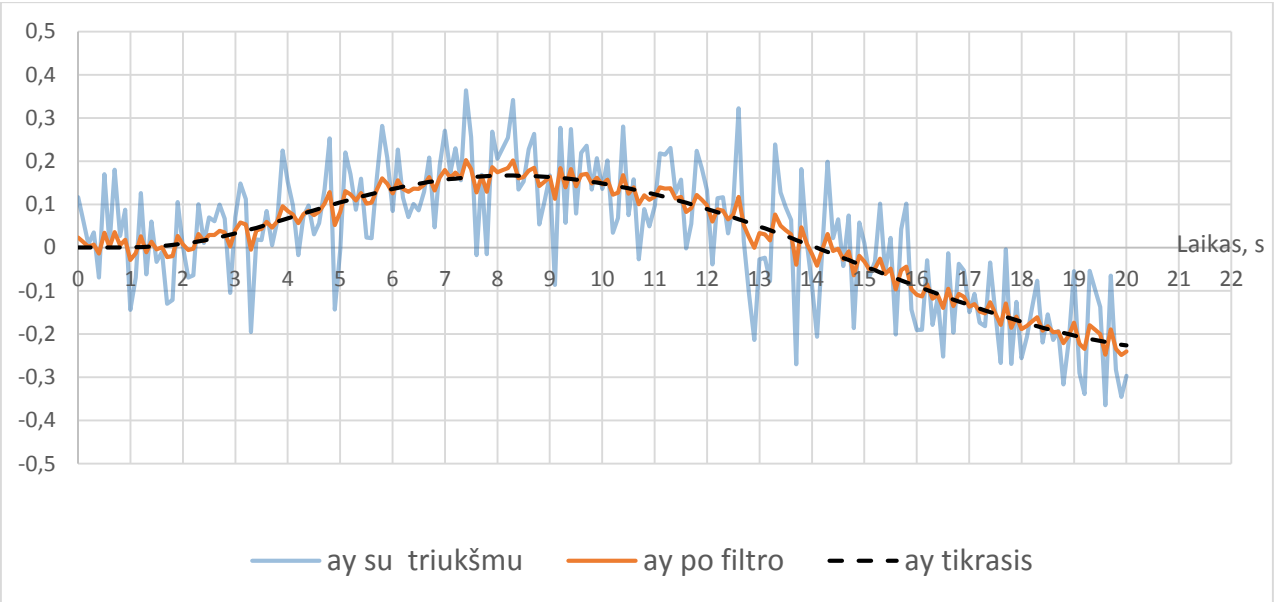
3.7 pav. Enkoderių matavimai prieš filtravimą ir parodymai po filtravimo

Toliau palyginame kitus matuojamus dydžius. Žemiau, 3.8 paveiksle, matome centro pagreičio, x projekcijos, kitimo grafiką, kur pavaizduoti matavimai be triukšmo, su triukšmu ir jau nufiltruoti matavimai.



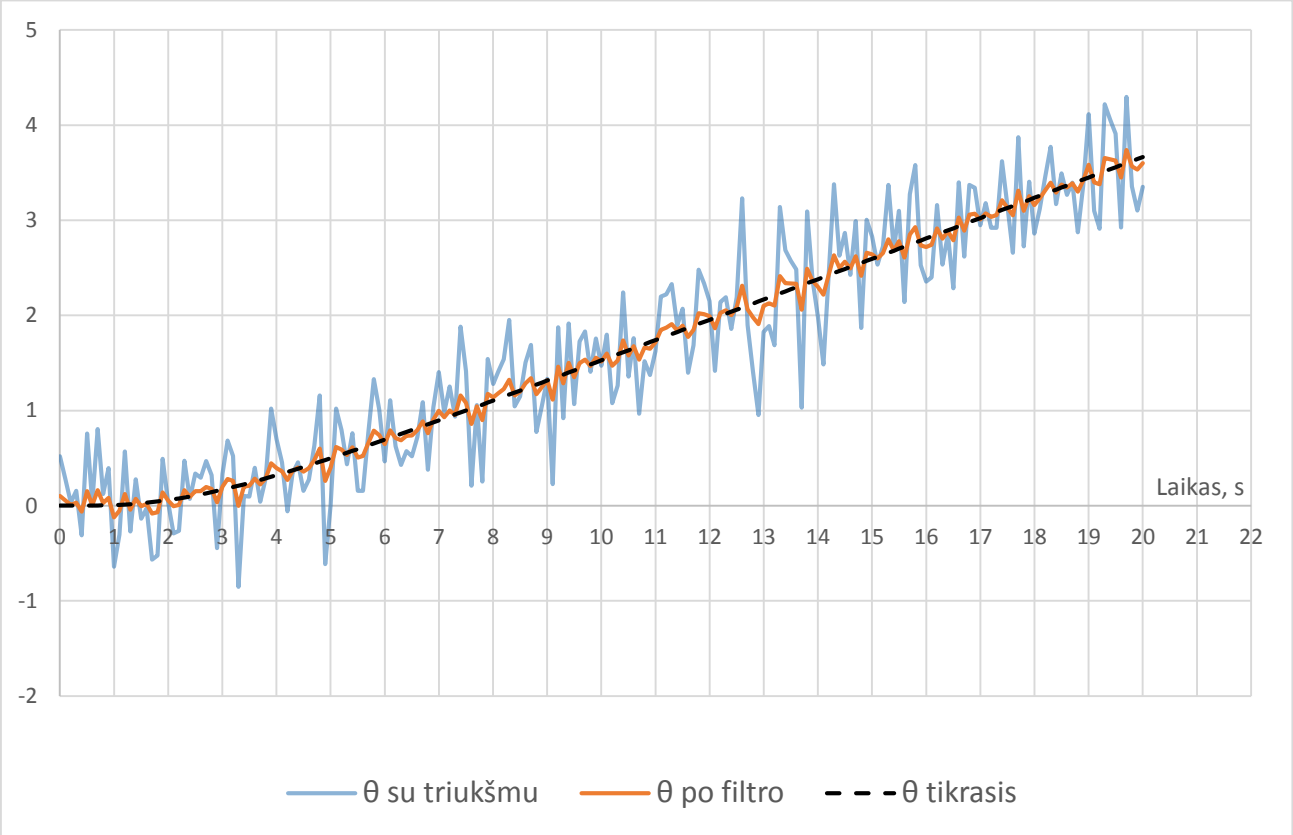
3.8 pav. Pagreičio x projekcija prieš filtravimą, po filtravimo ir tikroji

Taip pat yra gaunama pagreičio projekcija į y ašį, kuri pateikiama 3.9 paveiksle.



3.9 pav. Pagreičio y projekcija prieš filtravimą, po filtravimo ir tikroji

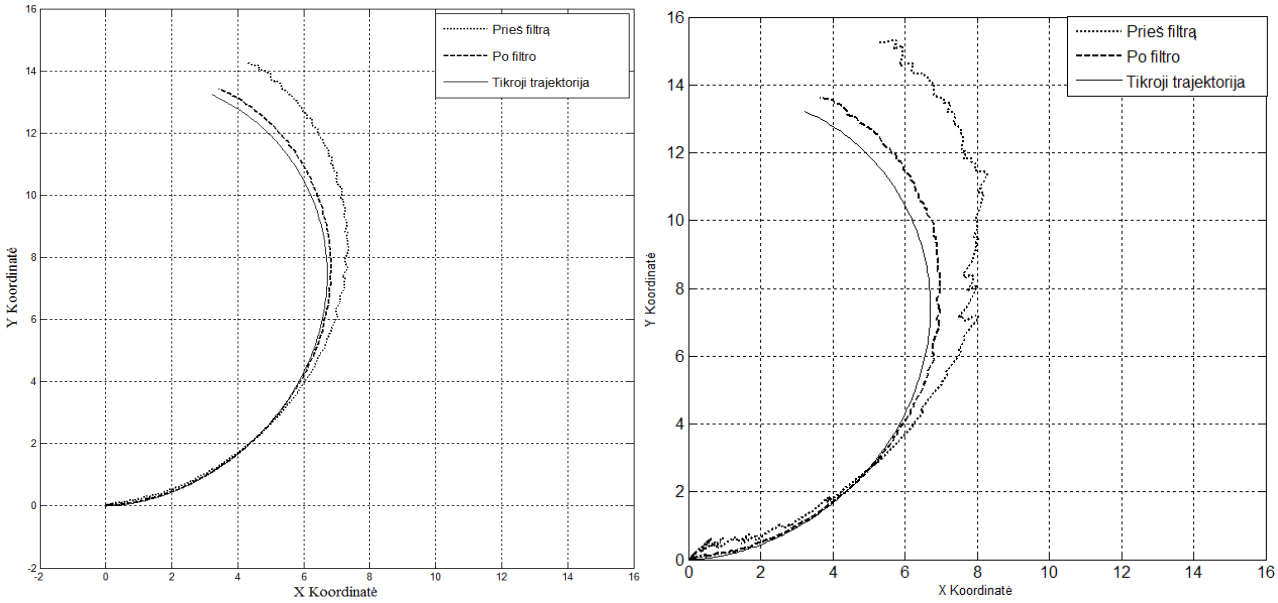
Analogiškai atliekamas posūkio kampo eksperimentas, kuris pateikiamas 3.10 paveiksle.



3.10 pav. Pasisukimo kampas θ prieš filtravimą, po filtravimą ir tikrasis

Gavus matavimus, galima apskaičiuoti triukšmo lygi prieš filtravimą ir po filtravimo. Gauti rezultatai rodo, kad bendru atveju triukšmo lygis signale sumažėja apie penkis kartus. Enkoderių duomenų palyginime matosi, kad filtras geba apskaičiuoti tikrąjį rato poslinkį. Apskaičiavus pagreičio projekcijos į x ašį triukšmo lygį, buvo gauta vidutinė – procentinė paklaida signalo prieš filtrą – 8,22% ir po filtro – 1,646 %. Maksimali absoliutinė paklaida prieš filtrą - 0,29 m/s^2, o po filtro – 0,06 m/s^2. Y ašies pagreičio projekcijos triukšmo lygio vidutinė – procentinė paklaida prieš filtrą – 8,217 % ir po filtro – 1,643 %. Maksimali absoliutinė paklaida prieš filtrą – 0,287 m/s^2, o po filtro – 0,0575 m/s^2. Posūkio kampo vidutinė – procentinė paklaida matavimo prieš filtrą – 36,74 % ir po filtro – 7,349%. Maksimali absoliutinė paklaida prieš filtrą – 1,2847 rad, o po filtro – 0,257rad. Todėl šitie rezultatai leidžia teigti, kad sukurtas santykinės entropijos filtras veikia tinkamai.

Kitame žingsnyje atliekamas roboto trajektorijos palyginimas tarp matavimų, naudojant triukšmingus, tikruosius ir nufiltruotus signalus. Rezultatai yra matomi 3.11 paveiksle.



3.11 pav. Trajektorijos palyginimas su skirtingomis triukšmo dispersijomis

Atlikus eksperimentą buvo nustatyta, kad per 20 sekundžių absoliutinis nuokrypis nuo tikrosios padėties su sukurtu savosios lokalizacijos metodu buvo 0,3 m, o naudojant matavimus su triukšmais – 1,51 m. Rezultatai buvo gauti, kai enkoderių triukšmo dispersija – 0,00003 [m/s]. Kai enkoderių matavimo triukšmo dispersija – 0,006 [m/s], per 20 sekundžių absoliutinis nuokrypis nuo tikrosios padėties su sukurtu savosios lokalizacijos metodu buvo 0,61 m, o naudojant matavimus su triukšmais – 2,97 m.

IŠVADOS IR REZULTATAI

1. Sukurtas keturračio roboto savosios lokalizacijos metodas, įvertinant enkoderių, akselerometro ir giroskopo matavimus bei ištirtas jo tinkamumas sukurtam matematiniam modeliui.
2. Eksperimentų metu buvo nustatyta, kad naudojant sukurtą santykinės entropijos filtrą, pagreičio projekcijos į x ir y ašį triukšmo lygio vidutinė – procentinė paklaida buvo prieš filtrą - 8,2 % ir po filtro – 1,6 %.
3. Eksperimentų metu buvo nustatyta, kad posūkio kampo vidutinė – procentinė paklaida matavimo prieš filtrą – 36,74 % ir po filtro – 7,349%.
4. Kai enkoderių matavimų triukšmo dispersija – 0,00003 m/s ir eksperimento trukmė - 20 s, absoliutinis nuokrypis nuo tikrosios padėties su sukurtu savosios lokalizacijos metodu buvo 0,3 m, o naudojant matavimus su triukšmu – 1,51 m. Dispersijai esant 0,006 m/s, absoliutinis nuokrypis nuo tikrosios padėties buvo 0,61 m po filtravimo ir 2,97 m su užtriukšmintomis vertėmis.

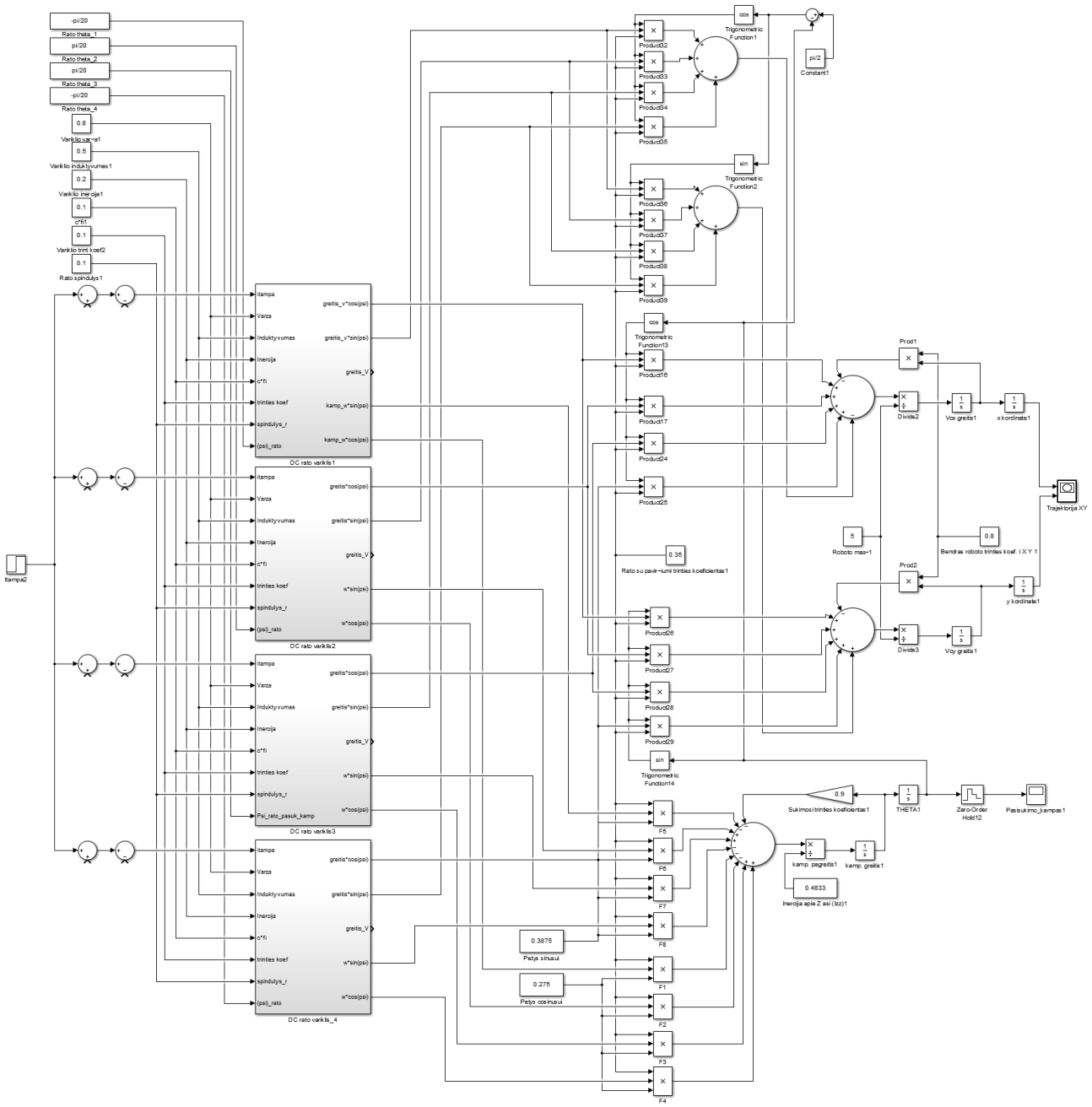
LITERATŪROS SĄRAŠAS

1. Global Positioning System. [Žiūrėta 2015-05-20]. Internetinė prieiga: http://goo.gl/YnLCG1
2. Biswas J., Veloso M. WiFi Localization and Navigation for Autonomous Indoor Mobile Robots. [Žiūrėta 2015-05-20]. Internetinė prieiga: http://goo.gl/Zagkhz
3. Navaro-Serment L. E., Paredis C. J., Khosla P. K. A Beacon System for the Localization of Distributed Robotic Teams. [Žiūrėta 2015-05-20]. Internetinė prieiga: http://goo.gl/ZJe51Z
4. Li L., Hu P., Peng C., Shen G., Zhao F. A Visible Light Based Positioning System. [Žiūrėta 2015-05-20]. Internetinė prieiga: http://goo.gl/mLe32F
5. Otsason V., Varshavsky A., Lamarca A., Lara E. Accurate GSM Indoor Localization. [Žiūrėta 2015-05-20]. Internetinė prieiga: https://goo.gl/T82B3k
6. Jung J., Lee S., Myung H. Indoor Mobile Robot Localization Using Ambient Magnetiic Fields and Range Measurements. [Žiūrėta 2015-05-20]. Internetinė prieiga: http://goo.gl/L7Ogpr
7. Woodman O. J. An introduction to inertial navigation. [Žiūrėta 2015-05-20]. Internetinė prieiga: http://goo.gl/DwUl8N
8. Foxlin E. NavShoe navigation technology. [Žiūrėta 2015-05-20]. Internetinė prieiga: https://goo.gl/onh1Ch
9. U-Blox positioning modules. [Žiūrėta 2015-05-20]. Internetinė prieiga: http://goo.gl/by5N0U
10. Jung D., Teixeira T., Savvides A. Towards Cooperative Localization of Wearable Sensors using Accelerometers and Cameras. [Žiūrėta 2015-05-20]. Internetinė prieiga: http://goo.gl/VIO85b
11. Faragher R. Understanding the Basis of the Kalman Filter. [Žiūrėta 2015-05-20]. Internetinė prieiga: http://goo.gl/klbYZf
12. Zunaidi I., Kato N., Nomura Y., Matsui H. Positioning System for 4-Wheel Mobile Robot: Encoder, Gyro and Accelerometer Data Fusion. [Žiūrėta 2015-05-20]. Internetinė prieiga: http://goo.gl/NyBnuU
13. Urniežius R. Research and development of dead reckoning localization method. [Žiūrėta 2015-05-20]. Internetinė prieiga: http://goo.gl/uKCaGR

14. Urniežius R., Griffin A. Simultaneous State and Parameter Estimation Using Maximum Relative Entropy with Nonhomogenous Differential Equation Constraints. [Žiūrėta 2015-05-20]. Internetinė prieiga: http://goo.gl/6fKy4L
15. Spong M. W., Hutchinson S., Vidyasagar M. Robot Dynamics and Control. [Žiūrėta 2015-05-20]. Internetinė prieiga: http://goo.gl/jNHX2X
16. Urniežius R., Griffin A. The Kalman Filter Revisited Using Maximum Relative Entropy. [Žiūrėta 2015-05-20]. Internetinė prieiga: http://goo.gl/vj4nH5

PRIEDAI

Priedas 1. Matlab – Simulink keturračio roboto matematinis modelis



Priedas 2. Filtro matematinės išraiškos

$$r_1 = \frac{1}{-\frac{32}{\sigma e^2}-\frac{259200}{L^2\pi^2\sigma\theta^2}}\left(\frac{2L\pi\Delta t^2(3\epsilon_{12}+2\epsilon_{23}+\epsilon_{34})}{45\sigma e^2} - \frac{360\left(c\theta_1+c\theta_2+c\theta_3+c\theta_4-\Delta t^2(3\epsilon_{12}+2\epsilon_{23}+\epsilon_{34})\right)}{L\pi\sigma\theta^2} + \right.$$

$$\frac{1}{\sigma e^2}(-2(ce1y1 + ce1y2 + ce1y3 + ce1y4 - ce2y1 - ce2y2 - ce2y3 - ce2y4 - ce3y1 -$$

$$ce3y2 - ce3y3 - ce3y4 + ce4y1 + ce4y2 + ce4y3 + ce4y4)\cos[\alpha] - 2(ce1x1 + ce1x2 +$$

$$ce1x3 + ce1x4 + ce2x1 + ce2x2 + ce2x3 + ce2x4 - ce3x1 - ce3x2 - ce3x3 - ce3x4 -$$

$$\left.ce4x1 - ce4x2 - ce4x3 - ce4x4)\sin[\alpha])\right)$$

$$sx_1 = \frac{1}{16}(-4(3ax_{12} + 2ax_{23} + ax_{34})\Delta t^2 + (ce1x1 + ce1x2 + ce1x3 + ce1x4 + ce2x1 + ce2x2 +$$

$$ce2x3 + ce2x4 + ce3x1 + ce3x2 + ce3x3 + ce3x4 + ce4x1 + ce4x2 + ce4x3 + ce4x4)\cos[\alpha] -$$

$$(ce1y1 + ce1y2 + ce1y3 + ce1y4 + ce2y1 + ce2y2 + ce2y3 + ce2y4 + ce3y1 + ce3y2 +$$

$$ce3y3 + ce3y4 + ce4y1 + ce4y2 + ce4y3 + ce4y4)\sin[\alpha])$$

$$sy_1 = \frac{1}{16}(-4(3ay_{12} + 2ay_{23} + ay_{34})\Delta t^2 + (ce1y1 + ce1y2 + ce1y3 + ce1y4 + ce2y1 +$$

$$ce2y2 + ce2y3 + ce2y4 + ce3y1 + ce3y2 + ce3y3 + ce3y4 + ce4y1 + ce4y2 + ce4y3 +$$

$$ce4y4)\cos[\alpha] + (ce1x1 + ce1x2 + ce1x3 + ce1x4 + ce2x1 + ce2x2 + ce2x3 + ce2x4 +$$

$$ce3x1 + ce3x2 + ce3x3 + ce3x4 + ce4x1 + ce4x2 + ce4x3 + ce4x4)\sin[\alpha])$$

$$r_2 = r_1 + \frac{1}{180}L\pi\Delta t^2\epsilon_{12};$$

$$sx_2 = sx_1 + ax_{12}\Delta t^2;$$

$$sy_2 = sy_1 + ay_{12}\Delta t^2$$

$$r_3 = \frac{1}{180}(180r_1 + L\pi\Delta t^2(\epsilon_{12} + \epsilon_{23}));$$

$$sx_3 = sx_1 + (ax_{12} + ax_{23})\Delta t^2;$$

$$sy_3 = sy_1 + (ay_{12} + ay_{23})\Delta t^2$$

$$r_4 = \frac{1}{180}(180r_1 + L\pi\Delta t^2(\epsilon_{12} + \epsilon_{23} + \epsilon_{34}));$$

$$sx_4 = sx_1 + (ax_{12} + ax_{23} + ax_{34})\Delta t^2;$$

$$sy_4 = sy_1 + (ay_{12} + ay_{23} + ay_{34})\Delta t^2;$$

$$H = -\frac{(\mathrm{ax}_{12}-\mathrm{cax}_{12})^2+(\mathrm{ay}_{12}-\mathrm{cay}_{12})^2}{\sigma \mathrm{a}^2} - \frac{(\mathrm{ax}_{12}+\Delta\mathrm{ax}_{23}-\mathrm{cax}_{23})^2+(\mathrm{ay}_{12}+\Delta\mathrm{ay}_{23}-\mathrm{cay}_{23})^2}{\sigma \mathrm{a}^2} -$$

$$\frac{(\mathrm{ax}_{12}+\Delta\mathrm{ax}_{23}+\Delta\mathrm{ax}_{34}-\mathrm{cax}_{34})^2+(\mathrm{ay}_{12}+\Delta\mathrm{ay}_{23}+\Delta\mathrm{ay}_{34}-\mathrm{cay}_{34})^2}{\sigma \mathrm{a}^2} -$$

$$\frac{(-\mathrm{cenc1x1}+\mathrm{enc1x1})^2+(-\mathrm{cenc1y1}+\mathrm{enc1y1})^2}{\sigma \mathrm{e}^2} - \frac{(-\mathrm{cenc1x2}+\mathrm{enc1x2})^2+(-\mathrm{cenc1y2}+\mathrm{enc1y2})^2}{\sigma \mathrm{e}^2} -$$

$$\frac{(-\mathrm{cenc1x3}+\mathrm{enc1x3})^2+(-\mathrm{cenc1y3}+\mathrm{enc1y3})^2}{\sigma \mathrm{e}^2} - \frac{(-\mathrm{cenc1x4}+\mathrm{enc1x4})^2+(-\mathrm{cenc1y4}+\mathrm{enc1y4})^2}{\sigma \mathrm{e}^2} -$$

$$\frac{(-\mathrm{cenc2x1}+\mathrm{enc2x1})^2+(-\mathrm{cenc2y1}+\mathrm{enc2y1})^2}{\sigma \mathrm{e}^2} - \frac{(-\mathrm{cenc2x2}+\mathrm{enc2x2})^2+(-\mathrm{cenc2y2}+\mathrm{enc2y2})^2}{\sigma \mathrm{e}^2} -$$

$$\frac{(-\mathrm{cenc2x3}+\mathrm{enc2x3})^2+(-\mathrm{cenc2y3}+\mathrm{enc2y3})^2}{\sigma \mathrm{e}^2} - \frac{(-\mathrm{cenc2x4}+\mathrm{enc2x4})^2+(-\mathrm{cenc2y4}+\mathrm{enc2y4})^2}{\sigma \mathrm{e}^2} -$$

$$\frac{(-\mathrm{cenc3x1}+\mathrm{enc3x1})^2+(-\mathrm{cenc3y1}+\mathrm{enc3y1})^2}{\sigma \mathrm{e}^2} - \frac{(-\mathrm{cenc3x2}+\mathrm{enc3x2})^2+(-\mathrm{cenc3y2}+\mathrm{enc3y2})^2}{\sigma \mathrm{e}^2} -$$

$$\frac{(-\mathrm{cenc3x3}+\mathrm{enc3x3})^2+(-\mathrm{cenc3y3}+\mathrm{enc3y3})^2}{\sigma \mathrm{e}^2} - \frac{(-\mathrm{cenc3x4}+\mathrm{enc3x4})^2+(-\mathrm{cenc3y4}+\mathrm{enc3y4})^2}{\sigma \mathrm{e}^2} -$$

$$\frac{(-\mathrm{cenc4x1}+\mathrm{enc4x1})^2+(-\mathrm{cenc4y1}+\mathrm{enc4y1})^2}{\sigma \mathrm{e}^2} - \frac{(-\mathrm{cenc4x2}+\mathrm{enc4x2})^2+(-\mathrm{cenc4y2}+\mathrm{enc4y2})^2}{\sigma \mathrm{e}^2} -$$

$$\frac{(-\mathrm{cenc4x3}+\mathrm{enc4x3})^2+(-\mathrm{cenc4y3}+\mathrm{enc4y3})^2}{\sigma \mathrm{e}^2} - \frac{(-\mathrm{cenc4x4}+\mathrm{enc4x4})^2+(-\mathrm{cenc4y4}+\mathrm{enc4y4})^2}{\sigma \mathrm{e}^2} -$$

$$\frac{(-\mathrm{c}\theta_1+\Delta\theta_1)^2}{\sigma\theta^2} - \frac{(-\mathrm{c}\theta_2+\Delta\theta_2)^2}{\sigma\theta^2} - \frac{(-\mathrm{c}\theta_3+\Delta\theta_3)^2}{\sigma\theta^2} - \frac{(-\mathrm{c}\theta_4+\Delta\theta_4)^2}{\sigma\theta^2}$$

$$\mathrm{enc1x1} = sx_1;$$

$$\mathrm{enc1y1} = r_1 + sy_1;$$

$$\mathrm{enc2x1} = sx_1 + r_1 \sin[2\alpha];$$

$$\mathrm{enc2y1} = sy_1 - r_1 \cos[2\alpha];$$

$$\mathrm{enc3x1} = sx_1;$$

$$\mathrm{enc3y1} = -r_1 + sy_1;$$

$$\mathrm{enc4x1} = sx_1 - r_1 \sin[2\alpha];$$

$$\mathrm{enc4y1} = sy_1 + r_1 \cos[2\alpha];$$

$$\mathrm{enc1x2} = sx_1 + \mathrm{ax}_{12}\Delta\mathrm{t}^2;$$

$$\mathrm{enc1y2} = r_1 + sy_1 + \mathrm{ay}_{12}\Delta\mathrm{t}^2 + \frac{1}{180}\mathrm{L}\pi\Delta\mathrm{t}^2\epsilon_{12}$$

$$\mathrm{enc2x2} = sx_1 + \mathrm{ax}_{12}\Delta\mathrm{t}^2 + \left(r_1 + \frac{1}{180}\mathrm{L}\pi\Delta\mathrm{t}^2\epsilon_{12}\right)\sin[2\alpha];$$

$$\mathrm{enc2y2} = sy_1 + \mathrm{ay}_{12}\Delta\mathrm{t}^2 - (r_1 + \frac{1}{180}\mathrm{L}\pi\Delta\mathrm{t}^2\epsilon_{12})\cos[2\alpha]$$

$$\text{enc3x2} = sx_1 + \text{ax}_{12}\Delta\text{t}^2,$$

$$\text{enc3y2} = -r_1 + sy_1 + \text{ay}_{12}\Delta\text{t}^2 - \frac{1}{180}\text{L}\pi\Delta\text{t}^2\epsilon_{12}$$

$$\text{enc4x2} = sx_1 + \text{ax}_{12}\Delta\text{t}^2 - \left(r_1 + \frac{1}{180}\text{L}\pi\Delta\text{t}^2\epsilon_{12}\right)\sin[2\alpha];$$

$$\text{enc4y2} = sy_1 + \text{ay}_{12}\Delta\text{t}^2 + (r_1 + \frac{1}{180}\text{L}\pi\Delta\text{t}^2\epsilon_{12})\cos[2\alpha]$$

$$\text{enc1x3} = sx_1 + (\text{ax}_{12} + \text{ax}_{23})\Delta\text{t}^2;$$

$$\text{enc1y3} = r_1 + sy_1 + \frac{1}{180}\Delta\text{t}^2(180\text{ay}_{12} + 180\text{ay}_{23} + \text{L}\pi(\epsilon_{12} + \epsilon_{23}))$$

$$\text{enc2x3} = sx_1 + (\text{ax}_{12} + \text{ax}_{23})\Delta\text{t}^2 + \frac{1}{180}\left(180r_1 + \text{L}\pi\Delta\text{t}^2(\epsilon_{12} + \epsilon_{23})\right)\sin[2\alpha];$$

$$\text{enc2y3} = sy_1 + (\text{ay}_{12} + \text{ay}_{23})\Delta\text{t}^2 - \frac{1}{180}\left(180r_1 + \text{L}\pi\Delta\text{t}^2(\epsilon_{12} + \epsilon_{23})\right)\cos[2\alpha];$$

$$\text{enc3x3} = sx_1 + (\text{ax}_{12} + \text{ax}_{23})\Delta\text{t}^2,$$

$$\text{enc3y3} = sy_1 + (\text{ay}_{12} + \text{ay}_{23})\Delta\text{t}^2 + \frac{1}{180}(-180r_1 - \text{L}\pi\Delta\text{t}^2(\epsilon_{12} + \epsilon_{23}))$$

$$\text{enc4x3} = sx_1 + (\text{ax}_{12} + \text{ax}_{23})\Delta\text{t}^2 - \frac{1}{180}\left(180r_1 + \text{L}\pi\Delta\text{t}^2(\epsilon_{12} + \epsilon_{23})\right)\sin[2\alpha];$$

$$\text{enc4y3} = sy_1 + (\text{ay}_{12} + \text{ay}_{23})\Delta\text{t}^2 + \frac{1}{180}\left(180r_1 + \text{L}\pi\Delta\text{t}^2(\epsilon_{12} + \epsilon_{23})\right)\cos[2\alpha];$$

$$\text{enc1x4} = sx_1 + (\text{ax}_{12} + \text{ax}_{23} + \text{ax}_{34})\Delta\text{t}^2;$$

$$\text{enc1y4} = sy_1 + (\text{ay}_{12} + \text{ay}_{23} + \text{ay}_{34})\Delta\text{t}^2 + \frac{1}{180}\left(180r_1 + \text{L}\pi\Delta\text{t}^2(\epsilon_{12} + \epsilon_{23} + \epsilon_{34})\right);$$

$$\text{enc2x4} = sx_1 + (\text{ax}_{12} + \text{ax}_{23} + \text{ax}_{34})\Delta\text{t}^2 + \frac{1}{180}\left(180r_1 + \text{L}\pi\Delta\text{t}^2(\epsilon_{12} + \epsilon_{23} + \epsilon_{34})\right)\sin[2\alpha];$$

$$\text{enc2y4} = sy_1 + (\text{ay}_{12} + \text{ay}_{23} + \text{ay}_{34})\Delta\text{t}^2 - \frac{1}{180}\left(180r_1 + \text{L}\pi\Delta\text{t}^2(\epsilon_{12} + \epsilon_{23} + \epsilon_{34})\right)\cos[2\alpha];$$

$$\text{enc3x4} = sx_1 + (\text{ax}_{12} + \text{ax}_{23} + \text{ax}_{34})\Delta\text{t}^2;$$

$$\text{enc3y4} = sy_1 + (\text{ay}_{12} + \text{ay}_{23} + \text{ay}_{34})\Delta\text{t}^2 + \frac{1}{180}(-180r_1 - \text{L}\pi\Delta\text{t}^2(\epsilon_{12} + \epsilon_{23} + \epsilon_{34}));$$

$$\text{enc4x4} = sx_1 + (\text{ax}_{12} + \text{ax}_{23} + \text{ax}_{34})\Delta\text{t}^2 - \frac{1}{180}\left(180r_1 + \text{L}\pi\Delta\text{t}^2(\epsilon_{12} + \epsilon_{23} + \epsilon_{34})\right)\sin[2\alpha];$$

$$\text{enc4y4} = sy_1 + (\text{ay}_{12} + \text{ay}_{23} + \text{ay}_{34})\Delta\text{t}^2 + \frac{1}{180}(180r_1 + \text{L}\pi\Delta\text{t}^2(\epsilon_{12} + \epsilon_{23} + \epsilon_{34}))\cos[2\alpha];$$

$$\Delta\theta_1 = \frac{180r_1}{\text{L}\pi};$$

$$\Delta\theta_2 \to \frac{180r_1}{\text{L}\pi} + \Delta\text{t}^2\epsilon_{12};$$

$$\Delta\theta_3 \to \frac{180r_1 + \text{L}\pi\Delta\text{t}^2(\epsilon_{12} + \epsilon_{23})}{\text{L}\pi};$$

$$\Delta\theta_4 \to \frac{180r_1 + \text{L}\pi\Delta\text{t}^2(\epsilon_{12} + \epsilon_{23} + \epsilon_{34})}{\text{L}\pi};$$

Priedas 3. Metodo C# programa

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Microsoft.Office.Interop.Excel;

namespace OptimMatlabModel
{
   public partial class Form1 : Form
   {
        short[] pagreiciai = new short[25]; // delta_ax23, delta_ay23..... delta_ax910, delta_ay910, epsilon12, epsilon23,
epsilon34....epsilon910
                    // ce1x2  1- rato numeris; 2 - matavimo zingsnis
        double          ce1x1, ce1y1, ce2x1, ce2y1, ce3x1, ce3y1, ce4x1, ce4y1,
                        cTheta1,
                        ce1x2, ce1y2, ce2x2, ce2y2, ce3x2, ce3y2, ce4x2, ce4y2,
                        cTheta2,
                        ce1x3, ce1y3, ce2x3, ce2y3, ce3x3, ce3y3, ce4x3, ce4y3,
                        cTheta3,
                        ce1x4, ce1y4, ce2x4, ce2y4, ce3x4, ce3y4, ce4x4, ce4y4,
                        cTheta4,
                        ce1x5, ce1y5, ce2x5, ce2y5, ce3x5, ce3y5, ce4x5, ce4y5,
                        cTheta5,
                        ce1x6, ce1y6, ce2x6, ce2y6, ce3x6, ce3y6, ce4x6, ce4y6,
                        cTheta6,
                        ce1x7, ce1y7, ce2x7, ce2y7, ce3x7, ce3y7, ce4x7, ce4y7,
                        cTheta7,
                        ce1x8, ce1y8, ce2x8, ce2y8, ce3x8, ce3y8, ce4x8, ce4y8,
                        cTheta8,
                        ce1x9, ce1y9, ce2x9, ce2y9, ce3x9, ce3y9, ce4x9, ce4y9,
                        cTheta9,
                        ce1x10, ce1y10, ce2x10, ce2y10, ce3x10, ce3y10, ce4x10, ce4y10,
                        cTheta10;
                        //------------------------

        const double    Sigma_e = 0.01,
                        Sigma_Theta = 0.01,
                        Sigma_a = 0.01,
                        aL = 0.475,
                        Delta_t = 0.1,
                        pi = Math.PI,
                        ilgis = 77.5,
                        plotis = 55d,
                        max_a_XY = 0.087323643 * 0.2,
                        max_a_Z = 0.021388556 * 0.4,
                        psi1 = 0,
                        psi2 = 0,
                        psi3 = 0,
                        psi4 = 0;
```

```
//-------------------------------------------------------------------------------
    //Kintamieji
    //-------------------------------------------------------------------------------
    double r1, sx1, sy1, ax12, ay12,
       alpha,
       enc1x1, enc1y1, enc2x1, enc2y1, enc3x1, enc3y1, enc4x1, enc4y1,
       enc1x2, enc1y2, enc2x2, enc2y2, enc3x2, enc3y2, enc4x2, enc4y2,
       enc1x3, enc1y3, enc2x3, enc2y3, enc3x3, enc3y3, enc4x3, enc4y3,
       enc1x4, enc1y4, enc2x4, enc2y4, enc3x4, enc3y4, enc4x4, enc4y4,
       enc1x5, enc1y5, enc2x5, enc2y5, enc3x5, enc3y5, enc4x5, enc4y5,
       enc1x6, enc1y6, enc2x6, enc2y6, enc3x6, enc3y6, enc4x6, enc4y6,
       enc1x7, enc1y7, enc2x7, enc2y7, enc3x7, enc3y7, enc4x7, enc4y7,
       enc1x8, enc1y8, enc2x8, enc2y8, enc3x8, enc3y8, enc4x8, enc4y8,
       enc1x9, enc1y9, enc2x9, enc2y9, enc3x9, enc3y9, enc4x9, enc4y9,
       enc1x10, enc1y10, enc2x10, enc2y10, enc3x10, enc3y10, enc4x10, enc4y10,
       Delta_Theta_1, Delta_Theta_2, Delta_Theta_3, Delta_Theta_4, Delta_Theta_5,
       Delta_Theta_6, Delta_Theta_7, Delta_Theta_8, Delta_Theta_9, Delta_Theta_10,
       Hamiltonian, cax12, cax23, cax34, cax45, cax56,
       cax67, cax78, cax89, cax910,
       cay12, cay23, cay34, cay45, cay56,
       cay67, cay78, cay89, cay910,
       cenc1x1, cenc1y1, cenc1x2, cenc1y2, cenc1x3, cenc1y3, cenc1x4, cenc1y4,
       cenc1x5, cenc1y5, cenc1x6, cenc1y6, cenc1x7, cenc1y7, cenc1x8, cenc1y8,
       cenc1x9, cenc1y9, cenc1x10, cenc1y10,
       cenc2x1, cenc2y1, cenc2x2, cenc2y2, cenc2x3, cenc2y3, cenc2x4, cenc2y4,
       cenc2x5, cenc2y5, cenc2x6, cenc2y6, cenc2x7, cenc2y7, cenc2x8, cenc2y8,
       cenc2x9, cenc2y9, cenc2x10, cenc2y10,
       cenc3x1, cenc3y1, cenc3x2, cenc3y2, cenc3x3, cenc3y3, cenc3x4, cenc3y4,
       cenc3x5, cenc3y5, cenc3x6, cenc3y6, cenc3x7, cenc3y7, cenc3x8, cenc3y8,
       cenc3x9, cenc3y9, cenc3x10, cenc3y10,
       cenc4x1, cenc4y1, cenc4x2, cenc4y2, cenc4x3, cenc4y3, cenc4x4, cenc4y4,
       cenc4x5, cenc4y5, cenc4x6, cenc4y6, cenc4x7, cenc4y7, cenc4x8, cenc4y8,
       cenc4x9, cenc4y9, cenc4x10, cenc4y10,
       ce11, ce12, ce13, ce14,
       ce15, ce16, ce17, ce18,
       ce19, ce110,
       ce21, ce22, ce23, ce24,
       ce25, ce26, ce27, ce28,
       ce29, ce210,
       ce31, ce32, ce33, ce34,
       ce35, ce36, ce37, ce38,
       ce39, ce310,
       ce41, ce42, ce43, ce44,
       ce45, ce46, ce47, ce48,
       ce49, ce410,
       old_hamiltonian, new_hamiltonian;

    public Form1()
    {
       InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
       OptimizeHamiltonian();
    }

    public void OptimizeHamiltonian()
```

```
{
   double //ax12, ay12, //ax23, ay23, ax34, ay34, epsilon12, epsilon23, epsilon34,
         delta_ax23, delta_ay23, delta_ax34, delta_ay34, delta_ax45, delta_ay45, delta_ax56, delta_ay56,
         delta_ax67, delta_ay67,delta_ax78, delta_ay78,delta_ax89, delta_ay89,delta_ax910, delta_ay910,
         epsilon12, epsilon23, epsilon34,epsilon45, epsilon56, epsilon67,epsilon78, epsilon89, epsilon910;
   //Priskiriu kintamiesiems reiksmes is excel
   //------------------------------------------------------------------------
   string path = @"C:\Users\Tomas\Desktop\3DOF\Pilna_simuliacija.xls";
   int k;
  //--------skaitymui-----
   Microsoft.Office.Interop.Excel.Application excel = new Microsoft.Office.Interop.Excel.Application();
   Workbook wb = excel.Workbooks.Open(path);
   Worksheet excelSheet = wb.ActiveSheet;
   //--------skaitymui iki cia-----

   //--------irasymui nuo cia-----
   Microsoft.Office.Interop.Excel.Application xlApp = new Microsoft.Office.Interop.Excel.Application();
   Microsoft.Office.Interop.Excel.Workbook xlWorkBook;
   Microsoft.Office.Interop.Excel.Worksheet xlWorkSheet;
   object misValue = System.Reflection.Missing.Value;
   xlWorkBook = xlApp.Workbooks.Add(misValue);
   //--------irasymui iki cia-----

   for (k = 0; k < (200-11); k++)  //nuskaitymo eiluciu sk.
   {

      // ce1x2  1- rato numeris; 2 - matavimo zingsnis
      //   [eilute, stulpelis]
   //pirmas ratas
      ce11 = Convert.ToDouble(excelSheet.Cells[2 + k, 2].Value.ToString()); //enkoderiai
      ce12 = Convert.ToDouble(excelSheet.Cells[3 + k, 2].Value.ToString()) ;
      ce13 = Convert.ToDouble(excelSheet.Cells[4 + k, 2].Value.ToString()) ;
      ce14 = Convert.ToDouble(excelSheet.Cells[5 + k, 2].Value.ToString()) ;
      ce15 = Convert.ToDouble(excelSheet.Cells[6 + k, 2].Value.ToString()) ;
      ce16 = Convert.ToDouble(excelSheet.Cells[7 + k, 2].Value.ToString()) ;
      ce17 = Convert.ToDouble(excelSheet.Cells[8 + k, 2].Value.ToString()) ;
      ce18 = Convert.ToDouble(excelSheet.Cells[9 + k, 2].Value.ToString()) ;
      ce19 = Convert.ToDouble(excelSheet.Cells[10 + k, 2].Value.ToString()) ;
      ce110 = Convert.ToDouble(excelSheet.Cells[11 + k, 2].Value.ToString()) ;

      ce1x1 = ce11 * Math.Cos(psi1);
      ce1y1 = ce11 * Math.Sin(psi1);
      ce1x2 = ce12 * Math.Cos(psi1);
      ce1y2 = ce12 * Math.Sin(psi1);
      ce1x3 = ce13 * Math.Cos(psi1);
      ce1y3 = ce13 * Math.Sin(psi1);
      ce1x4 = ce14 * Math.Cos(psi1);
      ce1y4 = ce14 * Math.Sin(psi1);
      ce1x5 = ce15 * Math.Cos(psi1);
      ce1y5 = ce15 * Math.Sin(psi1);
      ce1x6 = ce16 * Math.Cos(psi1);
      ce1y6 = ce16 * Math.Sin(psi1);
      ce1x7 = ce17 * Math.Cos(psi1);
      ce1y7 = ce17 * Math.Sin(psi1);
      ce1x8 = ce18 * Math.Cos(psi1);
      ce1y8 = ce18 * Math.Sin(psi1);
      ce1x9 = ce19 * Math.Cos(psi1);
      ce1y9 = ce19 * Math.Sin(psi1);
```

```
    ce1x10 = ce110 * Math.Cos(psi1);
    ce1y10 = ce110 * Math.Sin(psi1);

  //antras ratas
    ce21 = Convert.ToDouble(excelSheet.Cells[2 + k, 3].Value.ToString()); //enkoderiai
    ce22 = Convert.ToDouble(excelSheet.Cells[3 + k, 3].Value.ToString()) ;
    ce23 = Convert.ToDouble(excelSheet.Cells[4 + k, 3].Value.ToString()) ;
    ce24 = Convert.ToDouble(excelSheet.Cells[5 + k, 3].Value.ToString()) ;
    ce25 = Convert.ToDouble(excelSheet.Cells[6 + k, 3].Value.ToString()) ;
    ce26 = Convert.ToDouble(excelSheet.Cells[7 + k, 3].Value.ToString()) ;
    ce27 = Convert.ToDouble(excelSheet.Cells[8 + k, 3].Value.ToString()) ;
    ce28 = Convert.ToDouble(excelSheet.Cells[9 + k, 3].Value.ToString());
    ce29 = Convert.ToDouble(excelSheet.Cells[10 + k, 3].Value.ToString()) ;
    ce210 = Convert.ToDouble(excelSheet.Cells[11 + k, 3].Value.ToString()) ;

    ce2x1 = ce21 * Math.Cos(psi2);
    ce2y1 = ce21 * Math.Sin(psi2);
    ce2x2 = ce22 * Math.Cos(psi2);
    ce2y2 = ce22 * Math.Sin(psi2);
    ce2x3 = ce23 * Math.Cos(psi2);
    ce2y3 = ce23 * Math.Sin(psi2);
    ce2x4 = ce24 * Math.Cos(psi2);
    ce2y4 = ce24 * Math.Sin(psi2);
    ce2x5 = ce25 * Math.Cos(psi2);
    ce2y5 = ce25 * Math.Sin(psi2);
    ce2x6 = ce26 * Math.Cos(psi2);
    ce2y6 = ce26 * Math.Sin(psi2);
    ce2x7 = ce27 * Math.Cos(psi2);
    ce2y7 = ce27 * Math.Sin(psi2);
    ce2x8 = ce28 * Math.Cos(psi2);
    ce2y8 = ce28 * Math.Sin(psi2);
    ce2x9 = ce29 * Math.Cos(psi2);
    ce2y9 = ce29 * Math.Sin(psi2);
    ce2x10 = ce210 * Math.Cos(psi2);
    ce2y10 = ce210 * Math.Sin(psi2);

  //trecias ratas
    ce31 = Convert.ToDouble(excelSheet.Cells[2 + k, 4].Value.ToString()); //enkoderiai
    ce32 = Convert.ToDouble(excelSheet.Cells[3 + k, 4].Value.ToString());
    ce33 = Convert.ToDouble(excelSheet.Cells[4 + k, 4].Value.ToString()) ;
    ce34 = Convert.ToDouble(excelSheet.Cells[5 + k, 4].Value.ToString()) ;
    ce35 = Convert.ToDouble(excelSheet.Cells[6 + k, 4].Value.ToString()) ;
    ce36 = Convert.ToDouble(excelSheet.Cells[7 + k, 4].Value.ToString()) ;
    ce37 = Convert.ToDouble(excelSheet.Cells[8 + k, 4].Value.ToString()) ;
    ce38 = Convert.ToDouble(excelSheet.Cells[9 + k, 4].Value.ToString()) ;
    ce39 = Convert.ToDouble(excelSheet.Cells[10 + k,4].Value.ToString()) ;
    ce310 = Convert.ToDouble(excelSheet.Cells[11 + k, 4].Value.ToString()) ;

    ce3x1 = ce31 * Math.Cos(psi3);
    ce3y1 = ce31 * Math.Sin(psi3);
    ce3x2 = ce32 * Math.Cos(psi3);
    ce3y2 = ce32 * Math.Sin(psi3);
    ce3x3 = ce33 * Math.Cos(psi3);
    ce3y3 = ce33 * Math.Sin(psi3);
    ce3x4 = ce34 * Math.Cos(psi3);
    ce3y4 = ce34 * Math.Sin(psi3);
    ce3x5 = ce35 * Math.Cos(psi3);
    ce3y5 = ce35 * Math.Sin(psi3);
```

```
            ce3x6 = ce36 * Math.Cos(psi3);
            ce3y6 = ce36 * Math.Sin(psi3);
            ce3x7 = ce37 * Math.Cos(psi3);
            ce3y7 = ce37 * Math.Sin(psi3);
            ce3x8 = ce38 * Math.Cos(psi3);
            ce3y8 = ce38 * Math.Sin(psi3);
            ce3x9 = ce39 * Math.Cos(psi3);
            ce3y9 = ce39 * Math.Sin(psi3);
            ce3x10 = ce310 * Math.Cos(psi3);
            ce3y10 = ce310 * Math.Sin(psi3);

        //ketvirtas ratas
            ce41 = Convert.ToDouble(excelSheet.Cells[2 + k, 5].Value.ToString()); //enkoderiai
            ce42 = Convert.ToDouble(excelSheet.Cells[3 + k, 5].Value.ToString()) ;
            ce43 = Convert.ToDouble(excelSheet.Cells[4 + k, 5].Value.ToString());
            ce44 = Convert.ToDouble(excelSheet.Cells[5 + k, 5].Value.ToString()) ;
            ce45 = Convert.ToDouble(excelSheet.Cells[6 + k, 5].Value.ToString()) ;
            ce46 = Convert.ToDouble(excelSheet.Cells[7 + k, 5].Value.ToString()) ;
            ce47 = Convert.ToDouble(excelSheet.Cells[8 + k, 5].Value.ToString()) ;
            ce48 = Convert.ToDouble(excelSheet.Cells[9 + k, 5].Value.ToString()) ;
            ce49 = Convert.ToDouble(excelSheet.Cells[10 + k,5].Value.ToString()) ;
            ce410 = Convert.ToDouble(excelSheet.Cells[11 + k, 5].Value.ToString()) ;

            ce4x1 = ce41 * Math.Cos(psi4);
            ce4y1 = ce41 * Math.Sin(psi4);
            ce4x2 = ce42 * Math.Cos(psi4);
            ce4y2 = ce42 * Math.Sin(psi4);
            ce4x3 = ce43 * Math.Cos(psi4);
            ce4y3 = ce43 * Math.Sin(psi4);
            ce4x4 = ce44 * Math.Cos(psi4);
            ce4y4 = ce44 * Math.Sin(psi4);
            ce4x5 = ce45 * Math.Cos(psi4);
            ce4y5 = ce45 * Math.Sin(psi4);
            ce4x6 = ce46 * Math.Cos(psi4);
            ce4y6 = ce46 * Math.Sin(psi4);
            ce4x7 = ce47 * Math.Cos(psi4);
            ce4y7 = ce47 * Math.Sin(psi4);
            ce4x8 = ce48 * Math.Cos(psi4);
            ce4y8 = ce48 * Math.Sin(psi4);
            ce4x9 = ce49 * Math.Cos(psi4);
            ce4y9 = ce49 * Math.Sin(psi4);
            ce4x10 = ce410 * Math.Cos(psi4);
            ce4y10 = ce410 * Math.Sin(psi4);

        //pagreiciai cax
            cax12 =
               (Convert.ToDouble(excelSheet.Cells[3 + k, 6].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[2 +
k, 6].Value.ToString()))/2; //pagreicio vidurkis X
            cax23 =
               (Convert.ToDouble(excelSheet.Cells[4 + k, 6].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[3 +
k, 6].Value.ToString()))/2;
            cax34 =
              (Convert.ToDouble(excelSheet.Cells[5 + k, 6].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[4 +
k, 6].Value.ToString()))/2;
            cax45 =
               (Convert.ToDouble(excelSheet.Cells[6 + k, 6].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[5 +
k, 6].Value.ToString())) / 2;
            cax56 =
```

```
            (Convert.ToDouble(excelSheet.Cells[7 + k, 6].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[6 +
k, 6].Value.ToString())) / 2;
            cax67 =
              (Convert.ToDouble(excelSheet.Cells[8 + k, 6].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[7 +
k, 6].Value.ToString())) / 2;
            cax78 =
              (Convert.ToDouble(excelSheet.Cells[9 + k, 6].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[8 +
k, 6].Value.ToString())) / 2;
            cax89 =
             (Convert.ToDouble(excelSheet.Cells[10 + k, 6].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[9 +
k, 6].Value.ToString())) / 2;
            cax910 =
              (Convert.ToDouble(excelSheet.Cells[11 + k, 6].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[10
+ k, 6].Value.ToString())) / 2;

        //pagreiciai cay
            cay12 =
              (Convert.ToDouble(excelSheet.Cells[3 + k, 7].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[2 +
k, 7].Value.ToString()))/2; //pagreicio vidurkis Y
            cay23 =
              (Convert.ToDouble(excelSheet.Cells[4 + k, 7].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[3 +
k, 7].Value.ToString()))/2;
            cay34 =
             (Convert.ToDouble(excelSheet.Cells[5 + k, 7].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[4 +
k, 7].Value.ToString()))/2;
            cay45 =
             (Convert.ToDouble(excelSheet.Cells[6 + k, 7].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[5 +
k, 7].Value.ToString()))/2;
            cay56 =
             (Convert.ToDouble(excelSheet.Cells[7 + k, 7].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[6 +
k, 7].Value.ToString()))/2;
            cay67 =
             (Convert.ToDouble(excelSheet.Cells[8 + k, 7].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[7 +
k, 7].Value.ToString()))/2;
            cay78 =
             (Convert.ToDouble(excelSheet.Cells[9 + k, 7].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[8 +
k, 7].Value.ToString()))/2;
            cay89 =
             (Convert.ToDouble(excelSheet.Cells[10 + k, 7].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[9 +
k, 7].Value.ToString()))/2;
            cay910 =
             (Convert.ToDouble(excelSheet.Cells[11 + k, 7].Value.ToString()) + Convert.ToDouble(excelSheet.Cells[10
+ k, 7].Value.ToString()))/2;

            //pasisukimas
            cTheta1 = Convert.ToDouble(excelSheet.Cells[2 + k, 9].Value.ToString()); //pasisukimo kampas Z
            cTheta2 = Convert.ToDouble(excelSheet.Cells[3 + k, 9].Value.ToString());
            cTheta3 = Convert.ToDouble(excelSheet.Cells[4 + k, 9].Value.ToString());
            cTheta4 = Convert.ToDouble(excelSheet.Cells[5 + k, 9].Value.ToString());
            cTheta5 = Convert.ToDouble(excelSheet.Cells[6 + k, 9].Value.ToString());
            cTheta6 = Convert.ToDouble(excelSheet.Cells[7 + k, 9].Value.ToString());
            cTheta7 = Convert.ToDouble(excelSheet.Cells[8 + k, 9].Value.ToString());
            cTheta8 = Convert.ToDouble(excelSheet.Cells[9 + k, 9].Value.ToString());
            cTheta9 = Convert.ToDouble(excelSheet.Cells[10 + k, 9].Value.ToString());
            cTheta10 = Convert.ToDouble(excelSheet.Cells[11 + k, 9].Value.ToString());
        //-------------------------------------------------------------------------------------------------------------
```

```
int i, j;

for (i = 0; i <= 24; i++)
{
   pagreiciai[i] = 0;
}

old_hamiltonian = CalculateHamiltonian(pagreiciai[0], pagreiciai[1], pagreiciai[2], pagreiciai[3], pagreiciai[4],
pagreiciai[5], pagreiciai[6], pagreiciai[7], pagreiciai[8],
  pagreiciai[9], pagreiciai[10], pagreiciai[11], pagreiciai[12], pagreiciai[13], pagreiciai[14], pagreiciai[15],
pagreiciai[16], pagreiciai[17],
  pagreiciai[18], pagreiciai[19], pagreiciai[20], pagreiciai[21], pagreiciai[22], pagreiciai[23], pagreiciai[24]);
int candidate_j = -1;
short candidate_verte = 0;
int iteracijos = 0;
bool done = false;
while (!done)
{
   iteracijos++;
   done = true;


   for (j = 0; j < 50; j++)
   {
      int ind = j >> 1;
      short prev_value = pagreiciai[ind];
      if ((j & 0x01) == 0x01)
      {
         if (pagreiciai[ind] == -1)
            pagreiciai[ind] = 0;
         else
            pagreiciai[ind] = -1;
      }
      else
      {
         if (pagreiciai[ind] == 1)
            pagreiciai[ind] = 0;
         else
            pagreiciai[ind] = 1;
      }

      short nauja_verte = pagreiciai[ind];
                                    // max pokytis pagreitis XY
      delta_ax23 = max_a_XY * (double)pagreiciai[0];
      delta_ay23 = max_a_XY * (double)pagreiciai[1];
      delta_ax34 = max_a_XY * (double)pagreiciai[2];
      delta_ay34 = max_a_XY * (double)pagreiciai[3];
      delta_ax45 = max_a_XY * (double)pagreiciai[4];
      delta_ay45 = max_a_XY * (double)pagreiciai[5];
      delta_ax56 = max_a_XY * (double)pagreiciai[6];
      delta_ay56 = max_a_XY * (double)pagreiciai[7];
      delta_ax67 = max_a_XY * (double)pagreiciai[8];
      delta_ay67 = max_a_XY * (double)pagreiciai[9];
      delta_ax78 = max_a_XY * (double)pagreiciai[10];
      delta_ay78 = max_a_XY * (double)pagreiciai[11];
      delta_ax89 = max_a_XY * (double)pagreiciai[12];
      delta_ay89 = max_a_XY * (double)pagreiciai[13];
      delta_ax910 = max_a_XY * (double)pagreiciai[14];
```

```
            delta_ay910 = max_a_XY * (double)pagreiciai[15];
            epsilon12 = max_a_Z * (double)pagreiciai[16]; // max pagreitis Z
            epsilon23 = max_a_Z * (double)pagreiciai[17];
            epsilon34 = max_a_Z * (double)pagreiciai[18];
            epsilon45 = max_a_Z * (double)pagreiciai[19];
            epsilon56 = max_a_Z * (double)pagreiciai[20];
            epsilon67 = max_a_Z * (double)pagreiciai[21];
            epsilon78 = max_a_Z * (double)pagreiciai[22];
            epsilon89 = max_a_Z * (double)pagreiciai[23];
            epsilon910 = max_a_Z * (double)pagreiciai[24];

            new_hamiltonian = CalculateHamiltonian( delta_ax23, delta_ay23, delta_ax34, delta_ay34, delta_ax45,
delta_ay45,
               delta_ax56, delta_ay56, delta_ax67, delta_ay67, delta_ax78, delta_ay78, delta_ax89, delta_ay89,
delta_ax910, delta_ay910,
               epsilon12, epsilon23, epsilon34, epsilon45, epsilon56, epsilon67, epsilon78, epsilon89, epsilon910);

            if (new_hamiltonian > old_hamiltonian)
            {
               old_hamiltonian = new_hamiltonian;
               candidate_j = j;
               candidate_verte = nauja_verte;
               done = false;
            }
            pagreiciai[ind] = prev_value;
         }
         if (!done)
            pagreiciai[candidate_j >> 1] = candidate_verte;
      }

      // max pagreitis XY
      delta_ax23 = max_a_XY * (double)pagreiciai[0];
      delta_ay23 = max_a_XY * (double)pagreiciai[1];
      delta_ax34 = max_a_XY * (double)pagreiciai[2];
      delta_ay34 = max_a_XY * (double)pagreiciai[3];
      delta_ax45 = max_a_XY * (double)pagreiciai[4];
      delta_ay45 = max_a_XY * (double)pagreiciai[5];
      delta_ax56 = max_a_XY * (double)pagreiciai[6];
      delta_ay56 = max_a_XY * (double)pagreiciai[7];
      delta_ax67 = max_a_XY * (double)pagreiciai[8];
      delta_ay67 = max_a_XY * (double)pagreiciai[9];
      delta_ax78 = max_a_XY * (double)pagreiciai[10];
      delta_ay78 = max_a_XY * (double)pagreiciai[11];
      delta_ax89 = max_a_XY * (double)pagreiciai[12];
      delta_ay89 = max_a_XY * (double)pagreiciai[13];
      delta_ax910 = max_a_XY * (double)pagreiciai[14];
      delta_ay910 = max_a_XY * (double)pagreiciai[15];
      epsilon12 = max_a_Z * (double)pagreiciai[16]; // max pagreitis Z
      epsilon23 = max_a_Z * (double)pagreiciai[17];
      epsilon34 = max_a_Z * (double)pagreiciai[18];
      epsilon45 = max_a_Z * (double)pagreiciai[19];
      epsilon56 = max_a_Z * (double)pagreiciai[20];
      epsilon67 = max_a_Z * (double)pagreiciai[21];
      epsilon78 = max_a_Z * (double)pagreiciai[22];
      epsilon89 = max_a_Z * (double)pagreiciai[23];
      epsilon910 = max_a_Z * (double)pagreiciai[24];
```

```
        new_hamiltonian = CalculateHamiltonian(delta_ax23, delta_ay23, delta_ax34, delta_ay34, delta_ax45,
delta_ay45,
                delta_ax56, delta_ay56, delta_ax67, delta_ay67, delta_ax78, delta_ay78, delta_ax89, delta_ay89,
delta_ax910, delta_ay910,
                epsilon12, epsilon23, epsilon34, epsilon45, epsilon56, epsilon67, epsilon78, epsilon89, epsilon910);

    //    if (((k >> 3) << 3) == k)
    //     {
            textBox1.Text =
                String.Format("Max_hamiltonian = {0}", old_hamiltonian) + Environment.NewLine +
                String.Format("alpha = {0}", alpha) + Environment.NewLine +
                String.Format("r1 = {0}", r1) + Environment.NewLine +
                String.Format("sx1 = {0}", sx1) + Environment.NewLine +
                String.Format("sy1 = {0}", sy1) + Environment.NewLine +
                String.Format("delta_ax23 = {0}", delta_ax23) + Environment.NewLine +
                String.Format("delta_ay23 = {0}", delta_ay23) + Environment.NewLine +
                String.Format("delta_ax34 = {0}", delta_ax34) + Environment.NewLine +
                String.Format("delta_ay34 = {0}", delta_ay34) + Environment.NewLine +
                String.Format("delta_ax45 = {0}", delta_ax45) + Environment.NewLine +
                String.Format("delta_ay45 = {0}", delta_ay45) + Environment.NewLine +
                String.Format("delta_ax56 = {0}", delta_ax56) + Environment.NewLine +
                String.Format("delta_ay56 = {0}", delta_ay56) + Environment.NewLine +
                String.Format("delta_ax67 = {0}", delta_ax67) + Environment.NewLine +
                String.Format("delta_ay67 = {0}", delta_ay67) + Environment.NewLine +
                String.Format("delta_ax78 = {0}", delta_ax78) + Environment.NewLine +
                String.Format("delta_ay78 = {0}", delta_ay78) + Environment.NewLine +
                String.Format("delta_ax89 = {0}", delta_ax89) + Environment.NewLine +
                String.Format("delta_ay89 = {0}", delta_ay89) + Environment.NewLine +
                String.Format("delta_ax910 = {0}", delta_ax910) + Environment.NewLine +
                String.Format("delta_ay910 = {0}", delta_ay910) + Environment.NewLine +
                String.Format("epsilon12 = {0}", epsilon12) + Environment.NewLine +
                String.Format("epsilon23 = {0}", epsilon23) + Environment.NewLine +
                String.Format("epsilon34 = {0}", epsilon34) + Environment.NewLine +
                String.Format("epsilon45 = {0}", epsilon45) + Environment.NewLine +
                String.Format("epsilon56 = {0}", epsilon56) + Environment.NewLine +
                String.Format("epsilon67 = {0}", epsilon67) + Environment.NewLine +
                String.Format("epsilon78 = {0}", epsilon78) + Environment.NewLine +
                String.Format("epsilon89 = {0}", epsilon89) + Environment.NewLine +
                String.Format("epsilon910 = {0}", epsilon910) + Environment.NewLine +
                String.Format("iteracijos = {0}", iteracijos);
     //   }
//-----------------------------------------------------------------------------------------
//irasom duomenis
//-----------------------------------------------------------------------------------------
        xlWorkSheet = (Microsoft.Office.Interop.Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
      //ce1x2  1- rato numeris; 2 - matavimo zingsnis
      //[eilute, stulpelis]
        xlWorkSheet.Cells[1, 1] = "Laikas";
        xlWorkSheet.Cells[2 + k, 1] = k;
      //pirmas ratas
        xlWorkSheet.Cells[1, 2] = "enc1_po";
        xlWorkSheet.Cells[2 + k, 2] = Math.Sqrt(enc1x1 * enc1x1 + enc1y1 * enc1y1);
        xlWorkSheet.Cells[1, 12] = "enc1x_po";
        xlWorkSheet.Cells[2 + k, 12] = enc1x1;
        xlWorkSheet.Cells[1, 13] = "enc1y_po";
        xlWorkSheet.Cells[2 + k, 13] = enc1y1;

      //antras ratas
```

```
            xlWorkSheet.Cells[1, 3] = "enc2_po";
            xlWorkSheet.Cells[2 + k, 3] = Math.Sqrt(enc2x1 * enc2x1 + enc2y1 * enc2y1);
            xlWorkSheet.Cells[1, 14] = "enc2x_po";
            xlWorkSheet.Cells[2 + k, 14] = enc2x1;
            xlWorkSheet.Cells[1, 15] = "enc2y_po";
            xlWorkSheet.Cells[2 + k, 15] = enc2y1;

        //trecias ratas
            xlWorkSheet.Cells[1, 4] = "enc3_po";
            xlWorkSheet.Cells[2 + k, 4] = Math.Sqrt(enc3x1 * enc3x1 + enc3y1 * enc3y1);
            xlWorkSheet.Cells[1, 16] = "enc3x_po";
            xlWorkSheet.Cells[2 + k, 16] = enc3x1;
            xlWorkSheet.Cells[1, 17] = "enc3y_po";
            xlWorkSheet.Cells[2 + k, 17] = enc3y1;

        //ketvirtas ratas
            xlWorkSheet.Cells[1, 5] = "enc4_po";
            xlWorkSheet.Cells[2 + k, 5] = Math.Sqrt(enc4x1 * enc4x1 + enc4y1 * enc4y1);
            xlWorkSheet.Cells[1, 18] = "enc4x_po";
            xlWorkSheet.Cells[2 + k, 18] = enc4x1;
            xlWorkSheet.Cells[1, 19] = "enc4y_po";
            xlWorkSheet.Cells[2 + k, 19] = enc4y1;

        //pagreiciai X_Y
            xlWorkSheet.Cells[1, 7] = "ax_po";
            xlWorkSheet.Cells[2 + k, 7] = ax12; // delta_ax12;

            xlWorkSheet.Cells[1, 8] = "ay_po";
            xlWorkSheet.Cells[2 + k, 8] = ay12; // delta_ay12;

        // theta
            xlWorkSheet.Cells[1, 10] = "theta_po";
            xlWorkSheet.Cells[2 + k, 10] = Delta_Theta_1;
        }

            //uzdarom excelius
            xlWorkBook.SaveAs(@"C:\Users\Tomas\Desktop\3DOF\Po_filtro.xls",
Microsoft.Office.Interop.Excel.XlFileFormat.xlWorkbookNormal, misValue, misValue, misValue, misValue,
Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlNoChange, misValue, misValue, misValue, misValue,
misValue);
            xlWorkBook.Close(true, misValue, true);
            xlApp.Quit();
            wb.Close();
            MessageBox.Show("Baigta");
            //-------------------------------------------------------------------------------------------------------------------------
        }
        public double CalculateHamiltonian(double delta_ax23, double delta_ay23, double delta_ax34, double delta_ay34,
double delta_ax45, double delta_ay45, double delta_ax56, double delta_ay56, double delta_ax67, double delta_ay67,
double delta_ax78, double delta_ay78, double delta_ax89, double delta_ay89, double delta_ax910, double delta_ay910,
double epsilon12, double epsilon23, double epsilon34, double epsilon45, double epsilon56, double epsilon67, double
epsilon78, double epsilon89, double epsilon910)
        {

            //-----------------------------------------------------------------------------
            //Išraiškos (patikrinta)
            //-----------------------------------------------------------------------------
            alpha = Math.Atan(plotis / ilgis);
```

```
r1 = (-1 / ((-80 / (Sigma_e * Sigma_e)) - (648000 / (aL * aL * pi * pi * Sigma_Theta * Sigma_Theta)))) * (((-2 *
aL * pi * Delta_t * Delta_t * (9 * epsilon12 + 8 * epsilon23 + 7 * epsilon34 + 6 * epsilon45 + 5 * epsilon56 + 4 *
epsilon67 + 3 * epsilon78 + 2 * epsilon89 + epsilon910)) / (45 * Sigma_e * Sigma_e))
    + (360 * (cTheta1 + cTheta2 + cTheta3 + cTheta4 + cTheta5 + cTheta6 + cTheta7 + cTheta8 + cTheta9 +
cTheta10 - (Delta_t * Delta_t * (9 * epsilon12 + 8 * epsilon23 + 7 * epsilon34 + 6 * epsilon45 + 5 * epsilon56 + 4 *
epsilon67 + 3 * epsilon78 + 2 * epsilon89 + epsilon910))) / (aL * pi * Sigma_Theta * Sigma_Theta))
    + (((2 * Math.Cos(alpha) * (ce1y1 + ce1y2 + ce1y3 + ce1y4 + ce1y5 + ce1y6 + ce1y7 + ce1y8 + ce1y9 +
ce1y10 - ce2y1 - ce2y2 - ce2y3 - ce2y4 - ce2y5 - ce2y6 - ce2y7 - ce2y8 - ce2y9 - ce2y10 - ce3y1 - ce3y2 - ce3y3 -
ce3y4 - ce3y5 - ce3y6 - ce3y7 - ce3y8 - ce3y9 - ce3y10
    + ce4y1 + ce4y2 + ce4y3 + ce4y4 + ce4y5 + ce4y6 + ce4y7 + ce4y8 + ce4y9 + ce4y10)) + (2 *
Math.Sin(alpha) * (ce1x1 + ce1x2 + ce1x3 + ce1x4 + ce1x5 + ce1x6 + ce1x7 + ce1x8 + ce1x9 + ce1x10
    + ce2x1 + ce2x2 + ce2x3 + ce2x4 + ce2x5 + ce2x6 + ce2x7 + ce2x8 + ce2x9 + ce2x10 - ce3x1 - ce3x2 - ce3x3
- ce3x4 - ce3x5 - ce3x6 - ce3x7 - ce3x8 - ce3x9 - ce3x10 - ce4x1 - ce4x2 - ce4x3 - ce4x4 - ce4x5 - ce4x6 - ce4x7 -
ce4x8 - ce4x9 - ce4x10))) / (Sigma_e * Sigma_e)));

sx1 = (-1 * Sigma_e * Sigma_e / (-8800 * Delta_t * Delta_t * Delta_t * Delta_t - 240 * Sigma_e * Sigma_e /
(Sigma_a * Sigma_a))) * ((2 * (380 *Delta_t * Delta_t * Delta_t * Delta_t* Sigma_a*Sigma_a
    + 3 * Sigma_e * Sigma_e) / (Sigma_a * Sigma_a * Sigma_e * Sigma_e)) * (-4 * (36 * delta_ax23 + 28 *
delta_ax34 + 21 * delta_ax45 + 15 * delta_ax56 + 10 * delta_ax67 + 6 * delta_ax78+
    3*delta_ax89 +delta_ax910)*Delta_t*Delta_t + Math.Cos(alpha)*(ce1x1 + ce1x10 + ce1x2 + ce1x3 + ce1x4
+ ce1x5 + ce1x6 + ce1x7 + ce1x8 + ce1x9 + ce2x1 + ce2x10 + ce2x2 + ce2x3 + ce2x4 + ce2x5 +
    ce2x6 + ce2x7 + ce2x8 + ce2x9 + ce3x1 + ce3x10 + ce3x2 + ce3x3 + ce3x4 + ce3x5 + ce3x6 + ce3x7 + ce3x8
+ ce3x9 + ce4x1 + ce4x10 + ce4x2 + ce4x3 + ce4x4 + ce4x5 + ce4x6 + ce4x7 + ce4x8 + ce4x9)
    -Math.Sin(alpha)*(ce1y1 + ce1y10 + ce1y2 + ce1y3 + ce1y4 + ce1y5 + ce1y6 + ce1y7 + ce1y8 + ce1y9 +
ce2y1 + ce2y10 + ce2y2 + ce2y3 + ce2y4 + ce2y5 + ce2y6 + ce2y7 + ce2y8 + ce2y9 + ce3y1 + ce3y10 + ce3y2 + ce3y3
+
    ce3y4 + ce3y5 + ce3y6 + ce3y7 + ce3y8 + ce3y9 + ce4y1 + ce4y10 + ce4y2 + ce4y3 + ce4y4 + ce4y5 + ce4y6
+ ce4y7 + ce4y8 + ce4y9))
    - 60 * Delta_t * Delta_t * (2 * (cax12 + cax23 + cax34 + cax45 + cax56 + cax67 + cax78 + cax89 + cax910) /
(Sigma_a * Sigma_a) - 2 * (8 * delta_ax23 + 7 * delta_ax34 + 6 * delta_ax45 + 5 * delta_ax56 + 4 * delta_ax67 + 3 *
delta_ax78 + 2*delta_ax89+delta_ax910) / (Sigma_a * Sigma_a)
    - 8 * Delta_t * Delta_t * Delta_t * Delta_t*(240 * delta_ax23 + 196 * delta_ax34 + 154 * delta_ax45 + 115 *
delta_ax56 + 80 * delta_ax67 + 50 * delta_ax78 + 26 * delta_ax89 + 9*delta_ax910)/(Sigma_e*Sigma_e)
    + 2*Delta_t*Delta_t/(Sigma_e*Sigma_e)*((9 *ce1x10 + ce1x2 + 2 *ce1x3 + 3 *ce1x4 + 4 *ce1x5 + 5 *ce1x6
+ 6 *ce1x7 + 7 *ce1x8 + 8 *ce1x9 + 9* ce2x10 + ce2x2 + 2 *ce2x3 + 3 *ce2x4 + 4 *ce2x5
    + 5 *ce2x6 + 6 *ce2x7 + 7 *ce2x8 + 8 *ce2x9 + 9 *ce3x10 + ce3x2 + 2 *ce3x3 + 3 *ce3x4 + 4 *ce3x5 + 5
*ce3x6 + 6 *ce3x7 + 7 *ce3x8 + 8* ce3x9 + 9 *ce4x10 + ce4x2 + 2*ce4x3 + 3 *ce4x4 + 4 *ce4x5 + 5 *ce4x6 + 6
*ce4x7 + 7 *ce4x8 + 8* ce4x9)*Math.Cos(alpha)
    -(9 *ce1y10 + ce1y2 + 2 *ce1y3 + 3 *ce1y4 + 4 *ce1y5 + 5 *ce1y6 + 6 *ce1y7 + 7 *ce1y8 + 8 *ce1y9 + 9
*ce2y10 + ce2y2 + 2 *ce2y3 + 3 *ce2y4 + 4 *ce2y5 + 5* ce2y6 + 6* ce2y7 + 7* ce2y8 + 8 *ce2y9 + 9 *ce3y10 + ce3y2
+ 2 *ce3y3 +  3* ce3y4 + 4* ce3y5
    + 5* ce3y6 + 6 *ce3y7 + 7* ce3y8 + 8* ce3y9 +  9* ce4y10 + ce4y2 + 2* ce4y3 + 3* ce4y4 + 4* ce4y5 + 5
*ce4y6 + 6* ce4y7 +  7 *ce4y8 + 8 *ce4y9)*Math.Sin(alpha))));

sy1 = 0.025 * (-4 * Delta_t * Delta_t * (36 * delta_ay23 + 28 * delta_ay34 + 21 * delta_ay45 + 15 * delta_ay56
+ 10 * delta_ay67 + 6 * delta_ay78 + 3 * delta_ay89 + delta_ay910) + (ce1y1 + ce1y10 + ce1y2 + ce1y3 + ce1y4 +
ce1y5 + ce1y6 + ce1y7 +
    ce1y8 + ce1y9 + ce2y1 + ce2y10 + ce2y2 + ce2y3 + ce2y4 + ce2y5 + ce2y6 + ce2y7 + ce2y8 + ce2y9 + ce3y1
+ ce3y10 + ce3y2 + ce3y3 + ce3y4 + ce3y5 + ce3y6 + ce3y7 + ce3y8 + ce3y9 + ce4y1 + ce4y10 + ce4y2 + ce4y3 +
ce4y4 + ce4y5 + ce4y6 + ce4y7 + ce4y8 + ce4y9) * Math.Cos(alpha)
    + (ce1x1 + ce1x10 + ce1x2 + ce1x3 + ce1x4 + ce1x5 + ce1x6 + ce1x7 + ce1x8 + ce1x9 + ce2x1 + ce2x10 +
ce2x2 + ce2x3 + ce2x4 + ce2x5 +
    ce2x6 + ce2x7 + ce2x8 + ce2x9 + ce3x1 + ce3x10 + ce3x2 + ce3x3 + ce3x4 + ce3x5 + ce3x6 + ce3x7 + ce3x8
+ ce3x9 + ce4x1 + ce4x10 + ce4x2 + ce4x3 + ce4x4 + ce4x5 + ce4x6 + ce4x7 + ce4x8 + ce4x9) * Math.Sin(alpha))
```

```
+ (3 * Delta_t * Delta_t / (440 * Delta_t * Delta_t * Delta_t * Delta_t * Sigma_a * Sigma_a + 12 * Sigma_e * Sigma_e)) * (4 * Delta_t * Delta_t * Delta_t * Delta_t * Sigma_a * Sigma_a * (156 * delta_ay23 + 140 * delta_ay34 + 119 * delta_ay45 + 95 * delta_ay56 + 70 * delta_ay67 + 46 * delta_ay78 + 25 * delta_ay89 + 9 * delta_ay910)
- 2 * Sigma_e * Sigma_e * (cay12 + cay23 + cay34 + cay45 + cay56 + cay67 + cay78 + cay89 + cay910)
+ 2 * Sigma_e * Sigma_e * (8 * delta_ay23 + 7 * delta_ay34 + 6 * delta_ay45 + 5 * delta_ay56 + 4 * delta_ay67 + 3 * delta_ay78 + 2 * delta_ay89 + delta_ay910)
+ Delta_t * Delta_t * Sigma_a * Sigma_a * ((9 * ce1y1 - 9 * ce1y10 + 7 * ce1y2 + 5 * ce1y3 + 3 * ce1y4 + ce1y5 - ce1y6 - 3 * ce1y7 - 5 * ce1y8 - 7 * ce1y9 + 9 * ce2y1 - 9 * ce2y10 + 7 * ce2y2 + 5 * ce2y3 + 3 * ce2y4 + ce2y5 - ce2y6
- 3 * ce2y7 - 5 * ce2y8 - 7 * ce2y9 + 9 * ce3y1 - 9 * ce3y10 + 7 * ce3y2 + 5 * ce3y3 + 3 * ce3y4 + ce3y5 - ce3y6 - 3 * ce3y7 - 5 * ce3y8 - 7 * ce3y9 + 9 * ce4y1 - 9 * ce4y10 + 7 * ce4y2 + 5 * ce4y3 + 3 * ce4y4 + ce4y5 - ce4y6 - 3 * ce4y7 - 5 * ce4y8 - 7 * ce4y9) * Math.Cos(alpha)
+ (9 * ce1x1 - 9 * ce1x10 + 7 * ce1x2 + 5 * ce1x3 + 3 * ce1x4 + ce1x5 - ce1x6 -
3 * ce1x7 - 5 * ce1x8 - 7 * ce1x9 + 9 * ce2x1 - 9 * ce2x10 + 7 * ce2x2 + 5 * ce2x3 + 3 * ce2x4 + ce2x5 - ce2x6 - 3 * ce2x7 - 5 * ce2x8 - 7 * ce2x9 +
9 * ce3x1 - 9 * ce3x10 + 7 * ce3x2 + 5 * ce3x3 + 3 * ce3x4 + ce3x5 - ce3x6 - 3 * ce3x7 - 5 * ce3x8 - 7 * ce3x9 + 9 * ce4x1 - 9 * ce4x10 + 7 * ce4x2 +
5 * ce4x3 + 3 * ce4x4 + ce4x5 - ce4x6 - 3 * ce4x7 - 5 * ce4x8 - 7 * ce4x9) * Math.Sin(alpha)));

ax12 = (1 / (660 * Delta_t * Delta_t * Delta_t * Delta_t * Sigma_a * Sigma_a + 18 * Sigma_e * Sigma_e))*(-4*Delta_t*Delta_t*Delta_t*Delta_t*Sigma_a*Sigma_a*(156* delta_ax23 + 140*delta_ax34 + 119 * delta_ax45 + 95 * delta_ax56 + 70 * delta_ax67 + 46 * delta_ax78 + 25 * delta_ax89 + 9 * delta_ax910)
+ 2 * Sigma_e * Sigma_e * (cax12 + cax23 + cax34 + cax45 + cax56 + cax67 + cax78 + cax89 + cax910)
- 2 * Sigma_e * Sigma_e * (8 * delta_ax23 + 7 * delta_ax34 + 6 * delta_ax45 + 5 * delta_ax56 + 4 * delta_ax67 + 3 * delta_ax78 + 2 * delta_ax89 + delta_ax910)
+Delta_t*Delta_t*Sigma_a*Sigma_a*((-9 *ce1x1 + 9 *ce1x10 - 7 *ce1x2 - 5 *ce1x3 - 3 *ce1x4 - ce1x5 + ce1x6 + 3* ce1x7 + 5 *ce1x8 + 7 *ce1x9 - 9 *ce2x1 + 9 *ce2x10 - 7* ce2x2 -
5 *ce2x3 - 3* ce2x4 - ce2x5 + ce2x6 + 3* ce2x7 + 5 *ce2x8 + 7 *ce2x9 - 9 *ce3x1 + 9* ce3x10 - 7* ce3x2 - 5 *ce3x3 - 3* ce3x4 - ce3x5 + ce3x6 +
3* ce3x7 + 5* ce3x8 + 7* ce3x9 - 9* ce4x1 + 9 *ce4x10 - 7* ce4x2 - 5* ce4x3 - 3 *ce4x4 - ce4x5 + ce4x6 + 3 *ce4x7 + 5 *ce4x8 + 7 *ce4x9)*Math.Cos(alpha)
+(9 *ce1y1 - 9 *ce1y10 + 7* ce1y2 + 5* ce1y3 + 3* ce1y4 + ce1y5 - ce1y6 - 3 *ce1y7 - 5* ce1y8 - 7 *ce1y9 + 9 *ce2y1 - 9 *ce2y10 + 7* ce2y2 +
5 *ce2y3 + 3 *ce2y4 + ce2y5 - ce2y6 - 3 *ce2y7 - 5 *ce2y8 - 7 *ce2y9 + 9* ce3y1 - 9 *ce3y10 + 7 *ce3y2 + 5* ce3y3 + 3 *ce3y4 + ce3y5 - ce3y6 -
3* ce3y7 - 5 *ce3y8 - 7 *ce3y9 + 9 *ce4y1 - 9* ce4y10 + 7* ce4y2 + 5* ce4y3 + 3 *ce4y4 + ce4y5 - ce4y6 - 3* ce4y7 - 5* ce4y8 - 7* ce4y9)*Math.Sin(alpha)));

ay12 = (-1 / (660 * Delta_t * Delta_t * Delta_t * Delta_t * Sigma_a * Sigma_a + 18 * Sigma_e * Sigma_e)) * (4 * Delta_t * Delta_t * Delta_t * Delta_t * Sigma_a * Sigma_a * (156 * delta_ay23 + 140 * delta_ay34 + 119 * delta_ay45 + 95 * delta_ay56 + 70 * delta_ay67 + 46 * delta_ay78 + 25 * delta_ay89 + 9 * delta_ay910)
- 2 * Sigma_e * Sigma_e * (cay12 + cay23 + cay34 + cay45 + cay56 + cay67 + cay78 + cay89 + cay910)
+ 2 * Sigma_e * Sigma_e * (8 * delta_ay23 + 7 * delta_ay34 + 6 * delta_ay45 + 5 * delta_ay56 + 4 * delta_ay67 + 3 * delta_ay78 + 2 * delta_ay89 + delta_ay910)
+ Delta_t * Delta_t * Sigma_a * Sigma_a * ((9 * ce1y1 - 9 * ce1y10 + 7 * ce1y2 + 5 * ce1y3 + 3 * ce1y4 + ce1y5 - ce1y6 - 3 * ce1y7 - 5 * ce1y8 - 7 * ce1y9 + 9 * ce2y1 - 9 * ce2y10 + 7 * ce2y2 + 5 * ce2y3 + 3 * ce2y4 + ce2y5 - ce2y6
- 3 * ce2y7 - 5 * ce2y8 - 7 * ce2y9 + 9 * ce3y1 - 9 * ce3y10 + 7 * ce3y2 + 5 * ce3y3 + 3 * ce3y4 + ce3y5 - ce3y6 - 3 * ce3y7 - 5 * ce3y8 - 7 * ce3y9 + 9 * ce4y1 - 9 * ce4y10 + 7 * ce4y2 + 5 * ce4y3 + 3 * ce4y4 + ce4y5 - ce4y6 - 3 * ce4y7 - 5 * ce4y8 - 7 * ce4y9) * Math.Cos(alpha)
+ (9 * ce1x1 - 9 * ce1x10 + 7 * ce1x2 + 5 * ce1x3 + 3 * ce1x4 + ce1x5 - ce1x6 -
3 * ce1x7 - 5 * ce1x8 - 7 * ce1x9 + 9 * ce2x1 - 9 * ce2x10 + 7 * ce2x2 + 5 * ce2x3 + 3 * ce2x4 + ce2x5 - ce2x6 - 3 * ce2x7 - 5 * ce2x8 - 7 * ce2x9 +
9 * ce3x1 - 9 * ce3x10 + 7 * ce3x2 + 5 * ce3x3 + 3 * ce3x4 + ce3x5 - ce3x6 - 3 * ce3x7 - 5 * ce3x8 - 7 * ce3x9 + 9 * ce4x1 - 9 * ce4x10 + 7 * ce4x2 +
5 * ce4x3 + 3 * ce4x4 + ce4x5 - ce4x6 - 3 * ce4x7 - 5 * ce4x8 - 7 * ce4x9) * Math.Sin(alpha)));
```

```
        //--------------------------------------------------------------------------
        //Enkoderiai (patikrinta)
        //--------------------------------------------------------------------------
        enc1x1 = sx1;
        enc1y1 = r1 + sy1;
        enc2x1 = sx1 + r1 * Math.Sin(2 * alpha);
        enc2y1 = sy1 - r1 * Math.Cos(2 * alpha);
        enc3x1 = sx1;
        enc3y1 = -r1 + sy1;
        enc4x1 = sx1 - r1 * Math.Sin(2 * alpha);
        enc4y1 = sy1 + r1 * Math.Cos(2 * alpha);
        enc1x2 = sx1 + ax12 * Delta_t * Delta_t;
        enc1y2 = r1 + sy1 + ay12 * Delta_t * Delta_t + aL * pi * Delta_t * Delta_t * epsilon12 / 180;
        enc2x2 = sx1 + ax12 * Delta_t * Delta_t + (r1 + aL * pi * Delta_t * Delta_t * epsilon12 / 180) * Math.Sin(2 *
alpha);
        enc2y2 = sy1 + ay12 * Delta_t * Delta_t - (r1 + aL * pi * Delta_t * Delta_t * epsilon12 / 180) * Math.Cos(2 *
alpha);
        enc3x2 = sx1 + ax12 * Delta_t * Delta_t;
        enc3y2 = -r1 + sy1 + ay12 * Delta_t * Delta_t - aL * pi * Delta_t * Delta_t * epsilon12 / 180;
        enc4x2 = sx1 + ax12 * Delta_t * Delta_t - (r1 + aL * pi * Delta_t * Delta_t * epsilon12 / 180) * Math.Sin(2 *
alpha);
        enc4y2 = sy1 + ay12 * Delta_t * Delta_t + (r1 + aL * pi * Delta_t * Delta_t * epsilon12 / 180) * Math.Cos(2 *
alpha);
        enc1x3 = sx1 + (2*ax12 + delta_ax23) * Delta_t * Delta_t;
        enc1y3 = r1 + sy1 + Delta_t * Delta_t * (360 * ay12 + 180 * delta_ay23 + aL * pi * (epsilon12 + epsilon23)) /
180;
        enc2x3 = sx1 + (2*ax12 + delta_ax23) * Delta_t * Delta_t + Math.Sin(2 * alpha) * (180 * r1 + aL * pi * Delta_t
* Delta_t * (epsilon12 + epsilon23)) / 180;
        enc2y3 = sy1 + (2 * ay12 + delta_ay23) * Delta_t * Delta_t - Math.Cos(2 * alpha) * (180 * r1 + aL * pi * Delta_t
* Delta_t * (epsilon12 + epsilon23)) / 180;
        enc3x3 = sx1 + (2 * ax12 + delta_ax23) * Delta_t * Delta_t;
        enc3y3 = -r1 + sy1 + (360*ay12 + 180 * delta_ay23 - aL * pi * (epsilon12 + epsilon23)) * Delta_t * Delta_t /
180;
        enc4x3 = sx1 + (2*ax12 + delta_ax23) * Delta_t * Delta_t - Math.Sin(2 * alpha) * (180 * r1 + aL * pi * Delta_t
* Delta_t * (epsilon12 + epsilon23)) / 180;
        enc4y3 = sy1 + (2*ay12 + delta_ay23) * Delta_t * Delta_t + Math.Cos(2 * alpha) * (180 * r1 + aL * pi * Delta_t
* Delta_t * (epsilon12 + epsilon23)) / 180;
        enc1x4 = sx1 + (3*ax12 + 2*delta_ax23 + delta_ax34) * Delta_t * Delta_t;
        enc1y4 = r1 + sy1 + (540*ay12 + 360*delta_ay23 + 180*delta_ay34 + aL * pi* (epsilon12 + epsilon23 +
epsilon34)) * Delta_t * Delta_t / 180;
        enc2x4 = sx1 + (3*ax12 + 2*delta_ax23 + delta_ax34) * Delta_t * Delta_t + Math.Sin(2 * alpha) * (180 * r1 +
aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34)) / 180;
        enc2y4 = sy1 + (3*ay12 + 2*delta_ay23 + delta_ay34) * Delta_t * Delta_t - Math.Cos(2 * alpha) * (180 * r1 +
aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34)) / 180;
        enc3x4 = sx1 + (3*ax12 + 2*delta_ax23 + delta_ax34) * Delta_t * Delta_t;
        enc3y4 = sy1 + (3*ay12 + 2*delta_ay23 + delta_ay34) * Delta_t * Delta_t + (-180 * r1 - aL * pi * Delta_t *
Delta_t * (epsilon12 + epsilon23 + epsilon34)) / 180;
        enc4x4 = sx1 + (3*ax12 + 2*delta_ax23 + delta_ax34) * Delta_t * Delta_t - Math.Sin(2 * alpha) * (180 * r1 +
aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34)) / 180;
        enc4y4 = sy1 + (3*ay12 + 2*delta_ay23 + delta_ay34) * Delta_t * Delta_t + Math.Cos(2 * alpha) * (180 * r1 +
aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34)) / 180;
        enc1x5 = sx1 + (4 * ax12 + 3 * delta_ax23 + 2*delta_ax34 + delta_ax45) * Delta_t * Delta_t;
        enc1y5 = sy1 + (4 * ay12 + 3 * delta_ay23 + 2 * delta_ay34 + delta_ay45) * Delta_t * Delta_t + (180 * r1 + aL *
pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45)) / 180;
        enc2x5 = sx1 + (4 * ax12 + 3 * delta_ax23 + 2 * delta_ax34 + delta_ax45) * Delta_t * Delta_t + Math.Sin(2 *
alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45)) / 180;
        enc2y5 = sy1 + (4 * ay12 + 3 * delta_ay23 + 2 * delta_ay34 + delta_ay45) * Delta_t * Delta_t - Math.Cos(2 *
alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45)) / 180;
```

```
        enc3x5 = sx1 + (4 * ax12 + 3 * delta_ax23 + 2 * delta_ax34 + delta_ax45) * Delta_t * Delta_t;
        enc3y5 = sy1 + (4 * ay12 + 3 * delta_ay23 + 2 * delta_ay34 + delta_ay45) * Delta_t * Delta_t + (-180 * r1 - aL
* pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45)) / 180;
        enc4x5 = sx1 + (4 * ax12 + 3 * delta_ax23 + 2 * delta_ax34 + delta_ax45) * Delta_t * Delta_t - Math.Sin(2 *
alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45)) / 180;
        enc4y5 = sy1 + (4 * ay12 + 3 * delta_ay23 + 2 * delta_ay34 + delta_ay45) * Delta_t * Delta_t + Math.Cos(2 *
alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45)) / 180;
        enc1x6 = sx1 + (5 * ax12 + 4 * delta_ax23 + 3 * delta_ax34 + 2*delta_ax45 + delta_ax56) * Delta_t * Delta_t;
        enc1y6 = sy1 + (5 * ay12 + 4 * delta_ay23 + 3 * delta_ay34 + 2 * delta_ay45 + delta_ay56) * Delta_t * Delta_t
+ (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56)) / 180;
        enc2x6 = sx1 + (5 * ax12 + 4 * delta_ax23 + 3 * delta_ax34 + 2 * delta_ax45 + delta_ax56) * Delta_t * Delta_t
+ Math.Sin(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 +
epsilon56)) / 180;
        enc2y6 = sy1 + (5 * ay12 + 4 * delta_ay23 + 3 * delta_ay34 + 2 * delta_ay45 + delta_ay56) * Delta_t * Delta_t -
Math.Cos(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 +
epsilon56)) / 180;
        enc3x6 = sx1 + (5 * ax12 + 4 * delta_ax23 + 3 * delta_ax34 + 2 * delta_ax45 + delta_ax56) * Delta_t * Delta_t;
        enc3y6 = sy1 + (5 * ay12 + 4 * delta_ay23 + 3 * delta_ay34 + 2 * delta_ay45 + delta_ay56) * Delta_t * Delta_t
+ (-180 * r1 - aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56)) / 180;
        enc4x6 = sx1 + (5 * ax12 + 4 * delta_ax23 + 3 * delta_ax34 + 2 * delta_ax45 + delta_ax56) * Delta_t * Delta_t -
Math.Sin(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 +
epsilon56)) / 180;
        enc4y6 = sy1 + (5 * ay12 + 4 * delta_ay23 + 3 * delta_ay34 + 2 * delta_ay45 + delta_ay56) * Delta_t * Delta_t
+ Math.Cos(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 +
epsilon56)) / 180;
        enc1x7 = sx1 + (6 * ax12 + 5 * delta_ax23 + 4 * delta_ax34 + 3 * delta_ax45 + 2 * delta_ax56 + delta_ax67) *
Delta_t * Delta_t;
        enc1y7 = sy1 + (6 * ay12 + 5 * delta_ay23 + 4 * delta_ay34 + 3 * delta_ay45 + 2 * delta_ay56 + delta_ay67) *
Delta_t * Delta_t + (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56
+ epsilon67)) / 180;
        enc2x7 = sx1 + (6 * ax12 + 5 * delta_ax23 + 4 * delta_ax34 + 3 * delta_ax45 + 2 * delta_ax56 + delta_ax67) *
Delta_t * Delta_t + Math.Sin(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 +
epsilon45 + epsilon56 + epsilon67)) / 180;
        enc2y7 = sy1 + (6 * ay12 + 5 * delta_ay23 + 4 * delta_ay34 + 3 * delta_ay45 + 2 * delta_ay56 + delta_ay67) *
Delta_t * Delta_t - Math.Cos(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 +
epsilon45 + epsilon56 + epsilon67)) / 180;
        enc3x7 = sx1 + (6 * ax12 + 5 * delta_ax23 + 4 * delta_ax34 + 3 * delta_ax45 + 2 * delta_ax56 + delta_ax67) *
Delta_t * Delta_t;
        enc3y7 = sy1 + (6 * ay12 + 5 * delta_ay23 + 4 * delta_ay34 + 3 * delta_ay45 + 2 * delta_ay56 + delta_ay67) *
Delta_t * Delta_t + (-180 * r1 - aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 +
epsilon56 + epsilon67)) / 180;
        enc4x7 = sx1 + (6 * ax12 + 5 * delta_ax23 + 4 * delta_ax34 + 3 * delta_ax45 + 2 * delta_ax56 + delta_ax67) *
Delta_t * Delta_t - Math.Sin(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 +
epsilon45 + epsilon56 + epsilon67)) / 180;
        enc4y7 = sy1 + (6 * ay12 + 5 * delta_ay23 + 4 * delta_ay34 + 3 * delta_ay45 + 2 * delta_ay56 + delta_ay67) *
Delta_t * Delta_t + Math.Cos(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34
+ epsilon45 + epsilon56 + epsilon67)) / 180;
        enc1x8 = sx1 + (7 * ax12 + 6 * delta_ax23 + 5 * delta_ax34 + 4 * delta_ax45 + 3 * delta_ax56 + 2 * delta_ax67
+ delta_ax78) * Delta_t * Delta_t;
        enc1y8 = sy1 + (7 * ay12 + 6 * delta_ay23 + 5 * delta_ay34 + 4 * delta_ay45 + 3 * delta_ay56 + 2 * delta_ay67
+ delta_ay78) * Delta_t * Delta_t + (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 +
epsilon45 + epsilon56 + epsilon67 + epsilon78)) / 180;
        enc2x8 = sx1 + (7 * ax12 + 6 * delta_ax23 + 5 * delta_ax34 + 4 * delta_ax45 + 3 * delta_ax56 + 2 * delta_ax67
+ delta_ax78) * Delta_t * Delta_t + Math.Sin(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 +
epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78)) / 180;
        enc2y8 = sy1 + (7 * ay12 + 6 * delta_ay23 + 5 * delta_ay34 + 4 * delta_ay45 + 3 * delta_ay56 + 2 * delta_ay67
+ delta_ay78) * Delta_t * Delta_t - Math.Cos(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 +
epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78)) / 180;
```

```
enc3x8 = sx1 + (7 * ax12 + 6 * delta_ax23 + 5 * delta_ax34 + 4 * delta_ax45 + 3 * delta_ax56 + 2 * delta_ax67 + delta_ax78) * Delta_t * Delta_t;
enc3y8 = sy1 + (7 * ay12 + 6 * delta_ay23 + 5 * delta_ay34 + 4 * delta_ay45 + 3 * delta_ay56 + 2 * delta_ay67 + delta_ay78) * Delta_t * Delta_t + (-180 * r1 - aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78)) / 180;
enc4x8 = sx1 + (7 * ax12 + 6 * delta_ax23 + 5 * delta_ax34 + 4 * delta_ax45 + 3 * delta_ax56 + 2 * delta_ax67 + delta_ax78) * Delta_t * Delta_t - Math.Sin(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78)) / 180;
enc4y8 = sy1 + (7 * ay12 + 6 * delta_ay23 + 5 * delta_ay34 + 4 * delta_ay45 + 3 * delta_ay56 + 2 * delta_ay67 + delta_ay78) * Delta_t * Delta_t + Math.Cos(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78)) / 180;
enc1x9 = sx1 + (8 * ax12 + 7 * delta_ax23 + 6 * delta_ax34 + 5 * delta_ax45 + 4 * delta_ax56 + 3 * delta_ax67 + 2 * delta_ax78 + delta_ax89) * Delta_t * Delta_t;
enc1y9 = sy1 + (8 * ay12 + 7 * delta_ay23 + 6 * delta_ay34 + 5 * delta_ay45 + 4 * delta_ay56 + 3 * delta_ay67 + 2 * delta_ay78 + delta_ay89) * Delta_t * Delta_t + (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78 + epsilon89)) / 180;
enc2x9 = sx1 + (8 * ax12 + 7 * delta_ax23 + 6 * delta_ax34 + 5 * delta_ax45 + 4 * delta_ax56 + 3 * delta_ax67 + 2 * delta_ax78 + delta_ax89) * Delta_t * Delta_t + Math.Sin(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78 + epsilon89)) / 180;
enc2y9 = sy1 + (8 * ay12 + 7 * delta_ay23 + 6 * delta_ay34 + 5 * delta_ay45 + 4 * delta_ay56 + 3 * delta_ay67 + 2 * delta_ay78 + delta_ay89) * Delta_t * Delta_t - Math.Cos(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78 + epsilon89)) / 180;
enc3x9 = sx1 + (8 * ax12 + 7 * delta_ax23 + 6 * delta_ax34 + 5 * delta_ax45 + 4 * delta_ax56 + 3 * delta_ax67 + 2 * delta_ax78 + delta_ax89) * Delta_t * Delta_t;
enc3y9 = sy1 + (8 * ay12 + 7 * delta_ay23 + 6 * delta_ay34 + 5 * delta_ay45 + 4 * delta_ay56 + 3 * delta_ay67 + 2 * delta_ay78 + delta_ay89) * Delta_t * Delta_t + (-180 * r1 - aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78 + epsilon89)) / 180;
enc4x9 = sx1 + (8 * ax12 + 7 * delta_ax23 + 6 * delta_ax34 + 5 * delta_ax45 + 4 * delta_ax56 + 3 * delta_ax67 + 2 * delta_ax78 + delta_ax89) * Delta_t * Delta_t - Math.Sin(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78 + epsilon89)) / 180;
enc4y9 = sy1 + (8 * ay12 + 7 * delta_ay23 + 6 * delta_ay34 + 5 * delta_ay45 + 4 * delta_ay56 + 3 * delta_ay67 + 2 * delta_ay78 + delta_ay89) * Delta_t * Delta_t + Math.Cos(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78 + epsilon89)) / 180;
enc1x10 = sx1 + (9 * ax12 + 8 * delta_ax23 + 7 * delta_ax34 + 6 * delta_ax45 + 5 * delta_ax56 + 4 * delta_ax67 + 3 * delta_ax78 + 2 * delta_ax89 + delta_ax910) * Delta_t * Delta_t;
enc1y10 = sy1 + (9 * ay12 + 8 * delta_ay23 + 7 * delta_ay34 + 6 * delta_ay45 + 5 * delta_ay56 + 4 * delta_ay67 + 3 * delta_ay78 + 2 * delta_ay89 + delta_ay910) * Delta_t * Delta_t - (aL/180) *((-180 * r1/aL) - pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78 + epsilon89 + epsilon910));
enc2x10 = sx1 + (9 * ax12 + 8 * delta_ax23 + 7 * delta_ax34 + 6 * delta_ax45 + 5 * delta_ax56 + 4 * delta_ax67 + 3 * delta_ax78 + 2 * delta_ax89 + delta_ax910) * Delta_t * Delta_t + Math.Sin(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78 + epsilon89 + epsilon910)) / 180;
enc2y10 = sy1 + (9 * ay12 + 8 * delta_ay23 + 7 * delta_ay34 + 6 * delta_ay45 + 5 * delta_ay56 + 4 * delta_ay67 + 3 * delta_ay78 + 2 * delta_ay89 + delta_ay910) * Delta_t * Delta_t - Math.Cos(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78 + epsilon89 + epsilon910)) / 180;
enc3x10 = sx1 + (9 * ax12 + 8 * delta_ax23 + 7 * delta_ax34 + 6 * delta_ax45 + 5 * delta_ax56 + 4 * delta_ax67 + 3 * delta_ax78 + 2 * delta_ax89 + delta_ax910) * Delta_t * Delta_t;
enc3y10 = sy1 + (9 * ay12 + 8 * delta_ay23 + 7 * delta_ay34 + 6 * delta_ay45 + 5 * delta_ay56 + 4 * delta_ay67 + 3 * delta_ay78 + 2 * delta_ay89 + delta_ay910) * Delta_t * Delta_t + (aL/180)*((-180 * r1)/aL - pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78 + epsilon89 + epsilon910));
enc4x10 = sx1 + (9 * ax12 + 8 * delta_ax23 + 7 * delta_ax34 + 6 * delta_ax45 + 5 * delta_ax56 + 4 * delta_ax67 + 3 * delta_ax78 + 2 * delta_ax89 + delta_ax910) * Delta_t * Delta_t - Math.Sin(2 * alpha) * (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78 + epsilon89 + epsilon910)) / 180;
enc4y10 = sy1 + (9 * ay12 + 8 * delta_ay23 + 7 * delta_ay34 + 6 * delta_ay45 + 5 * delta_ay56 + 4 * delta_ay67 + 3 * delta_ay78 + 2 * delta_ay89 + delta_ay910) * Delta_t * Delta_t + Math.Cos(2 * alpha) * (180 * r1 + aL * pi *
```

```
Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 + epsilon56 + epsilon67 + epsilon78 + epsilon89 +
epsilon910)) / 180;

        Delta_Theta_1 = 180 * r1 / (aL * pi);
        Delta_Theta_2 = 180 * r1 / (aL * pi) + Delta_t * Delta_t * epsilon12;
        Delta_Theta_3 = (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23)) / (aL * pi);
        Delta_Theta_4 = (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34)) / (aL * pi);
        Delta_Theta_5 = (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45)) / (aL
* pi);
        Delta_Theta_6 = (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 +
epsilon56)) / (aL * pi);
        Delta_Theta_7 = (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 +
epsilon56 + epsilon67)) / (aL * pi);
        Delta_Theta_8 = (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 +
epsilon56 + epsilon67 + epsilon78)) / (aL * pi);
        Delta_Theta_9 = (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 +
epsilon56 + epsilon67 + epsilon78 + epsilon89)) / (aL * pi);
        Delta_Theta_10 = (180 * r1 + aL * pi * Delta_t * Delta_t * (epsilon12 + epsilon23 + epsilon34 + epsilon45 +
epsilon56 + epsilon67 + epsilon78 + epsilon89 + epsilon910)) / (aL * pi);
        //-------------------------------------------------------------------------------
        //Hamiltonian (patikrinta)
        //-------------------------------------------------------------------------------

        cenc1x1 = ce1x1 * Math.Cos(alpha) - ce1y1 * Math.Sin(alpha);
        cenc1y1 = ce1y1 * Math.Cos(alpha) + ce1x1 * Math.Sin(alpha);
        cenc1x2 = ce1x2 * Math.Cos(alpha) - ce1y2 * Math.Sin(alpha);
        cenc1y2 = ce1y2 * Math.Cos(alpha) + ce1x2 * Math.Sin(alpha);
        cenc1x3 = ce1x3 * Math.Cos(alpha) - ce1y3 * Math.Sin(alpha);
        cenc1y3 = ce1y3 * Math.Cos(alpha) + ce1x3 * Math.Sin(alpha);
        cenc1x4 = ce1x4 * Math.Cos(alpha) - ce1y4 * Math.Sin(alpha);
        cenc1y4 = ce1y4 * Math.Cos(alpha) + ce1x4 * Math.Sin(alpha);
        cenc1x5 = ce1x5 * Math.Cos(alpha) - ce1y5 * Math.Sin(alpha);
        cenc1y5 = ce1y5 * Math.Cos(alpha) + ce1x5 * Math.Sin(alpha);
        cenc1x6 = ce1x6 * Math.Cos(alpha) - ce1y6 * Math.Sin(alpha);
        cenc1y6 = ce1y6 * Math.Cos(alpha) + ce1x6 * Math.Sin(alpha);
        cenc1x7 = ce1x7 * Math.Cos(alpha) - ce1y7 * Math.Sin(alpha);
        cenc1y7 = ce1y7 * Math.Cos(alpha) + ce1x7 * Math.Sin(alpha);
        cenc1x8 = ce1x8 * Math.Cos(alpha) - ce1y8 * Math.Sin(alpha);
        cenc1y8 = ce1y8 * Math.Cos(alpha) + ce1x8 * Math.Sin(alpha);
        cenc1x9 = ce1x9 * Math.Cos(alpha) - ce1y9 * Math.Sin(alpha);
        cenc1y9 = ce1y9 * Math.Cos(alpha) + ce1x9 * Math.Sin(alpha);
        cenc1x10 = ce1x10 * Math.Cos(alpha) - ce1y10 * Math.Sin(alpha);
        cenc1y10 = ce1y10 * Math.Cos(alpha) + ce1x10 * Math.Sin(alpha);
        cenc2x1 = ce2x1 * Math.Cos(alpha) - ce2y1 * Math.Sin(alpha);
        cenc2y1 = ce2y1 * Math.Cos(alpha) + ce2x1 * Math.Sin(alpha);
        cenc2x2 = ce2x2 * Math.Cos(alpha) - ce2y2 * Math.Sin(alpha);
        cenc2y2 = ce2y2 * Math.Cos(alpha) + ce2x2 * Math.Sin(alpha);
        cenc2x3 = ce2x3 * Math.Cos(alpha) - ce2y3 * Math.Sin(alpha);
        cenc2y3 = ce2y3 * Math.Cos(alpha) + ce2x3 * Math.Sin(alpha);
        cenc2x4 = ce2x4 * Math.Cos(alpha) - ce2y4 * Math.Sin(alpha);
        cenc2y4 = ce2y4 * Math.Cos(alpha) + ce2x4 * Math.Sin(alpha);
        cenc2x5 = ce2x5 * Math.Cos(alpha) - ce2y5 * Math.Sin(alpha);
        cenc2y5 = ce2y5 * Math.Cos(alpha) + ce2x5 * Math.Sin(alpha);
        cenc2x6 = ce2x6 * Math.Cos(alpha) - ce2y6 * Math.Sin(alpha);
        cenc2y6 = ce2y6 * Math.Cos(alpha) + ce2x6 * Math.Sin(alpha);
        cenc2x7 = ce2x7 * Math.Cos(alpha) - ce2y7 * Math.Sin(alpha);
        cenc2y7 = ce2y7 * Math.Cos(alpha) + ce2x7 * Math.Sin(alpha);
        cenc2x8 = ce2x8 * Math.Cos(alpha) - ce2y8 * Math.Sin(alpha);
```

```
cenc2y8 = ce2y8 * Math.Cos(alpha) + ce2x8 * Math.Sin(alpha);
cenc2x9 = ce2x9 * Math.Cos(alpha) - ce2y9 * Math.Sin(alpha);
cenc2y9 = ce2y9 * Math.Cos(alpha) + ce2x9 * Math.Sin(alpha);
cenc2x10 = ce2x10 * Math.Cos(alpha) - ce2y10 * Math.Sin(alpha);
cenc2y10 = ce2y10 * Math.Cos(alpha) + ce2x10 * Math.Sin(alpha);
cenc3x1 = ce3x1 * Math.Cos(alpha) - ce3y1 * Math.Sin(alpha);
cenc3y1 = ce3y1 * Math.Cos(alpha) + ce3x1 * Math.Sin(alpha);
cenc3x2 = ce3x2 * Math.Cos(alpha) - ce3y2 * Math.Sin(alpha);
cenc3y2 = ce3y2 * Math.Cos(alpha) + ce3x2 * Math.Sin(alpha);
cenc3x3 = ce3x3 * Math.Cos(alpha) - ce3y3 * Math.Sin(alpha);
cenc3y3 = ce3y3 * Math.Cos(alpha) + ce3x3 * Math.Sin(alpha);
cenc3x4 = ce3x4 * Math.Cos(alpha) - ce3y4 * Math.Sin(alpha);
cenc3y4 = ce3y4 * Math.Cos(alpha) + ce3x4 * Math.Sin(alpha);
cenc3x5 = ce3x5 * Math.Cos(alpha) - ce3y5 * Math.Sin(alpha);
cenc3y5 = ce3y5 * Math.Cos(alpha) + ce3x5 * Math.Sin(alpha);
cenc3x6 = ce3x6 * Math.Cos(alpha) - ce3y6 * Math.Sin(alpha);
cenc3y6 = ce3y6 * Math.Cos(alpha) + ce3x6 * Math.Sin(alpha);
cenc3x7 = ce3x7 * Math.Cos(alpha) - ce3y7 * Math.Sin(alpha);
cenc3y7 = ce3y7 * Math.Cos(alpha) + ce3x7 * Math.Sin(alpha);
cenc3x8 = ce3x8 * Math.Cos(alpha) - ce3y8 * Math.Sin(alpha);
cenc3y8 = ce3y8 * Math.Cos(alpha) + ce3x8 * Math.Sin(alpha);
cenc3x9 = ce3x9 * Math.Cos(alpha) - ce3y9 * Math.Sin(alpha);
cenc3y9 = ce3y9 * Math.Cos(alpha) + ce3x9 * Math.Sin(alpha);
cenc3x10 = ce3x10 * Math.Cos(alpha) - ce3y10 * Math.Sin(alpha);
cenc3y10 = ce3y10 * Math.Cos(alpha) + ce3x10 * Math.Sin(alpha);
cenc4x1 = ce4x1 * Math.Cos(alpha) - ce4y1 * Math.Sin(alpha);
cenc4y1 = ce4y1 * Math.Cos(alpha) + ce4x1 * Math.Sin(alpha);
cenc4x2 = ce4x2 * Math.Cos(alpha) - ce4y2 * Math.Sin(alpha);
cenc4y2 = ce4y2 * Math.Cos(alpha) + ce4x2 * Math.Sin(alpha);
cenc4x3 = ce4x3 * Math.Cos(alpha) - ce4y3 * Math.Sin(alpha);
cenc4y3 = ce4y3 * Math.Cos(alpha) + ce4x3 * Math.Sin(alpha);
cenc4x4 = ce4x4 * Math.Cos(alpha) - ce4y4 * Math.Sin(alpha);
cenc4y4 = ce4y4 * Math.Cos(alpha) + ce4x4 * Math.Sin(alpha);
cenc4x5 = ce4x5 * Math.Cos(alpha) - ce4y5 * Math.Sin(alpha);
cenc4y5 = ce4y5 * Math.Cos(alpha) + ce4x5 * Math.Sin(alpha);
cenc4x6 = ce4x6 * Math.Cos(alpha) - ce4y6 * Math.Sin(alpha);
cenc4y6 = ce4y6 * Math.Cos(alpha) + ce4x6 * Math.Sin(alpha);
cenc4x7 = ce4x7 * Math.Cos(alpha) - ce4y7 * Math.Sin(alpha);
cenc4y7 = ce4y7 * Math.Cos(alpha) + ce4x7 * Math.Sin(alpha);
cenc4x8 = ce4x8 * Math.Cos(alpha) - ce4y8 * Math.Sin(alpha);
cenc4y8 = ce4y8 * Math.Cos(alpha) + ce4x8 * Math.Sin(alpha);
cenc4x9 = ce4x9 * Math.Cos(alpha) - ce4y9 * Math.Sin(alpha);
cenc4y9 = ce4y9 * Math.Cos(alpha) + ce4x9 * Math.Sin(alpha);
cenc4x10 = ce4x10 * Math.Cos(alpha) - ce4y10 * Math.Sin(alpha);
cenc4y10 = ce4y10 * Math.Cos(alpha) + ce4x10 * Math.Sin(alpha);
//-------------------------------------------------------------------------------
//---patikrintos israiskos
//-------------------------------------------------------------------------------
Hamiltonian =
    -1 * (((ax12 - cax12) * (ax12 - cax12) + (ay12 - cay12) * (ay12 - cay12)) / (Sigma_a * Sigma_a))
    - 1 * (((ax12 - cax23 + delta_ax23) * (ax12 - cax23 + delta_ax23) + (ay12 - cay23 + delta_ay23) * (ay12 -
cay23 + delta_ay23)) / (Sigma_a * Sigma_a))
    - 1 * (((ax12 - cax34 + delta_ax23 + delta_ax34) * (ax12 - cax34 + delta_ax23 + delta_ax34) + (ay12 - cay34
+ delta_ay23 + delta_ay34) * (ay12 - cay34 + delta_ay23 + delta_ay34)) / (Sigma_a * Sigma_a))
    - 1 * (((ax12 - cax45 + delta_ax23 + delta_ax34 + delta_ax45) * (ax12 - cax45 + delta_ax23 + delta_ax34 +
delta_ax45) + (ay12 - cay45 + delta_ay23 + delta_ay34 + delta_ay45) * (ay12 - cay45 + delta_ay23 + delta_ay34 +
delta_ay45)) / (Sigma_a * Sigma_a))
```

- 1 * (((ax12 - cax56 + delta_ax23 + delta_ax34 + delta_ax45 + delta_ax56) * (ax12 - cax56 + delta_ax23 + delta_ax34 + delta_ax45 + delta_ax56) + (ay12 - cay56 + delta_ay23 + delta_ay34 + delta_ay45 + delta_ay56) * (ay12 - cay56 + delta_ay23 + delta_ay34 + delta_ay45 + delta_ay56)) / (Sigma_a * Sigma_a))
- 1 * (((ax12 - cax67 + delta_ax23 + delta_ax34 + delta_ax45 + delta_ax56 + delta_ax67) * (ax12 - cax67 + delta_ax23 + delta_ax34 + delta_ax45 + delta_ax56 + delta_ax67) + (ay12 - cay67 + delta_ay23 + delta_ay34 + delta_ay45 + delta_ay56 + delta_ay67) * (ay12 - cay67 + delta_ay23 + delta_ay34 + delta_ay45 + delta_ay56 + delta_ay67)) / (Sigma_a * Sigma_a))
- 1 * (((ax12 - cax78 + delta_ax23 + delta_ax34 + delta_ax45 + delta_ax56 + delta_ax67 + delta_ax78) * (ax12 - cax78 + delta_ax23 + delta_ax34 + delta_ax45 + delta_ax56 + delta_ax67 + delta_ax78) + (ay12 - cay78 + delta_ay23 + delta_ay34 + delta_ay45 + delta_ay56 + delta_ay67 + delta_ay78) * (ay12 - cay78 + delta_ay23 + delta_ay34 + delta_ay45 + delta_ay56 + delta_ay67 + delta_ay78)) / (Sigma_a * Sigma_a))
- 1 * (((ax12 - cax89 + delta_ax23 + delta_ax34 + delta_ax45 + delta_ax56 + delta_ax67 + delta_ax78 + delta_ax89) * (ax12 - cax89 + delta_ax23 + delta_ax34 + delta_ax45 + delta_ax56 + delta_ax67 + delta_ax78 + delta_ax89) + (ay12 - cay89 + delta_ay23 + delta_ay34 + delta_ay45 + delta_ay56 + delta_ay67 + delta_ay78 + delta_ay89) * (ay12 - cay89 + delta_ay23 + delta_ay34 + delta_ay45 + delta_ay56 + delta_ay67 + delta_ay78 + delta_ay89)) / (Sigma_a * Sigma_a))
- 1 * (((ax12 - cax910 + delta_ax23 + delta_ax34 + delta_ax45 + delta_ax56 + delta_ax67 + delta_ax78 + delta_ax89 + delta_ax910) * (ax12 - cax910 + delta_ax23 + delta_ax34 + delta_ax45 + delta_ax56 + delta_ax67 + delta_ax78 + delta_ax89 + delta_ax910) + (ay12 - cay910 + delta_ay23 + delta_ay34 + delta_ay45 + delta_ay56 + delta_ay67 + delta_ay78 + delta_ay89 + delta_ay910) * (ay12 - cay910 + delta_ay23 + delta_ay34 + delta_ay45 + delta_ay56 + delta_ay67 + delta_ay78 + delta_ay89 + delta_ay910)) / (Sigma_a * Sigma_a))
- ((-cenc1x1 + enc1x1) * (-cenc1x1 + enc1x1) + (-cenc1y1 + enc1y1) * (-cenc1y1 + enc1y1)) / (Sigma_e * Sigma_e) -
((-cenc1x2 + enc1x2) * (-cenc1x2 + enc1x2) + (-cenc1y2 + enc1y2) * (-cenc1y2 + enc1y2)) / (Sigma_e * Sigma_e) -
((-cenc1x3 + enc1x3) * (-cenc1x3 + enc1x3) + (-cenc1y3 + enc1y3) * (-cenc1y3 + enc1y3)) / (Sigma_e * Sigma_e) -
((-cenc1x4 + enc1x4) * (-cenc1x4 + enc1x4) + (-cenc1y4 + enc1y4) * (-cenc1y4 + enc1y4)) / (Sigma_e * Sigma_e) -
((-cenc1x5 + enc1x5) * (-cenc1x5 + enc1x5) + (-cenc1y5 + enc1y5) * (-cenc1y5 + enc1y5)) / (Sigma_e * Sigma_e) -
((-cenc1x6 + enc1x6) * (-cenc1x6 + enc1x6) + (-cenc1y6 + enc1y6) * (-cenc1y6 + enc1y6)) / (Sigma_e * Sigma_e) -
((-cenc1x7 + enc1x7) * (-cenc1x7 + enc1x7) + (-cenc1y7 + enc1y7) * (-cenc1y7 + enc1y7)) / (Sigma_e * Sigma_e) -
((-cenc1x8 + enc1x8) * (-cenc1x8 + enc1x8) + (-cenc1y8 + enc1y8) * (-cenc1y8 + enc1y8)) / (Sigma_e * Sigma_e) -
((-cenc1x9 + enc1x9) * (-cenc1x9 + enc1x9) + (-cenc1y9 + enc1y9) * (-cenc1y9 + enc1y9)) / (Sigma_e * Sigma_e) -
((-cenc1x10 + enc1x10) * (-cenc1x10 + enc1x10) + (-cenc1y10 + enc1y10) * (-cenc1y10 + enc1y10)) / (Sigma_e * Sigma_e) -
((-cenc2x1 + enc2x1) * (-cenc2x1 + enc2x1) + (-cenc2y1 + enc2y1) * (-cenc2y1 + enc2y1)) / (Sigma_e * Sigma_e) -
((-cenc2x2 + enc2x2) * (-cenc2x2 + enc2x2) + (-cenc2y2 + enc2y2) * (-cenc2y2 + enc2y2)) / (Sigma_e * Sigma_e) -
((-cenc2x3 + enc2x3) * (-cenc2x3 + enc2x3) + (-cenc2y3 + enc2y3) * (-cenc2y3 + enc2y3)) / (Sigma_e * Sigma_e) -
((-cenc2x4 + enc2x4) * (-cenc2x4 + enc2x4) + (-cenc2y4 + enc2y4) * (-cenc2y4 + enc2y4)) / (Sigma_e * Sigma_e) -
((-cenc2x5 + enc2x5) * (-cenc2x5 + enc2x5) + (-cenc2y5 + enc2y5) * (-cenc2y5 + enc2y5)) / (Sigma_e * Sigma_e) -
((-cenc2x6 + enc2x6) * (-cenc2x6 + enc2x6) + (-cenc2y6 + enc2y6) * (-cenc2y6 + enc2y6)) / (Sigma_e * Sigma_e) -
((-cenc2x7 + enc2x7) * (-cenc2x7 + enc2x7) + (-cenc2y7 + enc2y7) * (-cenc2y7 + enc2y7)) / (Sigma_e * Sigma_e) -
((-cenc2x8 + enc2x8) * (-cenc2x8 + enc2x8) + (-cenc2y8 + enc2y8) * (-cenc2y8 + enc2y8)) / (Sigma_e * Sigma_e) -

```
            ((-cenc2x9 + enc2x9) * (-cenc2x9 + enc2x9) + (-cenc2y9 + enc2y9) * (-cenc2y9 + enc2y9)) / (Sigma_e *
Sigma_e) -
            ((-cenc2x10 + enc2x10) * (-cenc2x10 + enc2x10) + (-cenc2y10 + enc2y10) * (-cenc2y10 + enc2y10)) /
(Sigma_e * Sigma_e) -
            ((-cenc3x1 + enc3x1) * (-cenc3x1 + enc3x1) + (-cenc3y1 + enc3y1) * (-cenc3y1 + enc3y1)) / (Sigma_e *
Sigma_e) -
            ((-cenc3x2 + enc3x2) * (-cenc3x2 + enc3x2) + (-cenc3y2 + enc3y2) * (-cenc3y2 + enc3y2)) / (Sigma_e *
Sigma_e) -
            ((-cenc3x3 + enc3x3) * (-cenc3x3 + enc3x3) + (-cenc3y3 + enc3y3) * (-cenc3y3 + enc3y3)) / (Sigma_e *
Sigma_e) -
            ((-cenc3x4 + enc3x4) * (-cenc3x4 + enc3x4) + (-cenc3y4 + enc3y4) * (-cenc3y4 + enc3y4)) / (Sigma_e *
Sigma_e) -
            ((-cenc3x5 + enc3x5) * (-cenc3x5 + enc3x5) + (-cenc3y5 + enc3y5) * (-cenc3y5 + enc3y5)) / (Sigma_e *
Sigma_e) -
            ((-cenc3x6 + enc3x6) * (-cenc3x6 + enc3x6) + (-cenc3y6 + enc3y6) * (-cenc3y6 + enc3y6)) / (Sigma_e *
Sigma_e) -
            ((-cenc3x7 + enc3x7) * (-cenc3x7 + enc3x7) + (-cenc3y7 + enc3y7) * (-cenc3y7 + enc3y7)) / (Sigma_e *
Sigma_e) -
            ((-cenc3x8 + enc3x8) * (-cenc3x8 + enc3x8) + (-cenc3y8 + enc3y8) * (-cenc3y8 + enc3y8)) / (Sigma_e *
Sigma_e) -
            ((-cenc3x9 + enc3x9) * (-cenc3x9 + enc3x9) + (-cenc3y9 + enc3y9) * (-cenc3y9 + enc3y9)) / (Sigma_e *
Sigma_e) -
            ((-cenc3x10 + enc3x10) * (-cenc3x10 + enc3x10) + (-cenc3y10 + enc3y10) * (-cenc3y10 + enc3y10)) /
(Sigma_e * Sigma_e) -
            ((-cenc4x1 + enc4x1) * (-cenc4x1 + enc4x1) + (-cenc4y1 + enc4y1) * (-cenc4y1 + enc4y1)) / (Sigma_e *
Sigma_e) -
            ((-cenc4x2 + enc4x2) * (-cenc4x2 + enc4x2) + (-cenc4y2 + enc4y2) * (-cenc4y2 + enc4y2)) / (Sigma_e *
Sigma_e) -
            ((-cenc4x3 + enc4x3) * (-cenc4x3 + enc4x3) + (-cenc4y3 + enc4y3) * (-cenc4y3 + enc4y3)) / (Sigma_e *
Sigma_e) -
            ((-cenc4x4 + enc4x4) * (-cenc4x4 + enc4x4) + (-cenc4y4 + enc4y4) * (-cenc4y4 + enc4y4)) / (Sigma_e *
Sigma_e) -
            ((-cenc4x5 + enc4x5) * (-cenc4x5 + enc4x5) + (-cenc4y5 + enc4y5) * (-cenc4y5 + enc4y5)) / (Sigma_e *
Sigma_e) -
            ((-cenc4x6 + enc4x6) * (-cenc4x6 + enc4x6) + (-cenc4y6 + enc4y6) * (-cenc4y6 + enc4y6)) / (Sigma_e *
Sigma_e) -
            ((-cenc4x7 + enc4x7) * (-cenc4x7 + enc4x7) + (-cenc4y7 + enc4y7) * (-cenc4y7 + enc4y7)) / (Sigma_e *
Sigma_e) -
            ((-cenc4x8 + enc4x8) * (-cenc4x8 + enc4x8) + (-cenc4y8 + enc4y8) * (-cenc4y8 + enc4y8)) / (Sigma_e *
Sigma_e) -
            ((-cenc4x9 + enc4x9) * (-cenc4x9 + enc4x9) + (-cenc4y9 + enc4y9) * (-cenc4y9 + enc4y9)) / (Sigma_e *
Sigma_e) -
            ((-cenc4x10 + enc4x10) * (-cenc4x10 + enc4x10) + (-cenc4y10 + enc4y10) * (-cenc4y10 + enc4y10)) /
(Sigma_e * Sigma_e)
            - ((-cTheta1 + Delta_Theta_1) * (-cTheta1 + Delta_Theta_1) / (Sigma_Theta * Sigma_Theta))
            - ((-cTheta2 + Delta_Theta_2) * (-cTheta2 + Delta_Theta_2) / (Sigma_Theta * Sigma_Theta))
            - ((-cTheta3 + Delta_Theta_3) * (-cTheta3 + Delta_Theta_3) / (Sigma_Theta * Sigma_Theta))
            - ((-cTheta4 + Delta_Theta_4) * (-cTheta4 + Delta_Theta_4) / (Sigma_Theta * Sigma_Theta))
            - ((-cTheta5 + Delta_Theta_5) * (-cTheta5 + Delta_Theta_5) / (Sigma_Theta * Sigma_Theta))
            - ((-cTheta6 + Delta_Theta_6) * (-cTheta6 + Delta_Theta_6) / (Sigma_Theta * Sigma_Theta))
            - ((-cTheta7 + Delta_Theta_7) * (-cTheta7 + Delta_Theta_7) / (Sigma_Theta * Sigma_Theta))
            - ((-cTheta8 + Delta_Theta_8) * (-cTheta8 + Delta_Theta_8) / (Sigma_Theta * Sigma_Theta))
            - ((-cTheta9 + Delta_Theta_9) * (-cTheta9 + Delta_Theta_9) / (Sigma_Theta * Sigma_Theta))
            - ((-cTheta10 + Delta_Theta_10) * (-cTheta10 + Delta_Theta_10) / (Sigma_Theta * Sigma_Theta));
        return Hamiltonian;
    }
  }
}
```