VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Centrinių sekų generatorių algoritmu grįstų roboto judesių gerinimas

Improvement of Central Pattern Generator Based Robot Motion

Magistro baigiamasis darbas

Atliko:	Vaidas Budrys	(parašas)
Darbo vadovas:	asist. dr. Vytautas Valaitis	(parašas)
Recenzentas:	doc. dr. Aistis Raudys	(parašas)

Vilnius 2019

Centrinių sekų generatorių algoritmu grįstų roboto judesių gerinimas

Santrauka

Roboto judesių generavimas – robotikoje itin aktuali problema, kuriai spręsti dažnai pasitelkiamos idėjos iš gamtos. Vienas tokių gamtos ir robotikos bendradarbiavimo pavyzdžių yra centriniai sekų generatoriai – gyvūnuose randami neuroniniai mazgai, gebantys generuoti pasikartojančius signalus be aukštesnio lygio valdymo sistemos.

Šiame darbe centrinių sekų generatoriaus algoritmas pritaikytas šešiakojo roboto judesių generavimui. Atlikta išsami algoritmo analizė ir pasiūlyti alternatyvūs jo pritaikymo būdai įvairių judėjimo charakteristikų išgavimui. Pasiūlytas ir įgyvendintas centrinių sekų generatorių algoritmo pagerinimas pritaikius refleksyvinę sistemą, gebančią reaguoti į robotą supančią aplinką, taip išvengiant kliūčių ir planuojant tolimesnę trajektoriją. Taip pat darbe parinkta optimali roboto kojų konfigūracija, leidžianti išlaikyti didelį roboto stabilumą bei judėjimo greitį.

Darbe nagrinėtos idėjos pritaikytos simuliaciniame šešiakojo roboto modelyje, gebančiame naviguoti nežinomoje aplinkoje išvengiant kliūčių.

Raktiniai žodžiai: centrinis sekų generatorius, judesių generavimas, šešiakojis robotas, Braitenbergo mašina, roboto navigavimas.

Improvement of Central Pattern Generator Based Robot Motion

Abstract

Robot motion generation is an especially relevant problem in robotics, which is often solved by employing ideas from the Nature. One such example of robotics and Nature working together is central pattern generators – neural circuits found inside animals that are responsible for generating rhythmic, repetitive signals without the interference from a higher level control system.

The central pattern generator algorithm was adapted for the motion generation of a hexapod robot in this work. In-depth analysis of the algorithm was performed and alternative forms of its adaptation were proposed for generation of diverse movement characteristics. An improvement over the central pattern generator algorithm was proposed and implemented, which includes a reflexive system that is capable of reacting to the surroundings of the robot, in turn avoiding obstacles and planning its further trajectory. The optimal leg configuration of the robot was also selected in the work, which allows maintaining the optimal ratio between the robot's stability and movement speed.

The theoretical work was applied in a simulated model of the robot, that is capable of navigating through unknown environments and avoiding obstacles.

Keywords: Central Pattern Generator, motion generation, hexapod robot, Braitenberg's vehicle, robot navigation.

TURINYS

ĮVADAS

Judėjimas – vienas labiausiai ištobulintų gyvūnų gebėjimų, padedančių kasdienėse maisto, prieglobsčio, partnerių paieškose, taip pat išvengiant plėšrūnų. Ši gyvūnų savybė buvo milijonus metų tobulinama evoliucijos. Žmonių sukurtose judesį imituojančiose sistemose judėjimo problema yra itin svarbi, tam, kad šios sistemos gebėtų atlikti įvairias užduotis judėdamos aplinkoje. Nagrinėjant šią problemą, atrasta vis daugiau sąsajų tarp robotikos ir biologijos mokslo šakų. Dažniausiai šių sričių bendradarbiavimas naudą atneša robotikai – bandoma atkartoti gyvojoje gamtoje stebimų gyvūnų judėjimo, morfologijos bei valdymo metodus. Dažnas roboto modelis yra įkvėptas gyvūnų morfologijos – keturkojai robotai, robotai-gyvatės, netgi humanoidai. Tačiau pastaruoju metu, smarkiai tobulėjant robotų technologijoms, iš šių mainų vis daugiau naudos gauna ir biologijos sritis – kuriami nauji robotai, padedantys mokslininkams nagrinėti įvairias mokslines hipotezes [Ijs08].

Viena iš biologijos sričių, kuriai gali pasitarnauti robotika – neurologija. Dar 1950 metais, būtent neurofiziologas Viljamas Grėjus sukūrė pirmąjį autonominio roboto modelį [Wal50]. Jis įrodė, kad sudėtinga, tikslinga elgsena gali būti pasiekta keletu tarpusavyje sujungtų į neuronus panašių analoginių elektronikos prietaisų. Tačiau nervinės sistemos negali būti laikomos dalimi atskira nuo kūno ir organizmo elgsenos, nes fizikinės savybės (masė, trintis, tamprumas, laikiniai uždelstumai) turi didelę įtaką neuronų signalų poveikiui. Taip prieita prie išvados, kad norint išnagrinėti smegenų veiklą, būtina atlikti išplėstinę atskirų reikšminių kūno dalių, aplinkos bei neuronų sistemų analizę bei sintezę. Norint suprasti organizmų motoriką reikia perprasti transformacijas tarp nervinių signalų, raumenų ir aplinkos, taip supaprastinant sudėtingą gyvūnų kūnų struktūrą. Šiuo atveju robotai tampa tyrinėjamų gyvūnų biomechaniniais modeliais, tam, kad būtų ištirtos įvairios hipotezės [FIS14].

Palyginus su dabartiniais robotais, gyvūnai dažnai turi žymiai efektyvesnius ir lankstesnius judėjimo valdymo būdus. Kai kurie mokslininkai tiria gyvus organizmus tam, kad sukurtų geresnius robotus, o kiti kuria robotus tam, kad geriau perprastų organizmus [Rab15].

Biologija robotikai suteikia daug įkvėpimo – egzistuoja gyvūnai, galintys keliauti įvairiais paviršiais, kopti per kliūtis ar net vaikščioti aukštyn kojomis. Kadangi tokių robotų, kurie turi atkartoti gyvūnų judesius, valdymas yra itin sudėtingas dėl daugybės laisvės laipsnių ir kitų faktorių, kyla didelių iššūkių kuriant jiems valdymo sistemas [AGK+13]. Judėjimas yra didžiąją roboto energijos dalį suvartojantis veiksmas [CCH16]. Dėl šių priežasčių atsiranda judesių generavimo problema. Judesių generavimo sistemos rūpinasi judėjimo nurodymų planavimu ir vykdymu, tuo pat metu vengiant susidūrimų [Sie16].

Daugelį metų atliekami gyvūnų tyrimai atskleidė, kad jų judesiai generuojami centrinėje nervų sistemoje esančių neuroninių mazgų, vadinamų centriniais sekų generatoriais (angl. *Central Pattern Generator - CPG*). Šie mazgai geba generuoti signalus be papildomo jutiklinio atsako ar kitokios įvesties [Kat16] iš smegenų ar stuburo, t.y. nesant informacijos apie judesio specifiškumą, greitį ir t.t. Dėl paprastos *CPG* modelio struktūros, jis gali generuoti sudėtingas judesių komandas daug greičiau nei optimizaciniai metodai [RMM16]. Egzistuoja keletas *CPG* realizacijos metodų, naudojančių suporintų netiesinių osciliatorių sistemas, gebančius generuoti koordinuotas judesių sekas esant paprastiems valdymo parametrams [CLP08]. Skirtingi pritaikymo būdai leidžia pasiekti judėjimą, panašų į kirmėlės, voro ar net žmogaus [CP03].

Nors *CPG* mazgai geba generuoti signalus be valdymo iš smegenų, toks jų pritaikymas nėra idealus robotų sistemoms, nes neatsižvelgiama į aplinkos veiksnius, kliūtis ir t.t. Šią problemą galima būtų spręsti pritaikant aplinkos atpažinimo sistemas, siunčiančias valdymo signalus *CPG* mazgams, kurie nurodytų judėjimo greitį, kryptį bei judėjimo pobūdį (ėjimas, bėgimas) [Ijs08]. Pritaikius tokią sistemą, robotas galėtų dinamiškai reaguoti į aplinkos pokyčius ir atitinkamai koreguoti judėjimo pobūdį.

Darbo tikslas – pritaikyti centrinių sekų generatorių algoritmą dinaminę sąveika su aplinka turinčiam robotui.

Darbo uždaviniai:

1. Ištirti centrinių sekų generatorių algoritmo pranašumus ir trūkumus, lyginant su kitais judesių generavimo algoritmais.
2. Patobulinti centrinių sekų generatorių algoritmą, jį apjungiant su aukštesnio lygio roboto judesių valdymo sistema.
3. Sukurti roboto modelį, gebantį koordinuoti roboto judesių eigą atsižvelgiant į aplinkos veiksnius, bei dinamiškai prisitaikantį prie netikėtų kliūčių.

1. Biologiškai pagrįstų robotų ir sistemų analizė

Biologinių organizmų pagrindu sukurtų robotai – gan nauja robotikos šaka, kuri siekia išmokti gamtos naudojamus principus ir pritaikyti juos realiose inžinerinėse sistemose; kuriami robotai, įkvėpti biologinių sistemų. Šie robotai dažniausiai naudoja kokią nors judėjimo sistemą (angl. Locomotion System).

Pateiksime keletą gyvūnų judėjimo metodų, pritaikytų roboto judesių generavime.

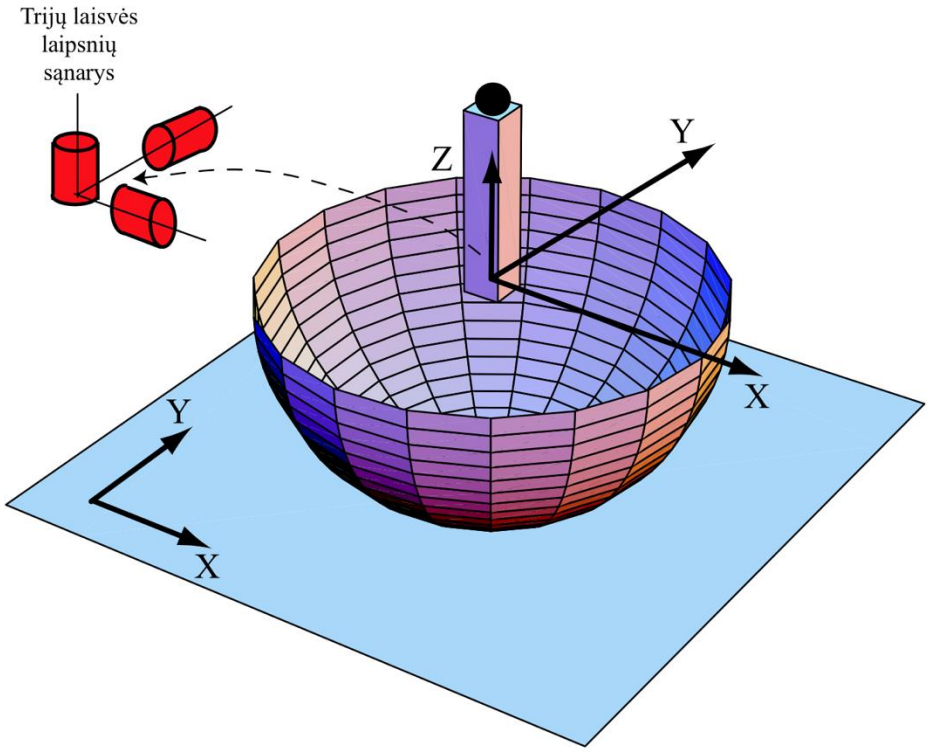
- Judėjimas kojų pagalba. Robotai, naudojantys kojas judėjimui gali turėti vieną, dvi, keturias ar daugiau kojų, priklausomai nuo jų taikymo. Vienas iš pagrindinių kojų pranašumų prieš kitą dažnai robotikoje naudojamą judėjimo būdą – ratus yra tas, kad galima žymiai efektyviau judėti nelygiais paviršiais. Dvikojis, keturkojis, šešiakojis bei aštuonkojis judėjimas yra vieni iš plačiausiai taikomų biorobotikoje (1.1 pav.) [CCB+01].

1.1 pav. *Robugtix T8* – aštuonkojis robotas, sukurtas voro pagrindu

- Bekojis judėjimas. Paviršiai, esantys siaurose, sunkiai prieinamose vietose gali kelti iššūkių daugeliui organizmų ir robotų. Tačiau tokiose vietose puikiai tvarkosi bekojai organizmai tokie kaip gyvatės. Keletas gyvūnų tokių kaip sliekai, gyvatės ar sraigės geba judėti nenaudodami kojų. Robotai, naudojantys ši judėjimo būdą gali būti skirstomi į robotus su aktyviais arba pasyviais ratais, banguojančius robotus naudojančius vertikalias bangas arba horizontalius susitraukinėjimus. Didžioji dalis gyvatiško pavidalo robotų naudoja arba susitraukiančius arba banguojančius judesius ir negali keliauti vertikaliais paviršiais [HC09]. Bekojis judėjimas gali būti atvaizduotas ir paprastu bendru modeliu (1.2 pav.), kuris vieną koją (kuri nekontaktuoja su paviršiais), įstatytą geometriniame sferos centre ant menamos ašies. Koja neturi masės, tačiau ant viršaus turi masyvų tašką aukštesniojoje dalyje. Koja gali suktis apie 3 ašis, t.y. turi tris laisvės laipsnius. 1 lentelėje pateikiamos galimos šio roboto judėjimo konfigūracijos [Bal04]:

1.1 lentelė. Kūno judesių gavimas naudojant bendrąjį bekojo judėjimo modelį

Kūno sukimasis	Kojos sukimosi ašis
Palinkimas	Y
Vinguriavimas	Dviem būdais: 1) X arba Y; tada Z. 2) Y arba X koja iš vertikalios pozicijos; tada Z
Ridenimasis	X
Palinkimas ir vinguriavimas	Y ir Z
Ridenimasis ir vinguriavimas	X ir Z
Ridenimasis, palinkimas ir vinguriavimas	X, Y ir Z



1.2 pav. Bekojo judėjimo bendras modelis, adaptuota iš [BAL04]

- Laipiojimas. Tai ypatingai sudėtinga užduotis, nes dėl lipančiojo padarytų klaidų jis gali prarasti sukibimą ir nukristi. Daugelis robotų gaminami pagal kurią nors konkrečią gyvūno, kurį bandoma atkartoti, savybę. Taip vadinami *Geckobot* (robotai-driežai) dažniausiai naudoja Van der Valso jėgas kurios veikia tik ant lygių paviršių. *Stickybots* [KST+07] (lipnūs robotai) naudoja kryptines lipnias medžiagas kurios taip pat tinka lygiems paviršiams. Kojas turintys laipiojantys robotai dažniausiai sunkiai dorojasi su didelėmis kliūtimis dėl lankstumo trūkumo ir didelės užimamos vietos. Jiems taip pat sunku vaikščioti ant lygių, šiurkščių paviršių ir pereiti tarp horizontalios bei vertikalios padėties.

- Šokinėjimas. Dažnai gyvų organizmų atliekamas veiksmas. Blusos, kengūros, žiogai, skėriai yra vieni geriausiai šokinėjančių gyvūnų. Skėrio įkvėptas miniatiūrinis, 7 gramus sveriantis EPFL sukurtas robotas, naudodamas suspaustą spyruoklę gali iššokti net 138 centimetrus į viršų.

2. Robotų judesių generavimo metodų apžvalga

Robotų judesių generavimui taikomi įvairūs metodai, priklausomai nuo to, kokio tipo aplinkoje jie turės veikti, koks naudojamas judėjimo metodas ir kitų svarbių faktorių. Aptarsime keletą robotikoje naudojamų robotų judesių generavimo metodų.

2.1. Jutiklinis robotų judesių generavimas

Vis daugiau mokslinių tyrimų ir industrinių interesų telkiasi ties sistemų autonomijos laipsnio didinimu. Kuriami robotai, kurie gali veikti įvairiomis sąlygomis, esant ilgiems darbo periodams be žmogaus įsikišimo. Tai gali būti ilgi, pasikartojantys veiksmai, arba darbas kenksmingose aplinkose. Autonominės navigacijos sistemos taikomos pavyzdžiui aptarnaujančiuose, stebėjimo sistemų ar aplinkos tyrinėjimo robotuose, kur mechanizmas gali judėti ir tuo pat metu atlikti jam paskirtą užduotį. Vienas pagrindinių reikalavimų šiems robotams – mobilumas, nes nuo jo priklausys kitų sistemų veiklos efektyvumas. Situacijose, kur reikalingas didelis tikslumas esant statiškai aplinkai galima taikyti jutiklinį roboto valdymo metodą. Jo tikslas – padėti mašiną nugabenti į norimą vietą išvengiant kliūčių.

Tokia sistema paremta suvokimo – veiksmo pagrindu, kartojamu dideliu dažniu. Jutikliai renka informaciją apie aplinką, o roboto procesorius apskaičiuoja darysimus judesius. Judesiai įvykdomi ir ciklas kartojamas. Galutinis rezultatas – pastovi judesių seka, kuri padeda robotui nukeliauti į norimą vietą išvengiant susidūrimų. Kad tokia sistema veiktų, ji turi atitikti tam tikrus reikalavimus:

- Informacijos integracija – visa jutiklių nuskaityta informacija turi būti išsaugoma arba integruota, tam kad būtų suformuota aiškus aplinkos apibrėžimas. Tai būtina norint išvengti kliūčių kurios nematomos dabartiniu momentu (pavyzdžiui kliūtis užstota kito objekto). Taip pat, sistema turi būti paruošta dinamiškiems scenarijams, antraip robotas vengs zonų, kuriose prieš tai būta kliūčių, nors jos ir buvo pašalintos, arba nevengs zonų, kuriose operavimo metu atsirado kliūtys.

- Cikliškos elgsenos vengimas – robotas privalo mokėti atpažinti ciklišką elgseną ir jos vengti. Pavyzdžiu galime laikyti simetrišką aplinką bei kitas įvairias kliūčių konfigūracijas. Joms esant robotas privalo pasiekti galutinį tikslą neužstrigdamas begaliniame cikle.
- Funkcionalumo integracija – visos roboto funkcijos turi būti integruotos architektūroje koordinacijai, netikslumų aptikimui ir jų taisymui. Sistema turi vykdyti suvokimo – veiksmo ciklą dideliu dažniu, nes sistema turi mokėti greitai pasitaisyti pagal aplinkos pokyčius [MM05].

2.2. Judesių generavimas naudojant atvirkštinę kinematiką bei dinamiką

Atvirkštinė kinematika dažniausiai naudojama robotų-humanoidų valdymui. To priežastis yra didelė robotų, kurie imituoja žmonių elgseną, paklausa. Atvirkštinė kinematika yra plačiausiai naudojama technika viso kūno judesių generavimui, deja sunkiai geba susidoroti su dinaminiais apribojimais. Išraiškos trūkumas riboja generuojamų judesių įvairovę. Tačiau šis metodas lengvai suprantamas, lengvai apdorojamas procesoriaus ir nesunkiai pritaikomas. Iš esmės atvirkštinė kinematika yra judesio lygties linearizacija, taip palengvinant skaičiavimus. Taip atsiranda akivaizdžių trūkumų, tačiau palengvėja procesoriaus darbas.

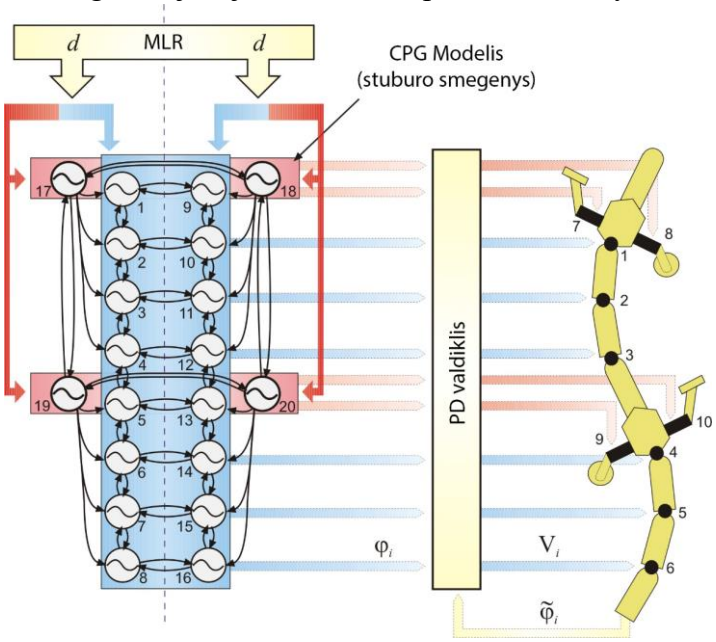
Kai optimali trajektorija atnaujinama realiu laiku naudojant jutiklių rodmenis, tai vadinama prognozuojamuoju modelio valdymu. Nepaisant galimų šio metodo privalumų, jo intensyvumas labai apkrauna sistemą naudodamas daug resursų ir kol kas nėra naudojamas.

Operacinės erdvės atvirkštinė dinamika yra laikoma tobulesne atvirkštinės kinematikos versija, kuri susidoroja su daugeliu jos apribojimų. Praplėsta išraiška leidžia didesnius apribojimus judesių tikslumui, greitesnius judesius ir svarbius momento pokyčius. Šis metodas taip pat naudoja daugiau resursų už atvirkštinę kinematiką, bet gerokai mažiau nei prognozuojamas valdymas. Palyginus du prieš tai minėtus operacinės erdvės atvirkštinė dinamika yra metodas kuris siūlo pakankamai daug galimybių su visai neblogu našumu pagrindinių apribojimų, pavyzdžiui humanoidams, apdorojimui.

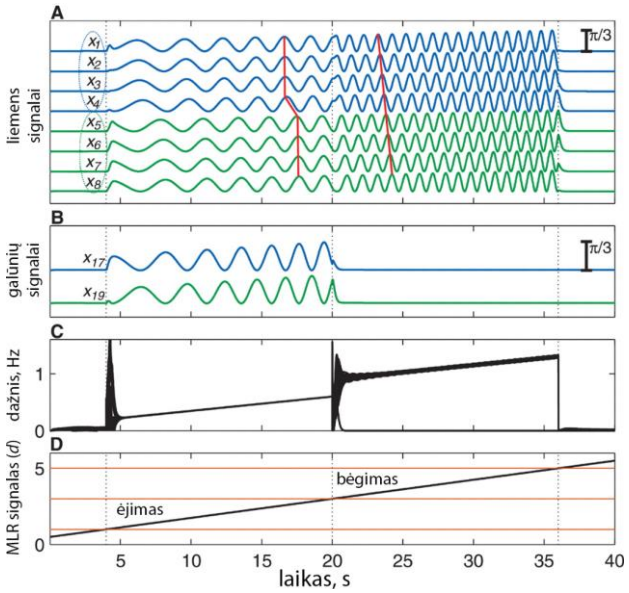
Judesių imitavimo tikslas yra atkartoti jam duotą žmogaus judesių konfigūraciją. Tačiau dėl kinematinių ir dinaminių skirtumų tarp žmonių ir robotų (struktūra, galia, forma, svorio pasiskirstymas) tiesioginis žmogaus sąnarių pozicijų perdavimas robotui tampa neįmanomas. Su šia problema susidoroja tik maži robotai, kurie dažnai būna pagaminti iš lengvų medžiagų [RMS+15]. Tam, kad tokios sistemos veiktų, organizmas ar objektas, kurio veiklą norima imituoti, turi teikti informaciją apie reikiamą trajektoriją. Tam tikslui naudojami jutikliai, kurie prisegami prie kūno ir seka kiekvieną sąnario judesį.

2.3. Roboto judėjimas naudojant centrinį sekų generatorių

Daugelis organizmų turi pasikartojančias arba banguojančias raumenų judesių sekas kurios sukuria ritmiškus judesius tokius kaip ėjimas, kvėpavimas, kasymasis. Konkretūs pavyzdžiai yra vėžio virškinimo sistema, nėgės plaukimas, ar staigūs sparnų judesiai skėrio šuolio metu. Netgi sudėtingesnis judėjimas, toks kaip katės eisena yra tokių banguojančių sekų padarinys. Neuronų grandinės, kurios leidžia vykti tokiems reiškiniams yra vadinamos *centriniais sekų generatoriais* (angl. *Central Pattern Generator - CPG*). Įvairūs eksperimentai, kurių metu tokios neuronų grandinės buvo atskirtos nuo išorinės įtakos parodė, kad jos gali be pagalbos generuoti pasikartojančius ritmiškos veiklos signalus. Tačiau gyvūnai judėdami prisitaiko prie aplinkos sąlygų ir už motoriką atsakingi neuronai kuria įvairius ritmus (ėjimas, bėgimas, plaukimas). Gebėjimas keistis tarp skirtingų motorinių elgsenų priklauso nuo atsako iš aukštesnių receptorių centrinėje nervų sistemoje, todėl *CPG* derėtų laikyti paskirstytos valdymo sistemos dalimi [CC98].



2.1 pav. Salamandros *CPG* modelis, adaptuota iš [FIS14]



2.2 pav. Eisenos pokytis iš ėjimo į plaukimą salamandros *CPG* modelyje, adaptuota iš [FIS14]

Priklausomai nuo tyrimų krypties, sukurta įvairių šį metodą naudojančių robotų – nuo itin detalių biofizinių modelių iki abstrakčių porinių osciliatorių. Osciliatorių modeliai paremti matematiniais porinių netiesinių osciliatorių modeliais. Kaip tokio metodo pritaikymo pavyzdį galime pateikti salamandrą (2.1 pav.). Šis modelis susideda iš dvidešimties fazės osciliatorių su valdoma amplitude. Osciliatoriai gauna signalą d iš Mesanfalinio Lokomotorinio Regiono simuliacijos. Generatoriaus išeiga – reikiamos sąnarių kampų pozicijos φ_i kurios perduodamos į valdiklius. Keičiant d vertę galima keisti salamandros judėjimo pobūdį. Salamandros modelio eisenos perėjimas iš ėjimo į

plaukimą matomas 2.2 pav. (kitos kūno pusės osciliatorių signalai yra priešingos fazės). Numeravimas sutampa su 2.1 pav. [FIS14].

3. Simuliacinių aplinkų apžvalga

Robotų sistemų testavimas realioje aplinkoje yra gan sudėtinga užduotis, nes tyrimams reikia sumodeliuoti robotų prototipus, bei jų testavimo aplinkas. Tai gali sukelti problemų su lėšomis ir reikalauja daug laiko, bei erdvės eksperimentams. Prieš modeliuojant realius robotus, galima simuliuoti juos virtualioje aplinkoje. Taip galima kontroliuoti įvairius aplinkos parametrus, taip pat lengvai atskirti programinės bei techninės įrangos problemas, pavyzdžiui jutiklių duomenys gali būti laikinai pašalinami iš simuliacijos algoritmo, pakeičiant juos įvairiais triukšmo modeliais, taip patikrinant algoritmo patikimumą. [LDB14]. Žemiau pateikti simuliacinių aplinkų privalumai ir trūkumai, lyginant su realiomis sistemomis [SR16].

Privalumai:

- Mažesni roboto gamybos kaštai.
- Įvairių aplinkų simuliavimas be papildomų išlaidų.
- Robotas ir komponentai gali būti ištestuoti prieš pritaikymą.
- Greitesnis prototipavimo procesas.

Trūkumai

- Simuliuojama tik tai, kas suprogramuota – išoriniai ar vidiniai faktoriai praleisti per programavimo stadiją neįtraukiami.
- Realioje aplinkoje robotas gali susidurti su scenarijais, kurių neįmanoma gerai susimuliuoti.

3.1. Specializuota robotų simuliavimo programinė įranga

Egzistuoja daugybė įvairių specializuotų programinės įrangos paketų, sukurtų robotų sistemų simuliacijai. Šios įrangos poreikis atsirado dėl gausėjančio robotų taikymo įvairiose veiklose, reikalaujančiose sąveikos su žmonėmis, pavyzdžiui medicinoje, arba robotams pagalbininkams. Tokioms sistemoms itin svarbu ištestuoti saugumą, patikimumą bei efektyvumą įvairiomis sąlygomis. Galimybė testuoti sistemos patikimumą reguliuojant tik specifinius aplinkos parametrus – vienas iš pagrindinių simuliacinės įrangos privalumų [SG11]. Toliau pateikti populiarūs robotų simuliacijos programinės įrangos paketai taikomi industrijoje.

- *Webots – C/C++, Java, Python, URBI, MATLAB* kalbas palaikantis simuliacinis paketas, leidžiantis sąveiką su kitomis sistemomis naudojant TCP/IP protokolą. Tai

vienas iš dažniausiai naudojamų paketų, turintis daug komponentų ir leidžiantis kurti naujus. Programa palaikoma *Microsoft Windows, Apple macOS* bei *Linux* operacinėse sistemose. Būtent šis paketas pasirinktas atlikti šio darbo modelio simuliacijai.

- *Virtual Robotics Toolkit – LEGO Mindstorms* ir *VEX* robotų simuliavimui skirtas programinis paketas, leidžiantis manipuliuoti modeliais, sukurtais *LEGO Digital Designer* modeliavimo programa. Šis paketas orientuotas į edukaciją bei dažnai naudojamas entuziastų. Pagrindiniai trūkumai – programuoti galima tik *Microsoft Windows* operacinėje sistemoje ir naudoti tik numatytus roboto komponentus.
- *LabVIEW* – sudėtinga programa skirta valdymo, simuliavimo, automatizacijos duomenų analizei, matavimui bei daugeliui kitų sričių. Egzistuoja daugybė realių robotų komponentų atitikmenų, taip pat leidžiama sąsaja su kitomis robotų valdymo sistemomis. Programinis paketas yra uždaro kodo, tačiau esama atviro kodo plėtinių lengvai integracijai su kitomis sistemomis. Plačiai naudojama inžinerijos ir tyrimų srityse.

3.2. Robotų sistemų simuliavimas žaidimų varikliais

Žaidimų varikliai naudingi ne tik žaidimų kūrimui – jie sėkmingai taikomi ir moksliniuose tyrimuose. Jie plačiai naudojami robotikoje, ypač įvairioms simuliacijoms ir vizualizacijoms. Didelę įtaką tam turi tai, jog šiandieniniai žaidimų varikliai turi modulius, kurie leidžia žaidimams apdoroti įvesties duomenis ne tik iš pelės ir klaviatūros, bet ir iš kamerų, mikrofonų, akcelerometrų ir t.t. Taip pat galima suprogramuoti žaidimų valdymą gestais arba balso komandomis [BSF+15].

Žaidimų kūrimo programinė įranga pradėta kurti vaizdo žaidimų kūrimo proceso palengvinimui. Įprastai žaidimų kūrimo programos turi įrankius greitam programavimui, 3D modelių atpažinimui, taip pat fizikos bei grafikos varikliui bei kitus privalumus, reikalingus robotų sistemų simuliavimui.

Žaidimų varikliai pritaikyti didelio našumo programų kūrimui. Taip pat iš daugumos žaidimų tikimasi aukšto realizmo lygio, didelio stabilumo bei realistiškų, dinamiškų aplinkų simuliavimo – bruožų, kurie reikalingi ir simuliuojant robotų sistemas.

Keletas populiariausių komercinių žaidimų variklių rinkoje: *CryEngine, Havok, Source, Unity, Unreal Engine*. Šie varikliai panaudoti daugybės rinkoje esamų žaidimo kūrime. Daugelis šių variklių kūrėjų leidžia nemokamai, arba už mažą kainą naudotis programine įranga, priešingai nei su daugeliu anksčiau pateiktomis specializuotomis robotų simuliavimo programomis, kurių licencijos gali kainuoti itin brangiai.

Esama ir atviro kodo žaidimų variklių, pavyzdžiui, *Godot, id Tech, Crystal Space, „OGRE“.* Šie varikliai dažnai palaikomi tik vartotojų, dėl to dažnai nėra dokumentacijos, trūksta įvairių savybių bei reikalingas geras procesų, vykstančių variklyje, supratimas [HAS11].

3.2.1. Žaidimų variklių apžvalga

Toliau apžvelgti keletas rinkoje dominuojančių žaidimų variklių, aptariant jų pagrindinius bruožus ir privalumus bei trūkumus simuliuojant robotų sistemas.

3.2.1.1. *Unreal Engine*

Epic Games kompanijos kuriamas Unreal Engine žaidimų variklis gan populiarus robotų simuliacijoms, remiantis šiuo varikliu atlikta nemažai mokslinių tyrimų [GM17]. Vienas pagrindinių šio variklio privalumų – kodas rašomas *C++* kalba, kuri yra viena plačiausiai naudojama robotikoje. Tai leidžia lengvesnį kodo perkėlimą iš simuliacijos į realią sistemą. Už šio variklio fiziką atsakinga *Nvidia PhysX* programinė įranga, pasižyminti itin tiksliomis ir realistiškomis fizikos simuliacijomis.

Privalumai:

- Realistiškas fizikos variklis.
- Skriptams naudojama *C++* kalba.
- Atviro kodo programa
- Nemokama licencija.

Trūkumai:

- Prastas atviro kodo platformų (Linux) palaikymas.
- Prasta/trūkstama dokumentacija.
- Gan sudėtinga vartotojo sąsaja.

3.2.1.2. *CryEngine*

Crytek kuriamas žaidimų variklis, palaikantis daugelį populiariausių platformų, taip pat leidžiantis programuoti *C#* bei *Lua,* kuri dažnai naudojama robotikoje. Ši programa naudoja savo fizikos variklį, kuris pasižymi didele sparta, tačiau turi mažesnį realistiškumą, palyginus su kitais rinkoje egzistuojančiais varikliais. Pagrindinis šio variklio privalumas – atviras kodas, leidžiantis modifikuoti variklio elgseną pagal esamus poreikius.

Privalumai:

- Atviras kodas.
- Didelis palaikomų platformų skaičius.
- Palaikoma *Lua* kalba.

Trūkumai:

- Sudėtinga vartotojo sąsaja.
- Netikslus fizikos variklis.
- Mažas papildomų bibliotekų pasirinkimas

3.2.1.3. Unity

Unity – plačiausiai rinkoje naudojamas žaidimų variklis, su didele gausa įvairių bibliotekų aplinkos atpažinimui, realistiškų aplinkų generavimui bei itin realistišku *Nvidia PhysX* fizikos varikliu. Šiam darbui pasirinktas būtent *Unity* žaidimų variklis. Šis variklis jau daugelį metų egzistuoja rinkoje, palaiko įvairius multimedijos formatus ir yra orientuotas į daug platformų, t.y. galima programuoti simuliaciją vienai platformai ir lengvai sukompiliuoti kodą kitai platformai, jo visiškai nepakeičiant. Kodas rašomas C# kalba, kuri, deja, nėra plačiai taikoma robotikoje. Dėl didelės vartotojų bazės, šis variklis turi itin detalią ir plačią dokumentaciją, palyginus su kitais žaidimų varikliais (pvz. *Unreal Engine*). Šis žaidimų variklis taip pat turi platų įvairių papildinių pasirinkimą, tarp jų ir *OpenCV* plėtinį, kuris gali būti naudojamas aplinkos atpažinimui.

Privalumai:

- Nemokama licencija.
- Itin detali dokumentacija.
- Galingas fizikos variklis.
- Paprasta vartotojo sąsaja.

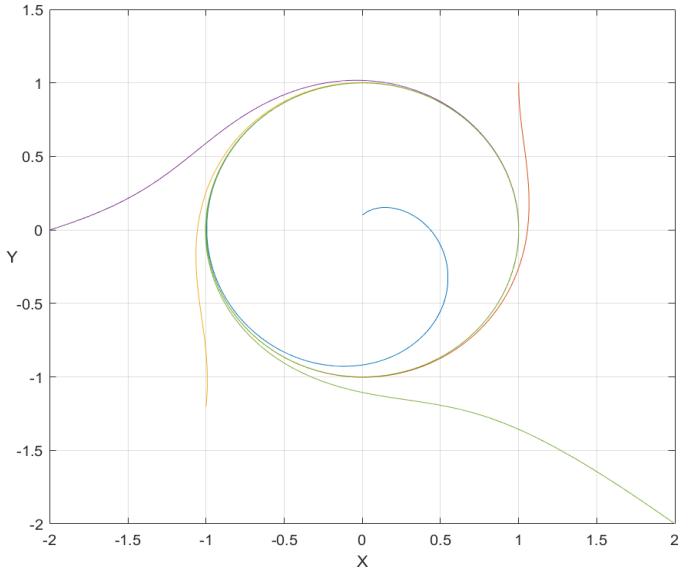
Trūkumai:

- Skriptai rašomi *C#* kalba.
- Didžioji dalis bibliotekų yra mokamos.

4. Judesių generavimo matematinio modelio sudarymas

Darbe naudojamam matematiniam modeliui keliamas reikalavimas – gebėjimas palaikyti sistemai duotas vertes, nepaisant išorinių trikdžių. Norint tai pasiekti, reikia taikyti matematinę sistemą, gebančią savaime grįžti į nustatytą būseną.

Yra daug modelių, skirtų simuliuoti tokioms sistemoms. Visų jų bendras bruožas yra tas, kad jos turi ribinę besikartojančią ciklo funkciją, kuri yra asimptotiškai stabili. Tai reiškia, kad esant bet kokioms pradinėms diferencialinės lygties sąlygoms, sprendinys anksčiau ar



4.1 pav. Izochroninio Hopfo virpesių generatoriaus fazių diagrama. Čia pavaizduota sistemos elgsena esant įvairioms pradinėms sąlygoms. Sistemos brėžiama kreivė visada konverguoja į ribinę funkciją

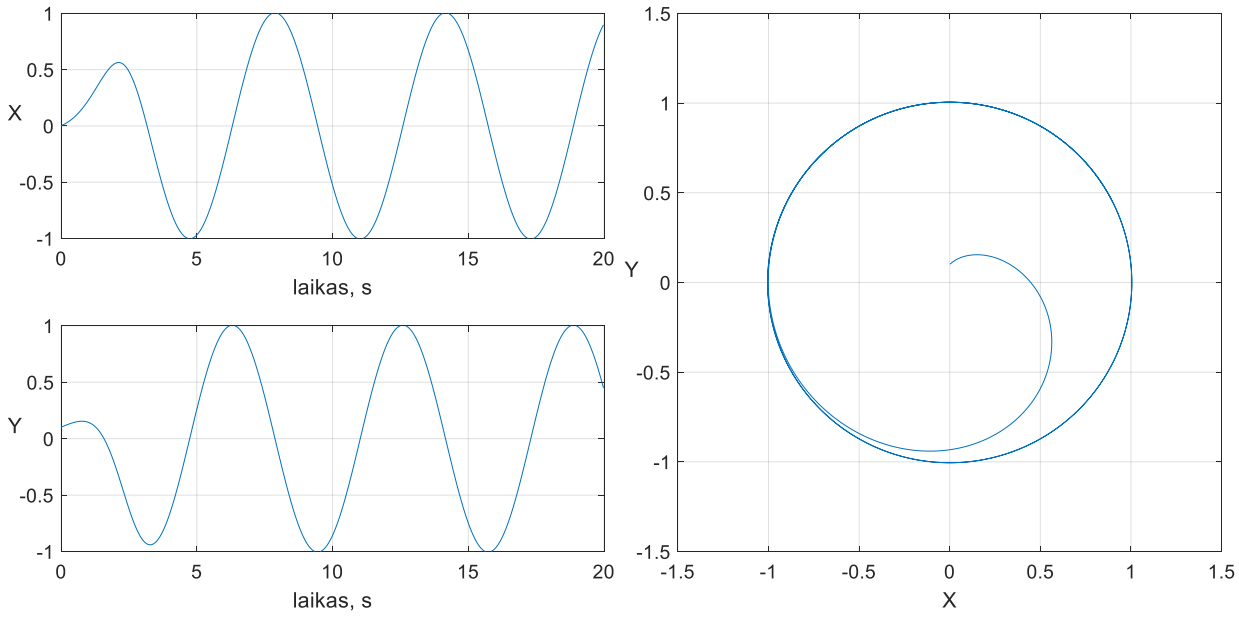
vėliau konverguoja į ribinę funkciją. Šiame darbe *CPG* pagrindą simuliuos paprastas izochroninis osciliatorius (4.1 pav.). Jis aprašomas lygčių sistema:

$$\begin{cases} \dot{\varphi} = \omega \ mod \ 2\pi \\ \dot{r} = r(\mu - r^2) \end{cases}. \tag{4.1}$$

Pateiktoje išraiškoje (4.1) φ atitinka fazę, o r – amplitudę. ω atitinka svyravimo dažnį. Kai $\dot{\varphi}$ lygu ω, fazė keisis ω dažniu. Kai $r^2 > \mu, \dot{r} < 0$ – r mažės. Priešingu atveju, kai $r^2 < \mu, \dot{r} > 0$ r didės. Taigi $r = \sqrt{\mu}$ yra fiksuotas taškas. Akivaizdu, kad šios dvi lygtys yra nesuporintos, taigi fazė ir amplitudė gali būti keičiamos nepriklausomai viena nuo kitos [20], žemiau įrodyta, kad ši savybė itin svarbi šiame darbe naudojamoje sistemoje. Lygčių sistema (4.1) yra pateikta polinėje koordinačių sistemoje, pavertus ją į Dekarto koordinačių sistemą, gaunama (4.2) formulė

$$\begin{cases} \dot{x} = (\mu - x^2 - y^2)x + \omega y \\ \dot{y} = (\mu - x^2 - y^2)y - \omega x \end{cases} \tag{4.2}$$

Kintamuosius $\dot{x}$ ir $\dot{y}$ laikant osciliatorių pozicijų pokyčiais, ir suprojektavus jų brėžiamas kreives viena kitos atžvilgiu gaunama į apskritimą konverguojančios kreivės grafikas (4.2 pav.):

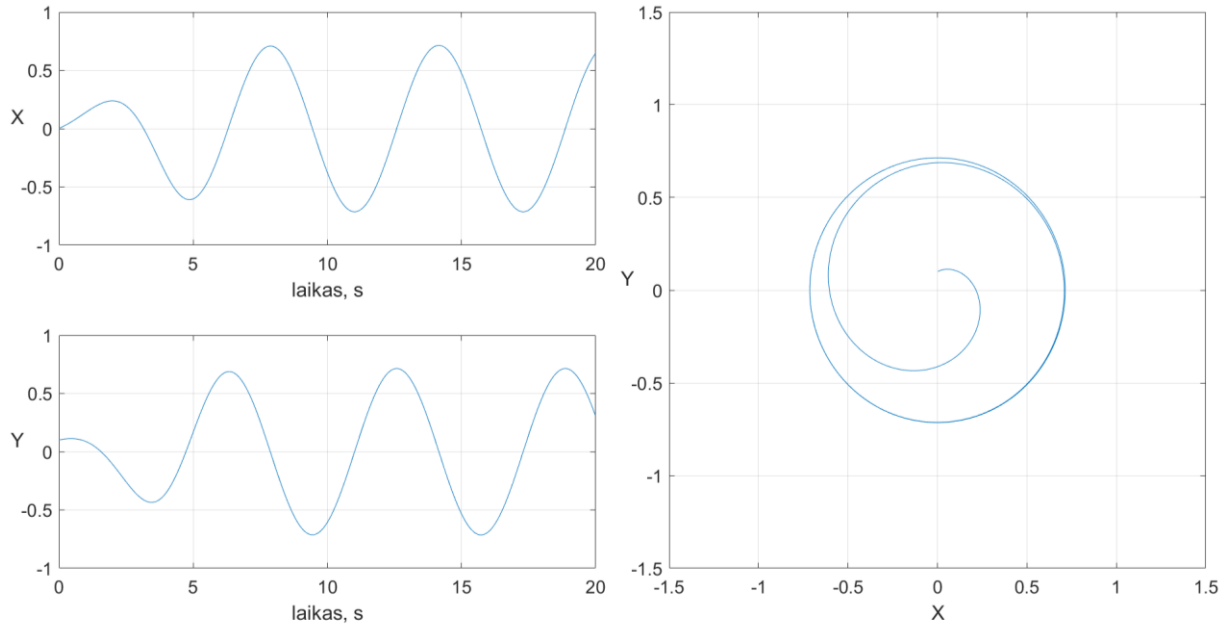


4.2 pav. Hopfo virpesių generatoriaus brėžiamos kreivės. Čia naudojamos osciliatorių parametrų vertės $\mu = \omega = 1$

Aukščiau pateiktame priklausomybių grafike matoma, kad sudėjus pavienių osciliatorių generuojamas kreives, gaunamas apskritimo formos grafikas. Pavienių osciliatorių kreivių formos

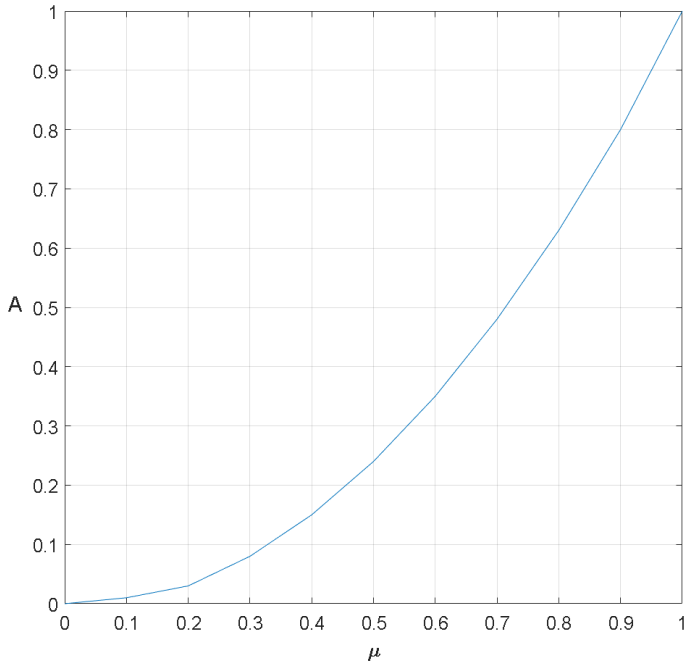
panašios į sinusinio pobūdžio kreives. Kai parametrų μ ir ω vertės lygios 1, maksimalios osciliatorių pasiekiamos vertės yra tarp -1 ir 1.

Dėl nesuporintų osciliatorių galimas lengvas sistemos parametrų – dažnio bei amplitudės individualioms ašims, keitimas. Pritaikius tokią sistemą realaus roboto valdymui, keičiant šiuos parametrus būtų galima keisti, pavyzdžiui, žingsnio aukštį, ilgį bei greitį. Šiame skyrelyje modeliuojamas sistemos judesių generavimo modelis, taikant įvairias amplitudės bei dažnio modifikacijas anksčiau nagrinėtam matematiniam modeliui.



4.3 pav. Osciliatoriaus fazių diagrama, kai $\mu = 0{,}5$

Pirmiausia nagrinėta μ įtaka sistemos charakteristikoms (4.3 pav.). Sumažinus μ vertę iki 0,5 generuojamų virpesių amplitudė sumažėja, tačiau, kaip matoma 4.4 pav., priklausomybė tarp μ ir amplitudės nėra tiesinė – sumažinus μ vertę nuo 1 iki 0,5 sistemos amplitudė išlieka apie 0,7. Sistemos amplitudės priklausomybė nuo μ pateikta 4.4 pav. Ši priklausomybė parodo, kad norint pasiekti tikslias amplitudės vertes, reikia atitinkamai sureguliuoti sistemai siunčiamą signalą.



4.4 pav. Sistemos amplitudės priklausomybė nuo μ

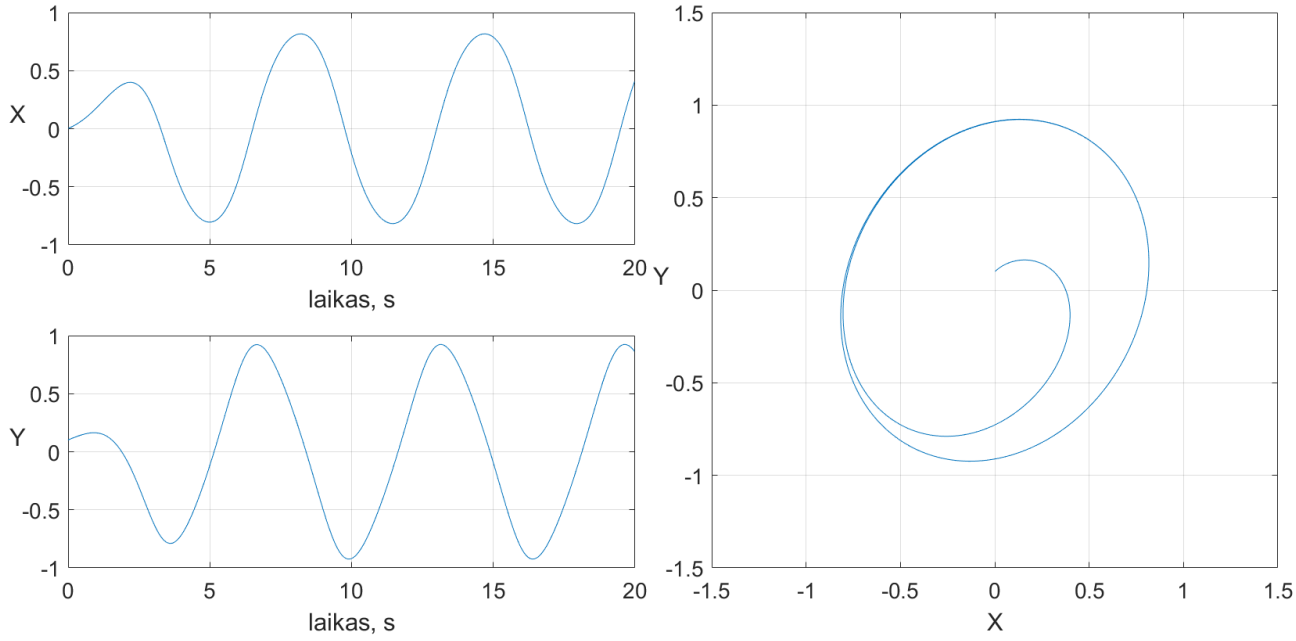
Sistemoje naudojami nesuporinti osciliatoriai leidžia manipuliuoti amplitudės vertes x ir y ašims

individualiai – tai būtina norint, pavyzdžiui, reguliuoti roboto žingsnio ilgį ir aukštį atskirai vienas nuo kito. Jei šį modelį pritaikytume realiai sistemai valdyti, x ir y koordinatės atitiktų valdomos galūnės žingsnio ilgį ir aukštį. Matematinė Hopfo osciliatoriaus išraiška, pritaikius skirtingas μ vertes x ir y ašims, pateikta (4.3) formulėje.

$$\begin{cases} \dot{x} = (\mu_x - x^2 - y^2)x + \omega y \\ \dot{y} = (\mu_y - x^2 - y^2)y + \omega x \end{cases}. \tag{4.3}$$

Toliau pateikti bandymų rezultatai, reguliuojant individualių ašių vertes, gauti remiantis anksčiau atlikta analize.

Duomenys, gauti sistemai nustačius amplitudės vertes, mažesnes už 1 ($\mu_x < 1$) (4.5 pav.) parodo, kad abiejų pavienių osciliatorių fazės pasislinko į dešinę, bei šiek tiek pasikeitė jų brėžiamų kreivių formos – sumažėjus x amplitudei, pastarasis osciliatorius maksimalias vertes pasiekia greičiau.



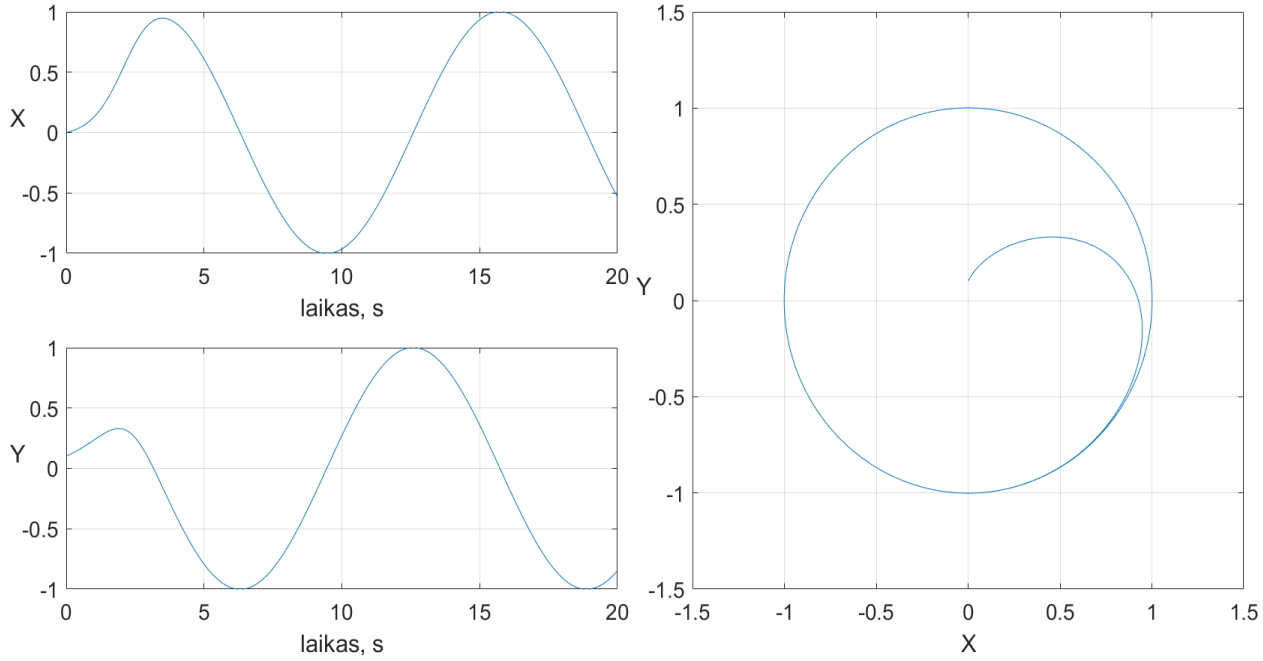
4.5 pav. Osciliatoriaus fazių diagrama, kai $\mu_x = 0{,}5$

Tam, kad sistema išliktų sinchronizuota, y parametro kitimas pasidarė greitesnis. Dėl šių pokyčių osciliatoriaus kreivė deformavosi – 4.5 pav. dešinėje pateiktas grafikas suskirsčius į keturis kvadrantus – ties $x = 0$ ir $y = 0$ ašimis, pirmame ir ketvirtame kvadrantuose ($x < 0, y > 0$; $x > 0, y < 0$) x vertės artėja link nulio kiek greičiau nei y vertės, o antrame ir trečiame kvadrantuose – atvirkščiai. Šis efektas neegzistuoja pavienių x ir y osciliatorių vertes keičiant tolygiai, nes y osciliatoriaus amplitudės mažinimas sukuria priešingą efektą (*pasvirimą* į kairę), taip sugrąžinant ribinei funkcijai apskritimo formą. Kadangi grafikas *pasviręs* į dešinę, toks

signalas netinkamas realiai sistemai – parametrų pokyčio modulis turėtų būti vienodas visuose kvadrantuose tam, kad būtų generuojamas natūralios formos žingsnis.

Kitas svarbus faktorius – keičiant x ašies amplitudės modifikatorių μ_x nukenčia y osciliatoriaus vertės – jos taip pat šiek tiek sumažėja. Akivaizdu, kad minėtos problemos trukdytų stabiliam ir nuspėjamam sistemos valdymui. Toliau pateiktų bandymų su sistemos dažniu rezultatai parodo, kad minėtas problemas gali padėti išspręsti dažnio ω modifikavimas.

Sistemos dažnio reguliavimo bandymai parodo, kad esant dažnio vertei, mažesnei už 1 (4.6 pav.), prailgėja svyravimų periodas. Kaip ir ankstesniuose bandymuose su amplitude, sistemos dažnio priklausomybė nuo dažnio modifikatoriaus ω nėra tiesinė – ω modifikavimo įtaka dažniui yra tiesiogiai proporcinga anksčiau nagrinėtai amplitudės modifikatoriaus μ įtakai sistemos amplitudei (4.4 pav.), tai bus įrodyta tolimesniais bandymais.

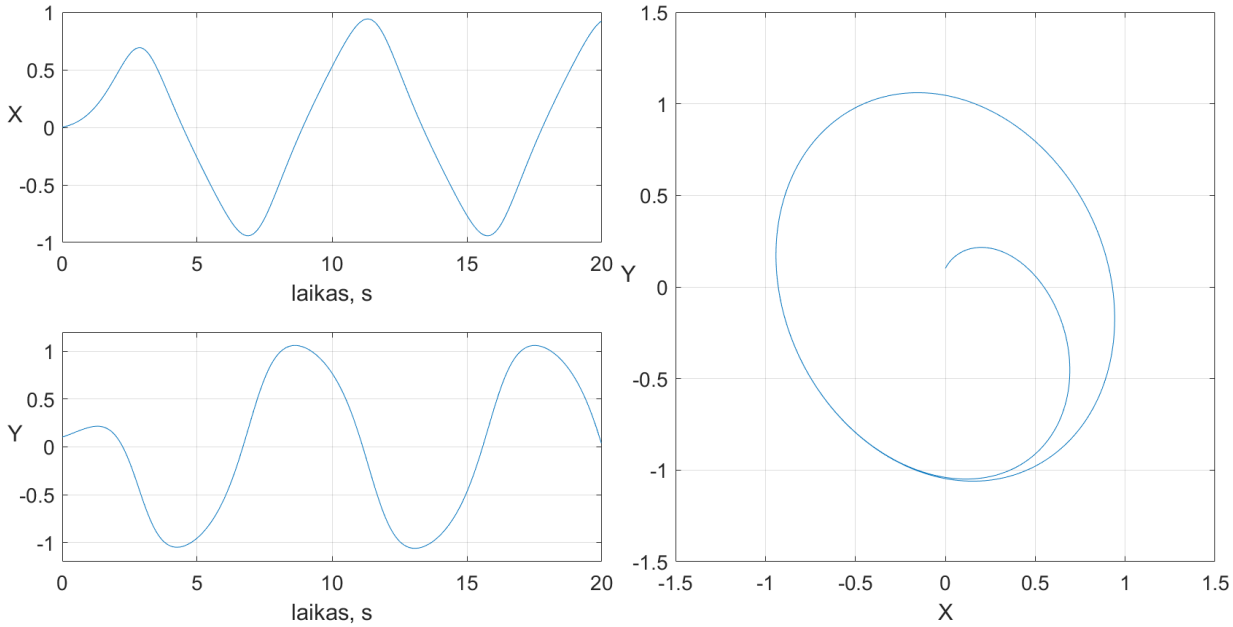


4.6 pav. Osciliatoriaus fazių diagrama, kai $\omega = 0{,}5$

Kaip ir amplitudės atveju, nesuporinti osciliatoriai leidžia modifikuoti jų dažnius individualiai. Tokios modifikacijos leistų greitinti ar lėtinti žingsnio greitį tik norima ašimi (pavyzdžiui, reguliuoti, kaip greitai roboto koja kyla ar leidžiasi nepaveikiant horizontalaus judėjimo greičio). Kadangi dažnis bus modifikuojamas kiekvienai ašiai atskirai, turimai Hopfo osciliatoriaus matematinei išraiškai (4.3) pritaikomi x ir y osciliatorių dažnio modifikatorių kintamieji - ω_x bei ω_y (4.4):

$$\begin{cases} \dot{x} = (\mu_x - x^2 - y^2)x + \omega_x y \\ \dot{y} = (\mu_y - x^2 - y^2)y + \omega_y x \end{cases}. \tag{4.4}$$

Toliau pateikti bandymų, keičiant individualių osciliatorių dažnio modifikatoriaus ω vertes, rezultatai.



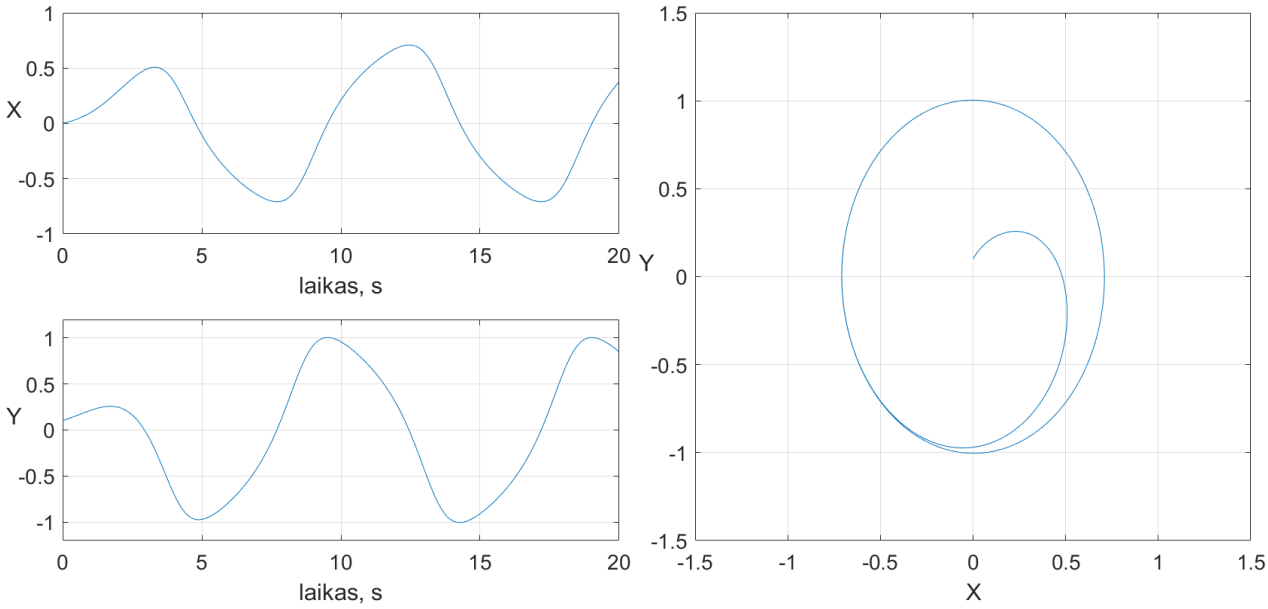
4.7 pav. Osciliatoriaus fazių diagrama, kai $\omega_x = 0{,}5$

Iš grafiko, gauto, pakeitus osciliatoriaus dažnio modifikatorių ω_x (4.7 pav.), matoma, kad, palyginus su anksčiau gautais amplitudės modifikatoriaus μ_x keitimo duomenimis (4.6 pav.), osciliatoriaus parametrai, mažinant dažnio modifikatorių ω_x, pasidaro priešingi – osciliatoriaus brėžiama kreivė *pasvyra* į priešingą (kairę) pusę. Taip pat matomas y amplitudės padidėjimas – priešingai nei mažinant ω_x.

Iš turimų duomenų, galima kelti hipotezę, kad vienu metu keičiant dažnį ir amplitudę, sistema nusistovės taip, kad x ir y parametrų kitimas bus simetriškas $x = 0$ bei $y = 0$ ašių atžvilgiu. Toliau bus nagrinėjami sistemos parametrai, lygiagrečiai keičiant sistemos modifikatorių - ω_x, ω_y ir μ_x, μ_y reikšmes.

Sulyginus dažnio ir amplitudės modifikatorius ω_x ir μ_x (4.8 pav.) matoma, kad osciliatoriai brėžia taisyklingos elipsės trajektoriją - x ašyje amplitudė siekia apie 0,7 – tokią pat vertę, kaip ir keičiant abu amplitudės modifikatorius (μ_x ir μ_y), todėl jos reguliavimui galėtų būti taikoma anksčiau atrasta priklausomybė (4.8 pav.). y ašyje amplitudė išlieka lygi 1 – kintant tik vienos ašies amplitudei sistema pasidaro žymiai lengviau valdoma. Grafikas nėra *pasviręs* ir atitinka natūralaus žingsnio trajektoriją.

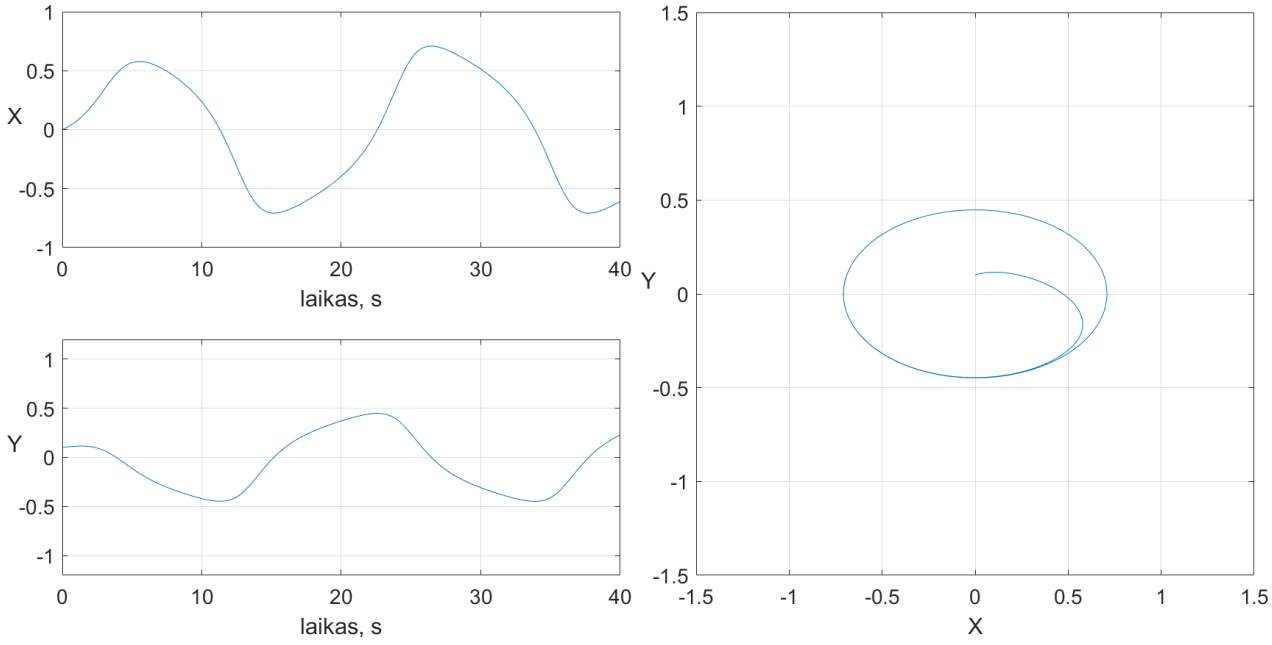
Bandymų metu prieita prie išvados, kad norint generuoti realistiškos žingsnio formos kreivę, užtenka proporcingai keisti kiekvieno iš osciliatorių dažnį ir amplitudę, t.y., jų vertės turi būti vienodos. Taip gauta galutinė išraiška, gauta pritaikius atskirus kintamuosius – x bei y osciliatorių modifikatorius m_x ir m_y (4.5).



4.8 pav. Osciliatoriaus fazių diagrama, kai $\omega_x = \mu_x = 0{,}5$

$$\begin{cases} \dot{x} = (m_x - x^2 - y^2)x + m_x y \\ \dot{y} = (m_y - x^2 - y^2)y + m_y x \end{cases}. \tag{4.5}$$

Pritaikius aukščiau pateiktą išraišką, galima išgauti įvairaus aukščio bei pločio kreives, kurių kitimo charakteristikos yra proporcingos visuose kvadrantuose (4.9 pav.).



4.9 pav. Osciliatoriaus fazių diagrama, kai $m_x = 0{,}5$, $m_y = 0{,}2$

Iš matematinės analizės metu gautų priklausomybių grafikų matoma, kad šios sistemos brėžiama kreivė turi šaknų kvadrantuose, kuriuose $y \in (0; -\infty)$. Tokios vertės būtų netinkamos realiai sistemai – roboto kojai esant žemiausioje įmanomoje pozicijoje, žemiausia y pasiekiama

vertė turėtų būti ne mažesnė nei 0. Ši problema daug paprasčiau išsprendžiama programiškai, nei matematiškai, todėl ji bus sprendžiama tolimesniuose skyriuose.

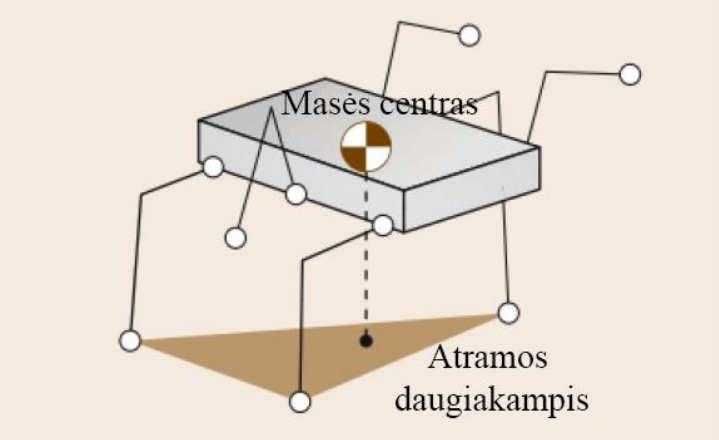
5. Roboto kojų konfigūracijos pagrindimas

Optimaliam roboto judėjimui, būtinas teisingas jo parametrų – matmenų, kūno formos, kojų skaičiaus, jutiklių tipų parinkimas. Prastai parinkti šie parametrai gali sudaryti sunkumus optimizuojant judėjimo algoritmus.

Parametrai turėtu būti parenkami atsižvelgiant į tai, jog simuliacija turėtų būti lengvai perkeliama į realią sistemą, t.y. visi naudojami simuliaciniai komponentai turėtų būti nesunkiai pakeičiami realiais komponentais – servo varikliais, jutikliais, kameromis ir t.t. Toliau bus parenkama optimali kojų konfigūracija bandomajam modeliui.

Pasirinkimas tarp skirtingų roboto kojų konfigūracijų yra milžiniškas – nuo dvikojų (humanoidų) iki keturkojų, šešiakojų ar net aštuonkojų robotų, imituojančių vorų eiseną. Nuo kojų skaičiaus priklauso eisenos pobūdis. Didėjant kojų skaičiui, daugėja galimų stabilių statinių eisenos kombinacijų. Robotų stabilumas yra fundamentalus gero judėjimo elementas. Trumpai tariant – stabilumas yra roboto balansavimo gebėjimas. Egzistuoja du stabilumo tipai – statinis ir dinaminis [Woe11].

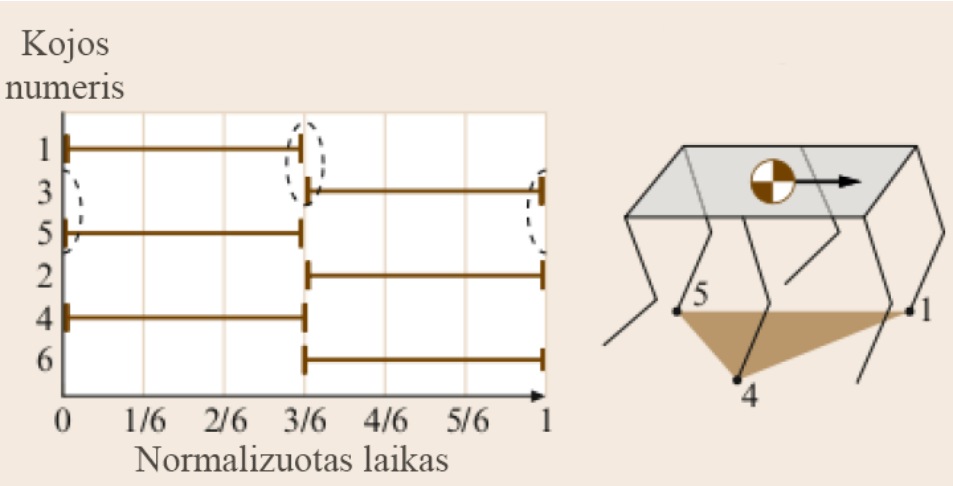
- **Statinis stabilumas** – svarbus, kai nereikia jokių papildomų judesių ar jėgų išlaikyti robotą neapvirtusį. Roboto masės centras nuolat išlieka atramos daugiakampio viduje, tarp kojų, kurios turi sąlytį su žeme (5.1 pav.).
- **Dinaminis stabilumas** – reikalingas, kai masės centras išeina už atramos daugiakampio ribų. Tokiu atveju, robotas nuvirstų, jei nebūtų papildomų jėgų ir kojų judesių, išlaikančių jo pusiausvyrą.



5.1 pav. Daugiakojo roboto atramos daugiakampis. Kai masės centras yra virš daugiakampio, robotas yra stabilus, adaptuota iš [Woe11]

Remiantis aukščiau pateiktais duomenimis, galima teigti, jog šešių ar daugiau kojų robotas suteikia didesnį statinį stabilumą nei keturkojis robotas dėl didesnio atramos daugiakampio. Pavyzdžiui, šešiakojis robotas gali statiškai vaikščioti bet kuriuo metu žemę liečiant penkioms kojoms, o keturkojis robotas statiškai vaikščioti gali tik trimis kojomis. Akivaizdu, jog tris ar mažiau kojų turintis robotas statiškai vaikščioti negali.

Keturkojų robotų trūkumas – mažas judėjimo greitis statinio vaikščiojimo metu. Kadangi trys iš keturių kojų privalo liesti žemę, kad būtų išlaikomas stabilumas, keisti poziciją gali tik likusi koja. Kitaip tariant, likusios kojos „laukia“ savo eilės judėti. Tuo metu šešiakojis robotas gali išlaikyti stabilumą trimis kojomis, o likusios trys gali atlikti žingsnį į priekį (5.2 pav.).



5.2 pav. Trikojė šešiakojo roboto eisena. Čia trys kojos juda vienu metu išlaikant statinį stabilumą, adaptuota iš [Woe11]

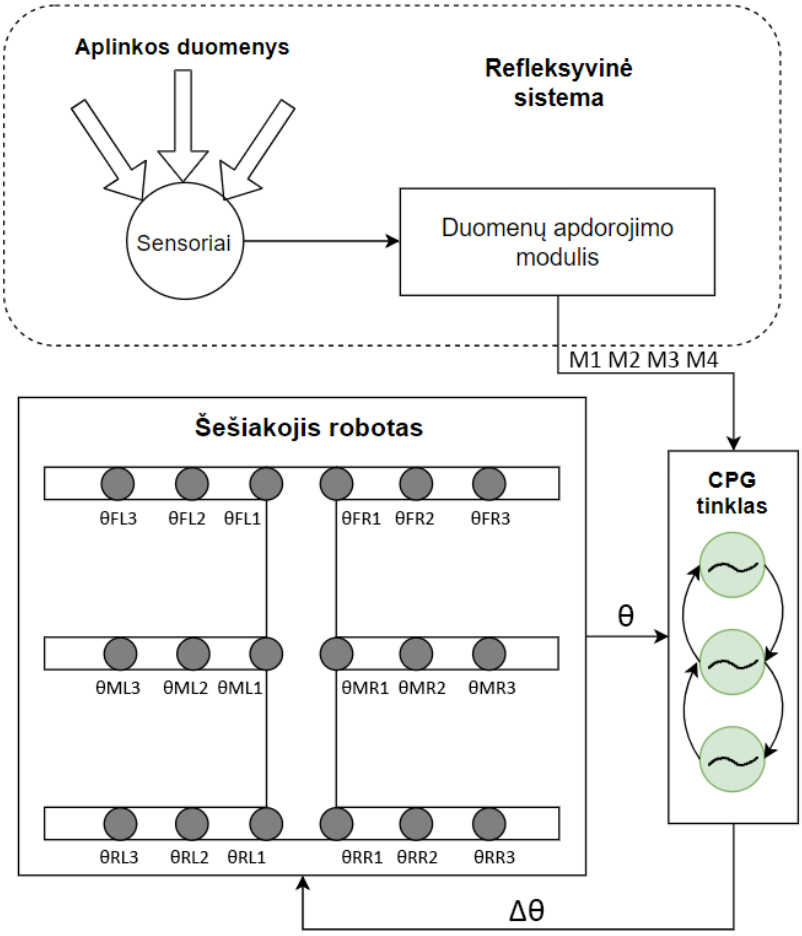
Siekiant didesnio roboto stabilumo, prasminga vengti dinaminio judėjimo. Tokiu atveju aukojamas potencialus judėjimo greičio padidėjimas, tačiau nereikia galvoti apie papildomų jėgų, padedančių išlaikyti roboto pusiausvyrą, generavimą (5.2. pav.). Esant tokiai eisenai, priekinė ir galinė kiekvienos pusės koja juda kartu su kitos pusės vidurine koja. Tai vienintelė įmanoma statiškai stabili šešiakojo roboto eisena, kur trys kojos juda vienu metu.

Taigi, išanalizavus galimas kojų konfigūracijas aišku, jog šešiakojis robotas suteikia gerą statinį stabilumą, bei didesnį judėjimo greitį palyginus su keturkoju robotu. Aštuonias ar net daugiau kojų turintis robotas galėtų pasiekti net didesnį greitį ir stabilumą, tačiau didėjant kojų skaičiui didėja ir valdymo kompleksiškumas, taip pat dalių skaičius. Remiantis šiomis išvadomis darbui atlikti bus modeliuojamas šešiakojis robotas.

6. Adaptyvus kombinatorinis judėjimo valdymas

Ankstesniame skyriuje įrodyta, kad *CPG* paremti algoritmai geba generuoti periodiškus, roboto valdymui tinkamus signalus. Tačiau tokia sistema nėra dinamiška – *CPG* generuojamas signalas nepriklauso nuo aplinkos faktorių, priešingai nei gyvuose organizmuose. Neišnaudojamas pilnas *CPG* potencialas – prisitaikymas prie kintančių aplinkos sąlygų.

Šią problemą galima spręsti pritaikant adaptyvų kombinatorinį valdymą sukuriamą panaudojant *CPG* algoritmus bei aukštesnio lygio signalus iš valdymo/koregavimo sistemos [CLW+16]. Ši sistema gali būti įgyvendinta apibrėžiant elgsenos modelius, kurie būtų taikomi esant tam tikromis situacijomis. Tačiau toks pritaikymas nėra dinamiškas – jei scenarijus nėra numatytas, sistema gali netinkamai interpretuoti aplinkos veiksnius ir siųsti klaidingą signalą *CPG* mazgams, generuojantiems judesius.



6.1 pav. Refleksyvinės sistemos integravimo schema

Tam, kad aplinkos interpretavimas būtų kuo tikslesnis, galima valdymo sistemai pasitelkti didesnio abstrakcijos lygio metodus, leidžiančius įvertinti aplinką dinamiškai, pavyzdžiui,

numatyti ir įvertinti artėjančią kliūtį (į robotą mestą objektą, artėjančią duobę ar pan.) bei atitinkamai koreguoti *CPG* generuojamus signalus. Tokio tipo sistemos, dar vadinamos refleksyviomis arba refleksais paremtomis, įgalina dalinę roboto autonomiją, galinčią pagerinti roboto valdymą.

Integravus refleksyvinę sistemą, iš jutiklių surenkama informacija interpretuojama duomenų apdorojimo modulio (6.1 pav.), iš kur siunčiamos „instrukcijos“ *CPG* mazgui. Šios instrukcijos galėtų būti siunčiamos *CPG* osciliatorių modifikatorių (M1-M4), aptartų ankstesniame skyriuje, pavidalu, taip efektyviai keičiant žingsniavimo parametrus.

Jutiklių surenkami duomenys (kamerų vaizdas, artumo jutiklių atstumai ir kt.) interpretuojami duomenų apdorojimo modulio. Šis modulis, remdamasis aplinkos duomenimis turi nuspręsti, kaip pakoreguoti *CPG* parametrus taip, jog būtų išlaikomas geriausias judėjimo pobūdis. Pavyzdžiui, jei artėjama prie kliūties, kuri kliudytų roboto viršutinę dalį, duomenų apdorojimo modulis turėtų nuspręsti, ar robotas gali „pasilenkti“ taip, kad įveiktų kliūtį ir nusiųstų atitinkamas modifikatorių vertes *CPG* mazgui, priešingu atveju, nustačius, kad robotas netelpa, turėtų būti mažinamas roboto greitis, taip išvengiant susidūrimo.

CPG modulis gauna informaciją apie esamas kojų aktuatorių (variklių, sąnarių) pozicijas ir paskaičiuoja, kaip toliau turėtų būti keičiamos aktuatorių pozicijos, kad būtų išlaikoma nustatyta judėjimo kryptis.

7. Centrinių sekų generatorių algoritmo pritaikymas daugiakryptei navigacijai

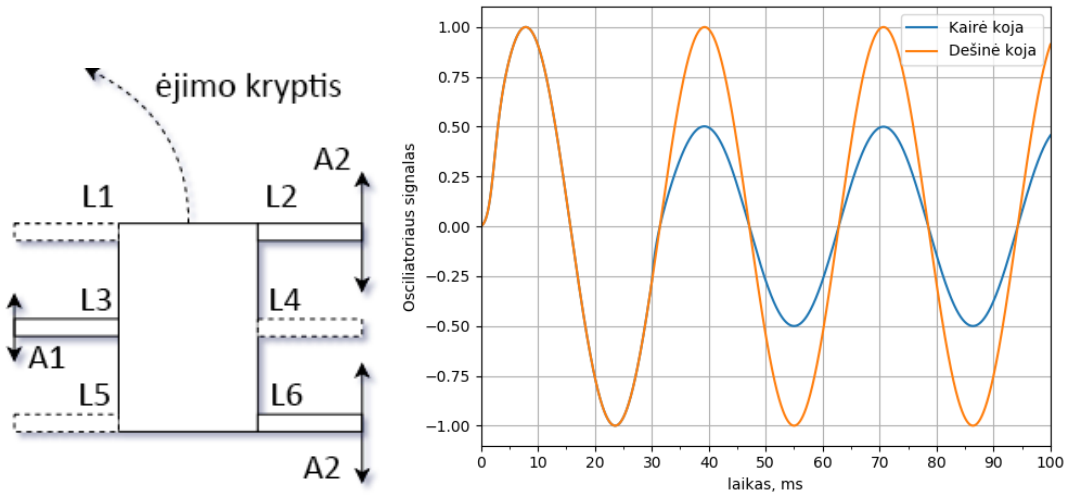
CPG paremti osciliatoriai turi daug naudingų ypatybių realaus laiko robotų trajektorijų generavimui. Trajektorijos generuojamos realiu laiku su mažais skaičiavimo kaštais. Taip pat gebėjimas konverguoti į nustatytas ribas leidžia integruoti atsako sistemas patikimam judesių valdymui. Generuojamų trajektorijų moduliavimas remiantis jų amplitude (žingsnio ilgiui), dažniu (žingsnio greičiui) ir poslinkiu vykdomas reguliuojant mažą parametrų rinkinį [SM12].

Esama tyrimų, kuriuose *CPG* funkcionalumas buvo pritaikytas vienakrypčiam judėjimui bei dinamiškam atsakui į aplinkos pokyčius [MPW08], tačiau toks judėjimo metodas nėra optimalus, jei judėjimo kryptis bei kiti parametrai nėra nustatyti iš anksto, pavyzdžiui, robotą kontroliuojant žmogaus pagalba.

7.1. Roboto sukiojimas judėjimo metu

Modifikuojant priešingose roboto pusėse atramos daugiakampį (turinčių sąlytį su žeme) sudarančių kojų svyravimo amplitudę, galima prailginti priešingų roboto pusių judėjimo greitį, taip priverčiant robotą suktis. Šis metodas leidžia paprastai keisti judėjimo kryptį robotui jau

judant. 7.1 pav. pateiktoje diagramoje (roboto kojos sunumeruotos L1-L6), kai žemę liečia kojos L2, L3 ir L6, sumažinus kojos L3 amplitudę perpus, roboto judėjimo kryptis pakrypsta į kairę. Šis pokytis matomas grafike, kai $t = 30\ ms$, kairės kojos *CPG* osciliatoriaus amplitudę sumažinus perpus ($A_1 = \frac{A_2}{2}$). Vienas iš svarbių *CPG* aspektų, padedančių generuoti stabilius signalus – keičiant amplitudę, svyravimo dažnis nekinta, todėl kojos juda sinchroniškai, nepaisant skirtingo žingsnio ilgio.



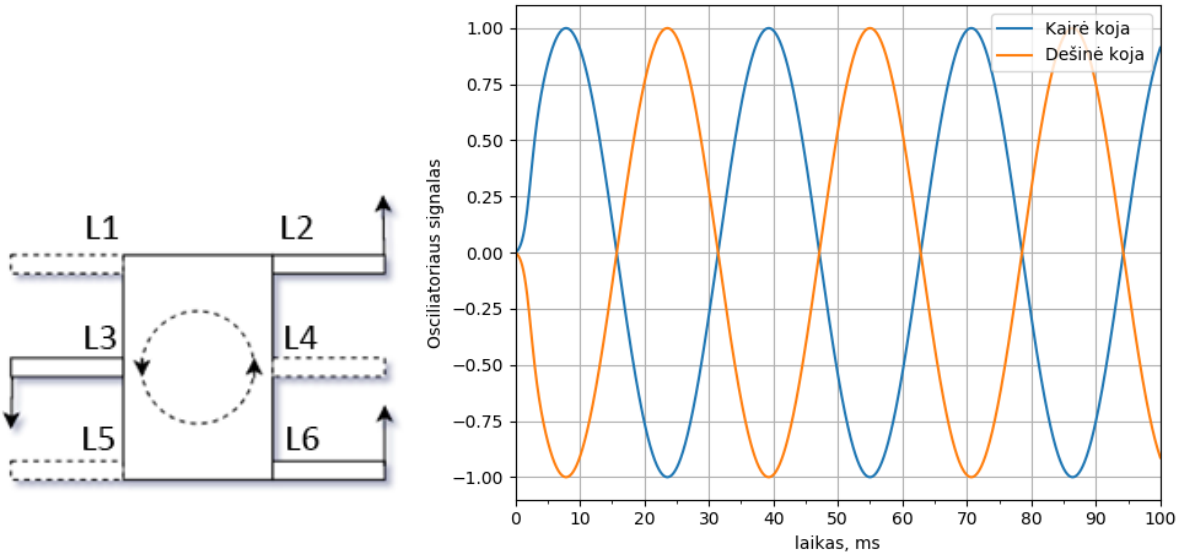
7.1 pav. Ėjimo kryptis, kairės pusės kojų osciliatorių amplitudę sumažinus ties $t = 30ms$

Esama ir žingsnio dažnio modifikavimu paremtų metodų ėjimo krypties keitimui [SM12], tačiau taip atsiranda kojų sinchronizavimo problema – vienu metu pakilus visoms vienos roboto pusės kojoms gali būti prarandamas statinis stabilumas ir robotas nuvirstų. Šiame darbe taikomas ėjimo krypties modifikavimo metodas leidžia išlaikyti statinį stabilumą bet kurioje žingsnio fazėje.

7.2. Sukimasis aplink fiksuotą tašką

Robotui esant ankštose erdvėse, apsisukimas judant ir pirmyn gali būti komplikuotas. Optimaliausias būdas apsukti robotą tokiu atveju – sukimasis aplink fiksuotą tašką. Toks judėjimo būdas gali būti pasiekiamas priešingose roboto pusėse esančioms kojoms siunčiamos tokios paties amplitudės bei dažnio virpesių signalus, tik vienai iš pusių apverčiant signalo kryptį.

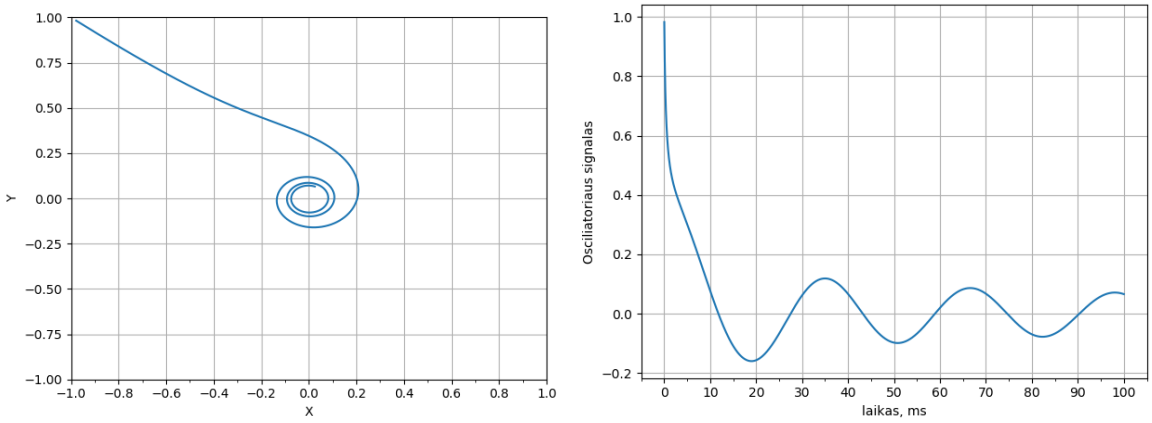
Tokio judėjimo schema pateikta 7.2 pav., kur invertavus kairės pusės kojų osciliatorių signalus gaunamas judėjimas aplink fiksuotą tašką. Kojoms L2 ir L6 judant į priekį, koja L3 juda priešinga kryptimi. Tuo metu kojos L1, L4 ir L5 grįžta į pradines padėtis. Galimas šio metodo trūkumas – kojų praslydimas, atsirandantis dėl nevienodu atstumu nuo roboto centro nutolusių kojų, turinčių sąlytį su žeme. Jį galima sušvelninti papildomu sąnariu, kuris naudotų papildomą *CPG* mazgą ir koreguotų atstumą tarp kojos, sąlyčio su žeme metu ir roboto centro.



7.2 pav. Sukamojo judėjimo generavimas, invertuojant *CPG* signalą ($\acute{x}_1 = \acute{x}_2$)

7.3. Stabdymas

Kita dinaminė roboto savybė, kuri nėra tokia akivaizdi – pilnas sustojimas, kai pasiekiamas tikslas arba sistema baigia darbą. *CPG* paremti osciliatoriai leidžia tolygiai mažėjančios amplitudės svyravimus, kai amplitudės modifikatorius nustatomas į nulinę vertę.

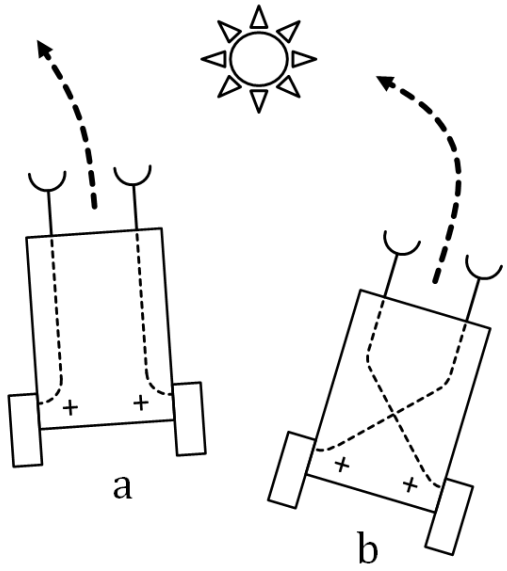


7.3 pav. Osciliatoriaus signalų konvergavimas link nulio, kai amplitudės modifikatorius nustatomas į $\omega_x = \omega_y = 0$

Aukščiau pateiktuose grafikuose (7.3 pav.) pateikta priklausomybė tarp osciliatorių dedamųjų (kairėje) ir osciliatoriaus x dedamosios laike (dešinėje). Iš priklausomybių matoma, kad įvyksta staigus pradinis amplitudės sumažėjimas ir vyksta tolimesnis lėtas konvergavimas link nulio. Stabdymo greitį galima koreguoti didinant arba mažinant osciliatorių konvergavimo greitį.

8. Navigavimas nežinomoje aplinkoje taikant Braitenbergo algoritmą

Braitenbergo algoritmas [Bra84] padeda agentui (robotui) naviguoti aplinkoje, remiantis jutiklių duomenimis. Paprasčiausiose konfigūracijose, jutiklis būna prijungtas tiesiogiai prie efektoriaus, taip generuojant judesį.



8.1 pav. Dviejų tipų Braitenbergo mašinos šalia šviesos šaltinio

Šiame darbe naudojamam modeliui panaudos idėjos, remiantis 8.1 pav. pateiktomis Braitenbergo mašinomis. Abi šios mašinos turi po du variklius, tiesiogiai pritvirtintus prie ratų ir du šviesos jutiklius. Mašinos skiriasi tik tuo, kaip šie jutikliai sujungti su varikliais. Mašinoje „a“, jutikliai prijungti prie tos pačios pusės ratų, o „b“ mašinoje, ratai sujungti su priešingos pusės jutikliais (kryžminis jungimas). Pliusas nurodo, kad kuo didesni jutiklio duomenys, tuo variklis greičiau sukasi. Saulės simbolis nurodo šviesos šaltinį, kurį gali aptikti jutikliai.

Mašinos „a“ konfigūracija, artėjant prie šviesos šaltinio, dešiniame jutiklyje aptinka didesnius rodmenis todėl dešinys ratas ima suktis greičiau už kairįjį ir mašina juda kairė – tolyn nuo šviesos šaltinio. Mašina „b“ priešingai, artėja prie šviesos šaltinio, nes didinamas priešingo rato sukimosi greitis. Iš 8.1 pav. aišku, jog „a“ ir „b“ mašinų principai gali būti panaudoti kliūčių vengime.

Tokia sąveika tarp jutiklių ir aktuatorių taikytina roboto navigavime aplinkoje [Bra84]. Remiantis Braitenbergo mašinų algoritmu, galima teigti, jog sėkmingam kliūčių vengimui taikytina abiejų mašinų kombinacija, į kiekvieno rato greičio skaičiavimus įtraukiant ir priešingos puses jutiklio rodmenis, tik apvertus jų kryptį. Toks metodas leistų pristabdyti roboto eigą priartėjus kliūčiai, nes akivaizdu, jog lėtesnis judėjimas kliūties link yra saugesnis, nei greitas

judėjimas. Jeigu jutikliai būtų primontuoti mašinos gale, jų signalas turėtų greitinti roboto judėjimą tolyn nuo kliūties. jutiklių poveikį kiekvieno rato greičiui galima išreikšti lygybe:

$$v_p = f(\sum_{i=1}^{n} \omega_i d_i). \tag{8.1}$$

Čia n – jutiklių skaičius, ω_i – unikalus jutiklio svoris, o d_i nurodo jutiklio vertę. Svorių vertės nėra žinomos ir priklauso nuo sprendžiamos problemos. Toliau pateiktoje lygybėje (8.2) pateikta roboto jutiklių aktyvumo funkcija [SM12]. Iš funkcijos matoma, kad roboto priekyje esantys jutikliai turės didesnę įtaką dėl mažesnio jutiklio kampo β_i nuo roboto judėjimo krypties:

$$\omega_i = k_1 e^{-\frac{d_i}{k_2}} \cdot e^{-\frac{\beta_i^2}{2\sigma^2}}. \tag{8.2}$$

Kur d_i – normalizuotas jutiklio nurodomas atstumas tarp kliūties ir roboto, k_1 ir k_2 – bendro stiprumo ir jutiklio signalo slopinimo modifikatoriai. Naudojamos aktyvavimo funkcijos vertės nuo 0 iki 1, taip normalizuojant vertes. Akivaizdu, kai robotas nukreiptas į kliūtį, jis suksis smarkiau, nei kliūčiai esant jo šone dėl jutiklio kampo įtakos aktyvumo funkcijos vertei. Galutinė aktyvumo funkcija pateikta (8.3):

$$\varphi_g = \sum_{i=1}^{n} \omega_{ik} \frac{r_i}{r_{max}} - \sum_{i=1}^{n} \omega_{id} \frac{r_i}{r_{max}}. \tag{8.3}$$

Čia φ_g – kliūčių vengimo atstumiamoji jėga, r_i ir r_{max} – esama ir maksimali leistina jutiklio rodmenų vertė. Individualios roboto pusės aktyvumo funkcija (8.4):

$$\varphi_p = \sum_{i=1}^{n} \omega_{ip} \frac{r_i}{r_{max}}. \tag{8.4}$$

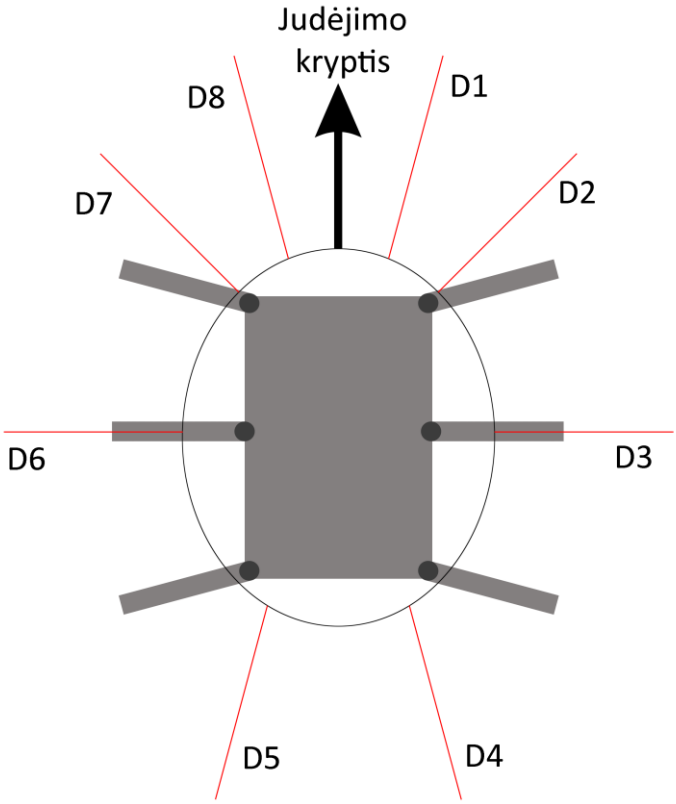
Pagal (8.4) funkciją apskaičiuojama, kaip stipriai turi būti modifikuojamas kiekvienos iš pusių judėjimo greitis. Aktyvumo funkcijos φ_p reikšmės priklauso nuo jutiklių, prijungtų prie atitinkamų roboto pusių.

9. Roboto valdymo schemos modeliavimas

Darbe sukurto modelio pademonstravimui pasitelkta šešis laisvės laipsnius turinčio šešiakojo roboto schema. Toliau pateikta aptartų komponentų – centrinio sekų generatoriaus mazgo, refleksyvinės sistemos (Braitenbergo algoritmo pavidalu) bei jutiklių apjungimo analizė.

9.1. Jutiklių konfigūracija

Tam, kad Braitenbergo algoritmas teisingai nustatytų judėjimo kryptį, išvengiant kliūčių, svarbu tinkamai išdėstyti jutiklius. Šiam darbui parinkti paprasti linijiniai infraraudonųjų spindulių jutikliai, su maksimaliu 5 metrų matymo atstumu.



9.1 pav. Infraraudonųjų spindulių jutiklių išdėstymo schema

Kiekvienas jutiklis pakreiptas tam tikru kampu nuo judėjimo krypties taip, kad būtų padengiamas kuo didesnis plotas aplink robotą. Galinėje roboto dalyje jutiklių mažiau, nes jų svarba mažesnė, nei priekyje esančių jutiklių.

9.1 lentelė. Infraraudonųjų spindulių jutiklių pasiskirstymo kampai

Jutiklis	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
Kampas, °	15	45	90	150	-150	-90	-45	-15

Žinant kampų dydžius, empiriškai nustatyti individualių jutiklių įtakos koeficientai (svoriai), kurie bus įstatomi į (8.2) išraišką [SM12]. Gautos vertės priderintos iš kitų, Braitenbergo algoritmą ratuotai navigacijai naudojančių šaltinių ir priderintos šiam darbui. Šios vertės nurodo, kokią įtaką roboto krypčiai turės kiekvienas jutiklis. Gautos vertės kairės ir dešinės roboto pusių modifikatoriams pateiktos 9.2 ir 9.3 lentelėse.

9.2 lentelė. Jutiklių svorių vertės kairės pusės greičio modifikavimui

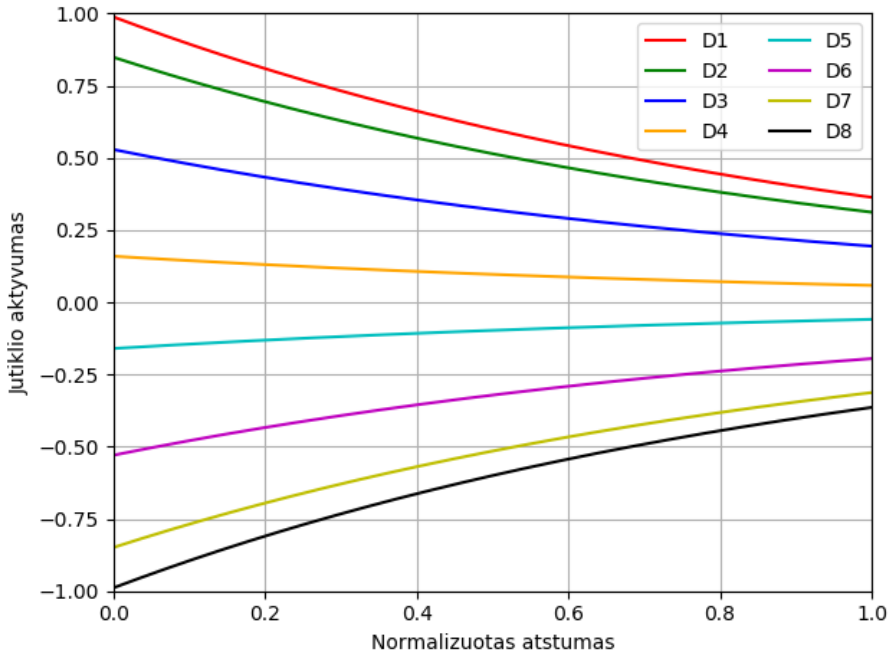
D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
-1	-0.85	-0.5	-0.15	0.15	0.5	0.85	1

9.3 lentelė. Jutiklių svorių vertės dešinės pusės greičio modifikavimui

D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
1	0.85	0.5	0.15	-0.15	-0.5	-0.85	-1

Turint šias vertes, roboto ėjimo krypčiai apskaičiuoti reikalingi tik jutiklių rodmenys, įstatant juos į (8.2) išraišką.

Jutikliai sukonfigūruoti grąžinti vertę tarp 0 (robotas liečia kliūtį) ir 1000 (atstumas iki kliūties – 5 metrai). Tokios vertės netinkamos Braitenbergo algoritmui ir privalo būti normalizuotos.



9.2 pav. Jutiklių aktyvumo priklausomybė nuo normalizuoto atstumo iki kliūties

Žinant kiekvieno jutiklio svorį, galima apskaičiuoti kiekvieno jutiklio aktyvumo vertę (8.4). Iš 9.2 pav. matoma, jog priekiniai jutikliai (D_1 ir D_8) turi didžiausią įtaką roboto krypčiai, dėl mažiausio kampo nuo judėjimo krypties ir didžiausio svorio. Galiniai jutikliai (D_4 ir D_5) įgyja labai nedideles aktyvumo vertes, todėl jų įtaka ėjimo greičiui taip pat maža. Esant statinei roboto

aplinkai, šie jutikliai gali būti pašalinti, bet esant dinaminei aplinkai, padidinus jų svorių vertes, būtų padidinamas roboto judėjimo tolyn nuo artėjančios iš galo kliūties greitis.

Jutiklių duomenys apdorojami Braitenbergo algoritmu ir siunčiami *CPG* mazgui, modifikatorių M_k ir M_d pavidalu, atitinkamai nurodančių kairių ir dešinių roboto kojų žingsnio ilgio modifikavimo koeficientus:

$$\begin{pmatrix} M_k \\ M_d \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} \omega_{ik} d_i \\ \sum_{i=1}^{n} \omega_{id} d_i \end{pmatrix} \tag{9.1}$$

Kur ω_{ik}, ω_{id} – jutiklių svorio modifikatoriai kairei ir dešinei roboto pusėms, d_i – normalizuotas jutiklio registruojamas atstumas. Braitenbergo algoritmu apdoroti duomenys, kartu su jutikliais sudaro refleksyvinę valdymo sistemą, aptartą 6-ame skyriuje.

9.2. Roboto variklių parametrų konfigūracija

Kiekviena roboto koja susideda iš trijų variklių, kurie apibūdinami maksimalia amplitude bei fazės poslinkiu. Amplitudė nusako variklio sukimosi ribas t.y. tokių parametrus kaip maksimalus žingsnio ilgis, kojos kėlimo aukštį ir pan. Fazės poslinkis nurodo, kiek vienas variklis „atsilieka“ nuo kito. Kiekvienos kojos variklių fazių poslinkiai P_i ir amplitudės A_i yra vienodos. Bandymų metu gauti natūralią tiesiaeigę eiseną gebantys generuoti poslinkiai ir amplitudės, jų vertės pateiktos 9.2 ir 9.3. Daugiakrypčiam judėjimui taikytinos tik nustatytos fazės poslinkio vertės, amplitudė kinta priklausomai nuo judėjimo krypties.

$$P_i = \begin{pmatrix} p_{i1} \\ p_{i2} \\ p_{i3} \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 2{,}5 \end{pmatrix} rad \tag{9.2}$$

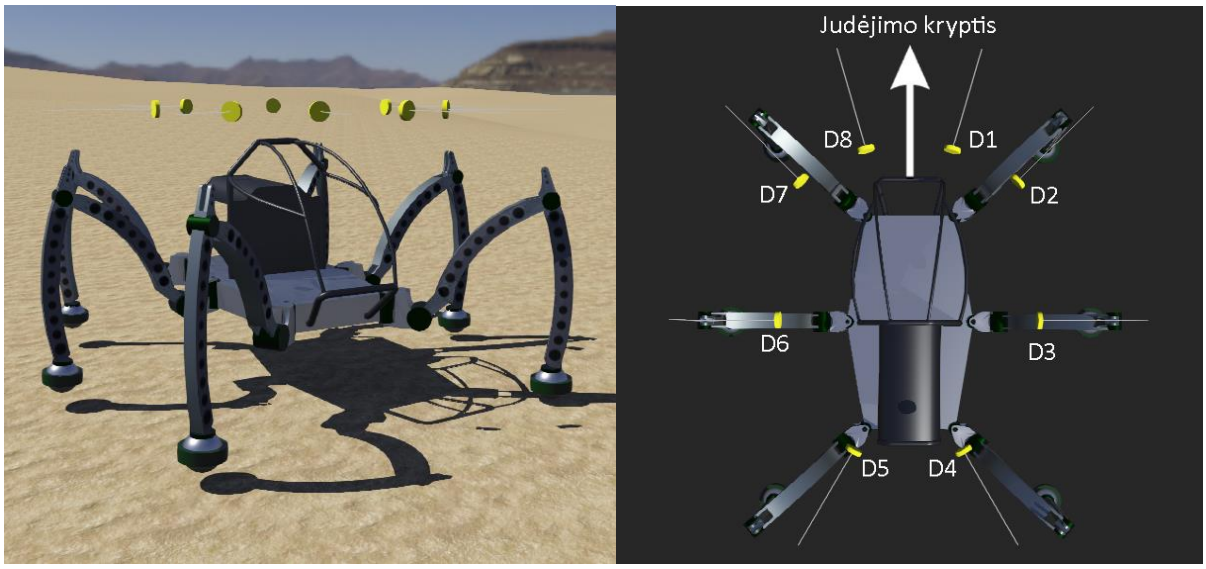
$$A_i = \begin{pmatrix} a_{i1} \\ a_{i2} \\ a_{i3} \end{pmatrix} = \begin{pmatrix} 0{,}25 \\ 0{,}2 \\ 0{,}05 \end{pmatrix} rad \tag{9.3}$$

9.3. Modelis simuliacinėje aplinkoje

Darbe analizuotų modelių ir idėjų patikrinimui simuliacinėje aplinkoje sukurtas šešiakojo roboto modelis, apjungiantis *CPG* ir Braitenbergo modulius, bei variklius, jutiklius. Šiam tikslui parinkta *Webots* programinė įranga dėl sąsajos paprastumo ir su įranga suteikiamų robotų modelių, leidžiančių greitai prototipuoti ir išbandyti idėjas.

Webots aplinkoje sukurti robotų modeliai susideda iš dviejų dalių – *proto* failo, kuriame nurodomos modelio vertės tokios, kaip komponentų tipai, dydžiai pradinės vertės (kampai, tarpusavio sąsajos), bei valdiklio dalies, kurioje aprašoma roboto logika. Valdiklis (angl. *controller*) – tai pasirinkta programavimo kalba parašytas kodo skriptas. Šiame roboto modelyje valdiklyje aprašyta jutiklių logika, Braitenbergo algoritmas bei *CPG* mazgas. Kodui pasirinkta *C* programavimo kalba dėl geriausiai dokumentuotos sąsajos *Webots* aplinkoje. Ši kalba tinkama ir didelėse robotų sistemose dėl didelės spartos, taip pat dėl didelės bibliotekų gausos.

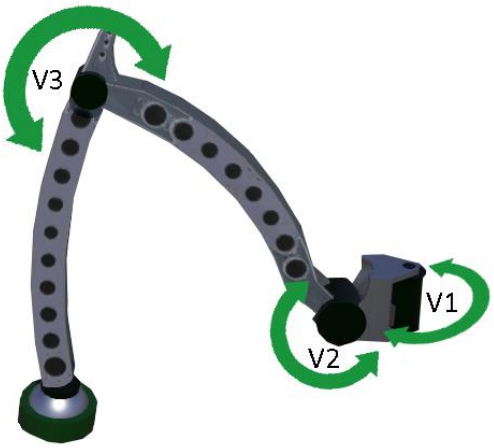
Iš *Webots* siūlomos robotų bibliotekos pasirinktas *Mantis* – šešiakojo roboto modelis [Cyb19], turintis 18 servo variklių (po 3 kiekvienai kojai).



9.3 pav. Adaptuotas *Mantis* roboto modelis *Webots* aplinkoje. Dešinėje pavaizduotas jutiklių (geltoni) pasiskirstymas aplink robotą

Roboto modelis papildytas aštuoniais infraraudonųjų spindulių jutikliais, pritaikius 9.1 schemą. jutikliai pakelti virš roboto modelio taip, kad judančios kojos neribotų jų matymo lauko.

Kiekviena roboto koja turi 3 variklius, kuriems tiesiogiai siunčiamas *CPG* mazgo generuojamas signalas. Varikliai žymimi V_1, V_2 ir V_3, tolstant nuo roboto kūno (9.4 pav.).



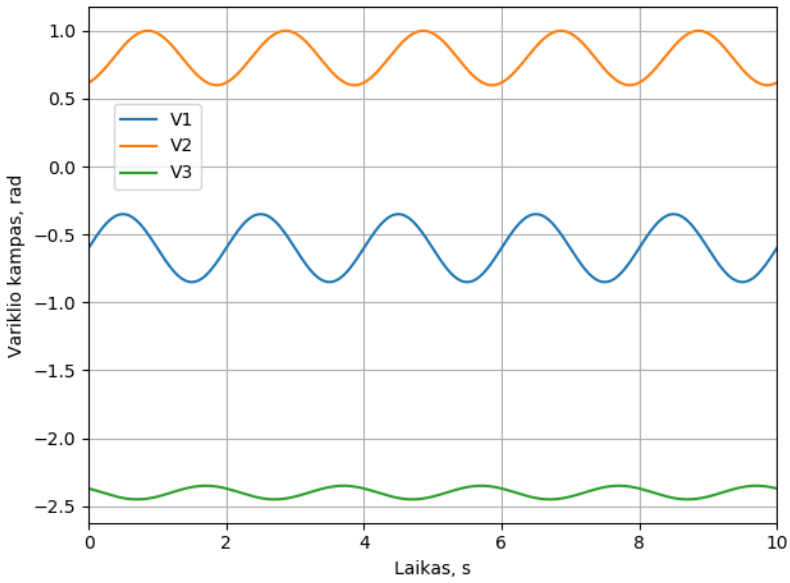
9.4 pav. Roboto variklių žymėjimo schema ir sukimosi kryptys. Varikliai žymimi didėjančia tvarka tolstant nuo roboto kūno

10. Simuliacinio modelio rezultatai

Toliau pateikti bandymų, atliktų panaudojant darbe aptartą judesių generavimo modelį, rezultatai. Roboto aplinka suprojektuota *Webots* programine įranga. Visi grafikai sudaryti iš realiu laiku vykdytų simuliacijų surinktų duomenų.

10.1. Judesių generavimo rezultatai nesant aukštesnio lygio valdymui

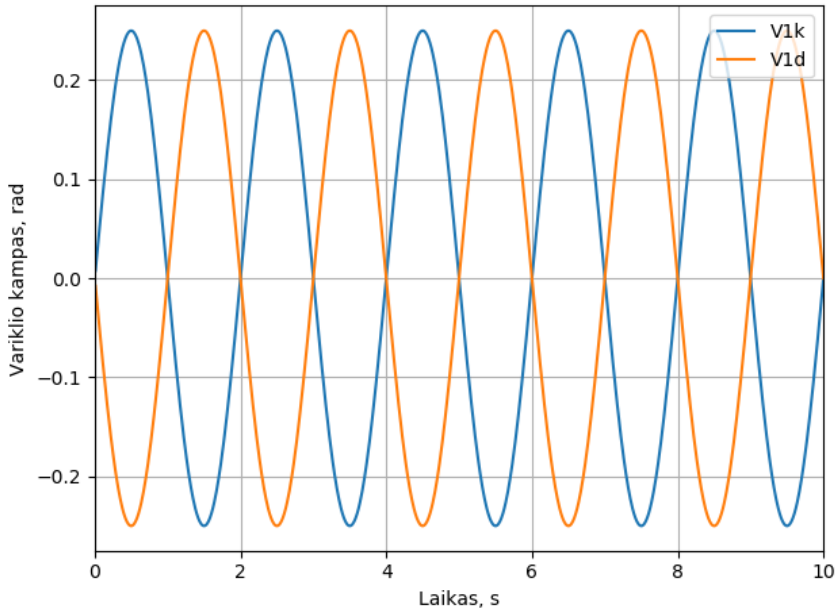
10.1.1. Tiesiaeigis judėjimas



10.1 pav. Priekinės kairės kojos variklių posūkio kampai esant tiesiaeigiam judėjimui

Pritaikius *CPG* algoritmą visoms roboto kojoms, išgautas tolygus tiesiaeigis judėjimas (10.1 pav.), t.y. robotas juda tiesia linija į pirmyn. Kiekvienas variklis veikia sinchroniškai, bet skiriasi vienas nuo kito fazės poslinkiu, nurodytu 9.2 išraiškoje. Grafike nepavaizduotų variklių priklausomybės yra identiškos, skiriasi tik jų poslinkiai.

10.1.2. Sukimasis aplink fiksuotą tašką

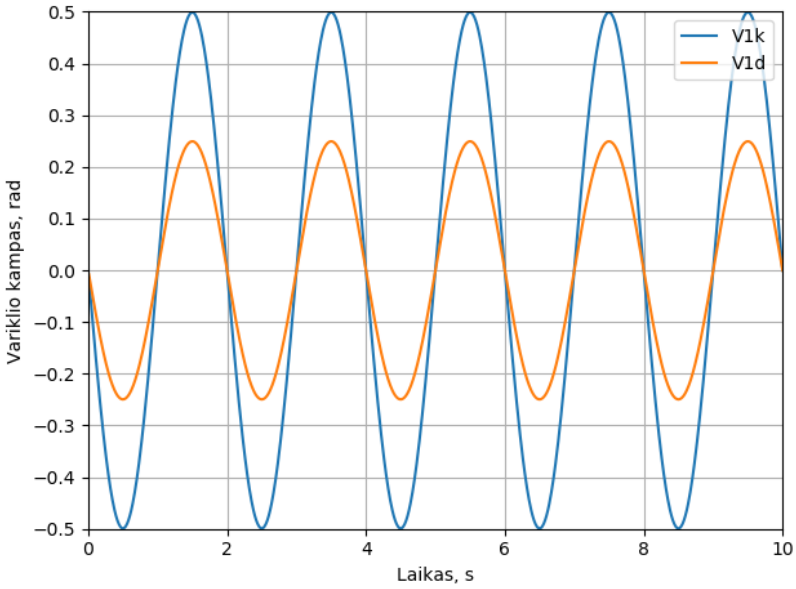


10.2 pav. Kairėje ir dešinėje roboto pusėse esančių vidinių (arčiausiai kūno esančių) variklių posūkio kampai robotui sukantis aplink fiksuotą tašką (*A = 0.5 rad*)

Vidinėms skirtingų roboto pusių kojoms invertavus *CPG* generuojamą signalą, robotas sukasi vietoje (žr. 7.2 pav.). Toks būdas suktis aplink savo ašį labai patogus tuo, jog nereikalingi papildomi skaičiavimai, visos kojos išlieka sinchronizuotos, išlaikomas roboto statinis stabilumas.

10.1.3. Sukimasis į šoną

Skirtingose roboto pusėse esantiems vidiniams (arčiausiai kūno) esantiems varikliams suteikus skirtingas amplitudės vertes, robotas juda link tos pusės, kurios amplitudė yra mažesnė. Iš 10.3 pav. matoma, kokias posūkio kampo vertes įgyja kairės ir dešinės roboto pusių vidiniai varikliai, lemiantys žingsnio ilgį. A_{V1k} čia lygus 1, o A_{V1d} – 0,5, tai reiškia, kad dėl didesnės kaires kojos amplitudės, kairė roboto pusė nukeliauja du kartus didesnį atstumą ir robotas suka į dešinę.



10.3 pav. Kairėje ir dešinėje roboto pusėse esančių vidinių (arčiausiai kūno esančių) variklių posūkio kampai robotui judant dešinėn. Amplitudžių vertės: $A_{V1} = 1$, $A_{V2} = 0{,}5$

Iš bandymų rezultatų matoma, kad robotas geba judėti tiesiai, atlikti posūkius į šonus bei suktis vienoje vietoje be aukštesnio lygio valdymo signalo, kaip aptarta 7-ame skyriuje. Toliau bus nagrinėjami rezultatai, gauti pasitelkus aukštesnio lygio valdymo sistemą.

10.2. Refleksyvinės sistemos koordinuojamo judesių generavimo rezultatai

Pasitelkiant refleksyvinę sistemą, aptartą 6-ame ir 8-ame skyriuose, sumodeliuotas roboto atsako lankas, gebantis reaguoti į roboto aplinkoje atsiradusias kliūtis. Integravus atsako lanką į sistemą, gautas roboto judesių generavimo ir valdymo algoritmas, jo schema pateikta 10.4 pav.

10.4 pav. Roboto judesių generavimo algoritmo schema

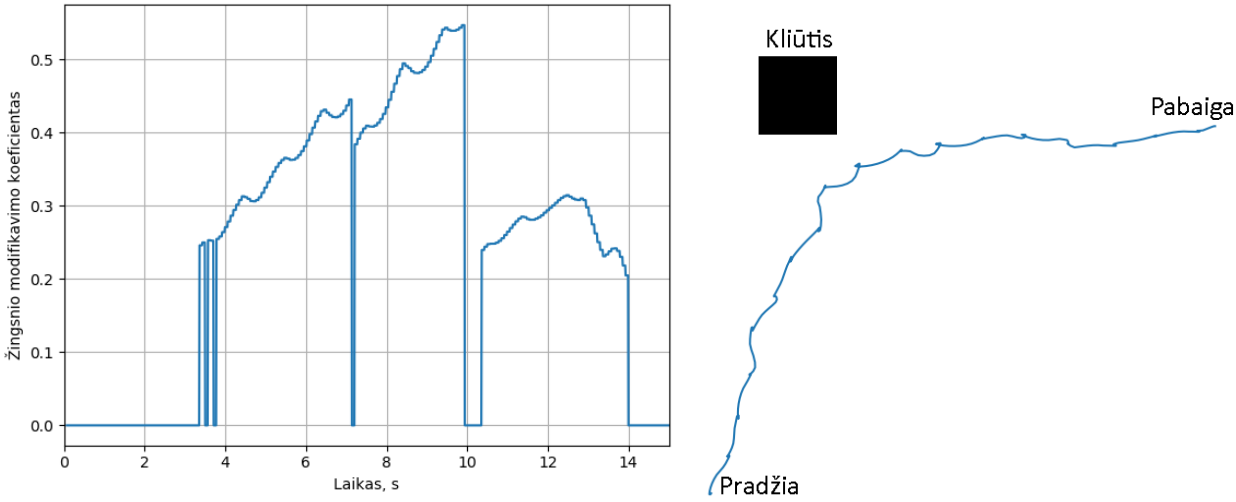
Judesių generavimo algoritmas:

Iš aštuonių jutiklių surenkama informacija apie atstumus iki aplink robotą esančias kliūtis, šie atstumai normalizuojami ir toliau apdorojami Braitenbergo mazge, kur jie paverčiami į kitą požymių erdvę – kairės ir dešinės roboto pusių kojų žingsnio ilgio modifikavimo koeficientus. Šie koeficientai *CPG* algoritmui nurodo amplitudės modifikavimą už žingsnio ilgį atsakingiems varikliams. *CPG* algoritmo sugeneruoti posūkio kampai siunčiami varikliams.

10.2.1. Pavienių kliūčių vengimas

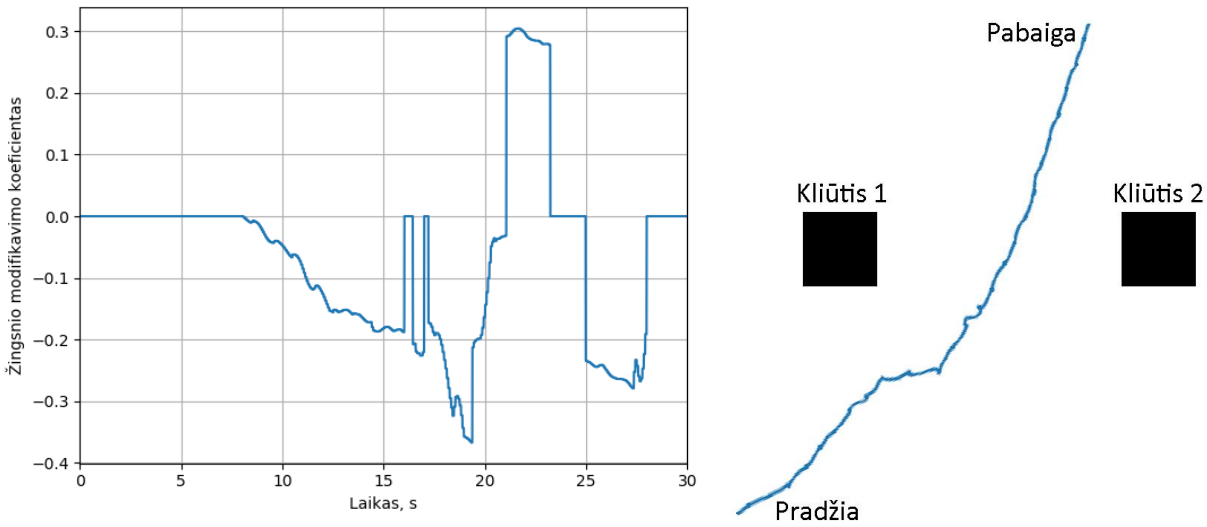
Toliau pateikti bandymų rezultatai, kai roboto simuliacija vykdoma aplinkose, turinčiose mažą kliūčių skaičių.

Robotą nukreipus tiesiai į kliūtį (10.5 pav.), kliūčiai patekus į jutiklių matymo lauką, žingsnio ilgio modifikatoriai įgyja vertes $M_k = 0{,}25$ ir $M_k = -0{,}25$, todėl kairės pusės roboto kojų žingsnio ilgis padidėja 25 %, o dešinės – sumažėja 25 %, priverčiant robotą suktis dešinėn. Robotui toliau artėjant kliūties link, modifikatorių vertės didėja, todėl sukama vis smarkiau, kylant iki 0,6. Ties 10 sekundžių riba, kliūtis išnyksta iš priekinių roboto jutiklių regos linijos, todėl modifikatorių vertės staigiai sumažėja. Trumpam laikui modifikatorių vertė tampa lygi nuliui, nes kliūties nemato nei vienas iš jutiklių, tai vadinama „akloji zona“. Kliūčiai vėl patekus į šoninių jutiklių regos lauką, modifikatorių vertės jau mažesnės, nesmarkiai viršija 0,3, nes kliūtis yra roboto šone, ją matančių jutiklių svorių vertės mažesnės.



10.5 pav. Pavienės kliūties vengimas. Kairėje pavaizduotas žingsnio ilgio modifikavimo koeficiento priklausomybė nuo laiko, dešinėje – roboto judėjimo trajektorija

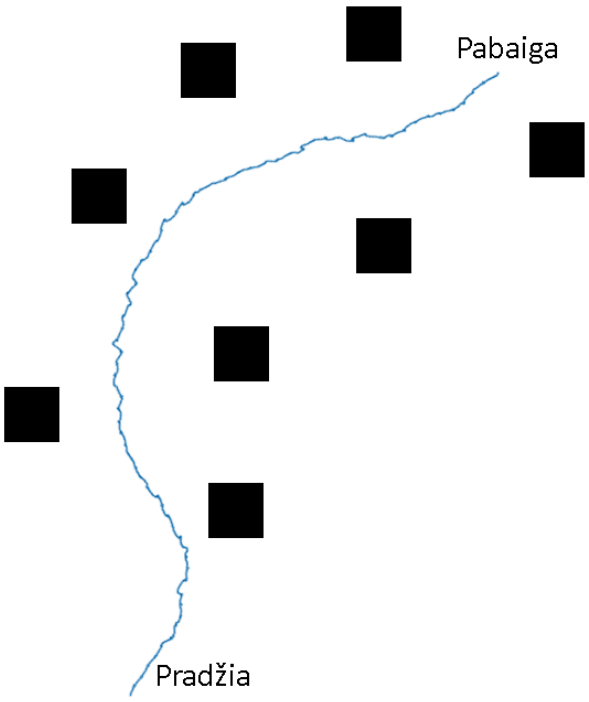
Kai robotas nukreipiamas link kliūčių, tarp kurių tarpas didesnis, nei roboto plotis (10.6 pav.), Braitenbergo algoritmas geba įvertinti, ar robotas tilps tarp kliūčių, ar jas reikėtų apeiti. Robotui artėjant link kliūčių, pirmiausiai į jutiklių regos lauką patenka 2 kliūtis, robotas ima švelniai sukti kairėn. Priartėjus prie 1 kliūties, ją pradeda aptikti šoniniai jutikliai, todėl trumpam nusukama dešinėn, kol kliūtis vėl nepatenka į „aklają zoną“. Robotui esant tarp kliūčių, atstumai iki abiejų kliūčių panašūs, todėl robotas neartėja prie nei vienos iš jų, įgyjamos modifikatoriaus vertės nedidelės.



10.6 pav. Dviejų kliūčių vengimas, kai tarpas tarp kliūčių didesnis, nei roboto plotis

10.2.2. Kelio planavimas sudėtingoje aplinkoje

Robotui esant aplinkoje su daug kliūčių (10.7 pav.), refleksyvinė sistema geba ne tik padėti išvengti roboto matymo lauke atsirandančių kliūčių, bet ir planuoja maršrutą. Nors Braitenbergo algoritmas ir neturi vertinimo metrikos, kuri artintų robotą link tam tikro tikslo, bet jo generuojama kryptis nėra linkusi kisti, jei tiesiai prieš robotą nėra kliūties. Ši charakteristika leidžia robotui naviguoti po aplinką, neužstringant uždaroje kilpoje, t.y. robotas nelinkęs sukti ratus aplink tą pačią kliūtį ar jų grupę.



10.7 pav. Roboto kelias kompleksiškoje aplinkoje

REZULTATAI IR IŠVADOS

Darbe sukurtas patobulintas centriniais sekų generatoriais paremto roboto judesių generavimo algoritmas. Pritaikyta refleksyvinė sistema, atsižvelgianti į robotą supančią aplinką ir gebanti atitinkamai koordinuoti roboto judesius, kad būtų išvengiama susidūrimo su kliūtimis. Taip pat remiantis statinio ir dinaminio stabilumo principais parinkta optimali roboto kojų konfigūracija, leidžianti išlaikyti didelį stabilumą judėjimo metu neaukojant judėjimo greičio.

Rezultatai:

1. Teorinis centrinių sekų generatorių modelis pritaikytas simuliacinėje roboto sistemoje, pasitelkiant aptartus judesių generavimo metodus.
2. Į judesių generavimo modelį integruota adaptyvaus kombinatorinio valdymo sistema, suteikianti galimybę atsižvelgiant į robotą supančią aplinką atitinkamai koordinuoti judesių eigą.
3. Atlikti roboto prototipo bandymai simuliacinėje aplinkoje, patikrinantys roboto gebėjimą išvengti kliūčių bei naviguoti kompleksiškose aplinkose.

Išvados:

1. Pritaikius alternatyvius centrinių sekų generatorių algoritmo patobulinimus, išnaudojamos lanksčios algoritmo charakteristikos dinamiškam judesių, tokių kaip sukimasis aplink fiksuotą tašką, posūkių atlikimas, generavimui.
2. Centrinių sekų generatorių algoritmą papildžius adaptyvaus kombinatorinio valdymo sistema, apjungiančia centrinių sekų generatorių mazgus ir aukštesnio valdymo sistemas, įgalinta dinamiška roboto sąveiką su aplinka.
3. Remiantis dinaminio ir statinio stabilumo principais parinkta optimali roboto kojų konfigūracija, įgalinanti robotą pasiekti santykinai didelį judėjimo greitį neaukojant roboto pusiausvyros. Tai leidžia išbandyti įvairius generuojamus kojų judesius negalvojant apie stabilumą.

LITERATŪROS SĄRAŠAS

[AGK+13] Aparna K., Geeta S., Kirtee K., Madhuri J. CPGs Inspired Adaptive Locomotion Control for Hexapod Robot. *International Journal of Engineering Science Invention*, 2013, No. 21, p. 13-18.

[Bal04] Balasubramanian R. Legless Locomotion: Concept and Analysis. Carnegie Mellon University, 2004, p. 36.

[Bra84] Braitenberg V. Vehicles: Experiments in synthetic psychology. 1984, *Cambridge, MA: MIT Press*, 168 p.

[BSF+15] Bartneck C., Soucy M., Fleuret K., Sandoval E. B. The Robot Engine - Making The Unity 3D Game Engine Work for HRI. *IEEE International Symposium on Robot and Human Interactive Communication*. 2015, p. 431-437.

[CC98] Crook S., Cohen A. *The Book of Genesis*, 1998, New York: Springer New York, p. 458.

[CCB+01] Clark J., Cham J., Bailey S., Froehlich E., Nahata P., Cutkosky M. Biomimetic design and fabrication of a hexapedal running robot. *International Conference on Robotics and Automation*, 2001, p. 3643-3649.

[CCH16] Chang C., Chang C., Hsiao C. Data Collection for Robot Movement Mechanisms in Wireless Sensor and Robot Networks. *International Computer Symposium 2016*, 2016, p. 435-441.

[Cyb19] Webots User Guide: micromagic's Mantis. https://cyberbotics.com/doc/guide/mantis. 2019. 891 KB

[CLG+16] Canio G., Larsen J. C., Worgotter F., Manoonpong P. A combination of central pattern generator-based and reflex-based neural networks for dynamic, adaptive, robust bipedal locomotion. 2016. In Proceedings of the First International Symposium on Swarm Behavior and Bio-Inspired Robotics SWARM. 4 p.

[CLP08] Crespi A., Lachat D., Pasquier A., Ijspeert A. J. Controlling swimming and crawling in a fish robot using a central pattern generator. *Autonomous Robots*, 2008, Vol 25, No 1-2, p. 1-2.

[CP03] Conradt J., Varshavskaya P., Distributed Central Pattern Generator Control for a Serpentine Robot. *CSAIL*, 2003

[FIS14] Floreano D., Ijspeert A. J., Schaal S. Robotics and Neuroscience. *Current Biology*, 2014, Lausanne: Elsavier, No. 24, p. 910-920.

[GM17] Ganono O., Mukundan R. A Framework for Visually Realistic Multi-robot Simulation in Natural Environment. *WSCG 2017 conference*, 2017.

[HAS11] Hernandez-Belmonte U. H, Ayala-Ramirez V., Sanchez-Yanez R. E. A Mobile Robot Simulator Using a Game Development Engine. *Rossum*, 2011.

[HC09] Hatton R., Choset H. Generating gaits for snake robots: annealed chain fitting and keyframe wave extraction. *Autonomous Robots*, 2009, p. 271-281.

[Ijs08] Ijspeert A. J. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 2008, Lausanne: Elsevier, No. 21, p. 643-653.

[Kat16] Katz P. S. Evolution of central pattern generators and rhythmic behaviours. *Philosophical Transactions of Royal Society*, 2016, No. 371.

[KST+07] Kim S., Spenko M., Trujillo S., Heyneman B., Mattoli V., Cutkosky M. R. Whole body adhesion: hierarchical, directional and distributed control of adhesive forces for a climbing robot. *2007 IEEE International Conference on Robotics and Automation*, 2007, p. 1268-1269.

[LDB14] Le B. S., Dang V., Bui T. Swarm Robotics Simulation Using Unity. *Integrated Circuits, Design, and Verification Conference*, Vietnam, 2014.

[MM05] Minguez J., Montano L. Sensor-Based Robot Motion Generation in Unknown, Dynamic and Troublesome Scenarios. *Robotics and Autonomous Systems*, 2005, Zaragoza, Elsavier, Vol 52, No 4, p. 290-311.

[MPW08] Sensor-driven neural control
for omnidirectional locomotion and versatile reactive behaviors of walking machines. *Robotics and Autonomous Systems*, 2008, Elsavier, Vol 56, No 3, p 256-288.

[Poa11] Parts of a Robot. http://www.mind.ilstu.edu/curriculum/medical_robotics/parts_of_robots.php. 2007. 11 KB.

[Rab15] Robots and Biology. *National Science Foundation*. 2015 https://www.nsf.gov/news/special_reports/robotics/robots.jsp.

[RMM16] Razavian R. S., Mehrabi N., McPhee J. A Neuronal Model of Central Pattern Generator to Account for Natural Motion Variation. *Journal of Computational and Nonlinear Dynamics*, 2016, Vol. 11, No 2.

[RMS+15] Ramos O. E., Mansard N., Stasse O., Benazeth C., Hak S., Saab L. Dynamic Whole Body Motion Generation for the Dance of a Humanoid Robot. *IEEE Robotics and Automation Magazine*, 2015, Toulouse, LAAS, p. 10.

[SG11] Staranowicz A., Mariottini G. L. A Survey and Comparison of Commercial and Open-Source Robotic Simulator Software. 2011, *Proceeding* No. 56.

[Sie16] Sieverling A. *Motion Generation*. 2016. http://www.robotics.tu-berlin.de/menue/research/motion_generation/.

[SM12] Santos C. P., Matos V. CPG modulation for navigation and omnidirectional quadruped locomotion. *Robotics and Autonomous Systems*, 2012, Elsavier, Vol 60, No 6, p 912-927.

[SR16] Smashing Robotics. Most Advanced Robotics Simulation Software Overview. https://www.smashingrobotics.com/most-advanced-and-used-robotics-simulation-software/. 150 KB.

[Wal50] Walter W. G. An Imitation of Life. *Scientific American*, 1950, Vol 182, No 5, p. 42-45.

[Woe11] Woering I. R. Simulating the “first steps” of a walking hexapod robot. 2011, *Technische Universiteit Eindhoven*, p 6-11.