Pikosatelliidi ADCS simulatsioonikeskkonna arendamine

Bakalaureusetöö

Üliõpilane: Liisa Kõiv
Üliõpilaskood:213494YAFB
Juhendaja: Ivo Müürsepp,
Thomas Johann Seebecki
Elektroonikainstituut,
vanemlektor
Õppekava: Rakendusfüüsika

Tallinn 2025

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Liisa Kõiv

Töö vastab bakalaureusetööle/magistritööle esitatavatele nõuetele.
Juhendaja: Ivo Müürsepp

Sisukord

Sissejuhatus

Tänapäeva kosmosetehnoloogia muutub ajas aina kallimaks ja keerukamaks, mistõttu üritatakse võimalikult palju vältida juba varases arendusetapis tehtavaid vigu. Iga toodetud ja loodud elemendi jaoks on vaja teha töökindluse kontroll, aga iga sammu testimiseks reaalsetes tingimustes midagi kosmosesse saata oleks liialt keeruline, kulukas ja suure riskiga. Kulude ja riskide vähendamise eesmärgil kasutatakse reaalsete tingimuste loomiseks ja nii riistvara kui tarkvara töökindluse kontrollimiseks erinevaid simulatsiooni keskkondi.

Asendi määramise ja juhtimissüsteem (*Attitude Determination and Control Subsystem*, ADCS) on satelliidi jaoks oluline, kuna see võimaldab täita mitmeid ülesandeid, näiteks suunata antennid või laserid Maa poole, tagada tõrgeteta suhtlus Maaga, viia läbi teaduslikke eksperimente ning sooritada autonoomseid manöövreid. Olemasolevad simulatsioonikeskkonnad on enamasti MATLAB-i ja Simulink-põhised mudelid või C++-il ja Pythonil põhinevad numbrilised lahendajad, mis pakuvad erinevaid täpsus- ja arvutustõhususe tasemeid. Antud mudelitel ja numbrilistel meetoditel on mitmeid erinevusi. Simulink võimaldab modelleerimist plokkskeemide abil ning pakub visuaalset liidest, mille abil saab koostada süsteeme, mis kasutavad erinevaid arvutusmootoreid ja optimeerimisalgoritme, ilma vajaduseta käsitsi teksti lisada, sest seda tehakse graafiliselt kastikeste kujul, mis tähendab ka, et ka andmete liigutamine funktsioonide vahel on graafiline. Numbriliste lahendajate abil luuakse algoritm, mida kasutatakse valitud probleemi numbriliseks lahendamiseks. See aga tähendab, et C++ ja Python nõuavad käsitsi teksti lisamist. MATLAB-i ja Simulinki kasutamine on pigem kiire ja intuitiivne samal ajal kui C++ ja Pyhtoni puhul on suurem õppimiskõver, aga samas võimaldavad nad suuremat kontrolli, optimeeritust ja jõudlust [1], [2]. MATLAB on laialdaselt kasutusel kiireks prototüüpimiseks, kuna see sisaldab sisseehitatud kosmosetehnika tööriistakogusid, samas kui C++ ja Python sobivad paremini suure jõudlusega simulatsioonide ning reaalajas töötava pardatarkvara integreerimiseks. Siiski on igal lähenemisel ka omad puudused, sealhulgas arvutuslikud piirangud, litsentsikulud ja raskused integreerimisel manussüsteemidega.

Varasemalt on ka tudengisatelliidid kasutanud MATLAB-i ADCS-i simuleerimiseks, aga seoses kõrgete kuludega väljaspool ülikooli töötavatele inimestele on seekord otsus luua simulatsioonikeskkond Pythoni baasil, sest see on vabavara ja kõigile kättesaadav, mida MATLAB ei ole. Lisaks võimaldab Python paremat paindlikkust simulatsioonimudelite ülesehitamisel ning kergemat integreerimist kaasaegsete masinõppe ja andmeanalüüsi tööriistadega. Selle otsusega kaasnevad aga mitmed keerukused, sest Pythonis ei ole samal tasemel ja koguses kosmosetehnika teeke nagu on MATLAB-is, mis tähendab, et osa vajalikke elemente ja algoritme tuleb ise kirja panna, mis omakorda tähendab, et töö on kindlasti väljakutsuvam ja ajakulukam kui see oleks MATLAB-i kasutades.

Satelliidi asendi määramine ja juhtimine nõuab mitmete parameetrite täpset simulatsiooni, mille tulemusi võrreldakse võimalusel eksperimentaalsete või emuleeritud andmetega.

Käesoleva töö eesmärk on välja arendada kohandatud ADCS simulatsioonikeskkond, mis võimaldab modelleerida ja analüüsida satelliidi dünaamikat erinevates töörežiimides. Simuleeritavate suuruste hulka kuuluvad:

- Satelliidi asukoht ja orientatsioon ning arvutuslike ja reaalsete tulemuste erinevus ehk viga – see on vajalik, et satelliidiga suhelda, ja teada, kus satelliit orbiidil hetkel asub. Vea

arvutused on kasulikud tulemuste usaldusväärsuse hindamisel ja õige orientatsiooni ning originaal orbiidist kõrvalekalde määramiseks.

- Nurkkiirus ja selle muutus ajas – selle teadmine aitab määrata täiturite, hoorataste ja vajaliku jõumomendi poolt tehtava töö suuruse. Samuti aitab see vältida satelliidi kontrollimatut pöörlemist.
- Kontrollimatu pöörlemise peatamise (*detumbling*) efektiivsus – näitab, kui efektiivne on loodud tarkvara satelliidi stabiilsuse taastamisel ja kas saadud tulemus on usaldusväärne.
- Täiturite poolt genereeritud asendi muutmiseks vajalik jõumoment (*torque*) – muudab satelliidi orientatsiooni, kontrollib ja stabiliseerib satelliidi liikumist
- Väliste häirivate jõudude mõju (õhutakistus, gravitatsioon, Maa magnetväli) – nende teadmine aitab korrigeerida satelliidi käitumist.
- Andurite müra – selle abil saab hinnata, kui palju võivad mõõtmistulemused sensorite tõttu aja jooksul muutuda.

Simulatsioonikeskkonna tulemusi saab kasutada ADCS algoritmide täpsuse ja töökindluse hindamiseks enne pardarakendustes kasutuselevõttu.

Kirjanduse ülevaade

Väikesatelliitide asendi määramise ja juhtimissüsteemi (ADCS) arendamine on oluline satelliidi stabiliseerimiseks, selle suuna ja orientatsiooni määramiseks ning hoidmiseks, Maaga suhtlemiseks ja orbiidil liikumiseks. Nende süsteemide planeerimine ja testimine reaalsetes stsenaariumides on aga kallis, keeruline ja riskantne. On ka elemente, mida on nii väikeste projektide puhul peaaegu võimatu praktikas luua, näiteks kaaluta olek. Mis tähendab, et tegelikult on vaikimisi juba algusest peale kõikide asjade praktikas katsetamine võimatu või ebapraktiline. Selleks võetakse sageli appi simulatsioonikeskkond.

Aja jooksul on ADCS testimine arenenud numbrilistest mudelitest erinevate reaalajas muutuvate platvormideni. Vastavalt võimalustele kasutatakse testimiseks erinevaid keskkondi.

Üks levinumaid ja laialdaselt kasutatavaid keskkondi ADCS testimiseks on MATLAB ja Simulink. MATLABi eeliseks on selle ulatuslikud sisseehitatud tööriistakogud, täpsed numbrilised lahendajad ja lihtne integreeritavus kontrollisüsteemidega [3]. Peamised tööriistad, mis kergendavad ADCS simulatsioone MATLABis, on:

- *Aerospace Toolbox* & *Aerospace Blockset* – sisaldab orbiidi dünaamika modelleerimise tööriistu, koordinaatmuundureid ja lennusimulatsioone.
- *Control System Toolbox* – võimaldab PID, LQR, Kalmani filtrite ja muude juhtimisalgoritmide lihtsat kasutamist.
- Simulink – võimaldab graafilist modelleerimist, mis lihtsustab süsteemi simulatsiooni ülesehitamist.

MATLAB/Simulink on eelistatud variant kiireks prototüüpimiseks ja akadeemilisteks uuringuteks, kuid sellel on ka miinuseid. Nendeks on, näiteks kõrged litsentsikulud ning piiratud võimalused reaalajas süsteemide jaoks.

Alternatiivina MATLABile kasutatakse C++ ja Pythoni põhiseid simulatsioonikeskkondi, mis pakuvad suuremat kiirust, paindlikkust ja reaalajas töötamise võimekust [4]. Peamised eelised nende keelte kasutamisel ADCS simulatsioonis on:

- Kiirem täitmiskiirus ja parem mäluhaldus – võimaldab tõhusamat Monte Carlo simulatsioonide jooksutamist ja keerukamate juhtimissüsteemide testimist.
- Avatud lähtekood ja litsentsivabadus – Python on vabavara, samas kui MATLABi kasutamine nõuab kallist litsentsi.
- Parem integratsioon reaalajas töötavate manussüsteemidega – C++ on sageli satelliitide pardatarkvara peamine keel, mistõttu võimaldab see sujuvat üleminekut simulatsioonist reaalse süsteemi rakendusse.

Siiski on C++/Pythonil põhinevatel simulatsioonidel ka olulisi väljakutseid:

- Puuduvad spetsiaalsed tööriistakogud kosmosetehnikale – erinevalt MATLABist tuleb orbiidi dünaamika ja juhtimissüsteemide mudelid sageli nullist arendada.
- Pikem arendusaeg ja suurem keerukus – süsteemi ülesehitamine nõuab rohkem programmeerimisoskuseid ja süvitsi minekut numbrilisse lahendamisse.

Hoolimata nendest raskustest on Python muutumas üha populaarsemaks akadeemilistes ja tööstuslikes projektides, sest see võimaldab paindlikku integreerimist kaasaegsete masinõppe ja andmeanalüüsi tööriistadega.

Lisaks puhtalt tarkvarapõhistele simulatsioonidele kasutatakse testimisel ka lähenemisi, mis hõlmavad osaliselt reaalse riistvara kasutamist, nagu *Processor-in-the-loop* (PIL) ja *Hardware-in-the-loop* (HIL) [5]. PIL võimaldab reaalsetel pardaprotsessoritel juhtimisalgoritmide testimist, et kontrollida nende ajakriitilist täitmist ja ressursikasutust. Seda kasutatakse tavaliselt enne ADCS tarkvara lõplikku pardaimplementatsiooni, et tagada koodi optimeeritus ja sobivus. Samas HIL ühendab päris riistvara (reaktsioonirattad, magnetväändurid, andurid) simulatsiooniga, et kontrollida, kas satelliidi süsteemid töötavad ootuspäraselt. See lähenemine on kasulik täiturite ja andurite dünaamika testimiseks ning reaalsete häiritud jõudude modelleerimiseks.

Teoreetilised alused

Antud peatükk toob välja ideed, millel põhineb satelliidi asend ja selle kontrollimine.

Satelliidi pöörlemissuuna kirjeldamine

Antud simulatsioonikeskkonnas kasutatakse satelliidi orientatsiooni kirjeldamiseks kvaternioone. Jäiga keha olekut kosmoses kirjeldab asukoht ja orientatsioon, mis kirjeldab satelliidi orientatsiooni valitud koordinaatsüsteemis.

Jäikus on keha võime koormuse all vastu panna kuju ja mõõtmete muutumisele ehk deformatsioonile. Jäigaks kehaks loetakse keha, mille mõõtmed ei deformeeru kui tahes suure jõu korral. Satelliiti saab pidada jäigaks kehaks, sest see on valmistatud materjalidest, mis taluvad kosmosekeskkonnas tekkivaid pingeid ilma olulise deformatsioonita. Kosmoses mõjutavad satelliidi kesta peamiselt gravitatsioonijõud ja võimalikud temperatuurierinevused, aga need ei põhjusta suuri struktuurimuutusi. Satelliidi pidamist jäigaks kehaks võimaldab kasutada jäiga keha dünaamika valemeid, mis lihtsustavad oluliselt orbitaal- ja pöörlemisdünaamika arvutusi.

Asukoha kirjeldamise all peetakse tavaliselt silmas satelliidi massikeskme asukohta kosmoses, selle kirjeldamiseks on olemas erinevad koordinaatide süsteemid nagu [6]:

- ECI koordinaadistik (*Earth-centred inertial coordinate frame*) ehk Maakeskne inertsiaalne koordinaatsüsteem – selle keskmeks on Maa geomeetriline keskpunkt.
- ECEF koordinaadistik (*Earth-centred Earth-fixed*)– on fikseeritud Maapinna suhtes, ehk pöörleb Maaga kaasa.
- Orbitaalraamistik – satelliidi põhine koordinaatsüsteem, mille abil saab satelliidi asendit kirjeldada pöördega satelliidi põhise koordinaatsüsteemi ja taustsüsteemi vahel.

Pöörlemise kirjeldamiseks on samuti mitmeid viise [7]:

- Euleri nurgad – Pöörlemine toimub ümber iga telje ja satelliidi juhtimiseks kombineeritakse need pöörlemised.
- Pööramismaatriksid - 3 × 3 maatriksid, milles iga veerg on ühikvektor piki keha telgi mis on määratletud navigeerimistelgede kaudu.
- Telje-nurga seos (*Axis-Angle Representation*) – ühe raamistiku pöörlemist teise suhtes arvestatakse pöörlemist ümber määratud telje määratud nurga all.
- Kvaternioonid – nelja elemendiga vektorid, mis kirjeldavad keha pöörlemist 3D ruumis.

Satelliidi pöörlemise kirjeldamisel kvaternioonide kasutamisel on mitmeid eeliseid. Võrreldes Euleri nurkadega ei teki singulaarsuseid ehk olukordi, kus pöördenurkade kirjeldus muutub matemaatiliselt määramatuks või arvutuslikult ebastabiilseks. Euleri nurkade kasutamisel võib tekkida „teljestike lukustumine“ (*gimbal lock*) [8]. See juhtub siis, kui üks pöörlemistelgedest langeb kokku teisega, põhjustades vabadusastmete kaotuse. Vabadusastme kaotamine võib osutuda probleemiks, sest antud hetkel ei saa enam ühte pöörlemistelge sõltumatult juhtida, mis omakorda tähendab, et satelliidi orientatsiooni ei ole antud hetkel võimalik määrata ainult Euleri nurkade abil. Teljestike lukustumise korral võib satelliidi orienteerimine olla ebatäpne või isegi kontrolli alt väljuda. Kvaternioonid väldivad singulaarsust selle kaudu, et nad ei kasuta järjestikuseid pöördeid vaid kirjeldavad pöörlemist otse ühe ühikvektori ja nurga kaudu. Kuna kvaternioonid ei kasuta kolme järjestikust pöördtelge ei saa tekkida teljestike kokkulangemisi ja lukustumist. Euleri nurkade ees on samuti eelis arvutamisel, sest arvutamisel ei ole vajalikud trigonomeetrilised funktsioonid ja keerulised

ümberarvutused. Samuti nõuavad kvaternioonid vähem arvutusi kui pööramismaatriksid (4 vs 3x3), mis vähendab süsteemi mälu kasutust ja keerukust. Pööramismaatriksid on küll intuitiivsemad kui kvaternioonid ja selletõttu ka lihtsamad kasutada, kuid nende maatriksite pikaajalisel kasutamisel võivad nad kaotada ortogonaalsuse. Ortogonaalsus on maatriksi omadus, mis näitab, et pööramismaatriksi transponeeritud versioon on võrdne tema pöördmaatriksiga. Ortogonaalsuse tõttu säilitavad pööramismaatriksid kaugused ja nurgad, mistõttu saab neid kasutada pööramiste ja peegelduste kirjeldamiseks. Teisisõnu, see kirjeldab ainult pööramist, ilma et objekt veniks või deformeeruks. Satelliidi asendi määramisel on see oluline, sest kui maatriks kaotab ortogonaalsuse (näiteks väikeste arvutusvigade tõttu), ei kirjelda see enam ainult pööramist. Selle asemel hakkab see justkui „moonutama“ asendit, mis toob kaasa ebatäpsused satelliidi asendi mudelis. Selle vältimiseks tuleb maatriksit regulaarselt korrigeerida, et see jääks kirjeldaks siiski ainult pööramist. Ortogonaalsuse kaotamine võib tekkida näiteks asendi sensorite vigade akumuleerumise tõttu, mille tulemusel asendimaatriks ei kirjelda enam puhast pööramist satelliidi taustsüsteemis, vaid võib hakata moonutama satelliidi suuna- või orientatsioonivektorite transformatsiooni ehk esindama ka venitust või nihkumist kolmemõõtmelises ruumis. Seetõttu tuleb pööramismaatriksit regulaarselt normeerida või ortonormaliseerida, et säilitada selle omadused. Kuna pööramismaatriksid nõuavad rohkem arvutusi kui kvaternioonid, nõuavad rohkem arvutusi ka nende kombinatsioonid ehk järjestikused pööramised mis omakorda teeb need arvutused n-ö kallimaks. Tegelikkuses kasutavad paljud satelliidid kvaternioone ja pööramismaatrikseid koos: kvaternioone kasutatakse pöörlemiste kombineerimiseks ehk sisemiste arvutuste tegemiseks, aga väljundite jaoks teisendatakse need pööramismaatriksiks. See teisendus võib olla vajalik, sest osa andureid, nt güroskoobid, annavad andmeid kas Euleri nurkadena või pööramismaatriksitena mis tähendab, et anduritega suhtlemiseks on lihtsam kasutada pööramismaatriksit.

Töö praktiline sisu

Antud peatükk kirjeldab, millest koosneb satelliidi juhtimine ja kuidas näeb välja juhtimissüsteem.

Kasutatavad algoritmid

B-dot algoritm

B-dot algoritmi kasutatakse satelliidi kontrollimatu pöörlemise vähendamiseks ja kontrollitavuse taastamiseks jõumomendi abil [3]. Kontrollimatu pöörlemise ajal koosneb nurkkiirus kahest komponendist, üks madalal orbitaalsagedusel ja teine kõrgel pöörlemissagedusel. Pöörlemise vähendamiseks üritatakse B-dot algoritmiga vähendada kõrgel pöörlemissagedusel olevat komponenti. Kõige tõhusam viis seda teha on tagada pöördemoment, mis on igal hetkel suuruselt võrdeline ja suunalt vastupidine kosmoselaeva hetkelise nurkkiirusega. Üldjuhul saab kasutada sellise pöördemomendi tekitamiseks magnetväändurit, välja arvatud juhul, kui selle tekitatud magnetväli langeb täpselt kokku välise magnetväljaga, antud juhul Maa magnetväljaga , mille korral vajalikku jõumomenti ei teki.

Nadiiri suunamine

Nadiiri suunamine on juhtimisviis, kus satelliidi valitud külg on kogu aeg suunatud maapinna poole. Nadiiri suund on vektor, mis osutab satelliidilt otse Maa poole ja on maapinnaga risti. Kui satelliit on alati Maa suhtes fikseeritud asendis saab lihtsustada pöörlemise dünaamikat, kuna satelliit saab passiivseks stabiliseerimiseks kasutada Maa gravitatsioonigradienti, vähendades energiakulu ja kontrolli keerukust.

Olekumasin (*State machine*)

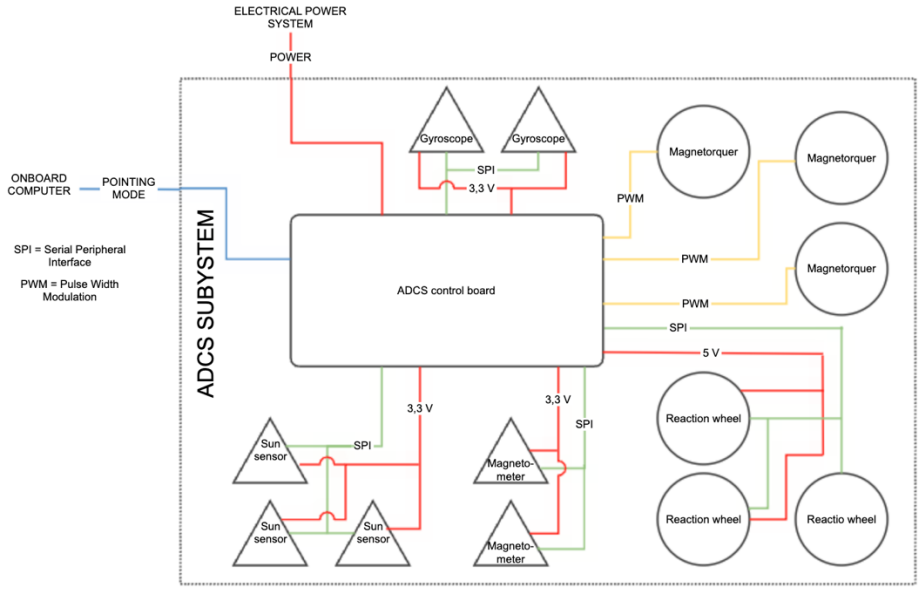
Olekumasin on juhtimisloogika, mida kasutatakse süsteemides, kus on vaja hallata erinevaid olekuid ja nende vahelisi üleminekuid, näiteks tarkvaras, riistvaras ja tööstusautomaatikas. Olekumasin jagab asendi määramise ja juhtimissüsteemi toimimise erinevatesse režiimidesse ning määrab, kuidas satelliit liigub ühest olekust teisele vastavalt sensorite sisenditele.
Olekumasin võimaldab selget režiimide [9] haldust, mis on vajalik, sest satelliidi käitumist peab optimeerima mitmete olukordade jaoks. Samuti aitab olekumasin lihtsustada juhtimist, kuna ADCS saab otsuseid vastu võtta reaalajas ja toimida järk-järgult.

Satelliidi SUTS asendikontrolli režiimid on:

- Ohutusrežiim - vähendab energiatarbimist miinimumini ja taastab baastaseme juhtimise, kasutades sageli magnetväändureid või passiivset stabiliseerimist. Kui satelliit kaotab kontrolli oma orientatsiooni või trajektoori üle või tuvastatakse anomaalia, lülitub see automaatselt ohutusseisundisse.
- Asendi stabiliseerimise režiim (*Detumble mode*) - kui satelliit eraldub raketist, võib selle pöörlemiskiirus olla liiga suur. Sageli kasutatakse siis satelliidi pöörlemise kontrollimiseks magnetväändureid ja reaktsioonirattaid.
- Orientatsiooni saavutamise režiim (*Acquisition Mode*) - satelliit püüab leida õige orientatsiooni Maa või Päikese suhtes, kasutades päikesesensoreid ja magnetomeetreid.
- Nadiiri suunamine – satelliit hoiab ühte külge suunatud Maa keskpunkti poole, et tagada andmete kogumine või signaali suunamine õigesse kohta.

ADCS tööpõhimõte

ADCS-i toimimiseks on vajalikud nii andurid, täiturid, juhtimisalgoritmid kui ka elektritoide. ADCS-i juhtplaat on peamine juhtplokk mis koordineerib nii andmete töötlemist kui ka täiturite juhtimist.



Joonis 1 Satelliidi SUTS ADCS-i ülesehitus

ADCS saab sisendina andmeid mitmetelt sensoritelt:

- güroskoobid mõõdavad satelliidi nurkkiirust;
- magnetomeetrid määravad Maa magnetvälja suunda, aidates hinnata satelliidi orientatsiooni viimase suhtes;
- päikesesensorid tuvastavad Päikese asukoha, andes olulist teavet satelliidi suuna määramiseks.

Need sensorid edastavad andmeid SPI-liidese (*Serial Peripheral Interface*) kaudu ADCS juhtplaadile, kus toimub orientatsiooni arvutamine ja stabiliseerimisotsuste tegemine. SPI on sünkroonse jadaliidese standard, mida kasutatakse enamasti andmete ülekandeks väikeste vahemaade taha ning mis leiab sageli kasutust manussüsteemides. [10]

Kui süsteem tuvastab, et satelliidi orientatsioon vajab korrigeerimist, aktiveeritakse täiturid:

- hoorattad kasutavad pöördemomendi muutmist, et täpselt kontrollida satelliidi asendit.
- magnetväändurid loovad magnetvälja mille vastastikmõju Maa magnetväljaga tekitab jõumomendi, mis hakkab mõjutama satelliidi asendit. Juhtimine toimub impulsslaiusmodulatsiooni abil, kus reguleeritakse nelinurkse juhtsignaali töötsüklit, mis määrab, kui suure osa ajast on signaal kõrgel pingetasemel ühe tsükli jooksul. Kuigi signaal on kujult binaarne, ei kanta sellega üle digitaalset infot, vaid edastatakse analoogset juhtimisinfot kestuse kaudu. Impulsslaiusmodulatsioon on signaali juhtimismeetod, mida

kasutatakse võimsuse reguleerimiseks elektroonikas. See põhineb signaali impulsside kestvuse muutmisel, hoides samal ajal signaali sageduse konstantsena.

ADCS-i juhtalgoritm jälgib pidevalt satelliidi orientatsiooni ja teeb vajadusel korrektuure. Kogu protsessi juhib pardaarvuti, mis saadab ADCS-i juhtplaadile käske, näiteks "*pointing mode*", määrates sihtsuuna või stabiliseerimisrežiimi.

Tänu nendele komponentidele suudab ADCS hoida satelliidi soovitud orientatsioonis, vähendada soovimatut pöörlemist ning võimaldada täpset suunamist teadusmissioonide, side- või navigatsioonisüsteemide jaoks.

Jäiga keha ühemõõtmelise liikumise kirjeldamine

Lihtsustatud lähenemisel (ühemõõtmeline liikumine), kui ei ole vaja arvestada keha pöörlemisega, jõu jaotumisega ning keha mõõtmed on liikumise ulatusega võrreldes tühised, võib jäika keha käsitleda kui punktmassi.

Oletame, et meil on punktmass m, mis saab liikuda ühes dimensioonis, mööda sirgjoonelist telge (näiteks x-telge). Punktmassi liikumist kirjeldavad kaks peamist suurust:

- asukoht $x[m]$ – keha kaugus mingist nullpunktist
- kiirus $v[\frac{m}{s}]$ – keha liikumise kiirus mööda x-telge.

Keha liikumist saame mõjutada rakendades välist jõudu F. Oluline on aga see, et rakendatav jõud on piiratud ehk see ei saa ületada mingisugust maksimaalset väärtust F_{max}.

Ülesandeks on nihutada kirjeldatud punktmass algpunktist x_0 sihtkohta x_1, kasutades võimalikult optimaalset juhtimist. Optimaalne juhtimine antud ülesandes on defineeritud kui minimaalne üle reguleerimine ja jõu kasutamine võimalikult efektiivselt. See tähendab, et:

- peame rakendama jõudu nii, et keha hakkaks liikuma soovitud sihtkoha suunas,
- mingil hetkel peame rakendama vastupidises suunas jõudu, et keha õigel ajal pidurdada ja sihtkohas peatada.

Punktmassi liikumist kirjeldab Newtoni II seadus:

$$m\frac{dv}{dt} = F \tag{1}$$

ehk

$$\frac{dv}{dt} = \frac{F}{m}. \tag{2}$$

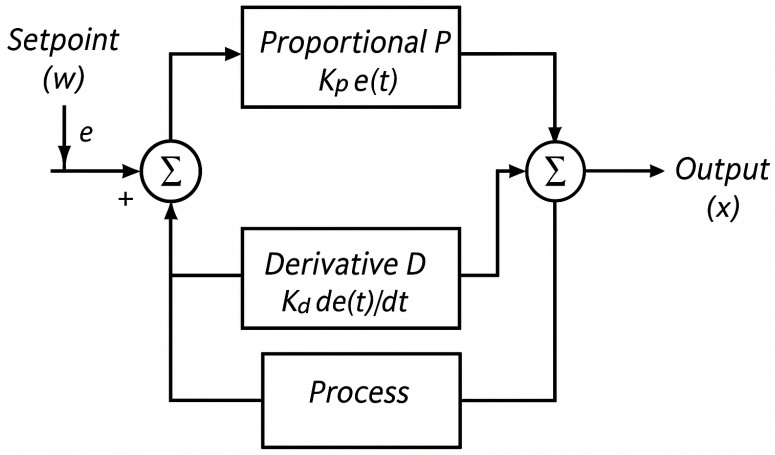
Kiirus ja asukoht on omavahel seotud võrrandiga

$$\frac{dx}{dt} = v. \tag{3}$$

Seega on tegemist teist järku diferentsiaalvõrrandiga, mille lahendamiseks peame määrama sobiva funktsiooni $F(t)$.

Kuna meie eesmärk on punktmassi võimalikult optimaalselt juhtida, peaksime kasutama mingisugust optimeerimisstrateegiat [11], näiteks:

- Bang-bang juhtimine (*on-off* kontroll, maksimaalse jõu kontroll) – kõige lihtsam juhtimismeetod, kus süsteem rakendab alati maksimaalset võimalikku jõudu ühes või teises suunas, sõltuvalt asukoha veast ja liikumise suunast. Rakendame alguses maksimaalset võimalikku jõudu F_{max}, et keha kiiresti kiirendada. Poole tee peal rakendame vastupidist jõudu $-F_{max}$, et punktmassi aeglustada ja peatada sihtpunktis x_1. See on küll lihtsaim ja kiireim lahendus, kuid see võib viia järskude liikumisteni ja võimaliku üleminekuni sihtpunktis.
- Proportsionaal-diferentsiaal juhtimine (PD) – töötab põhimõttel, et kui keha on sihtpunktist kaugel rakendatakse suurt jõudu sihtkoha poole. Kui keha liigub väga kiiresti sihtkoha poole vähendatakse jõudu, et vältida liiga suurt hoogu. Kui keha jõuab sihtkoha lähedale, siis proportsionaalne jõud väheneb järk-järgult samal ajal üritades ära hoida liigset võnkumist. PID - kontrollerit illustreerib joonis 2, kus
  - *w* on süsteemi soovitud väärtus ehk sihtväärtus,
  - *x* on süsteemi tegelik väljund, mida mõõdetakse ja võrreldakse sihtväärtusega,
  - *e* on viga ehk erinevus sihtväärtuse ja tegeliku väljundi vahel,
  - K_p on proportsionaalne komponent, mis arvutab juhtsignaali osa, mis on võrdeline veaga,
  - K_d on diferentsiaalkomponent, mis arvutab juhtsignaali osa, mis on võrdeline vea muutumiskiirusega ajas,
  - *process* esindab juhitavat süsteemi.



Joonis 2 PD kontrolleri plokk-skeem

PID juhtimine kasutab funktsiooni

$$F = K_p(x_1 - x) - K_d v, \tag{4}$$

kus:

$x_1 - x$ – näitab kui kaugel on keha sihtkohast

$v -$ kiirus, millega keha liigub sihtkohale lähemale või kaugemale

Proportsionaal-diferentsiaal kontrolleri kasutamine tagab sujuvama liikumise, kuid nõuab õigete K_p ja K_d parameetrite valimist.

Proportsionaal-diferentsiaalkontroller (PD kontroller)

PD-kontroller (Proportsionaal-Diferentsiaalne kontroller) on juhtelektroonikas ja juhtsüsteemides kasutatav algoritm, mille eesmärk on juhtida süsteemi olekut soovitud väärtuseni. PD kontrolleri kasutamise eesmärk on tõsta süsteemi stabiilsust, parandades juhtimist kuna sellel on võime ennustada tulevast viga vastavalt süsteemi reaktsioonile [13]. See koosneb kahest komponendist:

P-proportsionaalne komponent: reageerib veale ehk tegeliku ja soovitud oleku erinevusele. Mida suurem on vahe kahe oleku vahel, seda tugevam on proportsionaalse komponendi signaal. Määrab ära, kui tugevalt süsteem veale reageerib

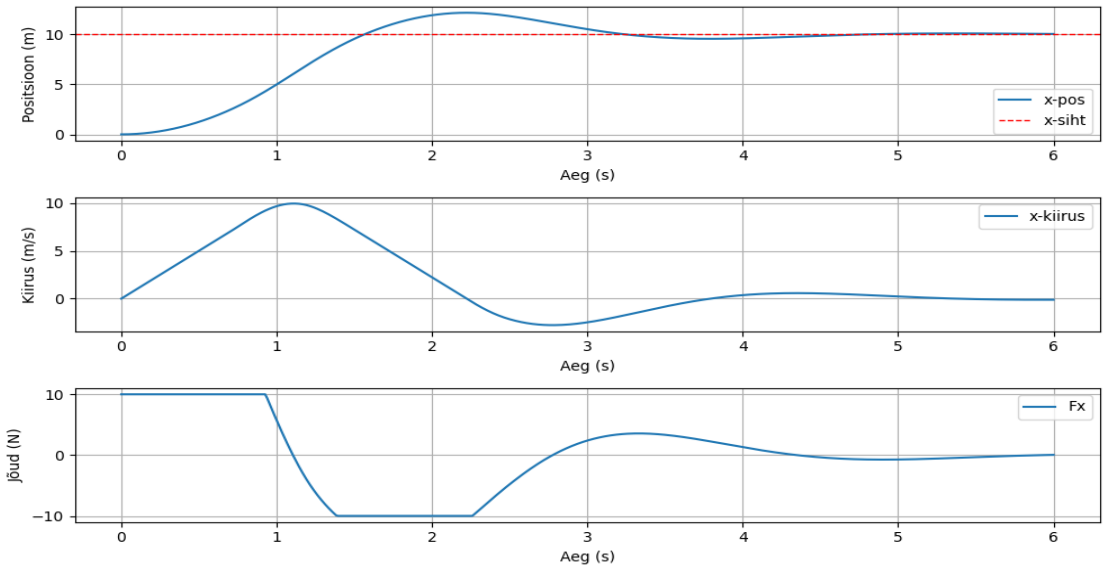
D-diferentsiaalne komponent: tekitab signaali, mis on võrdeline vea muutumise kiirusega ehk vea tuletisega aja suhtes. Parameeter K_d määrab, kui tugevalt süsteem reageerib vea muutumisele, aidates seeläbi vähendada ülemäärast võnkumist ja parandada süsteemi stabiilsust. Aitab pidurdada ja stabiliseerida liikumist enne kui viga nulli jõuab

Antud süsteemis on kasutatud PD kontrollerit PID asemel, sest I-osa parandab püsiviga, aga süsteemile pole antud ülesandes püsiviga ega väliseid häireid lisatud ehk PI või PID kontroller oleks ülearune.

Kui kasutada punktmassi optimeeritud liigutamiseks PD juhtimist ja vastavaid algtingimusi ning PD juhtimise parameetreid:

$$m = 1\ [kg]$$
$$x_0 = 0$$
$$x_1 = 10\ [m]$$
$$F_{max} = 10\ [N]$$
$$K_p = 5$$
$$K_d = 2$$

tuleksid vastavad asukoha, kiiruse ja rakendatud jõu graafikud sellised nagu on kujutatud alloleval joonisel 2.



Joonis 3 Punktmassi ühesuunalise sirgjoonelise liikumise kontrollimine PD juhtimisega

Positsiooni graafik $x(t)$ näitab, et punktmass alustab liikumist algpunktist ja liigub sihtpositsiooni x_1 suunas. Alguses tekib ülevõnge, mis on PD kontrollerile omane. Lõpuks on näha, et süsteem stabiliseerub asukohas x_1.

Kiiruse $v(t)$ graafikul on näha, et alguses kasvab kiirus järsult kuna rakendatakse maksimaalset jõudu. Umbes 2 sekundi juures hakkab kiirus kahanema ja muutub negatiivseks, et vältida sihtkohast üleliikumist. Peale väikest võnkumist kiirus stabiliseerub.

Rakendatud jõu $F(t)$ graafik näitab, et alguses rakendatakse maksimaalset jõudu, et punktmass kiiresti liikuma saada. Seejärel pöörub jõud negatiivseks, et punktmassi hoogu pidurdada ning hiljem jääb mõjuv jõud aina väiksemaks, et positsiooni korrigeerida.

PD simulatsioon on teostatud Pythoni keskkonnas , modelleerides punktmassi liikumist sirgjoonelisel trajektooril PD-kontrolleri abil. Simulatsiooni eesmärk on viia objekt sihtpositsioonile **x**, kasutades jõu rakendamist vastavalt positsiooniveale ja liikumiskiirusele. PD kontroller reguleerib objekti liikumist võttes arvesse kahte tegurit: proportsionaalne komponent (vea suurus siht- ja hetkeasukoha vahel) ja diferentsiaalne komponent (keha hetk-kiirus). Antud simulatsioonis lahendatakse süsteemi dünaamikat kirjeldavad diferentsiaalvõrrandid solve_ivp funktsiooni abil, mis kasutab vaikimisi RK45 meetodit (Runge-Kutta meetod neljanda ja viienda järgu kombinatsiooniga). Funktsioon solve_ivp (*solve initial value problem*) võimaldab algväärtusülesandena ehk ülesandena, kus on teada funktsiooni väärtus alghetkel. Kasutaja defineerib võrrandid funktsiooni kujul:

$$\frac{dy}{dt} = f(t, y), \tag{5}$$

algväärtused ja ajaintervalli, mille jooksul lahendust soovitakse. RK45 on numbriline meetod, mida kasutatakse diferentsiaalvõrrandite ligikaudseks lahendamiseks. Selle eesmärk on hinnata lahenduse täpsust jooksvalt ehk meetod arvutab iga sammu kohta kaks erineva täpsusega lahendit (4. ja 5. järgu), nende erinevuse põhjal hinnatakse viga ja seejärel kohandatakse ajasammu, näiteks kui viga on liiga suur, siis tehakse samm väiksemaks. RK45 võimaldab süsteemi oleku arvutamist aja jooksul diskreetsete sammudena vastavalt oma määratlusele. Parameetrid K_p ja K_d määravad, kui agressiivselt süsteem püüab sihtpunkti jõuda. Kui proportsionaalne võimendus (K_p) on liiga madal, jõuab objekt sihtpunkti väga aeglaselt. Ja vastupidi, kui K_p on liiga kõrge, võib süsteem hakata

võnkuma sihtpunkti ümber. Liiga madal diferentsiaalne võimendus (K_d) tähendab, et süsteem võib sihtpunktist mööda sõita ja liiga suur võimendus võib süsteemi muuta liiga aeglaseks. Praeguses simulatsioonis piirame jõudu vahemikus $-10\,[N]$ kuni $10\,[N]$. Jõu piiramine on vajalik, sest reaalsuses on saadaolev jõud alati piiratud ja ilma piiranguta võib PD-kontroller tahta rakendada ebarealistlikult suuri jõude. Kui F_{max} on liiga väike, võib süsteem sihtpunkti jõudmiseks liiga kaua aega võtta. Või vastupidi, kui F_{max} on liiga suur, võib süsteem olla väga kiire, aga võivad tekkida ülevõnked. PD kontrolleri parameetrite seadistamiseks on mitu viisi:

- Väikese süsteemi puhul võib saada sobiva tulemuse katsetamise teel. Saab suurendada K_p-d kuni süsteem liigub kiiresti, aga võnkumata, sihtmärgi suunas. Vähendada K_d-d kui liikumine muutub liiga aeglaseks ja suurendada kui tekivad võnkumised või liiga suur ülevise ehk soovitud sihtväärtusest minnakse üle.
- Ziegler-Nichols häälestamine [14]. Eeldab, et süsteemil on lubatud võnkuda ja töötab kõige paremini, kui süsteemi dünaamika ei ole teada. Seadistatakse ainult proportsionaalne kontroller, ehk on ainult parameeter K_p.Viimast suurendatakse seni kuni süsteem hakkab võnkuma kindla amplituudiga, selle abil saab leida Zieler-Nicholsi häälestamise jaoks vajalikud parameetrid: K_u (*ultimate gain* – viimane stabiilne $K_p$väärtus enne ebastabiilsust) ja T_u(võnkeperiood). Nende parameetrite abil saab arvutada süsteemile sobivaimad K_p ja K_d väärtused valemitega:

$$K_p = 0{,}8K_u\,, \tag{5}$$

$$K_d = 0{,}125T_u\,. \tag{6}$$

- Mudelipõhine optimeerimine (nt LQR – *Linear Quadratic Regulator*, PID *autotune* meetodid nt ITAE - *Integral of Time-weighted Absolute Error* , ISE - *Integral of Squared Error*). Antud süsteemis töötaks mudelipõhine optimeerimine väga hästi, sest süsteemi dünaamika on teada (määratud Newtoni II seadusega), antud süsteem on lineaarne, mistõttu saab kontrolleri disainida matemaatiliselt optimaalselt, ilma katse-eksituse häälestuseta, lisaks saab arvestada jõupiiranguid.

Pööramine ümber ühe telje

Modifitseerime sama ülesannet ümber ühe telje pööramise jaoks.

Pöördliikumise korral kirjeldame süsteemi (keha) olekut selle pöördenurga, nurkkiiruse ja nurkkiirenduse kaudu. Siiski võib lihtsustatud lähenemisel (ühemõõtmeline pööramine ümber ühe telje) käsitleda vaadeldavat süsteemi kui jäika keha, mille pöörlemist määrab inertsimoment ja sellele rakendatud väline moment. Kuna punktmassil ei ole teda läbiva telje sutes inertsimomenti, siis võib võtta vaadeldavaks süsteemiks näiteks varda, mille pikkus ja mass on võrdsed satelliidi omaga, ehk näiteks 15cm. Selle pöörlemist kirjeldavad kolm suurust:

- pöördenurk $\varphi\ [rad]$ – keha pöördenurk polaartelje suhtes
- nurkkiirus $\omega\ [\frac{rad}{s}]$ – keha pöörlemise nurk kiirus
- nurkkiirendus $\alpha\ [\frac{rad}{s^2}]$ – nurkkiiruse muutumise kiirus.

Keha pööramist saame mõjutada, rakendades pöördemomenti M, kuid selle maksimaalne väärtus on piiratud väärtusega M_{max}. See tähendab seda, et:

- rakendatav moment Mpeab alguses keha pöörama soovitud suunas;
- mingil hetkel tuleb rakendada vastupidist momenti, et keha õigel ajal pidurdada ja soovitud pöördenurgal peatuda;

- süsteemil ei lubata täispööret teha – eeldame, et φ jääb vahemikku $[-\pi;\ \pi]$.

Sirgjoonelisel liikumisel oli liikumine kirjeldatav valemiga (1). Kuna nüüd käsitleme pöörlemist, asendub see pöördemomendi võrrandiga:

$$M = I\alpha, \tag{7}$$

kus:
M – rakendatud pöördemoment [Nm],
I – keha inertsimoment [$\mathrm{kg} \cdot \mathrm{m}^2$],
α – nurkkiirendus [$\frac{\mathrm{rad}}{\mathrm{s}^2}$].
Inertsimoment iseloomustab keha võimet vastu seista nurkkiirendusele. Kui süsteemiks valitud varras pöörleb ümber oma keskpunkti, saab inertsimomenti arvutada valemiga:

$$I = \frac{1}{12}ML^2, \tag{8}$$

kus:
M – varda mass $[kg]$,
L – varda pikkus $[m]$.

Lihtsustame antud ülesannet ja eeldame, et punktmass liigub fikseeritud kaugusel pöörlemisteljest, ehk inertsimoment on konstantne.

Pöördenurk ja nurkkiirus muutuvad ajas vastavalt võrranditele

$$\frac{d\varphi}{dt} = \omega, \tag{9}$$

$$\frac{d\omega}{dt} = \alpha\ . \tag{10}$$

PD kontroller kasutab pöördemomenti M analoogselt sirgjoonelisele liikumisele rakendatud jõule:

$$M = K_p(\varphi_1 - \varphi) - K_d\omega. \tag{11}$$

Lisaks on vaja piirata maksimaalset pöördemomenti:

$$M = clip(M, -M_{max}, M_{max}). \tag{12}$$

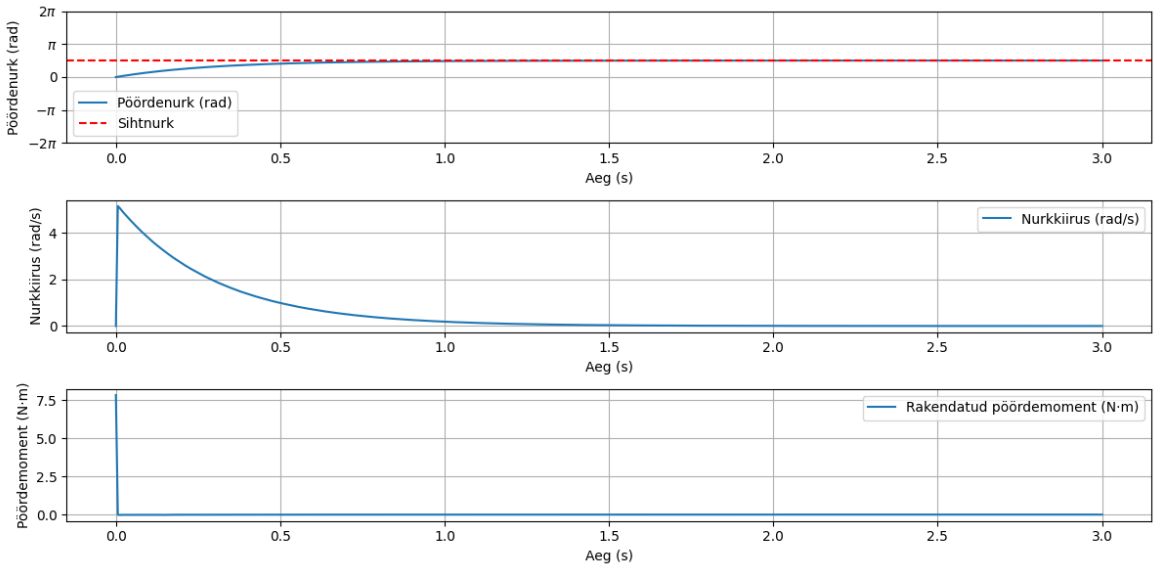
Kui maksimaalset pöördemomenti ei piirata võib juhtuda, et süsteem nõuab suuremat pöördemomenti, kui tegelikult võimalik on ning PD kontroller ei tööta nii nagu peaks. Lisaks võimaldab pöördemomendi kontrollimine sujuvamat liikumist ja kontrolli punktmassi üle. Liiga suur pöördemoment võib põhjustada järske ja ettearvamatuid liikumisi, mis võivad olla süsteemile kahjulikud või muuta juhtimise ebastabiilseks.

Kui antud ülesandes kasutada punktmassi optimeeritud liigutamiseks PD juhtimist ja vastavaid algtingimusi ning PD juhtimise parameetreid:

$$M = 0,75\ [\mathrm{kg}]$$
$$L = 0,15\ [\mathrm{m}]$$
$$\varphi_0 = 0\ [\mathrm{rad}]$$
$$\varphi_1 = \frac{\pi}{2}\ [\mathrm{rad}]$$
$$M_{max} = 7{,}85\ [Nm]$$
$$K_p = 5$$

$$K_d = 1{,}5$$

tuleksid vastavad pöördenurga, nurkkiiruse ja rakendatud inertsimomendi graafikud sellised nagu on kujutatud alloleval joonisel 3.



Joonis 4 Punktmassi pöörlemise ümber ühe telje kontrollimine PD kontrolleriga

Pöördenurga $\varphi(t)$ graafik näitab, et satelliit alustab pöörlemist nullist ja liigub sihtnurga φ_1 suunas. Alguses tekib ületõus, mis on PD-kontrollerile omane. See tähendab, et punktmass ületab hetkeks soovitud pöördenurga enne, kui hakkab stabiliseeruma. Lõpuks pöördenurk stabiliseerub soovitud väärtuse φ_1 juures.

Nurkkiiruse $\omega(t)$ graafikul on näha, et alguses kasvab nurkkiirus järsult, kuna kontroller rakendab maksimaalset pöördemomenti. Umbes 2 sekundi juures hakkab kiirus vähenema ja muutub hetkeks negatiivseks, et vältida sihtnurgast üleliikumist. Peale väikest võnkumist nurgakiirus stabiliseerub nulli lähedal, mis näitab, et satelliit jääb sihtnurka paigale.

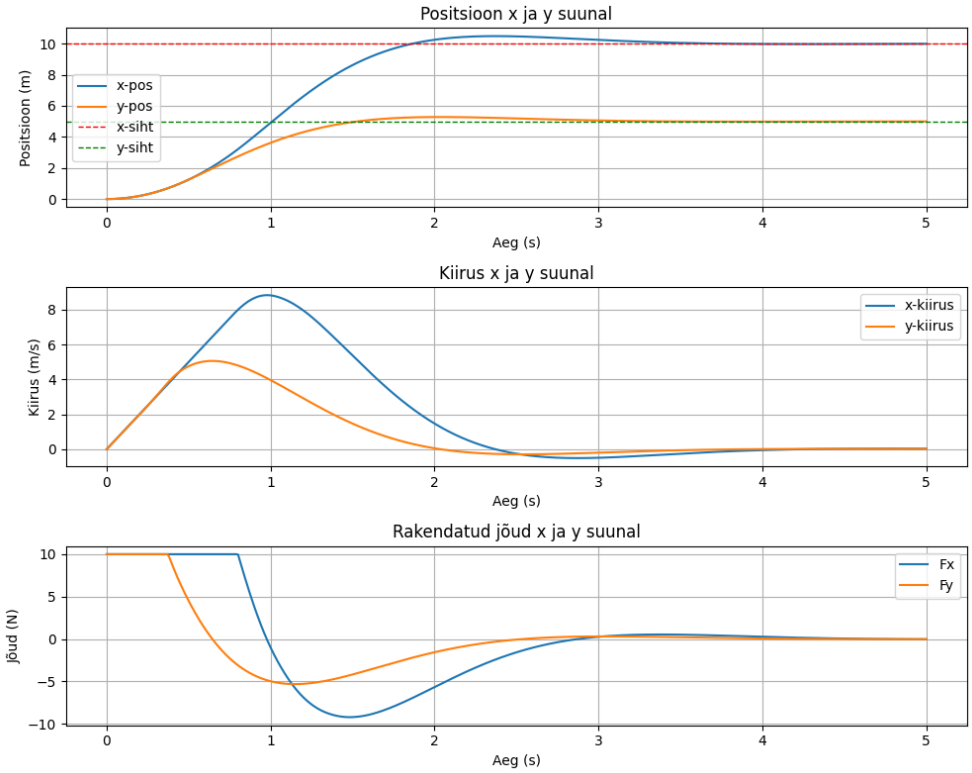
Rakendatud pöördemomendi $M(t)$ graafik näitab, et alguses rakendatakse maksimaalset pöördemomenti, et punktmass kiiresti pöörlema saada. Seejärel pöördub pöördemoment vastupidiseks, et vähendada satelliidi pöörlemiskiirust ja vältida ületõusu. Hiljem jääb pöördemoment väiksemaks, aidates süsteemil sihtnurga ümber stabiliseeruda.

Sirgjooneline liikumine tasandil

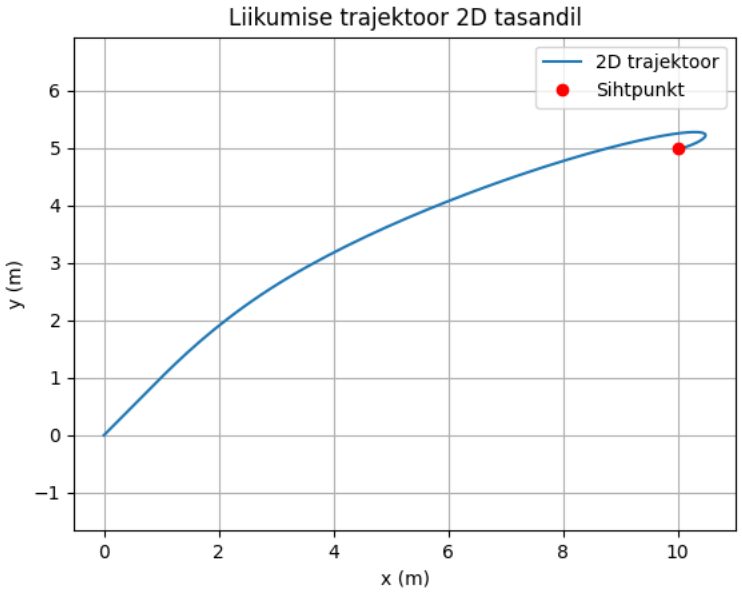
Analoogselt üheteljelisele liikumisele saab simulatsiooni edasi arendada kahe ja kolme telje jaoks. Suurimaks muudatuseks on enamike sisendandmete muutmine skalaaridest vektoriteks, sest nüüd on olemas vähemalt kaks koordinaati ehk selle kohta võiks öelda ka dimensioonide laiendamine, sest kõik dünaamika võrrandid ja kontrolleri rakendamine jäid idee poolest samaks. Kui panna punktmass liikuma punktist A(0;0) punkti B(10;5) ja anda ette PD kontrolleri parameetrid

$$K_{px}, K_{dx} = 5$$
$$K_{py}, K_{dy} = 3,$$

tuleksid punktmassile vastava telje sihilise positsiooni, kiiruse, rakendatud jõu ja trajektoori graafikud sellised, nagu on kujutatud joonistel 5 ja 6.



Joonis 5 Punktmassi kontrollitud liikumine kahe telje sihis



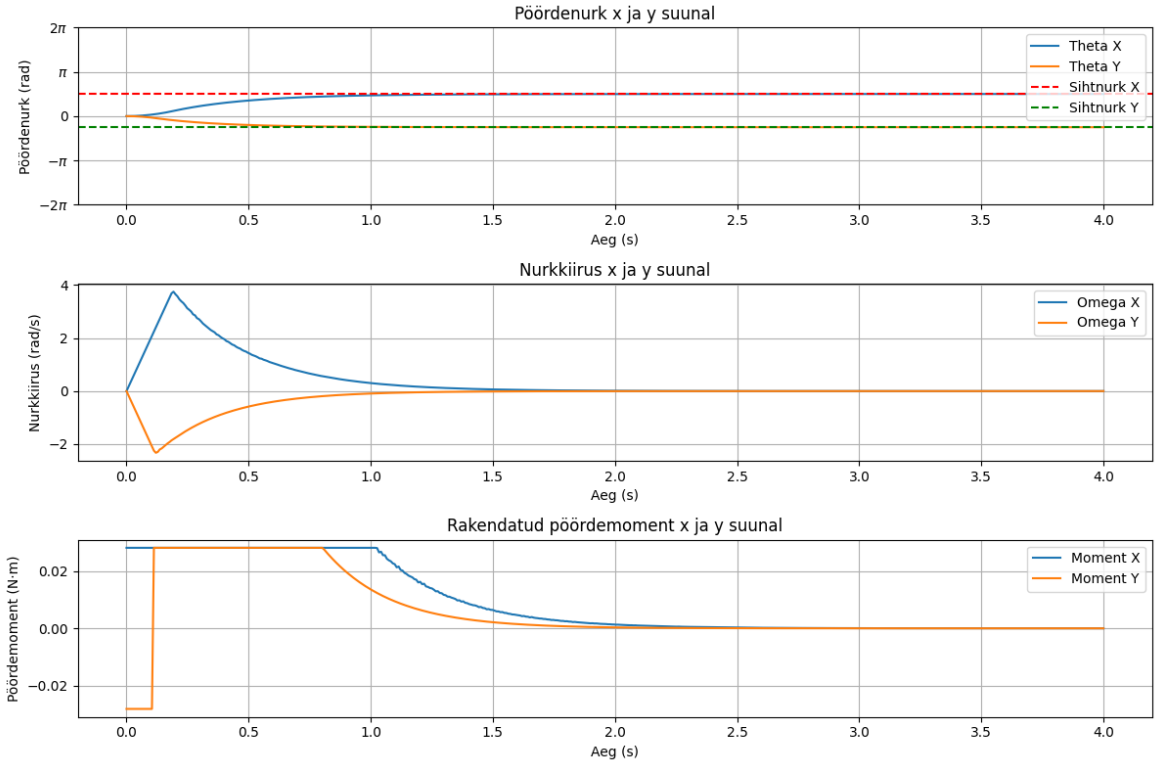
Joonis 6 Punktmassi trajektoor xy-tasandil

Positsiooni graafikul on jällegi näha, et alguses kiirustatakse sihtpunkti suunas ja seejärel liigutakse sellest veidi mööda. Seejärel korrigeeritakse mõjuva jõu suunda ja selle kaudu ka punktmassi liikumise suunda ning jõutakse sihtpunkti. Kiirus algab nullist ja kasvab järsult kuni sihtpunkt

ületatakse. Seejärel kiirus kahaneb ja pöördub negatiivseks, et punktmassi asukohta korrigeerida. Lõpuks kiirus kahaneb aja jooksul ja stabiliseerub nulli juures, mis tähendab, et punktmass peatus sihtpunktis. Trajektoori graafik näitab, et liikumine on sujuv kaar, sest mõlema telje jaoks on eraldi PD kontroller. Samuti on sellel graafikul näha ka sihtpunkti ületamine ja tagasipöördumine

Pööramine ümber kahe ristuva telje

Analoogselt üheteljelisele pöörlemisele saab simulatsiooni edasi arendada kahe ja kolme telje jaoks. Kui satelliidil on võimalik pöörata ümber mitme telje, tuleb iga telje sihilist pööramist eraldi kontrollida, et saada täpne ja stabiilne juhtimine. Selle jaoks tuleb jällegi dimensioone laiendada, sest iga telje sihis võib satelliidi dünaamika olla erinev näiteks vastavalt massijaotusele võivadb olla erinevad nii inertsimomendid kui ka mõjuvad jõud. Antud simulatsioonis, kui varras on alghetkel algtingimustega $\varphi_x, \varphi_y = 0,0$; $\omega_x, \omega_y = 0$. Sihtnurkadeks on $\varphi_x, \varphi_y = \frac{\pi}{2}; -\frac{\pi}{4}$ ning teljed x ja y on omavahel risti ja läbivad varda keskpunkti, tulevad varda pöördenurga, nurkkiiruse ja rakendatud pöördemomendi graafikud sellised, nagu on kujutatud joonisel 7.



Joonis 7 Varda kontrollitud pöörlemine ümber kahe telje

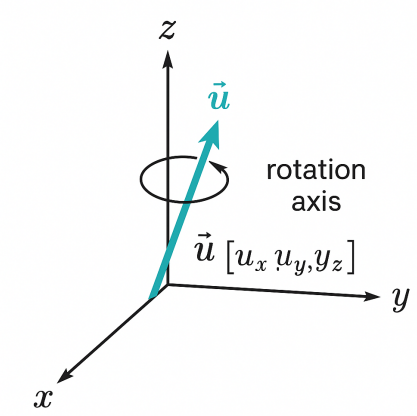
Alguses süsteem kiireneb ja liigub sihtnurkade poole. ω_x alustab nullist ja saavutab maksimumkiiruse ca 3,5 $\frac{rad}{s}$, seejärel väheneb ja läheneb nullile. ω_y käitub sarnaselt, kuid väiksema maksimaalse kiirusega, sest soovitud muutus algse ja sihtnurga vahel on väiksem. Pöördemoment on simulatsiooni alguses suur, et keha pöörlemist kiirendada. Seejärel kontroller muudab momenti vastavalt erinevusele sihtnurga ja hetkenurga vahel, et pidurdada. Pöörlemise lõpupoole moment peaaegu kaob, sest süsteem on stabiilne ja ei vaja enam lisajõudu.

Satelliidi asendi kirjeldamine kvaternioonide kaudu

Kvaternionid on kompleksarvude laiendus neljamõõtmelisse ruumi, mida kasutatakse pöörete kirjeldamiseks kolmemõõtmelises ruumis [15]. Kvaternioone kujutatakse kujul:

$$\vec{q} = [q_0, q_1, q_2, q_3] = \left[\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right) \cdot \mathrm{u_x}, \sin\left(\frac{\theta}{2}\right) \cdot \mathrm{u_y}, \sin\left(\frac{\theta}{2}\right) \cdot \mathrm{u_z}\right],$$

kus θ on pöördenurk ja $\hat{u} = [u_x, u_y, u_z]$ on pöördetelje ühikvektor, mis kirjeldab pöörlemise suunda kolmemõõtmelises ruumis. Joonisel 8 on kujutatud ühikvektor, kui pöörlemine toimub ümber z-telje.



Joonis 8 ühikvektor 3D ruumis

Kui on olemas soovitud- ja hetkeorientatsioon kvaternioonide kujul, saab välja arvutada pöördemomendi või nurkkiiruse, mida satelliidi juhtimissüsteem, näiteks PD-kontroller, kasutab pööramiseks.

Satelliidi liikumise optimeerimisel saab sõltuvalt süsteemi eesmärkidest ja piirangutest optimeerida mitmesuguseid suurusi, näiteks:

- pöördenurk - väikseima vajaliku nurga leidmine satelliidi pööramiseks, mis tavaliselt aitab vähendada pöörlemise kestust ja energiakulu;
- aeg – võimalikult kiire jõudmine soovitud asendisse, näiteks peab satelliidi suunama kindlale objektile mõne sekundi jooksul;
- energiakulu - paljudel satelliitidel on piiratud energiavaru Kuna mitmed satelliidi komponendid tarbivad voolu, siis energiasäästlik juhtimine võimaldab missiooni kestust pikendada ja paremat termilist stabiilsust, sest vähem töötavad täiturid toodavad vähem soojust ja ei põhjusta seadmes lokaalset ülekuumenemist;
- vibratsioonide minimeerimine - järsud pöörded võivad kahjustada sensoreid, kaameraid või mehhanisme näiteks avanevaid antenne.

Pöördenurga optimeerimine

Pöördenurga optimeerimist võib defineerida kui kvaternioonivea kaudu alati väikseima pöördenurgaga trajektoori valimine soovitud orientatsioonini, kus tulemuseks on kiire ja energiasäästlik manööver, mis väldib ebavajalikke pöördeid [15].

Satelliidi olekut saab esitada kvaterniooniga $\mathrm{q} = [\mathrm{q_0}, \mathrm{q_1}, \mathrm{q_2}, \mathrm{q_3}]$, kus q_0 on kvaterniooni skalaarosa $\cos\left(\frac{\theta}{2}\right)$, mis näitab pöördenurga suurust ja $[q_1, q_2, q_3]$ on vektorosa $\sin\left(\frac{\theta}{2}\right) \cdot \hat{u}$, mis määrab

pöörlemise suuna ning kus û on pöördetelje ühikvektor. Kui on olemas soovitav orientatsioon, on võimalik kvaternioonide vaheline viga leida valemiga:

$$q_e = q^{-1} \otimes q_d, \tag{13}$$

kus q^{-1}on olemasoleva orientatsiooni pöördväärtus, mis kvaternioonide puhul on defineeritud kui kvaternioon, mida algse kvaterniooniga korrutades on tulemuseks ühikkvaternioon, q_d on soovitav orientatsioon ja $\otimes$ tähistab kvaternioonide korrutamist ehk Hamiltoni korrutist . Kui kvaternioonide vaheline viga on leitud, saab välja arvutada vajaliku pöördenurga:

$$\theta = 2 \cdot \arccos(q_{e0}), \tag{14}$$

kus q_{e0} on veakvaterniooni skalaarkomponent. Pöördenurga optimeerimiseks peab süsteem hoidma $q_{e0} < 0$, et pöördenurk oleks väiksem kui π. Kvaternioonipõhise juhtimise korral võimaldab pöördenurga optimeerimist fakt, et füüsiline orientatsioon on kvaternioonide ruumis kaheväärtuseline ehk sama orientatsiooni kirjeldavad kvaternioonid *q* ja *-q*. Need kaks kvaterniooni vastavad samal teljel pööretele θ ja $2\pi - \theta$ ning juhtimissüsteem, mis optimeerib pöördenurka peab oskama neist valida väiksema pöördenurga. Juhtimisseadus, mis on kvaternioonivea põhjal disainitud, valib automaatselt väikseima pöördenurgaga trajektoori, muutes juhtimise nii ajaliselt kui ka energiamahult optimaalseks [15].

Punktmassi liikumise optimeerimine

Antud ülesanne käsitleb lihtsustatud punktmassi liikumist Newtoni dünaamika raames, kus liikumist mõjutavad piiratud suurusega jõud. Süsteemi juhitakse PD-kontrolleriga, mille parameetreid optimeeritakse, et saavutada võimalikult soodne dünaamiline käitumine, võttes arvesse mitut füüsikalist muutujat, nagu näiteks energiakulu, reageerimiskiirus ja liikumise sujuvus.

Antud metoodikat saab hiljem üldistada keerukamatele süsteemidele, näiteks satelliidi asendi juhtimisele kvaternioonide abil.

Energiakulu optimeerimine

Üks optimeerimise eesmärk on juhtimissüsteemi energiakulu minimeerimine, mida hinnatakse rakendatud jõu ruutnormi integraali kaudu [16] :

$$E = \int_0^T \left\| \vec{F}(t) \right\|^2 . \tag{15}$$

See valem on erijuht, mis keskendub ainult rakendatud jõule, kulufunktsiooni üldisemast vormist, mida kasutatakse optimeerimisprobleemide lahendamisel. Antud valem väljendab kogu juhtimistoiminguteks kulunud energiat ajavahemikus [0; *T*], kus 0 on juhtimisprotsessi algus- ja *T* selle lõppaeg.

Diskreetses simulatsioonis hinnatakse seda suurust, summeerides vastavad, diskreetsetel ajahetkedel mõõdetud, väärtused. Saadud suurus annab kvantitatiivse hinnangu sellele, kui intentsiivselt juhtimissüsteem peab töötama, et täita oma ülesannet. Kuigi see ei vasta otseselt füüsikalisele mehaanilisele tööle, annab see meetod usaldusväärse hinnangu kontrolleri aktiivsuse ja energiakasutuse tasemele kuna suuremad ruutnormi väärtused viitavad suuremale jõu kasutamisele ja potentsiaalselt suuremale energiatarbele.[17]

Teiseks optimeerimiseesmärgiks on süsteemi reageerimiskiirus ehk aeg, mille jooksul süsteem saavutab soovitud oleku. See on tihedalt seotud klassikalise juhtimisteooria mõistega reageerimisaeg, mis tähistab minimaalset aega, mille järel süsteemi olek jääb püsima soovitud sihtväärtuse lähedusse etteantud tolerantsi piires:

$$T_{settle} = \min\{t: \|\vec{x}(t) - \vec{x}_{target}\| < \varepsilon \text{ for all } t' > t\}, \text{kus} \quad (16)$$

$\vec{x}(t)$ tähistab süsteemi olekut ajamomendil t,

$\vec{x}_{target}$ tähistab soovitud sihtolekut, mille poole süsteem liigub,

ε tähistab lubatud tolerantsi,

$t' > t$ on tingimus, et süsteem peab jääma tolerantsi piiresse ka kõigil järgnevatel hetkedel, mitte ainult hetkeks.

Reageerimisaja minimeerimine on eriti oluline rakendustes, kus süsteem peab reageerima kiiresti ja täpselt – näiteks trajektoori korrigeerimisel või liikuva sihtmärgi jälgimisel. Lühike reageerimisaeg viitab suurele dünaamilisele jõudlusele, vähendades üle- või alasiirdeid ning parandades süsteemi stabiilsust ja juhitavust.[18].

Kolmas optimeerimiseesmärk on minimeerida kiirenduse tuletist ajas (*jerk*), et tagada juhtimissignaali sujuvus. See lähenemine ei piira kiirenduse suurust iseenesest, vaid vähendab kiirenduse järske muutusi, mis võivad põhjustada vibratsiooni või ülekoormusi mehaanilistes süsteemides. Sujuvust saab kvantitatiivselt hinnata järgmise funktsiooni abil:

$$S = \int_0^T \left\| \frac{d\vec{F}}{dt} \right\|^2 dt, \text{kus} \quad (17)$$

$\vec{F}(t)$ on rakendatud jõu vektor ajas,

$\frac{d\vec{F}}{dt}$ on selle ajatuletis ehk kui kiiresti jõusignaal ajas muutub (*jerk*).

Selle kriteeriumi minimeerimine aitab vältida äkilisi muutusi juhtimiskäskudes, mis võivad tekitada soovimatuid vibratsioone, mehaanilisi pingeid või kulumist süsteemi komponentides. Lisaks on juhtsignaali sujuvus oluline praktilistel põhjustel: paljud aktuaatorid (nt mootorid ja ajamid) omavad dünaamilisi piiranguid ega suuda efektiivselt järgida kiirelt muutuvaid või järske juhtimiskäske[17].

Et vältida optimaalseid, kuid praktiliselt ebaefektiivseid lahendusi, rakendatakse täiendavat karistusfunktsiooni juhtudel, kui süsteemi lõppolek erineb sihtolekust rohkem kui lubatud piirväärtus $\delta = 0{,}1$ ühikut. Sellisel juhul lisatakse eesmärgifunktsioonidele karistusliige kujul:

$$P = \lambda \cdot \|\vec{x}(T) - \vec{x}_{target}\|, \qquad \lambda = 100, \text{kus} \quad (18)$$

$\vec{x}(T)$ on süsteemi olek simuleerimisperioodi lõpuhetkel,

$\vec{x}_{target}$ on soovitud sihtolek,

λ on karistuskoefitsient, mis määrab tingimuse rikkumise mõju üldisele optimeerimistulemusele.

Karistusliige lisatakse kõigile kolmele optimeerimiseesmärgile — energiakulule, juhtimisajale ja juhtsignaali sujuvusele — eesmärgiga suunata otsing ainult selliste lahendusteni, mis lisaks optimeerimiskriteeriumitele täidavad ka probleemi põhieesmärgi: sihtpunkti saavutamise.

Selline lähenemine on levinud mitme eesmärgiga optimeerimise puhul, kus rangete piirangute asemel rakendatakse niinimetatud pehmeid piiranguid (*soft constraints*), mis võimaldavad lahendusteruumi paremat uurimist, kuid välistavad sobimatud tulemused karistusteguri kaudu.

Optimeerimiseks kasutatakse mitme eesmärgiga evolutsioonilist algoritmi NSGA-II (*Non-dominated Sorting Genetic Algorithm II*), mis on üks enim kasutatud meetodeid Pareto-efektiivsete lahenduste leidmiseks. NSGA-II tugineb geneetiliste algoritmide raamistikule, kus lahenduste populatsioon areneb iteratiivselt läbi valiku, ristamise ja mutatsiooni. Erinevalt klassikalistest, ühe eesmärgiga optimeerimismeetoditest, ei otsi NSGA-II ühtainsat optimaalset lahendit, vaid genereerib lahenduste hulga, mis moodustavad kompromisside kogumi ehk Pareto-piiri — näiteks vahetasakaalu energiatõhususe ja juhtimiskiiruse vahel.

NSGA-II põhineb domineerimattuse printsiibil: lahendus A domineerib lahendust

B, kui A ei ole ühegi eesmärgi osas halvem ja on vähemalt ühe eesmärgi osas parem. Sellise printsiibi kaudu järjestatakse lahendused mitmesse mitte-domineeritud tasemesse (Pareto-kihtidesse), mis võimaldab efektiivselt hinnata lahenduste kvaliteeti mitme eesmärgi suhtes.

Säilitamaks lahenduste mitmekesisust ja vältida populatsiooni koondumist kitsasse lahenduste piirkonda, rakendab NSGA-II lisaks *crowding distance* meetodit. See mõõdab iga lahenduse "lähedust" naabritele eesmärgiruumi sees ning eelistab valikus hajutatud lahendusi. Selline mehhanism tagab, et lõppvalim hõlmab erinevaid kompromisslahendusi kogu Pareto-fronti ulatuses.

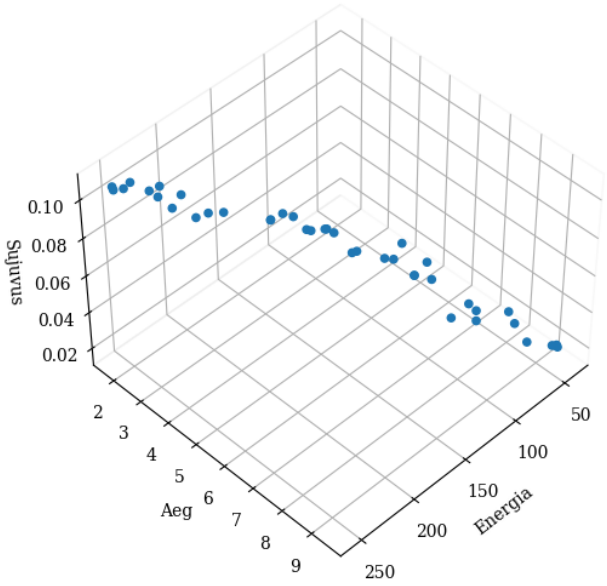
Ülesandes defineeritakse kuue vaba muutujaga optimeerimisprobleem — kolm proportsionaalkoefitsienti ja kolm derivatiivkoefitsienti mis mõjutavad PD-regulaatori käitumist. Optimeeritakse kolme eesmärki:

- juhtimisjõu energiakulu,
- jõudmise aeg,
- kontrolljõu sujuvus.

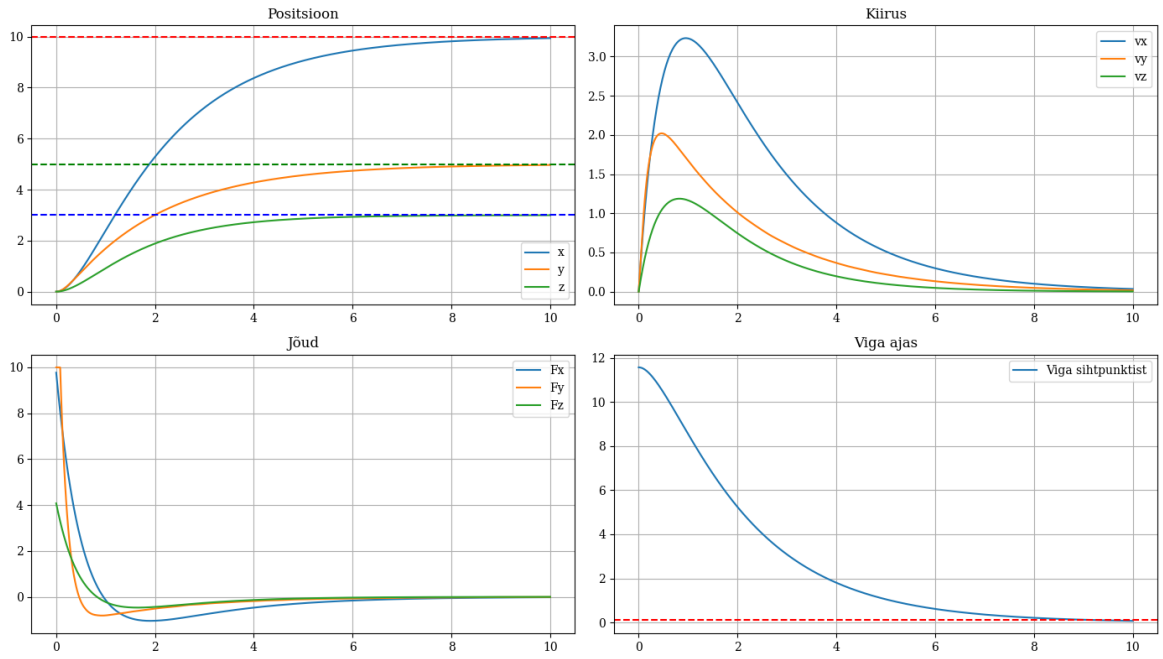
Algoritm seadistatakse järgmiste parameetritega:

- populatsiooni suurus on 40 indiviidi,
- generatsioonide arv on 100,
- initsialiseerimine: ühtlasele jaotusele alluv juhuslik suurusväärtustega vahemikus [0 ; 20].

Optimeerimise tulemuseks on Pareto optimaalsete lahenduste hulk, mis sisaldab kompromisslahendusi erinevate eesmärkide vahel. Lahenduste seast saab hiljem valida sobivaima vastavalt konkreetsele rakendusele — näiteks minimaalset energiat tarbiv või kiireim lahendus. Antud simulatsiooni tulemused on kujutatud joonistel 9 ja 10.



Joonis 9 Lahendite jaotus kolme näitaja suhtes



Joonis 10 Optimeerimise simulatsiooni tulemused

Joonis 9 on kolmemõõtmeline hajusdiagramm, kus iga punkt kujutab ühte optimeerimisprotsessi käigus leitud Pareto-optimaalset lahendust. Punktid on esitatud kolmel teljel: energia, aeg ja sujuvus. Diagrammilt on näha, et lahendused paiknevad kaldu mööda tasandit, mitte ei koondu ühte punkti. See on oodatav tulemus mitme eesmärgiga optimeerimise puhul kuna paremus ühe muutuja suhtes tähendab sageli halvenemist teise suhtes. Lahendused, mille energiakulu on väike on sageli sujuvamad, kuna agressiivne juhtimine nõuab tavaliselt rohkem energiat. See viitab sellele, et mõningane energiatõhusus saavutatakse loomupäraselt sujuva juhtimise kaudu. Lahendused, mis saavutavad sihtpunkti kiiresti, kipuvad nõudma suuremat energiakulu. See tähendab, et kiire

jõudmine toimub sageli jõulisema juhtimise kaudu, mis ei pruugi olla energiatõhus ega sujuv. See on oluline, kui optimeerimise kontekstis tuleb teha kompromiss kiiruse ja kulutuste vahel.

Lahendused ei ole koondunud ühte klastrisse, vaid katavad laia spektri eesmärgiruumis. See näitab, et NSGA-II on suutnud säilitada lahenduste mitmekesisuse, pakkudes erinevaid kompromisse. See on oluline, kuna lõppkasutaja saab valida just oma rakenduse jaoks optimaalse tasakaalu.

Joonisel 10 on kirjeldatud juhitud süsteemi käitumine nelja erineva graafiku kaudu, mis kirjeldavad vastavalt positsiooni, kiirust, juhtimisjõudu ning sihtpunktist kõrvalekallet ajas.

Positsiooni graafikul on kujutatud süsteemi positsioonikomponentide x, y ja z muutus ajas. Kõverad näitavad, kuidas süsteemi asukoht läheneb järk-järgult etteantud sihtpunktile. Kõik komponendid liiguvad sujuvalt oma sihtväärtuse poole, ilma nähtavate võnkumiste või kõikumisteta. See viitab hästi häälestatud kontrollerile ja stabiilsele juhtimisprotsessile, mis saavutab soovitud tulemuse kontrollitult ning täpselt.

Kiiruse graafikul on toodud kiiruskomponentide v_x, v_y ja v_z muutus ajas. Alguses on näha kiiruse kiiret kasvu, mis on ootuspärane reaktsioon algsele suurele positsiooniveale. Seejärel väheneb kiirus järk-järgult, lähenedes nullile, mis viitab sellele, et süsteem aeglustub stabiilselt sihtpunkti jõudes. Kiiruse muutumine on sujuv ja ilma järskude hüpeteta, mis näitab, et kontroller väldib ülekompensatsiooni ja tagab füüsikaliselt realistliku juhtimise.

Jõu graafikul on toodud kontrolleri poolt rakendatud jõu komponentide F_x, F_y ja F_z muutus ajas. Alguses on jõud suuremad, mis on loomulik, kuna süsteem peab kompenseerima suurt algviga. Aja möödudes jõud vähenevad ja stabiliseeruvad nulli lähedal, mis näitab, et kontroller lülitub järjest vaiksemale juhtimisele, kui sihtpunkt on saavutatud. Mõningane jõu muutlikkus võib tuleneda täpsuse saavutamiseks tehtud väikestest korrigeerimistest.

Vea graafikul on kujutatud kogu süsteemi kaugus sihtpunktist ehk viga ajas. Viga väheneb kiiresti ning jõuab tolerantsi alla umbes 7 sekundi juures. Pärast seda püsib viga selle piiri all, kinnitades, et süsteem jääb soovitud täpsuse piiresse. Vea eksponentsiaalne vähenemine kinnitab PD-kontrollerile iseloomulikku stabiilset ja järkjärgulist lähendamist sihtpositsioonile.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli arendada simulatsioonikeskkond väikese satelliidi, täpsemalt pikosatelliidi asendi määramise ja juhtimise süsteemi (ADCS) testimiseks. Loodud keskkond võimaldab kolmemõõtmelises ruumis modelleerida satelliidi translatoorset liikumist ja pöörlemist ümber kolme telje. Simulatsioon keskendub kineetiliste ja dünaamiliste mudelite rakendamisele ilma reaalsete keskkonnamõjude (nt magnetväli, andurimüra) kaasamiseta.

Pöörlemisliikumine on modelleeritud kvaternionide abil, et vältida Euleri nurkadega kaasnevaid singulaarsusi. Liikumise juhtimiseks on rakendatud PD-kontroller translatsioonilise liikumise jaoks ning simuleeritud satelliidi stabiliseerimist detumbling-protsessi kaudu. Simulatsioon on realiseeritud Pythonis ja pakub avatud lähtekoodiga alternatiivi levinud kommertstarkvaradele. Töö tulemus on modulaarne ja laiendatav keskkond, mida saab kasutada õppetöös ning edasiseks arenduseks satelliitide juhtimissüsteemide vallas.

Võtmesõnad: pikosatelliit, ADCS, simulatsioon, Python, kvaternionid, PD-kontroller, detumbling, 3D dünaamika

Abstract

The aim of this bachelor's thesis was to develop a simulation environment for testing the Attitude Determination and Control Subsystem (ADCS) of a small satellite, specifically a picosatellite. The developed system allows modeling of three-dimensional translational and rotational motion of the satellite, with a focus on kinematic and dynamic simulation without implementing real environmental effects such as magnetic field or sensor noise.

Rotational dynamics were represented using quaternions to avoid singularities related to Euler angles. A Proportional-Derivative (PD) controller was applied for translational control, and the satellite's stabilization was demonstrated through a simulated detumbling process. The simulation environment was implemented in Python as an open-source alternative to commercial software solutions. The result is a modular and extensible platform suitable for educational use and further development in satellite control systems.

Keywords: picosatellite, ADCS, simulation, Python, quaternions, PD controller, detumbling, 3D dynamics

Kasutatud kirjandus

[1] T. Andrews, „Computation Time Comparison Between Matlab and C++ Using Launch Windows“.

[2] S. B. Aruoba ja J. Fernández-Villaverde, „A Comparison of Programming Languages in Economics“, National Bureau of Economic Research, Cambridge, MA, w20263, juuni 2014. doi: 10.3386/w20263.

[3] Z. Pavanello, „Simulation of a Commercial Off-The-Shelf ADCS with Design of a Pitch Sun Tracking Attitude Mode for the ORCASat“.

[4] G. Chari, „CubeSat Attitude Determination and Control System (ADCS) Design, Analysis, and Validation“.

[5] K. Mahanti, „Hardware-in-the-loop simulation and testing of the ADCS of the Beyond Atlas CubeSat“.

[6] H. Sellik, M. Rebane, ja A. Leibak, „Kuupsatelliidi asendimääramine päikese- ja magnetvektoriga kasutades SVD meetodit“, juuni 2018, Vaadatud: 18. märts 2025. [Online]. Available at: https://digikogu.taltech.ee/et/item/aef79cd0-8187-44a7-a37e-70a52d8360f6

[7] J. Rodriguez Tersa, „Simulation of the ADCS subsystem for a VLEO satellite“, Master thesis, Universitat Politècnica de Catalunya, 2023. Vaadatud: 18. märts 2025. [Online]. Available at: https://upcommons.upc.edu/handle/2117/403752

[8] Justin Wyss-Gallifent, „MATH431: Gimbal Lock“. [Online]. Available at: https://math.umd.edu/~immortal/MATH431/book/ch_gimballock.pdf

[9] J. van Vuuren ja G. Hermann, „The design and simulation analysis of an attitude determination and control system for a small earth observation satellite“, Stellenbosch : Stellenbosch University, 2015. Vaadatud: 18. märts 2025. [Online]. Available at: http://hdl.handle.net/10019.1/96979

[10] S. C. Dhury, G. K. Singh, ja R. M. M. Hra, „Design and Verification Serial Peripheral Interface (SPI) Protocol for Low Power Applications“, *Int. J. Innov. Res. Sci. Eng. Technol.*, kd 03, nr 10, lk 16750–16758, okt 2014, doi: 10.15680/IJIRSET.2014.0310048.

[11] R. Muhammad ja F. Hadary, „Performance Comparison of Bang-Bang Controller, Single PID Controller, and Cascade PID Controller on Cart Pendulum“, *Proc. Natl. Conf. Electr. Eng. Inform. Ind. Technol. Creat. Media*, kd 4, nr 1, Art. nr 1, dets 2024.

[12] „Block diagram of PID closed loop control system“, ResearchGate. Vaadatud: 19. mai 2025. Available at: https://www.researchgate.net/figure/Block-diagram-of-PID-closed-loop-control-system_fig1_326068417

[13] S. Temel, „P, PD, PI, PID CONTROLLERS“.

[14] Z.-Y. Nie, Z. Li, Q.-G. Wang, Z. Gao, ja J. Luo, „A unifying Ziegler–Nichols tuning method based on active disturbance rejection“, *Int. J. Robust Nonlinear Control*, kd 32, nr 18, lk 9525–9541, 2022, doi: 10.1002/rnc.5848.

[15] R. Kristiansen ja P. J. Nicklasson, „Satellite attitude control by quaternion-based backstepping“, *Proceedings of the 2005, American Control Conference, 2005.*, juuni 2005, lk 907–912 kd 2. doi: 10.1109/ACC.2005.1470075.

[16] „Lewis optimal control 3rd edition 2012.pdf“. Vaadatud: 20. mai 2025. [Online]. Available at: https://lewisgroup.uta.edu/FL%20books/Lewis%20optimal%20control%203rd%20edition%202012.pdf

[17] K. Ogata, *Modern control engineering*, 5th ed. Prentice-Hall electrical engineering series. Instrumentation and controls series. Boston: Prentice-Hall, 2010.

[18]B. Siciliano ja O. Khatib, Toim, *Springer Handbook of Robotics*. Springer Handbooks. Cham: Springer International Publishing, 2016. doi: 10.1007/978-3-319-32552-1.