TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TOMMI UTTER

OHJUSSIMULAATTORIN TOTEUTUS OSAKSI REAALI-AIKAISTA HAJAUTETTUA SIMULAATIOTA

Diplomityö

Tarkastaja: Prof. Tommi Mikkonen
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekunta-
neuvoston kokouksessa 09.09.2015

TIIVISTELMÄ

TOMMI UTTER: Ohjussimulaattorin toteutus osaksi reaaliaikaista hajautettua simulaatiota
Tampereen teknillinen yliopisto
Diplomityö, 85 sivua
Joulukuu 2015
Tietotekniikan koulutusohjelma
Pääaine: Ohjelmistotuotanto
Tarkastajat: Prof. Tommi Mikkonen
Avainsanat: hajautettu simulaatio, ohjussimulaattori, ilmataisteluohjus

Sotaoperaatioissa hyödynnetyt laitteet ja niiden tarvitsemat resurssit kasvattavat operaatioista koituvia kustannuksia merkittävästi. Sotaharjoituksissa pyritään jäljittelemään oikeita sotatilanteita, mutta esimerkiksi satojen tuhansien eurojen arvoisten ilmataisteluohjusten käyttäminen harjoitustarkoituksiin ei ole järkevää. Simulaatioiden avulla osia harjoituksista voidaan mallintaa tietokoneella, jolloin resursseja kuluu vähemmän ja kustannuksia saadaan madallettua.

Teknologioita sekä kaupallisia toteutuksia sotaharjoitusten simulointiin on olemassa useita. Kaupalliset toteutukset kuitenkin myydään kokonaisuuksina eikä niiden yksittäisiä simulaattoreita, kuten ohjussimulaattoria, voida hyödyntää itsenäisinä komponentteina osana muita järjestelmiä. Työn kohdeyrityksellä oli tarvetta reaaliaikaiseen simulaatioon soveltuvalle ohjusmallinnukselle.

Tässä diplomityössä suunnitellaan sekä toteutetaan ilmataistelu- ja ilmatorjuntaohjuksia simuloiva ohjussimulaattori, joka voidaan liittää osaksi hajautettua simulaatiota. Työssä perehdytään hajautettujen simulaatioiden toimintaan sekä ohjusten ominaisuuksiin, jonka jälkeen oikean ohjuksen toiminnasta tehdään yksinkertaistavia oletuksia. Oletusten avulla muodostetaan maapallon ilmakehässä liikkuvan ohjuksen mallinnuksesta vastuussa oleva ohjusmalli. Ohjusmallia varten työssä johdetaan kuuden vapausasteen liikeyhtälöt jäykälle kappaleelle, joiden lisäksi esitetään yhtälöitä ohjukseen muodostuville voimille. Ohjusmallin toiminnassa keskitytään ohjuksen oikeankaltaisen lentoradan mallinnukseen sekä mallin suorituskykyisyyteen. Työn lopussa esitetään mittaustuloksia mallin toiminnasta ja suorituskyvystä.

Työssä toteutetun ohjussimulaattorin tarkkuus soveltui hyvin suunniteltuun käyttökohteeseen ja se otettiin käyttöön loppukäyttäjän käyttämässä tuotteessa, jossa se korvasi aiemman ohjusmallinnuksen. Toteutettu ohjusmalli sekä jäykkien kappaleiden mallinnus antavat hyvän pohjan tuleville simulaattoreille, jos tulevaisuudessa on tarvetta mallintaa esimerkiksi lentokoneiden liikettä tarkemmalla tasolla.

ABSTRACT

TOMMI UTTER: Implementation of a missile simulator for distributed real-time simulation
Tampere University of Technology
Master of Science Thesis, 85 pages
December 2015
Master's Degree Programme in Information Technology
Major: Software Engineering
Examiner: Prof. Tommi Mikkonen
Keywords: distributed simulation, missile simulator, air-to-air, surface-to-air, missile

Military operations require considerable amount of devices and resources which increase the costs of the operations. Military exercises are used to imitate the actual operations but using, for example, air-to-air missiles, which can cost hundreds of thousands of euros each, for training purposes is impractical. The expenses can be decreased by modeling parts of the exercises with simulators.

Several commercial solutions, which can be used to simulate military exercises, exist. However, the commercial implementations are usually sold as complete simulation software and their individual simulators, such as missile simulators, cannot be used as components inside other systems. This thesis was made for a company which had a need for real-time missile modeling.

In this thesis a missile simulator, which can be connected into a larger distributed simulation, is designed and implemented. The thesis examines features of distributed simulations and the behavior of missiles which are used to intercept aircrafts. A missile model which makes simplifying assumptions of the behavior of the real-life missiles will be defined in this thesis. The model uses six-degrees-of-freedom equations of motion for a rigid body which will be derived in this work. Forces acting on the missile will also be examined and equations for them will be presented. The missile model focuses on producing the correct trajectory and orientation of the missile with performance which is suitable for real-time simulation. Measurements and evaluations of the missile model are made at the end of this thesis.

Accuracy and performance of the implemented missile simulator suited well for its intended application. The simulator was introduced into the product used by the end-user, where it replaced the previous implementation which was used to simulate missiles. The implemented missile model, as well as the modeling of rigid bodies, gives a good foundation for upcoming simulators if in the future there is a need, for example, to model aircraft movement in a more detailed level.

ALKUSANAT

Tämä diplomityö on tehty Insta DefSec Oy:ssä lopputyönä Tampereeren teknillisen yliopiston tietotekniikan laitokselle vuoden 2015 aikana.

Haluaisin kiittää työn ohjaajaa Jarkko Hartikaista erinomaisesta ohjaamisesta sekä neuvoista, joita häneltä sain työn eri vaiheissa. Jarkolle kuuluu kiitos myös työn aiheen ideoinnista sekä avuliaista kirjoitusvinkeistä. Työn tarkastajana toiminutta professori Tommi Mikkosta haluaisin kiittää hyödyllisistä neuvoista diplomityön kirjoitukseen liittyen sekä hyvin toimineesta kommunikoinnista. Ripeät vastaukset sähköpostitiedusteluihin sekä nopeat palaverijärjestämiset edistivät työn valmistumista. Lisäksi haluaisin osoittaa kiitokseni diplomityön rahoituksessa mukana olleille osapuolille. Rahoituksesta oli suuri apu ja se nopeutti diplomityön valmistumista huomattavasti.

Haluaisin myös kiittää Teemu Salmivettä, joka auttoi paljon työn alkuun saamisessa etsimällä Instasta useita diplomityövaihtoehtoja. Vaihtoehtojen joukossa oli myös tässä diplomityössä käsiteltävä aihe. Kiitos kuuluu myös Sami Vuoriselle, joka diplomityön alkuvaiheissa kannusti minua aloittamaan työtäni sekä neuvoi monessa diplomityönteossa askarruttaneessa asiassa. Myös koko muulle Instan väelle kuuluu suuri kiitos mukavasta ja hyvin avuliaasta työilmapiiristä.

Lopuksi haluaisin kiittää vanhempiani sekä veljeäni, jotka ovat tukeneet minua opiskeluissani ensimmäisestä kouluvuodestani lähtien. Kiitos kuuluu myös rakkaalle avopuolisolleni Laura Tauriaiselle, jonka kannustuksen ansiosta jaksoin jatkaa työn tekoa raskaimpinakin hetkinä.

Kiitos!

Tampere, 30.10.2015

Tommi Utter

SISÄLLYS

KUVALUETTELO

TAULUKKOLUETTELO

LYHENTEET JA MERKINNÄT

cg	center of gravity, massakeskipiste
cp	center of pressure, painekeskipiste
CRC	Central RTI Component, itsenäinen RTI:n osa
ECEF	Earth-Centered Earth-Fixed, maapalloon kiinnitetty karteesinen koordinaatisto
ECI	Earth-Centered Inertial, maapallon keskipisteeseen kiinnitetty karteesinen inertiaalikoordinaatisto
FOM	Federation Object Model, federaation ominaisuuksia kuvaava määrittely
HLA	High Level Architecture, hajautettujen simulaatioiden luontiin kehitetty standardi
IP	Internet Protocol, tietoliikenneprotokolla
JMS	Java Message Service, Javalle määritetty komponenttien välinen viestinvälitysmenetelmä
LOS	Line of Sight, tähtäysviiva, ohjuksen ja kohteen välinen suora linja
LRC	Local RTI Component, federaatin osaksi liitettävä RTI:n osa
NACA	National Advisory Committee for Aeronautics, Yhdysvaltain ilmailu- ja avaruushallinto
NASA	National Aeronautics and Space Administration, ilmailualaan liittyvä Yhdysvaltain valtion virasto
NED	North, East and Down, maanpinnan suuntainen karteesinen koordinaatisto
OMT	Object Model Template, HLA:n määrittelemä tapa kuvata SOM ja FOM
PN	Proportional Navigation, suhteellinen navigointi
PPN	Pure Proportional Navigation, puhdas suhteellinen navigointi
RPR FOM	Real-time Platform-level Reference Federation Object Model, referenssi FOM toteutus
RTI	Runtime Infrastructure, toteuttaa HLA:n rajapintamäärittelyn ja välittää federaattien välistä tietoa
SOM	Simulation Object Model, federaatin ominaisuuksia kuvaava määrittely
TCP	Transport Control Protocol, yhteydellinen ja luotettava tietoliikenneprotokolla

TCP/IP	Transmission Control Protocol / Internet Protocol, tietoliikenneprotokollaperhe
UDP	User Datagram Protocol, yhteydetön ja epäluotettava tietoliikenneprotokolla
US DoD	United States Department of Defence, Yhdysvaltain puolustusministeriö
WGS-84	World Geodetic System, koordinaatistojärjestelmä
XML	Extensible Markup Language, merkintäkieli

α	kohtauskulma, joka muodostuu kappalekoordinaatiston ja tuulikoordinaatiston väliin
β	sivuluisukulma, joka muodostuu kappalekoordinaatiston ja tuulikoordinaatiston väliin
λ_{lon}	geodeettinen pituusaste
$\boldsymbol{\tau}$	vääntömomentti
$\boldsymbol{\tau}_a$	aerodynaamisista voimista aiheutuva vääntömomentti
$\boldsymbol{\tau}_c$	ohjuksen ohjausjärjestelmän tuottama vääntömomentti
ϕ_{lat}	geodeettinen leveysaste
ψ, ϕ, θ	Tait-Bryan rotaatiokulmat z-akselin, y-akselin ja x-akselin ympäri
$\boldsymbol{\omega}$	pyörimisnopeus
ω_e	WGS-84 järjestelmän mukainen maan pyörimisnopeus sen z_e akselin ympäri
$\boldsymbol{\omega}_e$	maapalloon kiinnitetyn koordinaatiston pyörimisnopeus
$\boldsymbol{\omega}_b$	kappaleeseen kiinnitetyn koordinaatiston pyörimisnopeus
$\boldsymbol{\Omega}_{LOS}$	ohjuksen ja kohteen välisen tähtäysviivan suuntautumisen muutos

a	WGS-84 järjestelmän mukainen isoakselin puolikas
$\boldsymbol{a}_{ppng}$	suhteellinen navigointi -ohjauslain tarvitsema kiihtyvyys
$\boldsymbol{a}_{vp}$	takaa-ajo-ohjauslain tarvitsema kiihtyvyys
b	WGS-84 järjestelmän mukainen pikkuakselin puolikas
$\boldsymbol{e}$	koordinaatiston kannan kuvaava yksikkövektori
f	WGS-84 järjestelmän mukainen litistyssuhde
$\boldsymbol{F}$	voimavektori
$\boldsymbol{F}_a$	aerodynaaminen voimavektori
F_d	ilmanvastuksesta aiheutuva voima vastasuoraan tuulikoordinaatiston x-akselia
$\boldsymbol{F}_g$	painovoimaa kuvaava vektori

F_l	aerodynaaminen nostevoima tuulikoordinaatiston z-akselin suuntaisesti
F_s	aerodynaaminen sivuttaissuuntainen voima tuulikoordinaatiston y-akselin suuntaisesti
$\boldsymbol{F}_t$	ohjuksen moottorin aiheuttama työntövoima
g	painovoiman aiheuttama kiihtyvyys
h	geodeettinen korkeus maata kuvaavan ellipsoidin pinnasta
$\boldsymbol{I}$	kappaleen inertiamatriisi
I_{ii}	inertiamatriisin hitausmomentit
I_{ij}, I_{ji}	inertiamatriisin hitaustulot
$\boldsymbol{L}$	kappaleen pyörimismäärä
m	massa
$\boldsymbol{q}$	kvaternio
$\boldsymbol{r}_{cg\to p}$	vektori kappaleen massakeskipisteestä painekeskipisteeseen
$\boldsymbol{r}_{eb}$	kappaleen sijainti maapalloon kiinnitetyn koordinaatiston origosta
$\boldsymbol{r}_m$	ohjuksen sijainti maan keskipisteestä
$\boldsymbol{r}_{ob}$	kappaleen sijainti inertiaalikoordinaatiston origosta
$\boldsymbol{r}_{oe}$	maapalloon kiinnitetyn koordinaatiston origon sijaitni inertiaalikoordinaatistosta
$\boldsymbol{r}_r$	ohjuksen ja kohteen välinen etäisyys
$\boldsymbol{r}_t$	kohteen sijainti maan keskipisteestä
$\boldsymbol{R}$	muunnosmatriisi, joka muuntaa vektorin esityksen koordinaatistosta toiseen
$\boldsymbol{v}$	nopeusvektori
$\boldsymbol{v}_m$	ohjuksen nopeusvektori
$\boldsymbol{v}_r$	ohjuksen ja kohteen nopeusvektoreiden erotus
$\boldsymbol{v}_t$	kohteen nopeusvektori
$\boldsymbol{v}_w$	virtaavan ilman nopeusvektori
x_b, y_b, z_b	kappaleeseen kiinnitetyn koordinaatiston akselit
x_e, y_e, z_e	maapalloon kiinnitetyn ECEF-koordinaatiston akselit
x_i, y_i, z_i	paikallaan pysyvän inertiaalikoordinaatiston akselit
x_m, y_m, z_m	ohjukseen kiinnitetyn koordinaatiston akselit
x_n, y_n, z_n	maanpinnan suuntaisen NED-koordinaatiston akselit
x_w, y_w, z_w	tuulikoordinaatiston akselit
$(\dot{\boldsymbol{x}})_b$	vektorin $\boldsymbol{x}$ aikaderivaatta kappalekoordinaatiston suhteen
$(\dot{\boldsymbol{x}})_e$	vektorin $\boldsymbol{x}$ aikaderivaatta maapalloon kiinnitetyn koordinaatiston suhteen

$(\ddot{\boldsymbol{x}})_e$	vektorin $\boldsymbol{x}$ aikaderivaatta kahdesti suoritettuna maapalloon kiinnitetyn koordinaatiston suhteen
$(\dot{\boldsymbol{x}})_i$	vektorin $\boldsymbol{x}$ aikaderivaatta inertiaalikoordinaatiston suhteen
$\frac{\mathrm{d}(\dot{\boldsymbol{x}})_e}{\mathrm{d}t}\Big\|_i$	vektorin $\boldsymbol{x}$ aikaderivaatta inertiaalikoordinaatiston suhteen
$(\ddot{\boldsymbol{x}})_i$	vektorin $\boldsymbol{x}$ aikaderivaatta kahdesti suoritettuna inertiaalikoordinaatiston suhteen

1. JOHDANTO

Sotatilanteita jäljittelevissä sotaharjoituksissa hyödynnetään suurta määrää erilaisia laitteita, joiden käyttämät ammukset, kuluttama polttoaine sekä mahdolliset vikaantumiset voivat kasvattaa harjoituksista koituvia kustannuksia hyvin suuriksi (Fujimoto 2000, s. 9). Esimerkiksi hävittäjälentokoneiden käyttämät ohjukset voivat nousta kappalehinnoiltaan useisiin satoihin tuhansiin euroihin, jonka vuoksi niiden käyttäminen harjoitustarkoituksiin ei ole järkevää (US DoD 2014, s. 53–54). Jos harjoituksen päätarkoituksena on kouluttaa johtotehtävässä olevaa henkilöä, joka havainnoi harjoitusten tilaa kartalta, ei käytännössä ole merkitystä ovatko kartalla näkyvät pisteet oikeita laitteita vai mallinnettuja kohteita, jotka toimivat kuten oikeat laitteet. Tällaisissa tilanteissa suuri osa harjoituksesta voidaan mallintaa tietokoneiden avulla, jolloin harjoituksissa käytettävien resurssien määrää saadaan vähennettyä ja kustannuksia saadaan madallettua.

Tietokoneella simuloituja kokonaisia sotaharjoituksia, sekä niitä tukevia työkaluja, on kehitetty merkittävästi Yhdysvaltojen Armeijan toimesta. Etenkin armeijan kehittämät, usean toisistaan erillään sijaitsevan laitteen vastuulle jaetut, simulaatiot osoittautuivat jo ensimmäisissä kokeiluissa hyödyllisiksi koulutusmenetelmiksi. (Tolk 2012, s. 415–419) Simulaatioiden kehitystä jatkettiin pidemmälle ja niitä käytetään vielä nykyäänkin kokonaisten sotilasoperaatioiden mallintamiseen, jolloin erilaisia strategioita voidaan harjoitella ja operaatioissa tarvittavien aseiden sekä muiden resurssien määrää arvioida (Fujimoto 2000, s. 7–12). Kokonaisten sotaharjoitusten simulointiin on olemassa kaupallisia sovelluksia, mutta usein ne myydään kokonaisina paketteina eikä niiden yksittäisiä tehtäviä hoitavia osuuksia, kuten ohjuksen lennon mallinnusta, voida hyödyntää erillisinä kokonaisuuksina osana muita toteutuksia (MASA Group 2015, VT MÄK 2015).

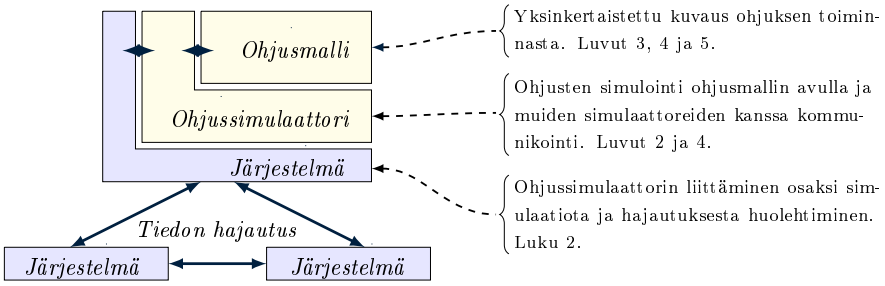
Tässä diplomityössä suunnitellaan sekä toteutetaan ilmatorjunta- ja ilmataisteluohjusten toimintaa mallintava ohjussimulaattori, joka voi toimia osana laajempaa, usean koneen vastuulle hajautettua, simulaatiota. Työssä myös selvitetään yleisellä tasolla hajautettujen simulaatioiden sekä ilmatorjunta- ja ilmataisteluohjusten ominaisuuksia. Ohjussimulaattori toteutetaan osaksi simulaatiota, jonka tarkoituksena on kouluttaa johtotason tehtävissä toimivia henkilöitä. Tämän vuoksi simulaattorin

tarkoituksena ei ole tuottaa äärimmäisen tarkkaa mallinnusta ohjuksen toiminnasta, vaan tärkempää on saavuttaa oikeankaltainen lentorata sekä toiminta tehokkuudella, joka sallii reaaliaikaisen mallinnuksen. Mallinnuksen tarkkuudelle on lisäksi asetettu joukko vaatimuksia, jotka on kasattuna taulukkoon 1.1.

Taulukko 1.1 *Toteutettavalle simulaattorille kerättyjä vaatimuksia.*

1. *Ohjuksen liikemäärän tulee säilyä luonnollisen mukaisesti eikä ohjus saa tehdä liian jyrkkiä käännöksiä.*
2. *Ohjuksen tulee lentäessään suuntautua oikein.*
3. *Kohteen on voitava väistää ohjus, jos se tekee onnistuneen väistöliikkeen.*
4. *Mallin tulee kyetä simuloimaan noin 50–100 ohjusta reaaliaikaisesti.*
5. *Mallin tulee olla modulaarinen ja helposti laajennettavissa.*
6. *Mallilla tulee voida simuloida sekä lyhyen matkan ohjuksia että keskipitkän matkan ohjuksia.*

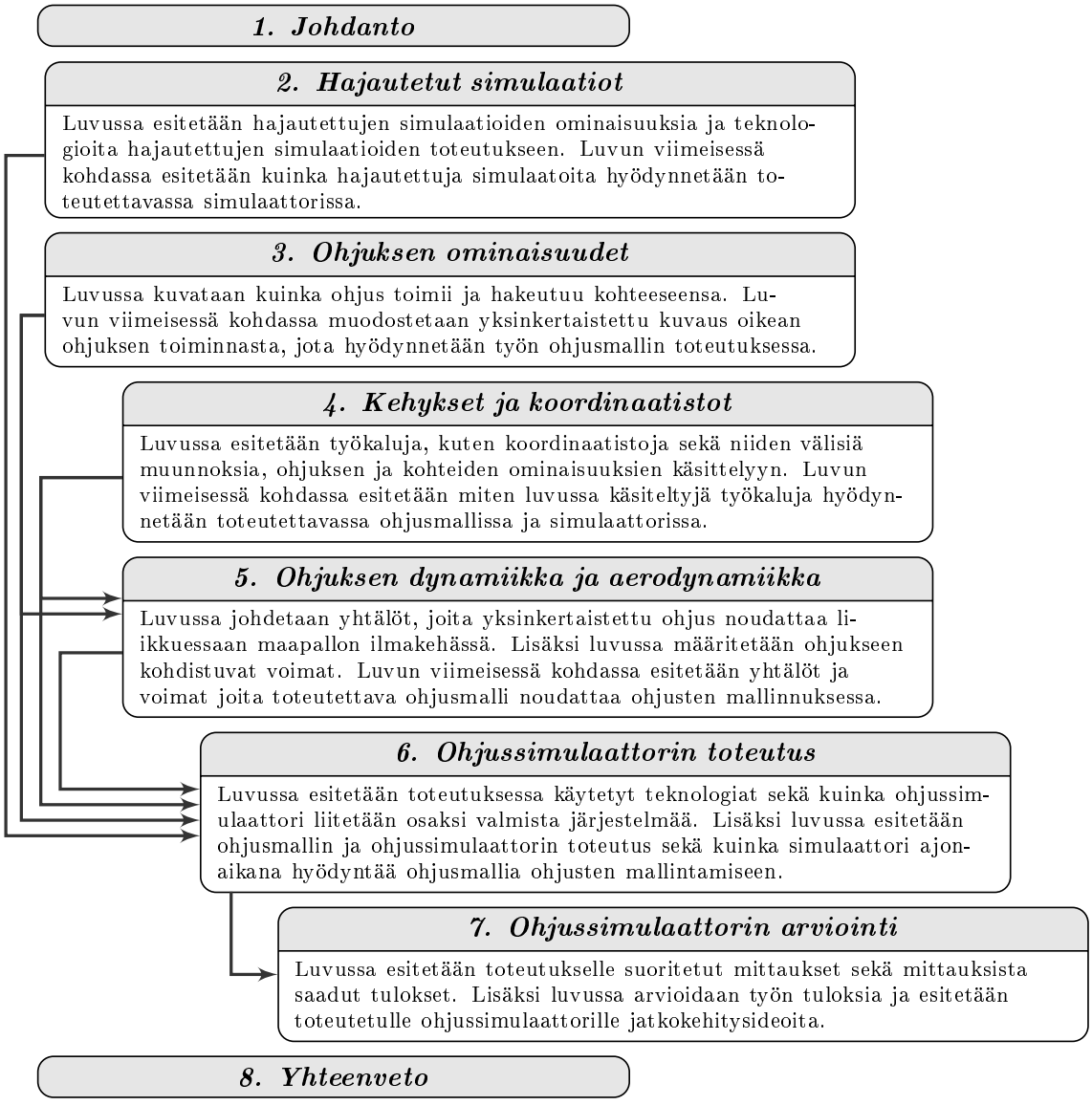
Simulaattori mallintaa ohjuksia hyödyntäen yksinkertaistavia oletuksia oikean ohjuksen toiminnasta tekevää ohjusmallia. Työssä tehdään tarvittavat oletukset, joiden jälkeen johdetaan ohjusmallin noudattamat liikeyhtälöt. Kuvassa 1.1 on esitettynä työssä toteutettavat osuudet keltaisella värillä ja muut työhon liittyvät, aiemmin toteutetut osuudet, sinisellä värillä. Kuvassa on myös esitettynä luvut, joissa käsitellään osuuksiin liittyviä asioita.



Kuva 1.1 *Tässä diplomityössä toteutettavat osuudet ja luvut, joissa esitellään osuuksiin liittyvää teoriaa. Keltaiset lohkot kuvaavat työssä toteutettavia osuuksia ja siniset lohkot aiemmin toteutettuja osuuksia, joita työssä hyödynnetään.*

Kuvassa 1.2 on esitettynä diplomityön rakenne sekä kuinka luvut hyödyntävät aiempien lukujen tietoja. Luvut 2 ja 3 sisältävät yleistä teoriaa hajautetuista simulaatioista sekä ilmatorjunta- ja ilmataisteluohjusten toiminnasta. Luvuissa 4 ja 5 esitellään tarvittavat työkalut, kuten koordinaatistot ja niiden väliset muunnokset, sekä yhtälöt, joiden avulla ohjusmalli mallintaa ohjusten liikettä. Kaikkien edellä

mainittujen lukujen viimeisessä kohdassa kuvataan, kuinka luvussa esitettyjä tietoja hyödynnetään työssä toteutettavassa ohjusmallissa ja ohjussimulaattorissa. Luvussa 6 esitellään ohjussimulaattorin toteutus sekä toteutukseen liittyneet teknologiat. Luvussa 7 kuvataan kuinka toteutusta mitattiin ja esitellään mittauksista saadut tulokset, jonka lisäksi luvussa arvioidaan toteutuksen toimintaa sekä esitellään ideoita jatkokehitykseen. Lopuksi luvussa 8 esitellään yhteenveto työssä käsitellyistä asioista.



Kuva 1.2 *Diplomityön rakenne.*

2. HAJAUTETUT SIMULAATIOT

Simulaatio on jostakin todellisesta tai kuvitteellisesta järjestelmästä luotu malli, jonka avulla suoritetaan kohdejärjestelmän toimintaa arvioivia laskutoimituksia. *Hajautettu simulaatio* on simulaatio, jonka suoritus on jaettu usean, mahdollisesti kaukana toisistaan sijaitsevan, laskentayksikön vastuulle. Työn ohjussimulaattori toteutetaan osaksi hajautettua simulaatiota, joka asettaa rajoitteita sekä vaatimuksia simulaattorin toiminnalle. Tämän vuoksi tässä luvussa esitellään hajautettujen simulaatioiden yleisiä ominaisuuksia sekä ongelmia, jonka lisäksi luvun lopussa tarkastellaan hajautettujen simulaatioiden toteutuksessa käytetyn menetelmän toimintaa.

2.1 Tarkoitus ja tyypit

Simulaation jakaminen useamman laitteen vastuulle, tekee simulaatiosta tehokkaamman ja vikasietoisemman. Jos esimerkiksi jokin laaja simuloitu kokonaisuus voidaan jakaa useammaksi pieneksi simulaatioksi, voidaan simulaation suoritusta tehostaa jakamalla simulaation osuudet eri laitteilla suoritettaviksi. Tämä kasvattaa myös simulaation vikasietoisuutta sillä yhden simulaattorin hajotessa, voi toinen simulaattori mahdollisesti korvata rikkoontuneen simulaattorin toiminnot. (Fujimoto 2000, s. 4–5)

Hajautetut simulaatiot voidaan jakaa kahteen tyyppiin: *analyyttiset simulaatiot* ja *virtuaaliset ympäristöt*. Analyyttisissä simulaatiossa simulaatio pyrkii suorittamaan laskutoimitukset mahdollisimman nopeasti saapuen johonkin lopputulokseen. Analyyttisiä simulaatoita voivat olla esimerkiksi ammuksen lentoradan määrittäminen joidenkin annettujen alkuarvojen perusteella. Virtuaalisissa ympäristöissä simulaatio muodostaa kellon tahtiin etenevän virtuaalisen maailman, johon ihmiset ja laitteet voivat liittyä. Simulaatioon liittyneet jäsenet voivat suorittaa toimenpiteitä virtuaalisen maailman sisällä, jotka vaikuttavat simulaation kulkuun ja sen lopputulokseen. Yhdysvaltojen Armeijalla on ollut suuri rooli hajautettujen simulaatioiden kehityksessä ja simulaatioiden toteutusta tukemaan on määritelty useita standardeja. Uusin niistä on *HLA (High Level Architecture)*, jonka tarkoituksena on yhdistää kaikki aiemmat simulaatioita koskevat standardit ja käytännöt yhdeksi, kaikkia aiempia simulaatioita tukevaksi simulaatioksi. (Fujimoto 2000, s. 6–12) Koska tässä

työssä toteutettava ohjussimulaattori toteutetaan osaksi reaaliaikaisesti etenevää simulaatiota, keskitytään seuraavissa kohdissa tarkemmin virtuaalisten ympäristöjen toimintaan.

2.2 Hajautetut virtuaaliset ympäristöt

Kuten edellisessä kohdassa mainittiin, virtuaalisissa ympäristöissä simulaatioon osallistuvat simulaattorit muodostavat virtuaalisen maailman, jossa simulaattorit sekä simulaatioon liittyneet ihmiset voi tehdä simulaation kulkuun vaikuttavia toimintoja. Simulaatioon liittyneet ihmiset voivat esimerkiksi toimia yksittäisinä sotilaina tai lentokoneiden pilotteina keinotekoisessa taistelutilanteessa. (Fujimoto 2000, s. 6–8)

Virtuaalisten ympäristöjen tavoitteena on luoda riittävän tarkka kuvaus jostakin, simulaation kohteena olevasta, järjestelmästä. Riittävä tarkkuus riippuu käytettävän simulaation käyttökohteesta. Esimerkiksi jos simulaatiota hyödynnetään lentäjien koulutuksessa, tulee mallinnettujen lentokoneiden toiminnan vastata hyvin oikean lentokoneen toimintaa. Jos simulaatiolla koulutetaan lennonjohtajia, voi olla riittävää, että lentokoneet reagoivat käskyihin oikein ja varsinainen lentokoneen toiminta mallinnetaan karkeammalla tarkkuudella. Virtuaalisten ympäristöjen tulisi tarjota kaikille käyttäjille samanlaiset edellytykset toimintojen suorittamiseen. Käyttäjät eivät saisi saada etulyöntiasemaa toisiin käyttäjiin nähden, esimerkiksi tehokkaamman laitteiston avulla, ja käyttäjien tekemien toimintojen onnistumisen tulisi riippua vain käyttäjän omista taidoista. (Fujimoto 2000, s. 195–196)

2.2.1 Virtuaalisen ympäristön verkko

Jotta toisistaan erillään toimivat itsenäiset simulaattorit voidaan yhdistää, tarkoituksena muodostaa hajautettu virtuaalinen ympäristö, tulee määrittää millaisen verkon ylitse yhdistäminen tehdään, jonka lisäksi tulee selvittää tukevatko kaikki liitettävät laitteet samoja tiedonvälitysprotokollia (Tolk 2012, s. 190–191).

Simulaatioon osallistuvat jäsenet voidaan yhdistää usealla tavalla kiinni toisiinsa. Jäsenet voivat esimerkiksi yhdistää johonkin keskitettyyn palvelimeen, joka hallitsee simulaation tilaa ja välittää tiedot simulaation tilasta kaikille osallistuville jäsenille. Keskittämällä simulaation hallinta yhden koneen vastuulle helpottuu ylläpito, mutta simulaation skaalautuvuus heikentyy. Toinen menetelmä on jakaa simulaation tilan hallinta kaikkien osallistuvien jäsenten kesken, jolloin keskitettyä palvelinta ei tarvita. (Fujimoto 2000, s. 196–199) Myös muunlaisia verkkorakenteita voidaa toteuttaa, joissa esimerkiksi osa simulaation hallinnasta on keskitetyllä palvelimella ja osa tiedoista välitetään suoraan jäsenten välillä.

Nykyään virtuaalisissa ympäristöissä hyödynnetään *TCP/IP (Transmission Control Protocol / Internet Protocol)* -protokollaperheen tarjoamia menetelmiä viestien välitykseen (Fujimoto 2000, s. 223–257). IP-protokolla on paketteihin perustuva tietoliikenneprotokolla, jonka avulla avulla lähetetyt paketit saadaan ohjattua oikealle vastaanottajalle. IP-protokollan päälle toteutettu *TCP-protokolla (Transport Control Protocol)* on yhteydellinen ja luotettava. Se mahdollistaa tavujonojen lähettämisen verkon ylitse siten, että tiedon perillepääsy varmistetaan. Toinen IP-protokollan päälle toteutettu, ja hajautetuissa simulaatioissa TCP-protokollan lisäksi hyödynnetty, protokolla on *UDP-protokolla (User Datagram Protocol)*. UDP-protokolla on yhteydetön ja epäluotettava protokolla, jossa tietoliikennepaketit välitetään itsenäisinä ja niiden perille pääsyä ei varmisteta. (Jia & Zhou 2005, s. 65–78) Luotettavaa protokollaa voidaan hyödyntää simulaatioissa esimerkiksi tärkeiden viestien välittämiseen. Tällaisia viestejä ovat esimerkiksi simulaation ajoon tai jonkin simulaatioentiteetin luomiseen liittyvät viestit. Nopeampaan pakettiviestintään tarkoitettuja viestinvälitysmenetelmiä voidaan hyödyntää esimerkiksi entiteettien tilapäivitysten lähetyksissä, sillä tilapäivitysten nopean lähetystahdin vuoksi, ei yhden viestin katoamisella ole merkittävää vaikutusta simulaation kulkuun. (Fujimoto 2000, s. 223–257)

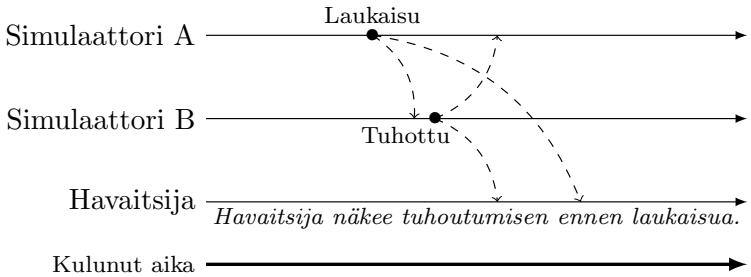
2.2.2 Viestien muoto ja ajoitukset

Kun useita simulaattoreita yhdistetään osaksi samaa simulaatiota tulee huomioida kaksi keskeistä ongelmaa: tiedonvälitys simulaattoreiden välillä ja simulaattoreiden välisen ajan synkronointi (Boer 2005, s. 20–21).

Samaan simulaatioverkkoon liitetyt simulaattorit voivat hyödyntää eri tarkkuudella toimivia malleja olioidensa simulointiin. Samassa simulaatiossa voi olla esimerkiksi tarkasti lentokoneen toimintaa mallintava simulaattori ja karkealla tarkkuudella panssariajoneuvoja simuloiva simulaattori. Kaikkien simulaatioon osallistuvien simulaattoreiden tulisi kuitenkin kyetä muodostamaan yhteinen ymmärrys simulaation tilasta. Tämän vuoksi simulaatiolle tulee määrittää jokin menetelmä kuvata verkossa kulkevat viestit, jotta itsenäiset simulaattorit osaavat tulkita simulaation tiedot samalla tavalla. (Tolk 2012, s. 198–199)

Samassa verkossa olevat simulaattorit voivat myös kuvata ajan etenemistä toisistaan poikkeavilla tavoilla, ja verkosta voi aiheutua viivettä simulaattoreiden välillä kulkeviin viesteihin. Tästä voi seurata tilanteita, joissa jokin simulaattori vastaanottaa viestejä järjestyksessä, joka on simulaation toiminnan kannalta epälooginen (Tolk 2012, s. 195–198). Kuvassa 2.1 on esitettynä tilanne, jossa epälooginen viestien

vastaanotto voi tapahtua. Kuvassa esitettävässä tilanteessa simulaatioon on liittynyt panssarivaunuja simuloiva simulaattori A, kohdetta simuloiva simulaattori B ja kolmas havainnoiva osapuoli. Kun panssarivaunu ampuu kohdettaan, lähetetään muille simulaattoreille tieto ammuksen laukaisusta. Kohdetta simuloiva simulaattori vastaanottaa tiedon, tuhoaa kohteen ja välittää tiedon räjähdyksestä sekä sen aiheuttamista vahingoista muille simulaattoreille. Tällöin verkosta muodostuneet viiveet viesteihin voivat aiheuttaa tilanteen, jossa tilannetta havainnoiva osapuoli vastaanottaa ensin tiedon kohteen räjähdyksestä ja vasta myöhemmin tiedon panssarivaunun laukaisemasta ammuksesta. Tämän vuoksi simulaattoreille tulee siis muodostaa jokin yhteinen menetelmä, jolla hallitaan simulaation ajoituksiin liittyviä ongelmia (Fujimoto 2000, s. 259–261).



Kuva 2.1 *Simulaattoreiden välinen ajoitusongelma, jossa havaitsija näkee kohteen tuhoutumisen ennen kuin sen tuhoava ohjus on laukaistu (Fujimoto 2000, s. 262).*

2.3 High Level Architecture

Jos edellisessä kohdassa esitettyjä virtuaalisten ympäristöjen luontiin liittyviä ongelmia tulisi ratkoa jokaista uutta hajautettua simulaatiota luodessa, kuluisi paljon aikaa ja rahaa. Tämän vuoksi on kehitetty valmiita standardeja ja niitä noudattavia toteutuksia, joiden avulla hajautettujen simulaatioiden muodostaminen on helpompaa. Eräs tällainen standardi on High Level Architecture (HLA) (IEEE Std 1516-2010 2010). HLA on simulaattoreiden yhdistämistä yksinkertaistamaan luotu standardoitu arkkitehtuuri. Sen tarkoituksena on yhdistää aiemmat simulaatioarkkitehtuurit yhdeksi, kaikkia simulaatiotilanteita tukevaksi, arkkitehtuuriksi. HLA muodostaa käytännöt, joiden avulla voidaan tukea eri simulaattoreiden välistä yhteensopivuutta, ja se pyrkii kasvattamaan simulaattoreiden uudelleenkäyttöä. (Fujimoto 2000, s. 209–211)

HLA koostuu kolmesta sen määrittelevästä osasta: *HLA-säännöistä*, *Object Model*

Template (OMT) -mallista ja *HLA:n rajapintamäärittelystä*. HLA-säännöt kuvaavat simulaatioiden suunnittelun pääperiaatteet ja federaattien sekä federaatioiden vastuut. OMT kuvaa tavan, jolla simulaatiossa oleva data tulisi esittää. Rajapintamäärittely kuvaa joukon palveluita, joita rajapinnan toteuttavan *Runtime Infrastructuren (RTI)* tulisi tarjota. (IEEE Std 1516-2010 2010)

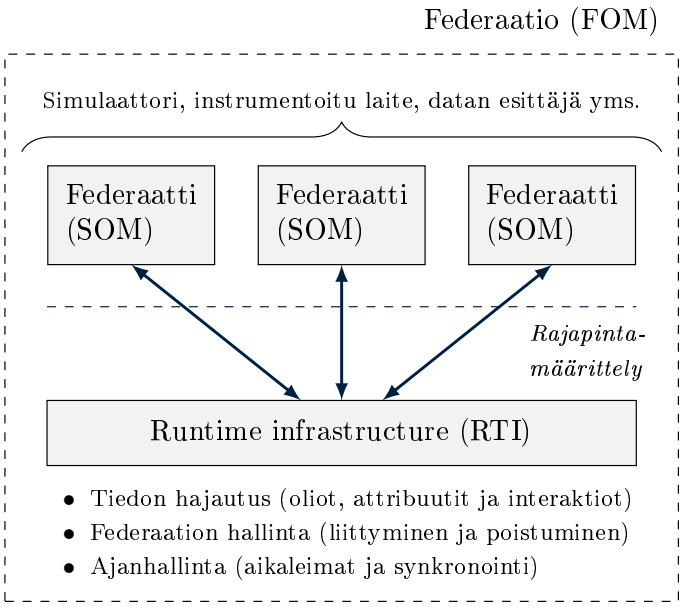
Simulaatiota kuvataan HLA:ssa termillä *federaatio* ja yksittäistä simulaattoria termillä *federaatti*. Federaatti voi, tietokoneella suoritetun simulaation lisäksi, olla esimerkiksi instrumentoitu fyysinen laite tai passiivinen datan esittäjä. Simulaatioon osallistuvia entiteettejä, kuten lentokoneita tai ajoneuvoja, kuvataan olioilla, jotka sisältävät yksilöivän tunnisteen, tilatietoja sisältäviä attribuutteja ja riippuvuuksia toisiin olioihin. Jokaisella olion attribuutilla on yksi, simulaation aikana mahdollisesti vaihtuva, omistaja, jonka vastuulla on ilmoittaa RTI:hin kun attribuutin arvo on muuttunut. Federaatit voivat ilmoittautua kuuntelemaan attribuuttiin tulevia muutoksia, jolloin muutoksen tapahtuessa RTI ilmoittaa kiinnostuneille federaateille tiedon muutoksesta. Muutokset attribuuteissa ovat yksi menetelmä, jolla federaatit voivat välittää tietoa toisilleen. Toinen menetelmä on interaktiot, jotka kuvaavat jotakin hetkellistä tapahtumaa, kuten räjähdystä tai ohjuksen laukaisua. Interaktiot voivat käynnistää jonkin toisen federaatin toimenpiteen, kuten uuden ohjuksen luomisen tai ajoneuvon tuhoamisen. (Fujimoto 2000, s. 211–213) Kuvassa 2.2 on esitettynä HLA:n toimintaan liittyviä osapuolia. Kuvan *Simulation Object Model (SOM)* kuvaa yksittäisen federaatin tietoja ja *Federation Object Model (FOM)* koko federaation ominaisuuksia. Tarkempi kuvaus SOM:ista ja FOM:ista esitetään alakohdassa 2.3.2.

2.3.1 HLA:n säännöt

HLA:n säännöt koostuvat viidestä federaatioita koskevasta säännöstä ja viidestä federaatteja koskevasta säännöstä. Sääntöjen tarkoituksena on mahdollistaa federaatin sisäisten federaattien välinen vuorovaikutus sekä määrittää federaatioiden ja federaattien vastuualueet (IEEE Std 1516-2010 2010). Taulukossa 2.1 on esitettynä federaatioille määritetyt säännöt ja taulukossa 2.2 on esitetty federaatteihin kohdistuvat säännöt. Tarkempi kuvaus sääntöjen merkityksestä, sekä niiden valintaan johtaneista perusteluista, on löydettävissä lähteestä (IEEE Std 1516-2010 2010).

2.3.2 HLA:n oliomallit

Alakohdassa 2.2.2 esitetty ongelma eri simulaattoreiden välillä lähetettyjen viestien ymmärtämiseen on ratkaistu HLA:ssa oliomallien avulla. Oliomallien avulla si-



Kuva 2.2 *HLA:n toimintaan liittyviä kokonaisuuksia.*

Taulukko 2.1 *HLA:n federaatioita koskevat säännöt (IEEE Std 1516-2010 2010).*

1. *Federaatioilla tulee olla HLA FOM dokumentoituna HLA OMT:n mukaisesti.*
2. *Federaatiossa kaikkien simulaatioon kuuluvien olioiden tiedot säilytetään federaateissa, ei RTI:ssä.*
3. *Federaation ajon aikana kaikki FOM-data liittyneiden federaattien välillä välitetään RTI:n lävitse.*
4. *Federaatioon liittyneet federaatit ovat vuorovaikutuksissa RTI:n kanssa HLA:n rajapintamäärittelyn mukaisesti.*
5. *Federaation olion attribuutilla voi olla vain yksi sen omistava federaatti kerrallaan.*

mulaattorit esittävät niiden tarvitseman sekä niiden tuottaman informaation ja ne mahdollistavat simulaattoreiden välisen vuorovaikutuksen määrittelemällä yhteisen muodon simulaatiossa liikkuvalle datalle. Oliomallit dokumentoidaan HLA:ssa taulukkojoukon avulla, joka on nimeltään *Object Model Template (OMT)*. OMT määrittelee mikä HLA:n oliomalli on ja sen avulla muodostetaan federaatin kykyjä kuvaava *Simulation Object Model (SOM)* ja federaation ominaisuuksia kuvaava *Federation Object Model (FOM)*. SOM määrittelee millaista tietoa yksittäinen federaatti voi tuottaa muille federaation federaateille ja millaista tietoa se voi vastaanottaa muilta federaateilta. FOM määrittelee federaatiossa federaattien välillä ajonaikana välitet-

Taulukko 2.2 *HLA:n federaatteja koskevat säännöt (IEEE Std 1516-2010 2010).*

1. *Federaateilla tulee olla HLA SOM dokumentoituna HLA OMT:n mukaisesti.*
2. *Federaattien tulee voida päivittää sekä vastaanottaa muutoksia attribuutteihin ja lähettää sekä vastaanottaa interaktioita niiden SOM-määrittelyn mukaisesti.*
3. *Federaattien tulee voida antaa ja vastaanottaa attribuutin omistajuus federaation ajon aikana dynaamisesti niiden SOM-määrittelyn mukaisesti.*
4. *Federaattien tulee voida muuttaa niiden SOM:eissa määriteltyjä ehtoja, joiden mukaan ne tuottavat päivityksiä attribuutteihin.*
5. *Federaattien tulee voida hallita lokaalia aikaa siten, että ne voivat vaihtaa dataa toisten federaation jäsenten kesken.*

tävät olioluokat, interaktioluokat ja niiden attribuutit. (IEEE Std 1516-2010 2010) SOM:it ja FOM on myös merkittynä aiemmin esitetyssä kuvassa 2.2.

OMT koostuu useasta taulukosta, joita tulee täydentää federaattien SOM:ien ja federaatioiden FOM:ien osalta. Taulukoita ovat seuraavat (IEEE Std 1516.2-2000 2001):

- **Oliomallin yksilöivä taulukko**, joka sisältää yleistä tietoa FOM:ista tai SOM:ista, kuten sen nimen, tarkoituksen ja version.
- **Olioluokkien rakennetaulukko**, joka määrittelee oliomallissa käytettävät luokat sekä kantaluokan ja luokkien väliset suhteet. Periytetyt luokat perivät niiden kantaluokassa määritetyt attribuutit. Luokalla voi olla HLA OMT:n mukaan vain yksi kantaluokka. Taulukko myös kuvaa federaatin kyvyn julkaista tietoa jonkin luokan attribuutteihin, ja tilata tietoa jonkin luokan attribuuteista. Taulukossa 2.3 on esimerkki olioluokkien rakennetaulukosta ajoneuvolle määriteltynä. Luokkien nimen perässä olevien sulkujen avulla ilmaistaan kyky tiedon julkaisusta tai tilaamisesta. Sulkujen sisällä oleva P (Publish) ilmaisee julkaisukykyä, S (Subscribe) ilmaisee tilaamiskykyä, PS ilmaisee sekä julkaisu että tilaamiskykyä ja N (Neither) ilmaisee ettei federaatti voi julkaista tai tilata luokan attribuutteja.
- **Attribuuttitaulukko**, joka määrittelee olioluokkien rakennetaulukon luokalle attribuutit. Attribuutit voivat olla määriteltynä mille vain rakennetaulukon luokalle. Attribuutit kuvaavat sellaisia nimettyjä osia olioiden tilasta, jotka voivat muuttua simulaation aikana. Taulukossa 2.4 on esitettynä osa attribuuttitaulukon sarakkeista ja lentokoneelle määritetyistä attribuuteista. Attribuuttitaulukon ensimmäinen *Object*-sarake osoittaa johonkin olioluokkien ra-

Taulukko 2.3 *Olioluokkien rakennetaulukko.*

Ajoneuvo (PS)	Lentokone (P)	Hävittäjä (P)	*F/A-18 Hornet (P)*
		Pommikone (PS)	*B-2 Spirit (PS)*
	Panssarivaunu (S)	Taistelu-panssarivaunu (S)	*Leopard 2A5 (S)*
		Rynnäkkö-panssarivaunu (S)	*BMP-2 (S)*

kennetaulukon luokkaan. Luokka voi olla millä vain hierarkiatasolla ja esimerkiksi kaikille ajoneuvoille voi määrittää joitakin yhteisiä attribuutteja. *Attribute*-sarakkeen avulla attribuutit voidaan nimetä ja *Datatype*-sarakkeessa attribuutille voidaan määrittää sen tyyppi. *Update type* -sarakeen ja *Update condition* -sarakkeen avulla federaatti voi ilmaista kuinka usein ja minkä ehtojen mukaan se päivittää attribuutin arvoa. Attribuuttitaulukko sisältää sarakkeen myös attribuutin omistajuusvaihdolle liittyvälle tiedolle sekä tiedolle, voiko federaatti julkaista tai tilata attribuutin dataa. Taulukko sisältää sarakkeet myös esimerkiksi kuormanrajaukseen ja päivitysviestien vastaanottotapaan liittyvien tietojen esittämiseen.

- **Interaktioluokkien rakennetaulukko**, joka on vastaavanlainen kuin olioluokkien rakennetaulukko. Se määrittelee interakioluokkien väliset perimissuhteet. Luokkiin lisätään, vastaavalla tavalla kuin olioluokkien rakennetaulukossa, tietoa federaation kyvyistä interaktioiden julkaisuun ja tilaamiseen.
- **Parametritaulukko**, joka sisältää parametrejä interaktioluokille vastaavasti kuin attribuuttitaulukko sisältää attribuutteja olioluokille. Monet parametritaulukon sarakkeista ovat vastaavia kuin attribuuttitaulukon sarakkeet, mutta federaatiot eivät voi julkaista tai tilata yksittäisiä interaktioiden parametrejä.

Lisäksi OMT sisältää esimerkiksi viestien suodatukseen liittyvän taulukon, datatyyppien määrittelyyn hyödynnettävän taulukon ja sanakirjamaisia taulukkoja, jotka sisältävät kaikkien käytettyjen luokkien nimet, attribuuttien nimet, interaktioiden nimet ja interaktioiden parametrit. Taulukot on tarkemmin esiteltynä OMT:n määrittelevässä standardissa lähteessä (IEEE Std 1516.2-2000 2001).

FOM:it voidaan kasata kokonaan federaattien SOM:eissa määriteltyjen tietojen avulla tai vaihtoehtoisesti FOM:ina voidaan hyödyntää referenssitoteutusta, jonka tietoja myös SOM:it käyttävät (IEEE Std 1516-2010 2010). Referenssi FOM:ien avulla voidaan säästää aikaa sekä rahaa, sillä valmista FOM runkoa voi laajentaa

Taulukko 2.4 *Osa attribuuttitaulukon sarakkeista.*

Object	Attribute	Datatype	Update type	Update condition
Lentokone	SijaintiX	*Float*	*Periodic*	10/second
	SijaintiY	*Float*	*Periodic*	10/second
	SijaintiZ	*Float*	*Periodic*	10/second
	Massa	*Float*	*Periodic*	10/second
	Tuniste	*String*	*Static*	n/a

tarvittavilla tiedoilla ilman että koko FOM:ia tarvitsee rakentaa tyhjästä. *Real-time Platform-level Reference Federation Object Model (RPR FOM)* on referenssi FOM, joka määrittelee olioluokat, interaktioluokat ja niiden attribuutit sotasimulaatioille. Se määriteltiin, jotta aiempien simulaatio standardien mukaan toteutettujen simulaattoreiden siirtäminen HLA:han olisi helpompaa ja se toimii edelleen yleisesti käytettynä referenssi FOM:ina. (Möller, Dubois, Leydour & Verhage 2014)

2.3.3 HLA:n rajapintamäärittely ja RTI

HLA:n rajapintamäärittely kuvaa rajapinnan federaattien ja RTI:n välillä, sekä tarvittavat palvelut, jotta federaatit voivat kommunikoida toistensa kanssa. Määrittely ei ota kantaa RTI:n toteutukseen tai toteutuksessa käytettäviin teknologoihin. Rajapintamäärittely on jaettu seitsemään ryhmään seuraavasti(IEEE Std 1516.1-2000 2001):

- **Federaation hallinta** palvelut sisältävät federaation hallintaan liittyviä operaatioita. Palvelut hoitavat esimerkiksi federaation käynnistykseen, tuhoamiseen, federaattien liittymiseen ja niiden poistumiseen liittyviä operaatioita. Se myös hoitaa federaattien väliseen synkronointiin sekä federaation tallennukseen ja lataamiseen liittyviä toimintoja.
- **Määrittelyn hallinta** palveluiden avulla federaatit ilmoittavat millaisia olioita, attribuutteja ja interaktioita ne luovat, sekä millaisia olioita, attribuuttimuutoksia ja interaktioita ne vastaanottavat. Federaattien tulee ilmoittaa mitä tietoa se käyttää ja millä tavalla se tietoa käsittelee ennen kuin se voi esimerkiksi päivittää olion attribuutteja.
- **Olioiden hallinta** palvelut hoitavat olioiden rekisteröinnin ja poiston sekä interaktioiden lähetyksen ja vastaanoton. Palvelut myös hoitavat olioiden attribuuttien muokkaamisen sekä muokkaustapahtuman välittämisen siitä kiinnos-

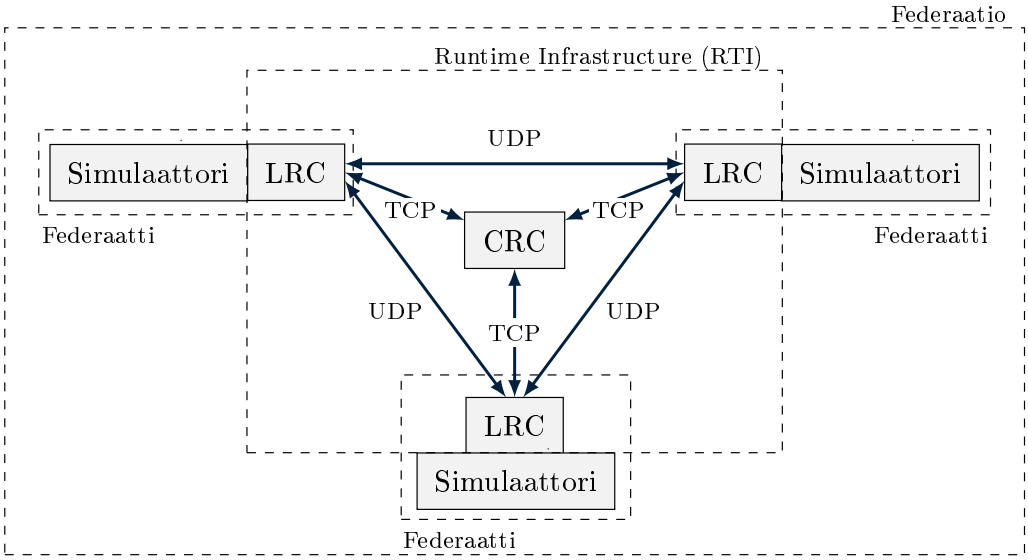
tuneille osapuolille. Lisäksi palvelut tarjoavat mahdollisuuden objektien hakemiseen.

- **Omistuksen hallinta** palveluita käytetään olion attribuuttien omistajuuteen liittyviin operaatioihin. Tällaisiä on esimerkiksi omistajuudesta luopuminen tai sen pyytäminen joltakin toiselta federaatilta. Vain federaatti, jolla on omistajuus johonkin olion attribuuttiin, voi muokata kyseistä attribuuttia ja vain yksi federaatti kerrallaan voi olla attribuutin omistaja.
- **Ajan hallinta** palveluilla pyritään takaamaan, että federaatioista lähetetyt viestit vastaanotetaan yhdenmukaisella tavalla kaikissa federaatioissa. Palvelut ylläpitävät virtuaalista hajautettua kelloa ja lisäävät aikaleimoja viesteihin, jotta alakohdan 2.2.2 kaltaiset ongelmatilanteet voidaan välttää.
- **Datan hajautuksen hallinta** palveluiden avulla voidaan vähentää sekä turhan tiedon lähettämistä että sen vastaanottoa. Attribuuttipäivitysten ja interaktioiden merkitys jollekin federaatille määritetään hyödyntäen käyttäjän määrittelemää avaruutta. Sekä päivityksen tuottaja että päivityksen kuuntelija kuvaavat jotkin rajat avaruudessa ja jos ne limittäytyvät, on tieto kuuntelijan kannalta merkittävää. Tällainen tilanne voi olla esimerkiksi jokin tapahtuma, jonka merkittävyys jollekin federaatille tarkistetaan etäisyytenä tapahtuman sijainnista.
- **Tukipalvelut** tarjoavat monenlaisia palveluita, joita simulaatiossa hyödynnetään. Palvelut sisältävät esimerkiksi toimintoja RTI:n käynnistämiseen ja sammuttamiseen.

RTI toteuttaa HLA:n rajapintamäärittelyn kuvaamat palvelut ja toiminnot. Koska rajapintamäärittely ei kuitenkaan ota kantaa itse RTI:n toteutukseen, on nykyään olemassa useita RTI-toteutuksia, joiden käytännöt eroavat toisistaan. Tämän vuoksi eri RTI-toteutukset eivät ole toistensa kanssa yhteensopivia ja kaikkien federaattien tulee käyttää samaa RTI-toteutusta, jotta federaatio toimii oikein. (Ross 2012)

RTI koostuu kahdenlaisista komponenteista: *Local RTI Component (LRC)* ja *Central RTI Component (CRC)*. CRC-komponentti on itsenäinen ajettava ohjelmisto ja sen vastuulla on koordinoida LRC:iden toimintaa. LRC-komponentti on federaatin osaksi liitettävä kirjasto ja se hoitaa yhteydenmuodostuksen CRC:hen sekä muihin LRC:ihin. Nykyisin yleinen viestinvälitysmenetelmä RTI-toteutuksissa on kohdassa 2.2.1 kuvattu TCP/IP, mutta joissakin toteutuksissa on mahdollista välittää viestejä esimerkiksi jaetun muistin yli. Välittäessään viestejä LRC:ltä LRC:lle useat RTI-toteutukset hyödyntävät UDP-protokollaa. Jos viestinvälitys tapahtuu

LRC:n ja CRC:n kesken, käytetään monessa toteutuksessa luotettavampaa TCP:tä. (Ross 2012) Kuvassa 2.3 on esitettynä LRC:t, CRC ja niiden väliset protokollat.



Kuva 2.3 *RTI:n rakenne.*

RTI voi järjestää LRC:t ja CRC:t erilaisilla tavoilla. Se voi esimerkiksi muodostaa alakohdassa 2.2.1 esitetyn rakenteen, jossa on yksi keskitetty palvelin, käynnistämällä jollekin palvelimelle yhden CRC:n, johon kaikki LRC:t yhdistävät. Tällöin CRC hallitsee simulaation yleisiä tietoja kuten tietoa simulaatioon liittyneistä federaateista. LRC:t voivat kuitenkin välittää yhä tietoa, kuten attribuuttipäivityksiä, toisten LRC:iden kesken. Vaihtoehtoisesti jokaisen federaatin LRC:n voi asettaa kommunikoimaan suoraan toisten LRC:iden kanssa, jolloin verkon rakenne vastaa kohdassa 2.2.1 esitettyä toista menetelmää, jossa ei ole yhtä keskitettyä palvelinta. Tällöin jokin LRC voi hoitaa CRC:n tehtäviä ja vastuu tehtävien hoitamisesta voi vaihtua simulaation aikana jollekin toiselle LRC:lle. (Ross 2012) Edellisessä kuvassa 2.3 on esitettynä RTI-toteutus, jossa CRC on suorituksessa keskitetyllä palvelimella, mutta LRC:t kommunikoivat UDP-protokollan avulla suoraan toisten LRC:iden kanssa.

2.4 Ohjussimulaattorin ympäristö

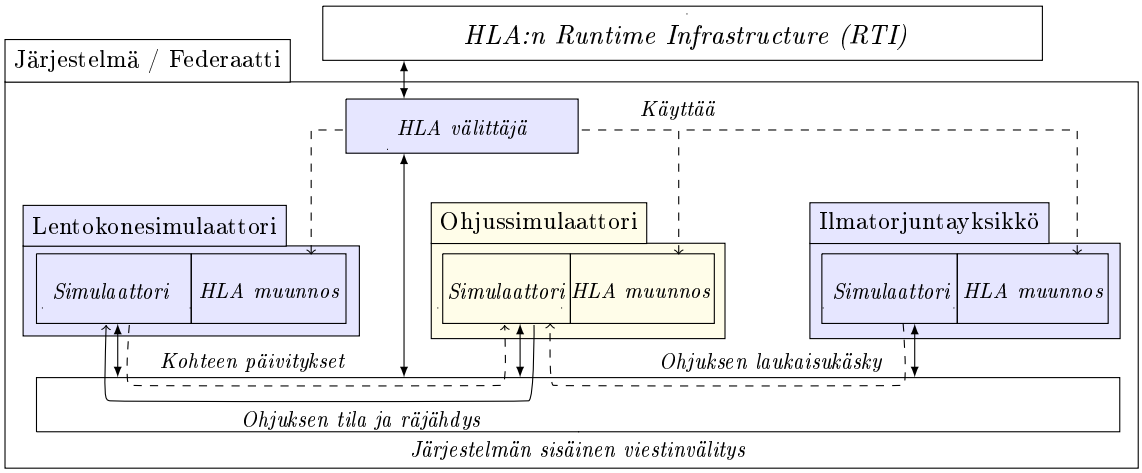
Työn ohjussimulattori toteutetaan osaksi isompaa simulaattorikokonaisuutta, johon on jo aiemmin toteutettu useita muita simulaattoreita. Työn kannalta oleelliset kokonaisuudessa valmiiksi olevat simulaattorit ovat ilmatorjuntasimulaattori, jonka vastuulla on ilmatorjuntayksikön simulointi sekä ilmatorjuntaohjuksen laukaisu, ja lentokonesimulaattori, jonka tehtävänä on simuloida lentokoneiden lentoa, laukaista

ilmataisteluohjuksia lentokoneista sekä suorittaa lentokoneiden tuhotarkastelu. Toteutettavan ohjussimulaattorin tehtävänä on vastaanottaa muilta simulaattoreilta saapuneita viestejä, tuottaa niiden perusteella ohjus-entiteettejä, ja julkaista niihin liittyviä tilatietoja sekä toimintoja muille simulaatioon osallistuville simulaattoreille.

Ympäristö, johon ohjussimulaattori toteutetaan, asettaa tiettyjä vaatimuksia simulaattorin toiminnalle. Hajautetun simulaation viestimäiseen kommunikointiin aiheutuu viiveitä simulaattoreiden välisestä verkosta, jonka lisäksi simulaattorit lähettävät tilapäivityksiä erilaisilla aikaväleillä. Tämä hankaloittaa ohjussimulaattorin simuloimien ohjusten osumista, sillä viiveiden vuoksi kohteet näyttävät liikkuvan askeltaen eteenpäin. Simulaation reaaliaikaisuus pakottaa myös tasapainoilemaan toteutettavan ohjusmallin tarkkuuden sekä sen suorituskyvyn välillä. Hyvin karkea ja suorituskykyinen simulaattori olisi helppo toteuttaa, mutta sen tuottamien ohjusten realistisuus jäisi vähäiseksi. Tarkasti ohjuksen toimintaa mallintava simulaattori soveltuisi tutkimustyökaluksi ja kehittämiseen, mutta sen vaatima suoritusteho estäisi sen käytön reaaliaikaisessa simulaatiossa. Toteutettavan simulaattorin tulee siis mallintaa kohteitaan sopivalla tarkkuudella, jotta saavutetaan reaaliaikaiseen simulointiin tarvittava tehokkuus. Osa ympäristön simulaattoreista myös kuvaa olioidensa tilaa ohjussimulaatorista poikkeavalla tavalla. Tämän vuoksi simulaattoreiden lähettämien tilapäivitysten tietoja, kuten kohteen nopeustietoa tai sijaintia, tulee muuntaa ohjussimulaattorin ymmärtämään muotoon.

Ohjussimulaattoria toteutettaessa otetaan huomioon aiempien simulaattoreiden toteutus, sekä niiden ajoympäristö. Toteutetussa ohjussimulaattorissa hyödynnetään valmista järjestelmää, joka toimii *ohjelmistokehyksenä* uusille simulaattoreille. Ohjelmistokehys on vaillinainen, määritetyissä kohdissa aukkoja sisältävä ohjelmisto. Ohjelmistokehyksen aukkoja voi täydentää tuotekohtaisilla komponenteilla, jolloin kehys muodostaa halutun toiminnallisuuden toteuttavan ohjelmiston. Kehys tarjoaa ohjelmistokomponenttien lisäksi esimerkiksi arkkitehtuuria ja perustoiminnallisuuksia, joita toteutetut komponentit voivat hyödyntää. (Koskimies & Mikkonen 2005, s. 185–194) Tässä työssä ohjelmistokehyksenä toimivaan järjestelmään toteutetaan ohjuksia simuloivat osuudet, jolloin kokonaisuudesta muodostuu hajautetuissa simulaatioissa toimiva ohjussimulaattori. Järjestelmä tarjoaa esimerkiksi konfiguraatiotiedostoja, jotka avustavat järjestelmän sisäisen ja RTI:n lävitse tapahtuvan viestinvälityksen toimintaan saamisessa. Kehys myös hoitaa alakohdassa 2.3.3 esitetyn LRC:n liittämisen osaksi simulaattoria sekä tarjoaa kantaluokkia ja rajapintoja, joiden avulla sisäisen viestinvälityksen viestit saadaan julkaistua HLA:n RTI:hin. Kuvassa 2.4 on esitettynä järjestelmä, jonka osaksi ohjussimulaattori toteutetaan. Täs-

sä työssä toteutettava osuus on korostettuna keltaisella värillä ja muut toimintaan liittyvät osuudet on korostettuna sinisellä värillä.



Kuva 2.4 *Järjestelmä, jonka osaksi ohjussimulaattori toteutettiin.*

Ohjelmistokehyksenä toimiva järjestelmä voi sisältää useita simulaattoreita, jotka on yhdistetty toisiinsa sisäisen viestinvälityksen avulla. Sisäisen viestinvälityksen ansiosta järjestelmän sisäiset simulaattorit voidaan sitoa löyhästi toisiinsa ja ne voivat toimia asynkronisesti toisistaan riippumatta. Osa simulaattoreista voisi olla sijoitettuna myös toisen vastaavanlaisen järjestelmän sisälle, jolloin järjestelmien välinen viestinvälitys tapahtuisi HLA RTI:n lävitse. Kuvan järjestelmä kuvaa siis yhtä federaattia ja federaatio koostuu useasta RTI:n yli kommunikoivasta järjestelmästä. Järjestelmät kommunikoivat toisten federaattien kanssa järjestelmään kuuluvan *HLA välittäjän* avustuksella. HLA välittäjä muuntaa viestejä järjestelmän sisäisen viestinvälityksen ja HLA:n määrittelemien oliomallien välillä. Se hyödyntää toiminnassaan järjestelmän sisäisten simulaattoreiden määrittelemiä kuvauksia siitä, kuinka kunkin simulaattorin lähettämät sisäisen viestinvälityksen viestit tulisi muuntaa HLA standardia noudattaviksi viesteiksi. Ohjelmistokehyksen vastuulle kuului myös järjestelmän sisäisten simulaattoreiden käynnistäminen sekä pysäyttäminen. Kehyksen toimintaan tai sen arkkitehtuuriratkaisuihin ei keskitytä tässä työssä tarkemmin.

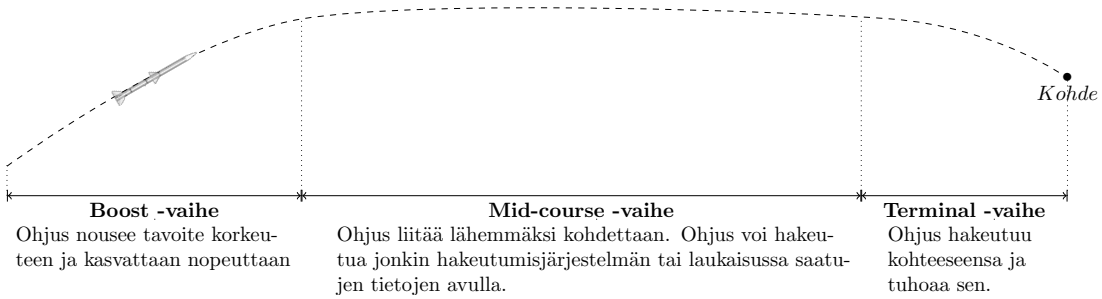
3. OHJUKSEN OMINAISUUDET

Ohjus on sylinterimäinen kappale, jonka tarkoituksena on pysäyttää tai tuhota sille määrätty kohde. Ohjukset koostuvat useasta komponentista, joista jokaisella on tärkeä rooli ohjuksen tehtävän suorituksessa. Ohjusten muodot sekä ominaisuudet vaihtelevat paljon riippuen käyttötarkoituksesta johon ohjus on suunniteltu. Tässä työssä toteutetaan ohjussimulaattori, joka simuloi ilmassa sijaitsevien kohteiden tuhoamiseen tarkoitettujen ohjusten toimintaa. Tämän vuoksi tässä luvussa esitellään yleisesti ilmatorjunta- ja ilmataisteluohjuksien rakennetta sekä niiden toimintaan liittyviä järjestelmiä. Luvussa käsiteltävät asiat perustuvat lähteissä (Strickland 2011) sekä (US Army Missile Command 1995) esitettyihin tietoihin, ellei tekstissä ole toisin mainittu.

3.1 Ohjuksen lennon vaiheet

Ohjuksen lento alkaa sen laukaisualustalta. Lentokoneiden tuhoamiseen tai pysäyttämiseen tarkoitetut ohjukset voidaan laukaista monenlaisilta alustoilta kuten ilmassa liikkuvasta lentokoneesta, laivasta tai maan pinnalla sijaitsevasta ilmatorjuntajärjestelmästä. Ilmasta ilmaan ammuttavat ilmataisteluohjukset ja maasta ilmaan ammuttavat ilmatorjuntaohjukset ovat rakenteeltaan melko samanlaisia. Joitakin ilmasta laukaistuja ohjuksia voidaan käyttää sellaisenaan myös maanpinnalla sijaitsevissa ilmatorjuntajärjestelmissä.

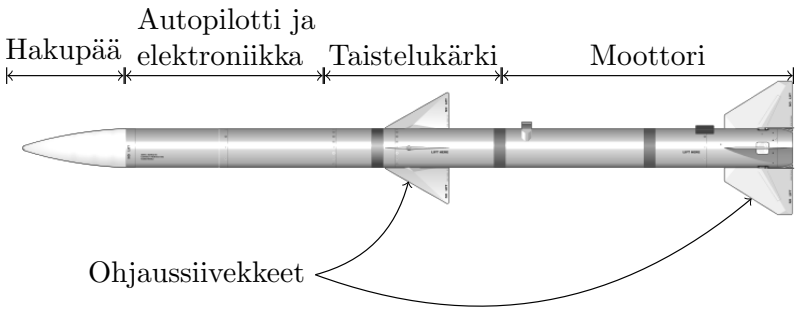
Ohjuksen lento jaetaan usein kolmeen vaiheeseen: *boost*, *mid-course* ja *terminal*. Boost-vaiheessa ohjus käyttää joko kaiken tai valtaosan sen polttoaineesta saavuttaakseen suuren nopeuden sekä halutun lentokorkeuden. Mid-course -vaiheessa ohjus liitää kohti kohdettaan hyödyntäen ohjuksen rungon aerodynaamisia muotoja ja pyrkii pääsemään tarpeeksi lähelle kohdetta, jotta terminal-vaihe voidaan käynnistää. Terminal-vaihe on ohjuksen lennon viimeinen ja tärkein vaihe, jossa ohjus tekee viimeiset liikkeet päästäkseen räjähdysetäisyydelle kohteesta. Lennon eri vaiheissa voidaan hyödyntää erilaisia hakeutumisjärjestelmiä ja esimerkiksi lyhyen matkan ohjuksissa ei välttämättä ole mid-course -vaihetta ollenkaan. (Siouris 2004, s. 177) Kuvassa 3.1 on esitettynä ohjuksen lennon eri vaiheet.



Kuva 3.1 *Ohjuksen lennon vaiheet.*

3.2 Rakenne

Ohjuksen runko on pitkä ja ohut sylinterimäinen kappale, joka kapenee kärkeään kohti. Ohjuksen rungon sisällä on ohjuksen hakupää, autopilotti, taistekukärki, moottori sekä muuta ohjuksen toimintaan tarvittavaa elektroniikkaa. Runkoon on lisäksi kiinnitetty siivekkeitä joiden avulla ohjus voi muuttaa liikesuuntaansa. Edellä mainitut hakeutuvan ohjuksen osat on esitetynä kuvassa 6.3 paikoilla, joissa ne yleisesti sijaitsevat. Työssä käytetty ohjuksen kuva on lähteestä (Wikimedia 2009).



Kuva 3.2 *Yleinen ilmatorjunta- ja ilmataisteluohjuksen rakenne.*

Ensimmäisenä ohjuksen kärjessä sijaitsee hakupää, joka pyrkii jäljittämään kohdetta, sekä rakenteet ja elektroniset komponentit, joita hakupää toiminnassaan tarvitsee. Hakupään jälkeen ohjuksen rungossa on autopilotti ja muu ohjuksen ohjaamiseen tarvittava elektroniikka. Autopilotin takana on taistelukärki sekä taistelukärjen laukaisemiseen tarvittavat komponentit. Viimeisenä ohjuksen perässä on ohjuksen työntövoiman tuottava moottori. Ohjuksen eri osien toiminnot on tarkemmin kuvattu seuraavissa alakohdissa.

3.2.1 Hakupää

Ohjuksen hakupää on usein ohjuksen kärjessä sijaitseva anturi, jonka tehtävänä on havaita kohde sen tuottaman tai heijastaman energian perusteella. Jos jokin ulkoinen järjestelmä huolehtii kohteen havaitsemisesta, voi anturin tilalla olla antenneja, joiden avulla vastaanotetaan ulkoisen järjestelmän antamia käskyjä. Hakupään anturi on kiinnitettynä kehykseen, joka voi pyöriä usean akselin ympäri. Kehyksen avulla anturi voidaan kohdistaa vakaasti tiettyyn suuntaan ja säilyttää anturin suuntautuminen ohjuksen suorittaessa liikeradan korjauksia. Kehys voi kääntyä ohjuksen keskiviivan suhteen maksimissaan noin 40–60 asteen kulmaan, jonka vuoksi kehystä ei aina voida suunnata kohdetta kohti ja ohjuksen hakeutuminen keskeytyy. Hakupäihin kiinnitetyt anturit ovat usein joko optisia tai radio-aaltoihin perustuvia ratkaisuja.

Optiset anturit havaitsevat ultraviolettisäteilyä, näkyvää valoa tai infrapunasäteilyä. Näkyvän valon aallonpituuksia hyödyntävillä antureilla voidaan kohteen havaitsemisessa hyödyntää kohteen pinnasta heijastuvaa auringon valoa. Ultravioletiaallonpituuksia käyttävät anturit havaitsevat kohteen taustalta säteilevää ultraviolettivaloa ja kohde näkyy tummana esteenä taustalta tulevan säteilyn edessä. Infrapuna-anturit havaitsevat kuumista kohdista, kuten pakokaasuista tai lämpenevistä metalliosista, säteilevää infrapunasäteilyä. Optisissa hakupäissä on lisäksi teleskooppi, jonka avulla kohteen säteily saadaan kohdistettua tarkasti anturiin.

Radioaaltoja hyödyntävät hakupäät sisältävät tutkan, joka vastaanottaa kohteesta heijastuvia radiosignaaleja. Signaalit voivat olla ohjuksessa sijaitsevan lähettimen tuottamia tai ne voivat olla peräisin ulkoisesta järjestelmästä, joka lähettää radioaaltoja kohteeseen. Vastaanotetut radioaallot voivat olla peräisin myös kohteen omista radiolähettimistä. Radioaallot voidaan lähettää kohteeseen pulsseina, jolloin kohteen etäisyys voidaan määrittää pulssin lähetyksen ja heijastuneen pulssin vastaanoton välissä kuluneen ajan avulla. Aallot voidaan lähettää myös jatkuvana, jolloin lähetetyssä ja vastaanotetussa signaalissa ilmenevän Doppler-ilmiön avulla voidaan määrittää kohteen suhteellinen nopeus.

3.2.2 Autopilotti ja ohjaus

Ohjuksen autopilotin tehtävänä on laskea joko hakupään tai jonkin muun järjestelmän antamien tietojen avulla kiihtyvyydet, joiden avulla ohjus saadaan lopulta osumaan kohteeseensa. Autopilotti pyrkii myös tasapainottamaan ohjuksen lentoa sekä varmistamaan, ettei siihen kohdistu suurempia voimia kuin mitä ohjuksen runko voi

enimmillään kestää.

Autopilotti välittää tarvittavat kiihtyvyydet lentoradan korjaamiseksi joko siipipintojen tai moottorin ohjaajalle. Siipipintojen ohjaaja muuntelee siipipintojen asentoja kunnes ohjukseen kohdistuvan nosteen aiheuttama kiihtyvyys vastaa autopilotin tarvitsemia kiihtyvyyksiä. Vastaavasti joissakin ohjuksissa moottorin tuottaman työntövoiman suuntaa on mahdollista muunnella siten, että ohjus kääntyy asentoon, jossa ohjukseen ilmasta aiheutuvat nostevoimat tuottavat autopilotin vaatimat kiihtyvyydet.

3.2.3 Taistelukärki ja sytytin

Taistelukärki on ohjuksen osa, jonka tarkoituksena on lamaannuttaa tai tuhota ohjuksen kohde. Yleisimmin taistelukärki koostuu räjähdysaineesta, joka on ympäröity metallisella kotelolla. Kun taistelukärki räjähtää, metallikotelosta irtoavat sirpaleet syöksyvät erittäin suurella nopeudella ohjusta ympäröiviin suuntiin. Terävät sirpaleet aiheuttavat vahinkoa paljon suuremmalla säteellä kuin mitä pelkällä räjähdyksen aiheuttamalla paineella voitaisiin saavuttaa. Sirpaleiden kasvattama räjähdyksen säde parantaa ohjuksen mahdollisuuksia kohteen tuhoamiseen, sillä useasti ohjukset eivät osu suoraan kohteeseen vaan ohittavat sen lähietäisyydeltä.

Jos taistelukärjen räjähdys annetaan vapaasti purkautua pallomaisena jokaiseen suuntaan, purkautuu räjähdyksen energiaa ja sirpaleita myös sellaisiin suuntiin, joissa kohde ei liiku. Tämän vuoksi räjähdyksen suunta ja sirpaleiden leviäminen usein kohdistetaan haluttuihin suuntiin, jolloin räjähdyksen voimakkuus kohteeseen saadaan maksimoitua. Yleinen tapa suunnata räjähdysenergia on ohjuksen rungon ympäri, jolloin sirpaleet lentävät kohtisuoraan poispäin ohjuksen rungosta kehämäisenä kasvavana muodostelmana. Koska kehämäinen räjähdys leviää rungon ympäriltä, ei taistelukärkeä tarvitse sijoittaa ohjuksen kärkeen vaan se voi sijaita vapaassa kohdassa ohjuksen rungossa.

Taistelukärjen sirpaleet voivat myös olla putkia, jotka on hitsattu päistään toisiinsa kiinni. Tällöin räjähdyksessä sirpaleet muodostavat kasvavan ketjumaisen kehän, joka leikkaa kohteeseen laajoja uria heikentäen sen rakennetta. Kohteen liikkuessa suurella nopeudella ilmassa voivat aerodynaamiset voimat repiä osia kohteen heikentyneestä rungosta.

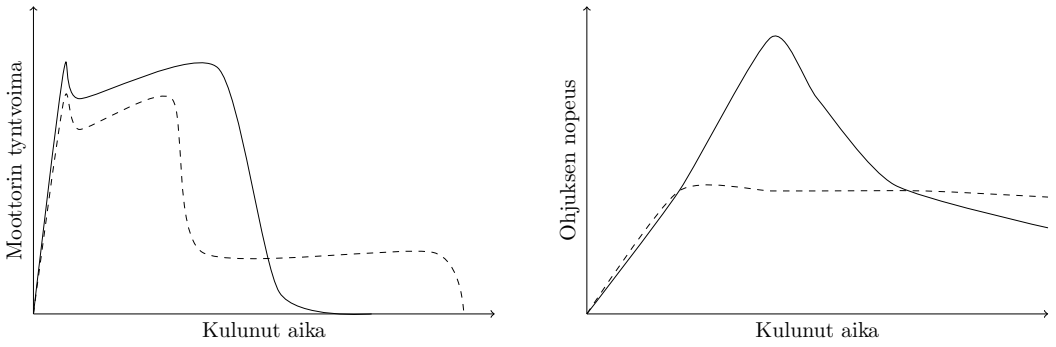
Taistelukärjen räjäytyksestä huolehtii ohjuksen sytytin. Sytyttimen tehtävänä on räjäyttää ohjuksen taistelukärki, kun kohde on ihanteellisessa kulmassa ja etäisyydellä taistelukärjen räjähdykseen nähden. Sytytin laskee ohjuksen sekä kohteen etäisyyksiä, nopeuksia ja kulmia, joiden avulla se päättää ajan jolloin räjäytys tuottaa

parhaimman lopputuloksen.

3.2.4 Moottori

Valtaosa ilmatorjuntaohjuksista käyttää moottorinaan kiinteää polttoainetta hyödyntävää rakettimoottoria. Kiinteä polttoaine on raemaista ainetta, jota poltetaan ohjuksen polttokammiossa. Palossa syntyneet kaasut ohjataan ohjuksen perässä olevasta suuttimesta ulos suurella nopeudella, jolloin ohjuksesta nopeasti poistuvat kaasuhiukkaset aiheuttavat Newtonin kolmannen lain mukaan vastakkaisen työntövoiman ohjukseen.

Joissakin ohjuksissa rakettimoottori tuottaa mahdollisimman paljon työntövoimaa lennon alussa, jonka jälkeen moottori sammuu ja ohjus liitää kohteeseensa hyödyntäen aerodynaamisia voimia. Osa polttoaineesta voidaan myös säästää, jolloin se voidaan käyttää pienen työntövoiman ylläpitämiseen alun suuren työntövoiman jälkeen. Kuvassa 3.3 on esitettynä molempien moottorityyppien työntövoimat ajan suhteen sekä ohjuksen nopeus ajan suhteen.



***Kuva** 3.3 Vasemmanpuoleinen kuva esittää moottorin työntövoimaa ajan suhteen ja oikeanpuoleinen kuva esittää ohjuksen nopeutta ajan suhteen. Tasainen viiva kuvaa kaiken polttoaineen alussa käyttävän ohjuksen ominaisuuksia ja katkoviiva kuvaa polttoainetta säästävän ohjuksen ominaisuuksia.*

Polttoaineen palamisnopeutta voidaan säädellä muuntamalla kiinteän polttoaineen muotoa polttokammion sisällä. Esimerkiksi jättämällä sylinterimäinen reikä polttoaineen keskelle, kasvaa polttopinta-ala ja työntövoima, mutta polttoaine kuluu nopeammin loppuun.

3.3 Hakeutuminen ja hakeutumisjärjestelmät

Hakeutumisjärjestelmä koostuu kaikista niistä komponenteista, jotka ovat osallisena ohjuksen ohjaamisessa räjähdysetäisyydelle kohteesta. Järjestelmän tehtävänä on

tunnistaa kohde, määrittää kohteen tilatietoja, kuten sijainti ja nopeus, sekä tuottaa ohjukselle komentoja, joiden avulla ohjus tekee korjauksia lentorataansa. Ohjuksen lentoa ohjaavat hakeutumisjärjestelmät voidaan jakaa kahteen ryhmään: ulkoisiin hakeutumisjärjestelmiin ja sisäisiin hakeutumisjärjestelmiin. Kun ohjuksen lentorataa muuttavat komennot välitetään ohjukselle jostakin ulkoisesta lähteestä, kuten maassa olevasta tutka-asemasta, on kyse ulkoisesta hakeutumisjärjestelmästä. Vastaavasti jos ohjus itse jäljittää kohdettaan ja tuottaa omalla laskentayksiköllään tarvittavat ohjauskäskyt, on kyseessä sisäinen hakeutumisjärjestelmä. Taulukossa 3.1 on esitettynä kohdassa käsiteltävät hakeutumisjärjestelmät sekä ohjauslait.

Taulukko 3.1 *Luvussa esitettäviä ohjuksen hakeutumisjärjestelmiä ja ohjauslakeja.*

Ulkoiset järjestelmät	*(laskenta ei ohjuksessa)*
Komennetut	Sensori ja prosessointi ulkoisessa järjestelmässä.
Sensori ohjuksessa	Sensori ohjuksessa ja prosessointi ulkoisessa järjestelmässä. Ohjus lähettää sensorin tiedot ulkoiselle järjestelmälle.
Sisäiset järjestelmät	*(laskenta ohjuksessa)*
Aktiiviset	Säteilyn lähetys ja vastaanotto ohjuksessa.
Puoliaktiiviset	Etsijä ohjuksessa mutta vastaanotettu säteily lähetetty ulkoisesta järjestelmästä.
Passiiviset	Kohde havaitaan sen tuottaman säteilyn tai äänen perusteella.
Ohjauslait	
Törmäyksen ennakointi	Ennakoidaan törmäyspiste johon suunnataan.
Takaa-ajo	Ohjuksen nopeusvektori suunnataan kohteeseen.
Sädeohjaus	Ohjus kulkee ulkopuolisen tuottamaa sädettä pitkin.
Suhteellinen navigointi	Kappaleiden välisen suoran suuntautuminen pysyy vakiona.

3.3.1 Ulkoiset hakeutumisjärjestelmät

Kaikki ohjuksen hakeutumiseen tarvittavat toiminnot voivat olla siirrettynä jonkin ulkoisen järjestelmän vastuulle. Tällaistä hakeutumista kutsutaan *komennetuksi*

hakeutumiseksi. Komennetussa hakeutumisessa kohteen jäljittäminen sekä lentoradan korjauskomentojen tuottaminen hoidetaan ulkoisen järjestelmän komponenttien avulla. Lopulliset komennot lähetetään ulkoisesta järjestelmästä ohjukselle, joiden avulla ohjus muuntaa lentorataansa. Komennetussa hakeutumisessa ulkoisen järjestelmän tulee jäljittää sekä ohjusta että kohdetta, jotta niiden välisiä etäisyyksiä ja nopeuksia voidaan laskea. Jos jäljitettävät kappaleet ovat kaukana jäljittävästä laitteesta, kasvaa mittausvirhe suureksi eikä ohjus osu tarkasti kohteeseensa.

Hakeutumista voidaan tarkentaa asentamalla kohdetta etsivä anturi ohjuksen sisään. Tällöin anturi on huomattavasti lähempänä mitattavaa kohdetta ja mittaustulokset tarkentuvat. Ohjuksen sisään sijoitetun anturin avulla voidaan lisäksi mitata suoraan ohjuksen ja kohteen välisiä suhteellisia suureita ilman erillisiä laskutoimituksia. Ohjus lähettää anturiltaan saamat suureet ulkopuoliselle järjestelmälle, joka laskee tarvittavat lentoradan muutoskomennot, jotta ohjus lopulta törmää kohteeseensa. Ulkoinen järjestelmä lähettää komennot takaisin ohjukselle, joka muuntaa niiden avulla lentorataansa.

3.3.2 Ohjuksen sisäiset hakeutumisjärjestelmät

Sijoittamalla hakeutumiseen tarvittavat komponentit ohjuksen sisään, voidaan mittaustuloksia saada tarkemmiksi ja ohjuksen reagointiaikoja pienemmiksi. Sisäiset hakeutumisjärjestelmät voidaan jakaa aktiivisiin, puoliaktiivisiin ja passiivisiin järjestelmiin.

Aktiivisissa hakeutumisjärjestelmissä tarvittavat komponentit kohteen havaitsemiseen ja jäljittämiseen sijaitsevat ohjuksen sisällä. Ohjus voi esimerkiksi lähettää radioaaltoja kohdetta kohti jolloin kohteesta heijastuvat radioaallot voidaan vastaanottaa ohjuksen hakupäässä sijaitsevalla tutkalla. Aktiiviset hakeutumisjärjestelmien vaatimat lisäkomponentit lisäävät ohjuksen kustannuksia sekä painoa. Ohjuksen lähettämät signaalit lisäksi tekevät siitä helpommin havaittavan, sillä lähetetty säteily voidaan havaita myös kohteessa sijaitsevilla antureilla.

Puoliaktiivisissa hakeutumisjärjestelmissä ohjuksen kärjessä oleva hakupää vastaanottaa kohteesta heijastavaa säteilyä, joka on alunperin lähetetty esimerkiksi maassa sijaitsevasta tutka-asemasta. Puoliaktiivisessa hakeutumisessa ulkoisen tutkan tulee jatkuvasta valaista kohdetta, jotta ohjus voi hakeutua siihen. Koska maassa olevien tutkien lähettämän signaalin teho voi olla hyvin suuri, voi puoliaktiivisten järjestelmien avulla hakeutua kohteisiin pitkien etäisyyksien päästä.

Passiivisissa hakeutumisjärjestelmissä kohde havaitaan sen tuottaman tai siitä heijastuvan energian avulla. Kohteesta voidaan havaita esimerkiksi moottorin läm-

pösäteilyä, kohteesta heijastuvaa auringonvaloa tai kohteen tuottamia ääniä.

3.4 Ohjauslait

Kun ohjus on saanut tietoja kohteestaan antureiden avulla, tulee vielä määrittää tarvittavat komennot, joilla ohjus saadaan osumaan kohteeseensa. Ohjuksen ohjaaminen kohteeseen on monimutkainen tehtävä, sillä sekä ohjus että sen kohde voivat liikkua suurella nopeudella tehden samalla useita väistöliikkeitä. Kappaleiden liikkeet huomioivia, sekä niiden avulla käskyjä tuottavia, prosesseja kutsutaan *ohjauslaeiksi*.

3.4.1 Ohjauslait yleisesti

Ohjauslakeja on olemassa paljon ja monesta ohjauslaista on tehty useita muunnelmia. Yksinkertaisessa *törmäyspisteen ennakoinnissa* lasketaan ohjuksen nopeuden, ohjuksen sijainnin, kohteen sijainnin ja kohteen nopeuden avulla piste, jossa törmäys tulee tapahtumaan ja suunnataan ohjus pisteeseen. Törmäyspisteen arvioiminen on kuitenkin hyvin hankalaa ohjuksen sekä kohteen lentoradan muutosten vuoksi. *Takaa-ajo* -ohjauslaissa ohjuksen nopeusvektori pyritään suuntaamaan jatkuvasti kohdetta kohti, jolloin ohjus lopulta saavuttaa kohteen, jos sen nopeus on suurempi kuin kohteen nopeus. *Sädeohjauksessa* jokin ulkopuolinen järjestelmä tuottaa säteen kohteeseen, jota pitkin ohjus pyrkii kulkemaan. Kun ohjuksen hakupää tai jokin ulkoinen sensori havaitsee, ettei ohjus kulje tarpeeksi lähellä sädettä, ohjauskäskyt ohjaavat ohjuksen kulkemaan jälleen sädettä pitkin. *Suhteellinen navigointi* on laajasti käytetty laki ja se perustuu ajatukseen, jossa kappaleet ovat törmäyskurssilla, jos ne lähestyvät toisiaan ja niiden välisen suoran suuntautuminen pysyy vakiona.

3.4.2 Takaa-ajo-ohjauslaki

Takaa-ajo-ohjausleissa ohjus pyrkii jatkuvasti suuntautumaan siten, että sen nopeusvektori osoittaa kohti ohjuksen kohdetta. Piste, johon ohjuksen nopeusvektori suunnataan, voi olla joko keskellä kohdetta tai kohteen edessä. Jos kohde tai ohjus liikkuu ja nopeusvektori ei enää osoita haluttuun pisteeseen, takaa-ajo-ohjauslaki määrittää ohjukselle tarvittavat kiihtyvyydet joiden avulla nopeusvektori saadaan jälleen osoittamaan oikeaan suuntaan. Jos ohjuksen sijaintia ilmaistaan termillä $\boldsymbol{r}_m$ ja kohteen sijaintia termillä $\boldsymbol{r}_t$, saadaan niiden suhteellinen sijainti kaavalla $\boldsymbol{r}_r = \boldsymbol{r}_t - \boldsymbol{r}_m$. Vastaavasti niiden suhteellinen nopeus saadaan kaavalla $\boldsymbol{v}_r = \boldsymbol{v}_t - \boldsymbol{v}_m$. Jos ohjuksen

nopeusvektori ja ohjuksen sekä kohteen suhteellinen sijaintivektori eivät ole yhdensuuntaiset, voidaan tarvittava suunta yhdensuuntaisuuden korjaukseen määrittää kaavalla

$$\boldsymbol{e}_l = \frac{\boldsymbol{v}_m \times \boldsymbol{u}}{\|\boldsymbol{v}_m \times \boldsymbol{u}\|}, \tag{3.1}$$

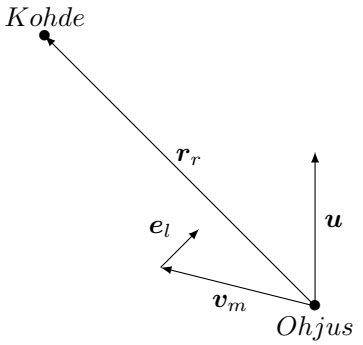
jossa vektori $\boldsymbol{u}$ on määritelty kaavan

$$\boldsymbol{u} = \frac{\boldsymbol{r}_r}{\|\boldsymbol{r}_r\|} \times \boldsymbol{v}_m \tag{3.2}$$

avulla. Lopullinen tarvittava kiihtyvyysvektori saadaan kaavalla

$$\boldsymbol{a}_{vp} = \frac{G\|\boldsymbol{u}\|}{max(t_{go}, 1)}\boldsymbol{e}_l, \tag{3.3}$$

jossa G on vahvistuskerroin ja $t_{go} = -(\boldsymbol{r}_r \cdot \boldsymbol{v}_r)/(\boldsymbol{v}_r \cdot \boldsymbol{v}_r)$ (Siouris 2004, s. 182–186). Suunnan muodostuminen on esitettynä myös kuvassa 3.4



Kuva 3.4 *Takaa-ajo-ohjauslaki.*

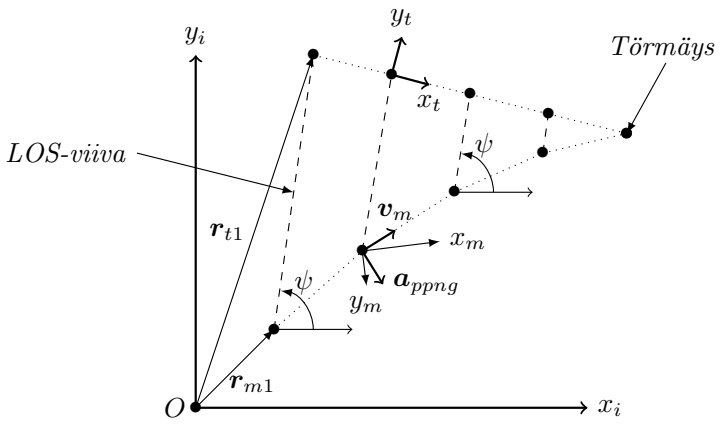
Takaa-ajo-ohjauslakia käytettäessä tulee ohjuksen nopeuden olla paljon kohteen nopeutta suurempi, jotta ohjus voi saavuttaa kohteensa. Ohjauslaki myös vaatii sitä suurempia ohjauskiihtyvyyksiä mitä lähemmäksi ohjus kohdettaan pääsee. Ohjus ei välttämättä kykene tuottamaan ohjauslain vaatimia kiihtyvyyksiä hakeutumisen viimeisillä hetkillä, josta seuraa suuria ohitusetäisyyksiä. (Siouris 2004, s. 182–186)

3.4.3 Suhteellinen navigointi -ohjauslaki

Suhteellinen navigointi -ohjauslaki, eli Proportional Navigation (PN) -ohjauslaki, on hakeutuvissa ohjuksissa laajasti käytössä oleva hakeutumisalgoritmi. Se perustuu ilmiöön, jossa kaksi toisiaan lähestyvää kappaletta lopulta törmäävät, jos niiden välinen *tähtäysviiva*, eli Line of Sight (LOS) -viiva, pysyy koko lähestymisen ajan samansuuntaisena paikallaan pysyvän koordinaatiston suhteen. PN-ohjauslaki pyrkii

siis määrittämään sellaisia kiihtyvyyksiä, joiden avulla LOS-viivan suuntautumisen muutokset saadaan mitätöityä. Ohjus tuottaa vaaditut kiihtyvyydet muuntamalla siivekkeidensä asentoa, jolloin aerodynaamiset voimat muuttavat ajan myötä ohjuksen sijaintia ja samalla myös LOS-viivan suuntautumista (Siouris 2004, s. 194).

PN-ohjauslakeja löytyy kirjallisuudesta useita, ja ne eroavat toisistaan riippuen mitä ominaisuuksia kohteen ja ohjuksen liikkeestä huomioidaan sekä mitä vektoria kohtisuoraan vaaditut kiihtyvyydet annetaan. *Puhdas suhteellinen navigointi* -ohjauslaissa, eli Pure Proportional Navigation (PPN) -ohjauslaissa, huomioidaan LOS-viivan muutosnopeus ja vaaditut kiihtyvyydet esitetään kohtisuorina kiihtyvyyksinä ohjuksen nopeusvektoria kohden (Siouris 2004, s. 195). Kuvassa 3.5 on esitettynä PPN-ohjauslain toimintaa.



Kuva 3.5 *PPN-ohjauslaki.*

Kuvan akselit y_i ja x_i kuvaavat paikallaan pysyvän koordinaatiston akseleita, akselit x_m ja y_m kuvaavat ohjukseen kiinnitetyn koordinaatiston akseleita ja akselit x_t sekä y_t kuvaavat kohteeseen liitetyn koordinaatiston akseleita. Kaikkien koordinaatistojen z-akselit osoittavat lukijaa kohden. Viivat kohteen ja ohjuksen reitillä olevien pisteiden välillä ovat LOS-viivoja, joiden kulma ψ pysyy vakiona koko lähestymisen ajan. Ohjuksen nopeusvektori on kuvattuna termillä $\boldsymbol{v}_m$ ja sitä kohtisuoraan on esitettynä PPN-ohjauslain tarvitsemat kiihtyvyydet termillä $\boldsymbol{a}_{ppng}$. Kuvassa on lisäksi esitettynä ohjuksen paikkavektori sen ensimmäisessä pisteessä termillä $\boldsymbol{r}_{m1}$ ja vastaavasti kohteen ensimmäinen paikkavektori samalla ajanhetkellä termillä $\boldsymbol{r}_{t1}$.

PPN-ohjauslain tarvitsemat kiihtyvyydet saadaan kaavalla

$$\boldsymbol{a}_{ppng} = N\boldsymbol{\Omega}_{LOS} \times \boldsymbol{v}_m, \tag{3.4}$$

jossa $\boldsymbol{a}_{ppng}$ on vaadittu kiihtyvyys LOS-viivan suunnan muutoksen kumoamiseen, N

on vahvistuskerroin, $\boldsymbol{v}_m$ on ohjuksen nopeusvektori ja LOS-viivan suuntautumisen muutosnopeus, eli kulmanopeus, saadaan kaavan

$$\boldsymbol{\Omega}_{LOS} = \frac{\boldsymbol{r}_r \times (\boldsymbol{v}_t - \boldsymbol{v}_m)}{\|\boldsymbol{r}_r\|^2}, \tag{3.5}$$

avulla. Kaavassa 3.5 termi $\boldsymbol{r}_r$ on kohteen paikkavektorin ja ohjuksen paikkavektorin välinen erotus $\boldsymbol{r}_m - \boldsymbol{r}_t$ (Guelman, Idan & Golan 1995, s. 839).

3.5 Simuloitujen ohjusten ominaisuudet

Työssä myöhemmin esitettävässä ohjussimulaattorissa toteutetaan ohjusmalli ja sille ohjuskonfiguraatiot lyhyille matkoille tarkoitetulle, vain terminal-vaiheen sisältävälle, ohjukselle sekä keskipitkille matkoille tarkoitetulle, terminal- ja mid-course -vaiheen sisältävälle, ohjukselle. Lyhyen matkan ilmataisteluohjus mukailee AIM-9 Sidewinder -ilmataisteluohjusta ja keskipitkän matkan ohjus pyrkii mukailemaan AIM-120 AMRAAM -ilmataisteluohjusta. Kummastakaan ohjuksesta ei ole saatavissa tarkkoja tietoja niiden mitoista, massoista tai muista ominaisuuksista, jonka vuoksi työssä käytettyjen ohjusten tiedot on koostettu löydetyistä arvioista sekä ohjuksista löydetyistä kuvista.

3.5.1 Ohjusmallin huomioimat ominaisuudet

Työssä toteutettavan ohjusmallin tavoitteena on tuottaa oikean ohjuksen kaltainen lentorata tehokkuudella, joka sopii virtuaaliseen ympäristöön. Tämän vuoksi malli huomioi vain osan oikean ohjuksen toiminnallisuudesta, jonka lisäksi toiminnallisuuksissa on tehty yksinkertaistavia oletuksia, jotta ohjuksen mallinnuksesta saadaan riittävän tehokas. Taulukossa 3.2 on esitettynä ohjuksen ominaisuudet sellaisena, miten ne ohjusmallissa huomioidaan.

Malli näkee simuloimiensa ohjusten rakenteen symmetrisenä ja muotoaan muuttamattomana. Ohjukset ovat todellisuudessa hyvin lähellä symmetristä kappaletta, kun niiden pinnassa olevia mahdollisia antureita ei huomioida. Lisäksi oikeiden ohjusten muoto pysyy muuttumattomana, jos useampivaiheiset ohjukset jätetään huomioimatta. Oikeaan ohjukseen voi muodostua eri kohtiin esimerkiksi taipumista tai värähtelyä, mutta tämän työn ja työssä toteutettavan ohjusmallin kannalta on tärkeämpää selvittää kuinka ohjukseen vaikuttavat voimat ja momentit muuttavat sen lentorataa sekä sen asentoa. Tämän vuoksi ohjuksen pienet taipumiset jätetään mallissa huomioimatta ja työssä ohjus oletetaan jäykäksi kappaleeksi, jonka massa ei

Taulukko 3.2 *Ohjuksen ominaisuudet, jollaisena ohjusmalli ne huomioi.*

Rakenne	Jäykkä, symmetrinen ja muotoaan muuttamaton kappale, jonka massa pysyy vakiona.
Hakupää	Määrittää kohteen sijainnin ja nopeuden riippumatta häiriöistä tai ominaisuudesta johon hakeudutaan.
Autopilotti	Yksinkertainen erosuureen huomioiva säädin. Ohjausmomentit tuotetaan suoraan runkoon.
Sytytin	Räjäytys etäisyyden perusteella.
Taistelukärki	Räjähtää ympyrämäisenä.
Moottori	Polttoaine palaa tasaisesti tuottaen tasaisen työntövoiman.
Hakeutumislaki	Suhteellinen navigointi.

muutu lennon aikana. Todellisuudessa ohjuksen massa vähenee polttoaineen kuluessa. Pienemmällä massalla ohjus voi tehdä jyrkempiä käännöksiä, mutta jättämällä massan muutos huomioimatta, yksinkertaistuvat työssä myöhemmin määriteltävät liikeyhtälöt.

Mallin ohjausjärjestelmä olettaa, että sen määräämistä ohjauskäskyistä aiheutuu välittömästi ohjuksen runkoon kohdistuvia ohjausmomentteja, joiden suuruus riippuu lineaarisesti ohjuksen nopeudesta. Oikeiden ohjusten ohjausjärjestelmät lähettävät ohjauskäskyjä ohjaussiivekkeille, jotka muuttavat asentoaan ja samalla aiheuttavat esimerkiksi ohjuksen nopeudesta riippuvia ohjausvoimia ohjukseen. Oikean ohjuksen ohjauskäskyjen ja tapahtuneen ohjauksen välissä on viivettä, jonka vaikutus ohjuksen lentorataan kuitenkin oletetaan pieneksi ja siten jätetään ohjusmallissa huomioimatta. Ohjausjärjestelmä on ohjusmallissa toteutettu yksinkertaisen säätimen avulla, joka ottaa sisäänmenonsa asetusarvon ja mitatun arvon välisen erotuksen. Ohjusmallissa erotus lasketaan ohjauslain tarvitsemien kiihtyvyyksien ja siipipintojen tuottamien nostevoimien aiheuttamien kiihtyvyyksien välillä. Säätimen ulostuloa käyttäen tuotetaan ohjausmomentteja, joiden avulla ohjuksen asentoa, ja siten myös ohjukseen kohdistuvien aerodynaamisten voimien suuruutta, voidaan säätää.

Ohjusmallissa ei keskitytä ohjuksen räjähdyksen mallinnukseen, jonka vuoksi malli olettaa ohjuksen sytyttimen huomioivan vain etäisyyden kohteeseen ja ohjuksen räjähdyksen ympyrämäiseksi sekä sirpaleettomaksi. Todennäköisyys kohteen tuhoamiselle tai sen vaurioittamiselle toteutetaan kohteen ja räjähdyksen välisen etäisyyden perusteella. Tämän lisäksi räjähdyksen aiheuttamien tuhojen tarkastelu

ei ole ohjussimulaattorin vastuulla. Räjähdyksestä julkaistaan tieto muille simulaation simulaattoreille, jotka päättävät räjähdyksen parametrien, kuten taistelukärjen tyypin perusteella millaista tuhoa räjähdys aiheutti sen simuloimille olioille.

Ohjusmallin moottori tuottaa tasaisen työntövoiman eikä esimerkiksi suuttimen muotoa tai polttoaineen epätasaista palamista huomioida työntövoiman suuruudessa. Todellisessa ohjuksessa työntövoimaan vaikuttaa moni tekijä, mutta koska työntövoima vaikuttaa vain ohjuksen lennon alkuvaiheessa, on tämän työn kannalta merkittävämpää lopullinen nopeus, johon moottori ohjuksen kiihdyttää.

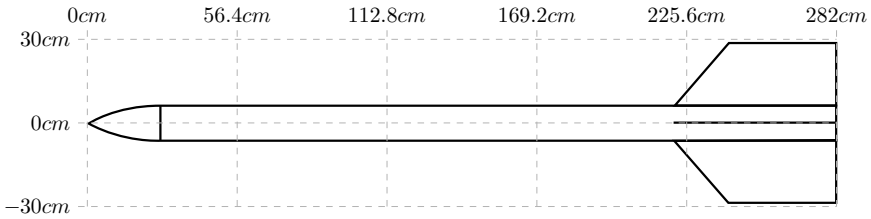
Ohjusmallin ohjukset hakeutuvat vain kohteen tietojen perusteella eikä esimerkiksi hakupään häiriöitä huomioida. Simulaatiossa kohteet eivät pyri häiritsemään ohjuksen lentoa ja kohteiden suorittamien väistöliikkeiden oletetaan olevan merkittävempiä kuin esimerkiksi kohinan aiheuttamat häiriöt, jonka vuoksi ne jätetään ohjusmallissa huomioimatta. Myös säteilyn tyyppi, johon ohjus hakeutuu, jätetään huomioimatta, sillä käytännössä ohjus hakeutuu samaan pisteeseen riippumatta säteilyn tyypistä. Lisäksi ohjuksen hakeutumisen vaatimat laskutoimitukset lasketaan ohjusmallissa eikä esimerkiksi ampuvan lentokoneen simulaattorissa, joka välittäisi hakeutumiskomentoja sisältäviä viestejä ohjukselle. Kehyksenä toimivan järjestelmän rakenteen vuoksi hakeutumiskomentoihin liittyvä kommunikaatio lentokoneen ja ohjuksen välillä olisi monimutkaista, jonka vuoksi laskutoimitusten suorittaminen lentokoneessa rajataan ohjusmallista pois.

3.5.2 Simuloidut ohjukset

Kuvassa 3.6 on esitettynä työn simulaatiossa käytettävän lyhyen matkan ohjuksen ominaisuudet, ja kuvassa 3.7 on esitettynä simulaatiossa käytettävän keskipitkän matkan ohjuksen ominaisuudet.

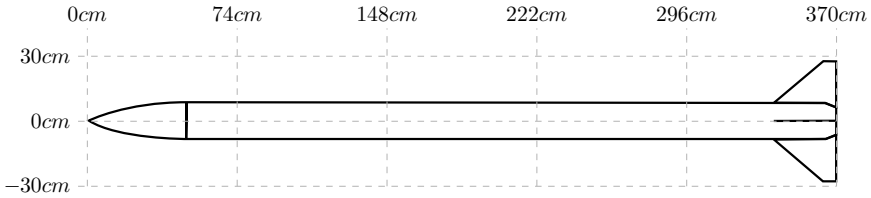
Lyhyen matkan ohjuksen moottori tuottaa ohjuksen lennon alkuvaiheessa työntövoiman, jonka jälkeen se liitää kohteeseen. Moottori pyrkii mukailemaan alakohdassa 3.2.4 esitettyä moottorityyppiä, jossa ei säästetä polttoainetta. Keskipitkän matkan ohjuksen moottori mukailee toista moottorityyppiä, jossa alussa tuotetaan suuri työntövoima ja myöhemmin ohjuksen nopeutta ylläpitävä työntövoima.

Lyhyen matkan ohjuksen mallina olevassa AIM-9:ssä käytetään alakohdassa 3.3.2 esitettyä passiivista infrapunasäteilyä hyödyntävää hakupäätä. Ohjusmallissa ohjus kuitenkin hakeutuu kohteen sijainnin perusteella eikä se huomioi kohteesta lähtevää säteilyä. Keskipitkän matkan ohjuksen mallina toimivassa AIM-120:ssa terminalvaihe toimii alakohdassa 3.3.2 esitetyn aktiivisen hakeutumisjärjestelmän avulla, jonka lisäksi ohjus hakeutuu mid-course -vaiheessa alussa kohteesta saatujen ja mah-



Massa	$85.28kg$
Pituus	$2.82m$
Rungon halkaisija	$0.13m$
Moottori	$0 - 3s : 23000N$
Hakeutuminen	Terminal-vaihe: lämpöhakuinen

Kuva 3.6 *Simulaatiossa käytetyn lyhyen matkan ohjuksen muoto ja ominaisuudet.*



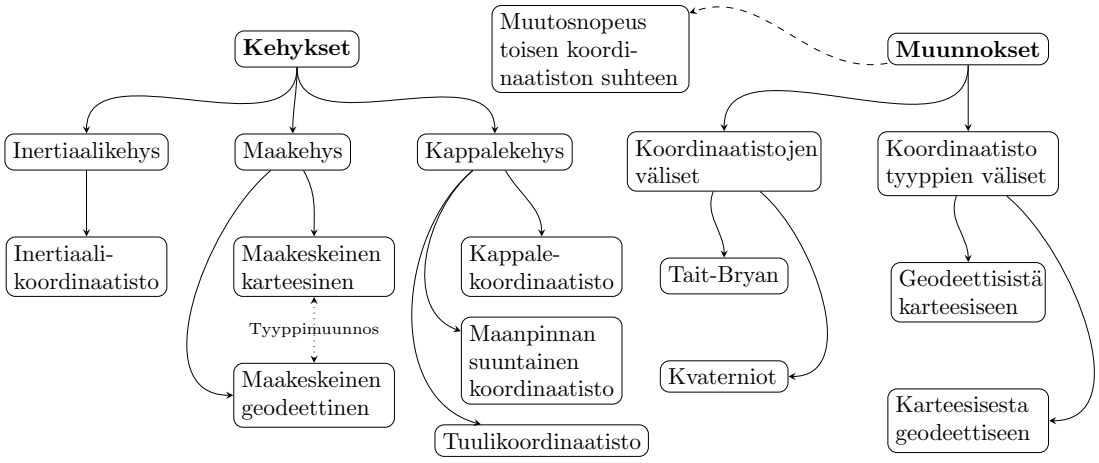
Massa	$152.00kg$
Pituus	$3.70m$
Rungon halkaisija	$0.18m$
Moottori	$0 - 10s : 16000N$
	$10 - 30s : 1800N$
Hakeutuminen	Midcourse-vaihe: komennettu
	Terminal-vaihe: aktiivinen tutka

Kuva 3.7 *Simulaatiossa käytetyn keskipitkän matkan ohjuksen muoto ja ominaisuudet.*

dollisten uusien, laukaisijasta lähetettyjen, tietojen perusteella. Ohjus myös pyrkii nousemaan mid-course -vaiheessa korkeammalle, jotta ilmanvastus olisi pienempi ja ohjus voisi lentää pidemmälle. Mid-course -vaihe keskipitkän matkan ohjukselle toteutetaan mallissa siten, että ohjus ennakoi kohteen nopeuden sekä oman nopeutensa avulla pisteen, jossa törmäys tapahtuu, ja suuntaa sitä kohden. Ennakoitua pistettä päivitetään muutaman sekunnin välein, joka vastaa tilannetta, jossa lentokone keräisi tutkalla tietoja kohteesta sekä ohjuksesta ja lähettäisi niiden perusteella ohjukselle uuden törmäyspisteen. Terminal-vaihe on toteutettu keskipitkän matkan ohjuksessa vastaavalla tavalla kuin lyhyen matkan ohjuksessa. Molempien ohjustyyppien kaikissa lentovaiheissa hyödynnetään alakohdassa 3.4.3 esitettyä suhteellista navigointia.

4. KEHYKSET JA KOORDINAATISTOT

Jotta ohjuksille voidaan muodostaa seuraavassa luvussa esiteltävät liikeyhtälöt tulee ensin muodostaa käsitteet, joiden avulla voidaan kuvata eri kappaleiden suhteen määriteltyjä ominaisuuksia. Nämä käsitteet pitävät sisällään *kehykset*, jotka kuvaavat fyysisiä kappaleita, ja niihin liitetyt *koordinaatistot*, jotka mahdollistavat kappaleiden ominaisuuksien ilmaisemisen sekä suuntien ja suuruuksien avulla laskemisen. (Zipfel 2007, s. 55–57) Kuvassa 4.1 on koottuna kaikki luvussa esiteltävät ja tässä työssä tarvittavat kehykset, koordinaatistot ja niiden väliset muunnokset. Luvussa esitettävät asiat ovat ohjusmallinnuksen kannalta välttämättömiä, sillä niitä hyödynnetään sekä ohjuksen liikkeen noudattamien yhtälöiden määrittämisessä että simulaatiossa olevien olioiden tilan, kuten sijainnin tai nopeuden, kuvaamisessa. Tilan kuvaamisen lisäksi luvussa esitettäviä työkaluja hyödynnetään muiden simulaattoreiden lähettämien tilatietojen muokkaamisessa ohjussimulaattorin ymmärtämään muotoon.

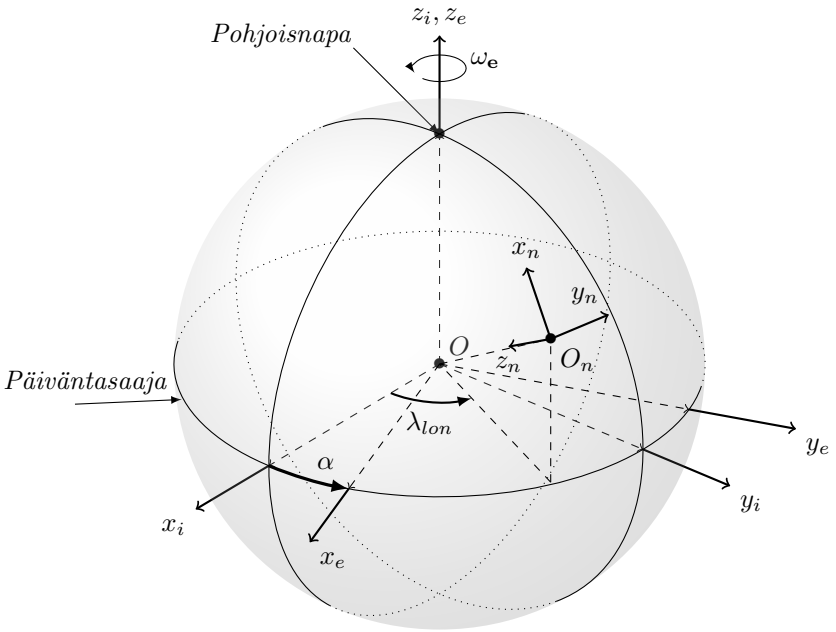


Kuva 4.1 *Työssä käytetyt kehykset, koordinaatistot ja niiden väliset muunnokset.*

4.1 Maakeskeinen inertiaalikehys

Inertiaalikehys on sellainen kehys, jonka pisteet eivät ole kiihtyvässä liikkeessä ja jossa Newtonin toinen laki on voimassa. Inertiaalikehykseen liitettyä koordinaatistoa kutsutaan *inertiaalikoordinaatistoksi*. Käyttötarkoitus usein määrittelee mikä kehys voidaan arvioida olevan riittävällä tarkkuudella sovelluksessa käytetty inertiaalikehys. Esimerkiksi planeettojen välisissä lennoissa inertiaalikehys voidaan määrittää auringon ja kaukaisten tähtien avulla, kun taas maan ilmakehässä tapahtuvissa lennoissa maan keskipisteeseen voidaan liittää riittävällä tarkkuudella toimiva inertiaalikehys. (Zipfel 2007, p. 57)

Inertiaalikoordinaatistoa, joka on kiinnitetty maakeskeiseen inertiaalikehykseen kutsutaan *Earth-Centered Inertial (ECI)* -koordinaatistoksi (Noureldin et al. 2013, s. 27–33). Kuvassa 4.2 ECI-koordinaatisto on esitettynä origon O sekä akseleiden x_i, y_i ja z_i avulla.



Kuva 4.2 *Työssä hyödynnettäviä kehyksiä sekä niihin liitettyjä koordinaatistoja.*

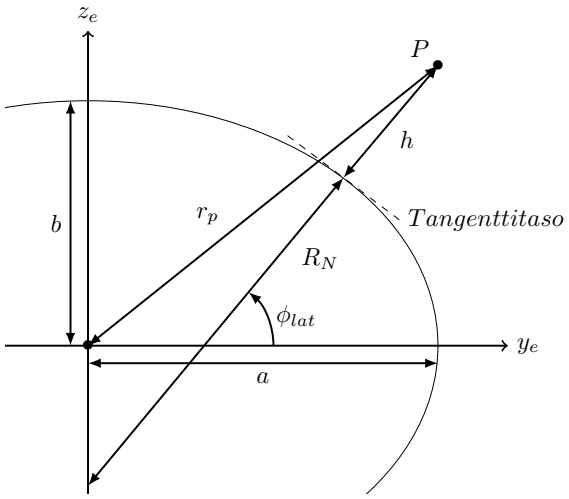
Koordinaatiston z_i-akseli kulkee maan keskipisteen sekä pohjoisnavan lävitse. x_i-akseli kulkee maan keskipisteen ja sellaisen pisteen lävitse, joka muodostuu kun päiväntasaaja ja taso, jolla maapallo kulkee auringon ympäri, leikkaavat. y_i-akseli valitaan siten, että koordinaatistosta muodostuu oikean käden koordinaatisto (Zipfel 2007, p. 59).

4.2 Maakehys

Maapalloa ja sen pisteitä kuvataan *maakehys* -nimisen kehyksen avulla. Maan muoto on monimutkainen ja se muuttuu jatkuvasti, minkä vuoksi sen pintaa tulee estimoida erilaisten geometristen kappaleiden avulla (Zipfel 2007, s. 386–391). Tässä työssä maan ajatellaan olevan *World Geodetic System (WGS-84)* -järjestelmän määrittelemä ellipsoidi, jolle on määritelty seuraavat ominaisuudet (Noureldin et al. 2013, s. 46–48):

- isoakselin puolikas $a = 6378137,0m$
- litistyssuhde $f = 0,00335281$
- pikkuakselin puolikas $b = a(1-f) = 6356752,3m$
- maan pyörimisnopeus $\omega_e = 7,2921150 \cdot 10^{-5}\ rad/s$.

Kuvassa 4.2 harmaa ellipsoidi esittää maakehystä, ja akseli z_e kuvaa akselia, jonka ympäri maapalloa kuvaava maakehys pyörii. Kuvassa 4.3 on esitettynä isoakselin sekä pikkuakselin puolikkaat.



Kuva 4.3 *WGS-84 järjestelmän ellipsoidi (Noureldin et al. 2013, p. 59).*

Kuvassa 4.2 origo O sekä akselit x_e, y_e ja z_e muodostavat koordinaatiston, joka on kiinnitetty maakehykseen ja siten pyörii maapalloa kuvaavan maakehyksen mukana. Tällaistä maakehykseen liitettyä karteesista koordinaatistoa kutsutaan *Earth-Centered Earth-Fixed (ECEF)* -koordinaatistoksi. ECEF-koordinaatisto muodostetaan siten, että sen z_e-akseli kulkee origosta pohjoisnavan lävitse, x_e-akseli on kohtisuorassa z_e-akselin kanssa sekä kulkee Greenwichin pituuspiirin lävitse ja y_e valitaan

siten, että koordinaatistosta muodostuu oikean käden koordinaatisto (Noureldin et al. 2013, s. 27–33).

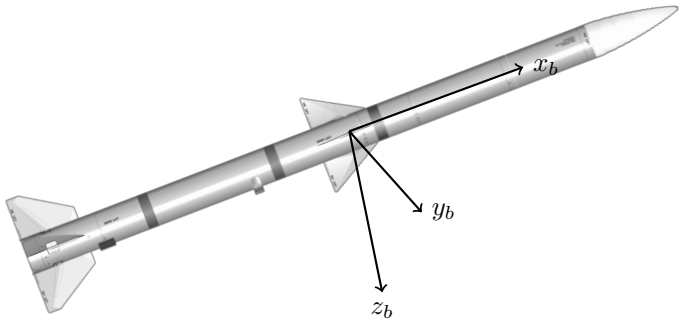
Edellä mainitun ECEF -koordinaatiston lisäksi maakehykseen liitetään usein myös toinen koordinaatisto nimeltään *geodeettinen koordinaatisto*. Geodeettinen koordinaatisto muodostaa maan pinnalle leveys- ja pituuspiirien avulla verkon, jonka avulla voidaan määrittää mikä vain piste maan pinnalta (Zipfel 2007, s. 71–74). Leveyspiirit ovat päiväntasaajan suuntaisia maapallon pinnalla olevia kehiä ja pituuspiirit ovat päiväntasaajaa vasten kohtisuorassa olevia kehiä.

Työssä käytettävä WGS-84 koordinaattijärjestelmä käyttää *geodeettisiä leveysasteita*. Geodeettiset leveysasteet määritellään kulmana, joka muodostuu pisteen ja ellipsoidipinnan normaalin läpi kulkevan suoran sekä päiväntasaajan väliin (Zipfel 2007, s. 386–391). Kuvassa 4.3 on esitettynä geodeettinen leveysaste termillä ϕ_{lat} ja pituusaste on esitettynä kuvassa 4.2 termillä λ_{lon}. Kappaleen korkeus esitetään geodeettisissa koordinaateissa korkeutena ellipsoidin pinnasta (Noureldin et al. 2013, s. 46–48). Kuvassa 4.3 geodeettinen korkeus on esitettynä termillä h.

4.3 Kappalekehys

Ohjuksia ja lentokoneita, kuvataan *kappalekehyksien* avulla. Kappalekehykseen voidaan kiinnittää useita erilaisia koordinaatistoja, joiden avulla esimerkiksi kappaleelle muodostettavia liikeyhtälöitä voidaan yksinkertaistaa.

Kappalekehykseen liitettävä karteesinen koordinaatisto, jota kutsutaan *kappalekoordinaatistoksi* muodostetaan kappaleen geometristen ominaisuuksien avulla (Zipfel 2007, s. 74–75). Kuvassa 4.4 on esitettynä ohjus, jonka massakeskipisteestä alkavat akselit x_b, y_b ja z_b muodostavat ohjuksen kappalekoordinaatiston.



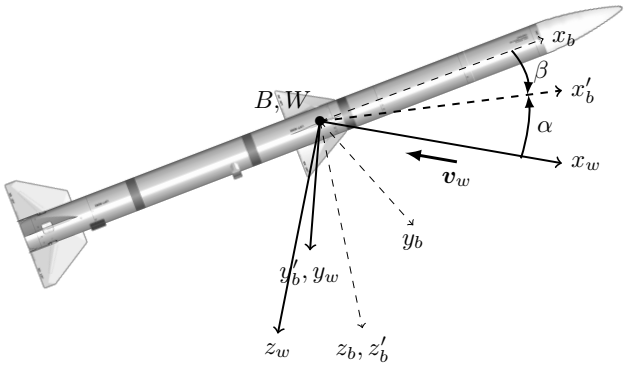
Kuva 4.4 *Ohjuksen karteesinen kappalekoordinaatisto B.*

Koordinaatiston x-akseli osoittaa ojuksen kärkeä kohti, y-akseli osoittaa oikealle

ja z-akseli osoittaa alaspäin. Kappalekoordinaatisto on kiinnitettynä kappalekehykseen ja siten liikkuu ja pyörii ohjuksen mukana. Kappalekoordinaatisto on hyödyllinen koordinaatisto kun lasketaan kappaleeseen vaikuttavien voimien sekä vääntömomenttien vaikutuksia ohjuksen liikkeeseen.

Maan ilmakehässä liikkuvien kappaleiden nopeudet ja suunnat ilmaistaan usein maanpinnan suhteen määriteltyinä. Esimerkiksi lentokoneen suuntautuminen on paljon luontevampaa ilmaista kulmina maanpinnasta kuin kulmina maan keskipisteeseen kiinnitetystä inertiaalikoordinaatistosta. Tämän vuoksi onkin hyödyllistä määritellä koordinaatisto, joka muodostaa maanpinnan suhteen tangenttitason kappaleen kussakin sijainnissa. Tällaistä koordinaatistoa, jonka origo on kiinnitetty kappaleen massakeskipisteeseen, x-akseli osoittaa aina pohjoiseen, y-akseli osoitttaa aina itään ja z-akseli osoittaa kohtisuoraan maanpinnan sisään, kutsutaan *North, East and Down (NED)* -koordinaatistoksi (Noureldin et al. 2013, s. 27–33). Kuvassa 4.2 on esitettynä NED-koordinaatisto akseleiden x_n, y_n ja z_n avulla.

Ohjukseen muodostuu ilmasta voimia, jotka riippuvat ohjuksen ohitse virtaavan ilman nopeudesta ja jotka ilmaistaan virtaavan ilman suuntaisina tai sitä kohtisuorina. Jotta voimia voidaan helposti laskea, liitetään ohjukseen koordinaatisto, jonka x-akseli osoittaa aina ohjuksen nopeusvektorin suuntaisesti. Tällaistä koordinaatistoa kutsutaan *tuulikoordinaatistoksi* (Stevens & Lewis 1992, s. 61–63). Kuvassa 4.5 on ohjukseen liitetty tuulikoordinaatisto.



Kuva 4.5 *Ohjuksen karteesinen kappalekoordinaatisto B sekä tuulikoordinaatisto W.*

Tuulikoordinaatiston origo W on samassa pisteessä kappalekoordinaatiston origon kanssa ja akseleita on kuvattu termein x_w, y_w sekä z_w. Akseli x_w on vastakkaisuuntainen ohjuksen ohitse kulkevan ilman nopeuden $\boldsymbol{v}_w$ kanssa. Kulmia kappalekoordinaatiston ja tuulikoordinaatiston välissä kutsutaan *kohtauskulmaksi* α ja

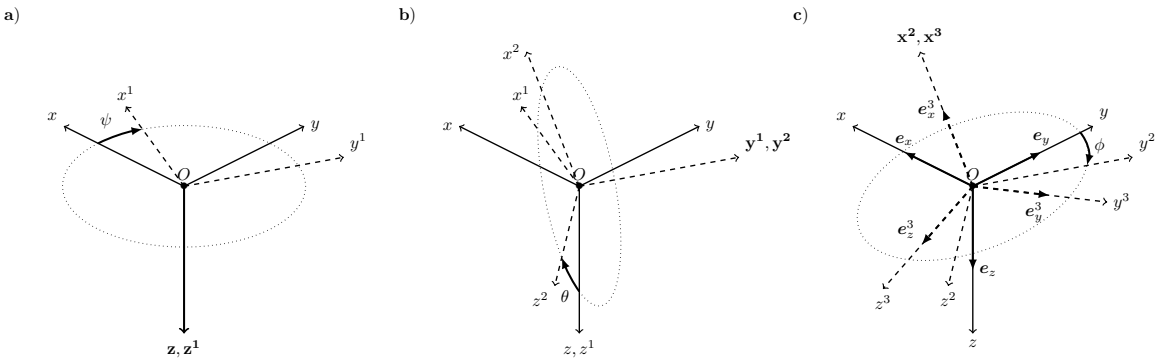
sivuluisukulmaksi β.

4.4 Koordinaatistomuunnokset

Usein vektorista tarvitaan sen esitys jonkin muun koordinaatiston suhteen kuin minkä suhteen sen esitys tunnetaan. Eräs tällainen tilanne on esimerkiksi nopeusvektori, joka on esitettynä kappalekehykseen kiinnitetyn koordinaatiston suhteen, mutta sen esitys tarvitaan maakehykseen kiinnitetyn koordinaatiston suhteen. Tällöin alkuperäinen vektori tulee muuntaa toisen koordinaatiston kantavektoreiden suhteen määritellyksi. Seuraavissa alakohdissa esitellään menetelmiä, joilla koordinaatistojen välinen muunnos voidaan suorittaa.

4.4.1 Tait-Bryan kulmat

Jonkin koordinaatiston B kantavektoreiden suuntautuminen jonkin toisen koordinaatiston O kantavektoreiden suhteen voidaan muodostaa kolmen peräkkäisen rotaation avulla. Rotaatiot suoritetaan kääntämällä koordinaatisto B yksitellen kukin akselin ymäri johonkin tiettyyn kulmaan kunnes saavutetaan lopullinen suuntautuminen. Kulmia, joihin akselit käännetään, merkitään symboleilla ψ, ϕ, θ ja niitä kutsutaan *Tait-Bryan kulmiksi*. (Wittenburg 2008, s. 9–14)



Kuva 4.6 *Koordinaatiston suuntautumisen esitys Tait-Bryan kulmien avulla. Kuvassa tummennettuina akselit, joiden ympäri koordinaatistoa käännetään.*

Kuvassa 4.6 on esitettynä Tait-Bryan kulmien avulla suoritettu muunnos koordinaatistosta toiseen. Kuvan kohdassa *a*) koordinaatisto O_{xyz} on kierretty z akselin ympäri kulman ψ verran, jolloin on saavuttu koordinaatistoon $O_{x^1y^1z^1}$. Kohdassa *b*) koordinaatistoa $O_{x^1y^1z^1}$ on kierretty akselin y^1 ympäri kulman θ verran, jolloin on saavuttu koordinaatistoon $O_{x^2y^2z^2}$. Lopuksi kohdassa *c*) koordinaatistoa $O_{x^2y^2z^2}$ on vielä kierretty akselin x^2 ympäri kulman ϕ verran, jolloin saadaan lopullinen

koordinaatisto $O_{x^3y^3z^3}$.

Ensimmäisessä muunnoksessa kääntyneen akselin x^1 kantavektori voidaan ilmaista vanhojen kantavektoreiden avulla, jolloin saadaan

$$\boldsymbol{e}_x^1 = cos(\psi)\boldsymbol{e}_x - sin(\psi)\boldsymbol{e}_y, \tag{4.1}$$

sillä vanha kantavektori $\boldsymbol{e}_x$ voidaan ajatella kolmion hypotenuusana ja kantavektori $\boldsymbol{e}_y$ kolmion yhtenä sivuna. Vastaavasti kantavektorille $\boldsymbol{e}_y^1$ saadaan kaava

$$\boldsymbol{e}_y^1 = sin(\psi)\boldsymbol{e}_x + cos(\psi)\boldsymbol{e}_y, \tag{4.2}$$

ja koska kantavektori $\boldsymbol{e}_z^1$ pysyy paikoillaan, vastaa se suoraan kantavektoria $\boldsymbol{e}_z$. Matriisimuodossa esitettynä saadaan

$$\begin{bmatrix} \boldsymbol{e}_x^1 \\ \boldsymbol{e}_y^1 \\ \boldsymbol{e}_z^1 \end{bmatrix} = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_x \\ \boldsymbol{e}_y \\ \boldsymbol{e}_z \end{bmatrix}, \tag{4.3}$$

jossa keskimmäistä matriisia kutsutaan *muunnosmatriisiksi* ja sen avulla voidaan muuntaa jonkin kannan suhteen määritelty vektori toisen kannan suhteen määritellyksi. Vastaavasti kantaan $\boldsymbol{e}_x^2$, $\boldsymbol{e}_y^2$, $\boldsymbol{e}_z^2$ siirtymiseksi saadaan muunnosmatriisi

$$\begin{bmatrix} \boldsymbol{e}_x^2 \\ \boldsymbol{e}_y^2 \\ \boldsymbol{e}_z^2 \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_x^1 \\ \boldsymbol{e}_y^1 \\ \boldsymbol{e}_z^1 \end{bmatrix}, \tag{4.4}$$

ja edelleen kantaan $\boldsymbol{e}_x^3$, $\boldsymbol{e}_y^3$, $\boldsymbol{e}_z^3$ siirtymiseen muunnosmatriisi

$$\begin{bmatrix} \boldsymbol{e}_x^3 \\ \boldsymbol{e}_y^3 \\ \boldsymbol{e}_z^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_x^2 \\ \boldsymbol{e}_y^2 \\ \boldsymbol{e}_z^2 \end{bmatrix}. \tag{4.5}$$

Kun edellä esitellyt muunnosmatriisit ilmaistaan termein $\boldsymbol{R}^1$, $\boldsymbol{R}^2$, $\boldsymbol{R}^3$ saadaan

kaava

$$\begin{bmatrix} \boldsymbol{e}_x^3 \\ \boldsymbol{e}_y^3 \\ \boldsymbol{e}_z^3 \end{bmatrix} = \boldsymbol{R}^3 \begin{bmatrix} \boldsymbol{e}_x^2 \\ \boldsymbol{e}_y^2 \\ \boldsymbol{e}_z^2 \end{bmatrix} = \boldsymbol{R}^3\boldsymbol{R}^2 \begin{bmatrix} \boldsymbol{e}_x^1 \\ \boldsymbol{e}_y^1 \\ \boldsymbol{e}_z^1 \end{bmatrix} = \boldsymbol{R}^3\boldsymbol{R}^2\boldsymbol{R}^1 \begin{bmatrix} \boldsymbol{e}_x \\ \boldsymbol{e}_y \\ \boldsymbol{e}_z \end{bmatrix} = \boldsymbol{R}^{31} \begin{bmatrix} \boldsymbol{e}_x \\ \boldsymbol{e}_y \\ \boldsymbol{e}_z \end{bmatrix}, \tag{4.6}$$

jossa $\boldsymbol{R}^{31}$ kuvaa muunnosta kannasta $\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z$ kantaan $\boldsymbol{e}_x^3, \boldsymbol{e}_y^3, \boldsymbol{e}_z^3$. Matriisimuodossa ilmaistuna saadaan kaava

$$\boldsymbol{R}^{31} = \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix}, \tag{4.7}$$

jossa *sin* on lyhennetty kirjaimella s ja *cos* kirjaimella c. Johdettu muunnosmatriisi $\boldsymbol{R}^{31}$ suorittaa siis kolme rotaatiota koordinaatiston akseleiden ympäri siirtäen vektorin esityksen joidenkin kantavektoreiden $\boldsymbol{e}_x$, $\boldsymbol{e}_y$ ja $\boldsymbol{e}_z$ avulla määritellystä koordinaatistosta toiseen koordinaatistoon, joka on määritelty kantavektoreiden $\boldsymbol{e}_x^3$, $\boldsymbol{e}_y^3$ ja $\boldsymbol{e}_z^3$ avulla. Kun saadusta muunnosmatriisista $\boldsymbol{R}^{31}$ otetaan inversio tai transpoosi kaavan

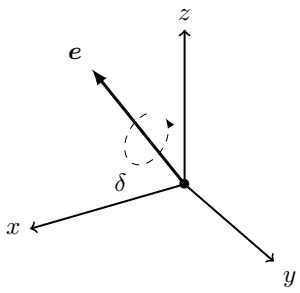
$$\boldsymbol{R}^{13} = (\boldsymbol{R}^{31})^T = (\boldsymbol{R}^{31})^{-1} \tag{4.8}$$

mukaisesti saadaan muunnosmatriisi $\boldsymbol{R}^{13}$, joka muuntaa vektorin esityksen takaisin alkuperäiseen koordinaatistoon. (Noureldin et al. 2013, s. 34–38)

Tait-Bryan kulmia käytettäessä on mahdollista joutua tilanteeseen, jossa ensimmäinen ja viimeinen rotaatio tapahtuvat samansuuntaisen akselin ympäri. Tällaistä tilannetta kutsutaan *Gimbal lock -ilmiöksi* ja se tapahtuu kun y-akselin ympäri tehtävän rotaation θ suuruus on $\pi/2 + n\pi$ $(n = 0, 1, ...)$. (Wittenburg 2008, s. 9–14)

4.4.2 Kvaterniot

Toinen menetelmä kappaleen asennon kuvaamiseen ja rotaation suorittamiseksi aseentoon pääsemiseen ovat *kvaterniot*. Kvaterniot ovat Sir William Rowan Hamiltonin kehittämä kompleksilukujen laajennus neliulotteiseen avaruuteen ja niitä voidaan hyödyntää *Eulerin parametrien* esittämiseen. Eulerin parametrit koostuvat akselista sekä akselin ympäri tehtävästä rotaatiosta kuten kuvassa 4.7 on esitettynä. Vapaasti valittavan akselin ja sen ympäri tehdyn rotaation avulla voidaan suorittaa mikä vain kolmiulotteisen avaruuden rotaatio. (Zipfel 2007, s. 121–127)



Kuva 4.7 *Akseli ja sen ympäri suoritettava rotaatio.*

Kvaternion komponenteille voidaan laskea arvot kaavalla

$$\boldsymbol{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} cos(\frac{\delta}{2}) \\ e_x sin(\frac{\delta}{2}) \\ e_y sin(\frac{\delta}{2}) \\ e_z sin(\frac{\delta}{2}) \end{bmatrix}, \tag{4.9}$$

jossa $\boldsymbol{e} = \begin{bmatrix} e_x & e_y & e_z \end{bmatrix}^T$ on valitun akselin määrittävä yksikkövektori ja δ on akselin ympäri tehtävä rotaatio. Kvaternion arvoista voidaan muodostaa muunnosmatriisi $\mathbf{R}$ kaavan

$$\boldsymbol{R} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{4.10}$$

avulla. Kvaternioille määritellyt laskutoimitukset yksinkertaistavat rotaatioiden käsittelyä ja niiden avulla tehdyt laskut ovat nopeita suorittaa tietokoneella. Myös esimerkiksi kulmanopeuden aiheuttama muutos kappaleen asentoon on helppo määrittää kvaternion derivaatan avulla. Toisin kuin Tait-Bryan kulmilla, kvaternioilla ei esiinny Gimbal lock -ilmiötä. (Zipfel 2007, s. 121–126)

4.4.3 Maakeskeisestä geodeettisestä karteesiseen

Monessa tilanteessa jonkin kappaleen koordinaatit on ilmaistu geodeettisten koordinaattien avulla, mutta laskemista varten tulisi saada selville kappaleen koordinaatit karteesisessa koordinaatistossa. Kappale voi olla esimerkiksi lentokone, jonka leveysaste, pituusaste ja korkeus on tiedossa, ja johon ohjuksen tulisi suunnistaa. Tällöin

on tarpeen muuntaa lentokoneen sijainti karteesisiksi, jotta ohjuksen hakeutuminen kohteeseen on helposti laskettavissa. Muunnos voidaan tehdä yhtälön

$$\begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} = \begin{bmatrix} (R_N + h)cos(\phi_{lat})cos(\lambda_{lon}) \\ (R_N + h)cos(\phi_{lat})sin(\lambda_{lon}) \\ \Big(R_N(1 - e^2) + h\Big)sin(\phi_{lat}) \end{bmatrix} \tag{4.11}$$

avulla, jossa R_N saadaan kaavalla

$$R_N = \frac{a}{\sqrt{1 - e^2 sin^2(\phi_{lat})}} \tag{4.12}$$

ja e kaavalla

$$e = \sqrt{\frac{a^2 - b^2}{a^2}}. \tag{4.13}$$

Edeltävissä kaavoissa a ja b ovat kohdassa 4.2 esitetyt pikkuakselin ja isoakselin puolikkaat. Termit ϕ_{lat} ja λ_{lon} ovat geodeettinen leveysaste sekä geodeettinen pituusaste. Edellä esitellyt kaavat on johdettu lähteessä (Noureldin et al. 2013, s. 53–59).

4.4.4 Maakeskeisestä karteesisesta geodeettiseen

Vaikka maakeskeisen kehyksen geodeettisestä koordinaatistosta voidaan helposti muuntaa vektorit karteesiseen koordinaatistoon, on muunnos toiseen suuntaan paljon monimutkaisempi. Usein ratkaisu tähän ongelmaan saadaan interatiivisen algoritmin avulla, mutta myös suoraan likiarvon antavia algoritmejä on olemassa. (Noureldin et al. 2013, s. 50–52)

Tässä työssä ei keskitytä muunnoksen tekeviin algoritmeihin syvemmin, eikä esitetä kuinka ne johdetaan, vaan hyödynnetään suoraan kirjallisuudessa esiintyviä ratkaisuja. Tarkemmin työssä hyödynnetään Donald K. Olson:in esittämää algoritmiä, jonka avulla voidaan laskea muunnoksen likiarvo ilman iterointia (Olson 1996).

4.5 Muutosnopeus eri koordinaatistoista havaittuna

Joissakin tilanteissa vektorin muutosnopeus on helposti laskettavissa jonkin koordinaatiston suhteen, mutta sen muutosnopeus tulisi saada selville jonkin toisen koordinaatiston suhteen. Vektorin derivaattaa ei voi suoraan laskea toisen koordinaatiston suhteen ja muuntaa tulosta takaisin alkuperäisen koordinaatiston suhteen määritellyksi, sillä tällöin menetettäisiin toisen koordinaatiston kantavektoreiden muutok-

sesta aiheutuva ero. Tällaisessä tilanteessa derivaatta tulee laskea kaavan

$$(\dot{\boldsymbol{v}})_i = (\dot{\boldsymbol{v}})_b + \boldsymbol{\omega} \times \boldsymbol{v} \tag{4.14}$$

avulla, jossa$(\dot{\boldsymbol{v}})_i$ on vektorin derivaatta alkuperäisen koordinaatiston suhteen, $(\dot{\boldsymbol{v}})_b$ on derivaatta toisen koordinaatiston suhteen ja $\boldsymbol{\omega}$ on koordinaatistojen suhteellinen pyörimisnopeus. (Schaub & Junkins 2003, s. 8–12) Edeltävää kaavaa hyödynnetään laajasti seuraavan luvun liikeyhtälöiden johtamisessa. Kaava on johdettuna tarkemmin lähteessä (Schaub & Junkins 2003, s. 8–12).

4.6 Ohjussimulaattorissa hyödynnetyt koordinaatistot

Työssä toteutettavassa ohjussimulaattorissa hyödynnetään alakohdassa 4.2 esitettyä maakehykseen liitettyä ECEF-koordinaatistoa ohjuksen sijainnin määrittämiseen. Myös kohdassa 2.3.2 mainituissa referenssi FOM:eissa olioiden ominaisuudet esitetään ECEF:in suhteen määriteltyinä, jonka vuoksi simulaattorin tulee julkaista tiedot ECEF-koordinaattien suhteen ilmaistuna. Geodeettinen koordinaatisto on yleinen tapa esittää maapallon pinnalla olevien kappaleiden sijaintitietoa, jos kappaleita tulee esittää karttatasolla tai niiden korkeus maanpinnasta tulee määrittää. Myös toteutettavassa simulaattorissa ohjuksen ECEF-koordinaatteja muunnetaan geodeettisiksi koordinaateiksi, jotta ohjuksen lentokorkeus maanpinnasta saadaan selvitettyä. Simulaattorissa ECEF-koordinaattien ja geodeettisten koordinaattien väliset muunnokset tehdään alakohdissa 4.4.3 ja 4.4.4 esitettyjen algoritmien avulla.

Kohdassa 4.3 esitettyä kappalekehykseen liitettyä kappalekoordinaatistoa hyödynnetään ohjuksen liikeyhtälöiden muodostamisessa. Yhtälöt muodostetaan siten, että voimia ja momentteja voidaan antaa ohjuksen omien koordinaatistojen suhteen määriteltynä, jolloin yhtälöiden muodostaminen ja niiden avulla laskeminen yksinkertaistuu. Kappalekoordinaatiston suhteen määriteltyjen yhtälöiden avulla voidaan esimerkiksi määrittää, että ohjukseen kohdistuu jatkuvasti jokin x-akselin suuntainen voima ja vaikka ohjuksen asento muuttuu, ei voimavektorille tarvitse suorittaa erillisiä muunnosoperaatioita. Kappalekehykseen kiinnitettyä NED-koordinaatistoa hyödynnetään simulaattorissa ohjuksen kurssin sekä ohjuksen maanpinnan suuntaisen nopeuden ilmaisemiseen. Simulaattorissa hyödynnetään myös tuulikoordinaatistoa ohjukseen kohdistuvien aerodynaamisten voimien laskemiseen. Ohjukseen kohdistuvat noste- ja ilmanvastusvoimat ilmaistaan tuulikoordinaatiston suhteen määriteltyinä, josta ne muunnetaan kappalekoordinaatiston suhteen määritellyiksi.

Simulaattorissa hyödynnetään alakohdassa 4.4.1 esitettyjä Tait-Bryan kulmia oh-

juksen laukaisuhetken asennon asettamiseen. Tait-Bryan kulmia käytetään, ECEF-koordinaattien tavoin, esimerkiksi RPR FOM:issa olioiden asennon kuvaamiseen, jonka vuoksi simulaattorin tulee muodostaa Tait-Bryan kulmat ohjuksen asennosta myös HLA:n RTI:hin julkaisua varten. Kulmia hyödynnetään myös muiden simulaattoreiden esittämien ominaisuuksien muuntamiseen ohjussimulaattorin ymmärtämään muotoon. Tällaisiä ovat esimerkiksi maanpinnan suuntaisen kulman ja vauhdin muuntaminen ECEF-koordinaatiston suhteen määritellyksi nopeusvektoriksi. Alakohdassa 4.4.2 esitettyjä kvaternioita hyödynnetään simulaattorin ohjusmallissa ohjuksen asennon integroimisessa ohjuksen kulmanopeudesta. Kvaternioita hyödynnetään myös simulaattorin ohjusmallissa käytetyissä koordinaatistomuunnoksissa.

5. OHJUKSEN DYNAMIIKKA JA AERODYNAMIIKKA

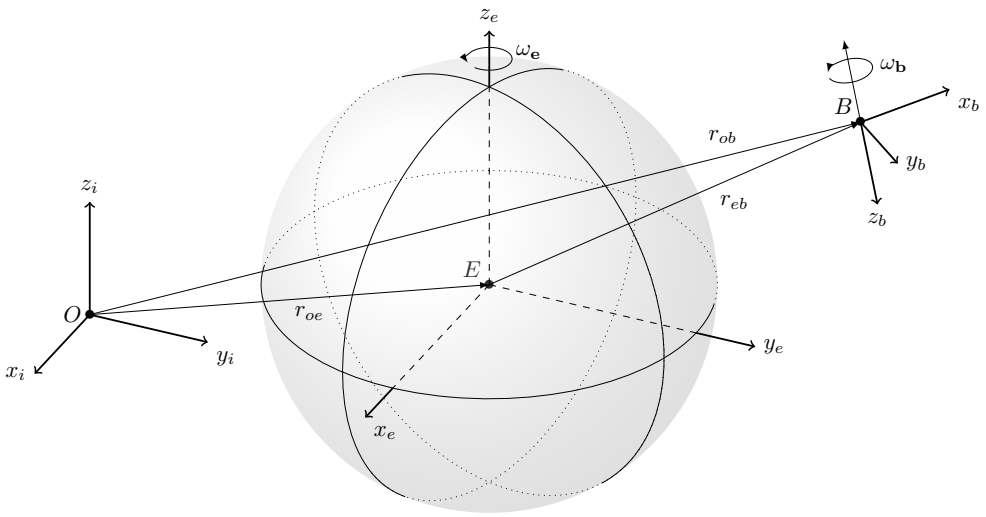
Jotta ohjuksen lentorataa voidaan simuloida, tulee sille ensin johtaa liikeyhtälöt, joita ohjus liikkuessaan noudattaa. Oikean ohjuksen lentoon vaikuttaa moni asia, jonka vuoksi tarkkojen liikeyhtälöiden muodostaminen ohjukselle on hankalaa. Liikeyhtälöiden muodostamisessa kuitenkin huomioidaan aiemmin alakohdassa 3.5.1 esitetyt oletukset ohjuksen toiminnalle, jolloin yhtälöiden muodostaminen yksinkertaistuu. Luvussa johdetaan maapallon ilmakehässä liikkuvalle, vakiomassaiselle ja jäykkänä kappaleena nähtävälle, ohjukselle kuuden vapausasteen liikeyhtälöt. Yhtälöitä johdettaessa maapallon pyörimisnopeus oletetaan vakioksi ja inertiaalikoordinaatiston origon sekä maapalloon kiinnitetyn koordinaatiston origon ajatellaan olevan kiinnitettynä samaan pisteeseen. Lisäksi luvussa esitellään, työn ohjusmallissa huomioitavat, ohjukseen kohdistuvat voimat sekä lähteet, joista voimia muodostuu.

5.1 Kappalekoordinaatiston kiihtyvyys inertiaalikoordinaatiston suhteen

Jotta jollekin maapallon ilmakehässä liikkuvalle jäykälle kappaleelle, kuten ohjukselle, voidaan määrittää liikeyhtälöt, tulee ensin selvittää kappaleen kiihtyvyydet inertiaalikoordinaatiston suhteen. Kuvassa 5.1 on esitettynä tilanne, jossa kappaleeseen kiinnitetty koordinaatisto liikkuu maapallon ilmakehässä. Kuvan koordinaatisto O esittää inertiaalikoordinaatistoa, koordinaatisto E esittää maapalloon liitettyä koordinaatistoa ja koordinaatisto B esittää kappaleeseen kiinnitettyä koordinaatistoa. Kuvan termi $\boldsymbol{\omega_e}$ kuvaa koordinaatiston E pyörimistä ja termi $\boldsymbol{\omega_b}$ kappalekoordinaatiston B pyörimistä (Stevens & Lewis 1992, s. 18–22).

Jos kappalekoordinaatiston origon sijainti tunnetaan maahan liitetyn koordinaatiston origon suhteen ja maahan liitetyn koordinaatiston origon sijainti tunnetaan inertiaalikoordinaatiston suhteen, voidaan kappalekoordinaatiston sijainti inertiaalikoordinaatiston suhteen esittää kaavalla

$$\boldsymbol{r}_{ob} = \boldsymbol{r}_{oe} + \boldsymbol{r}_{eb}, \tag{5.1}$$



Kuva 5.1 *Inertiaalikoordinaatisto O, maahan liitetty koordinaatisto E ja kappalekoordinaatisto B. Maahan kiinnitetty koordinaatisto ja inertiaalikoordinaatisto ovat erillään selkeyden vuoksi.*

jossa $\boldsymbol{r}_{ob}$ on kappalekoordinaatiston paikkavektori koordinaatiston O origosta, $\boldsymbol{r}_{oe}$ on koordinaatiston E paikkavektori inertiaalikoordinaatiston O origosta, ja termi $\boldsymbol{r}_{eb}$ on kappalekoordinaatiston origon sijainti koordinaatiston E origon suhteen. Kun yhtälöstä otetaan molemmilta puolilta aikaderivaatta inertiaalikoordinaatiston O suhteen, saadaan

$$(\dot{\boldsymbol{r}}_{ob})_i = (\dot{\boldsymbol{r}}_{oe})_i + (\dot{\boldsymbol{r}}_{eb})_i, \tag{5.2}$$

jossa $(\dot{\boldsymbol{r}}_{oe})_i$ on koordinaatiston E sijainnin muutos koordinaatiston O suhteen nähtynä koordinaatistosta O, $(\dot{\boldsymbol{r}}_{eb})_i$ on koordinaatiston B sijainnin muutos koordinaatiston E suhteen nähtynä koordinaatistosta O ja $(\dot{\boldsymbol{r}}_{ob})_i$ on koordinaatiston B sijainnin muutos koordinaatiston O suhteen nähtynä paikallaan pysyvästä inertiaalikoordinaatistosta O. Viimeiseen termiin $(\dot{\boldsymbol{r}}_{eb})_i$ voidaan soveltaa kaavaa 4.14, jolloin edellinen yhtälö muuntautuu muotoon

$$(\dot{\boldsymbol{r}}_{ob})_i = (\dot{\boldsymbol{r}}_{oe})_i + (\dot{\boldsymbol{r}}_{eb})_e + \boldsymbol{\omega}_e \times \boldsymbol{r}_{eb}, \tag{5.3}$$

jossa $(\dot{\boldsymbol{r}}_{eb})_e$ kuvaa siis kappalekoordinaatiston B sijainnin muutosta maahan liitetyn koordinaatiston E suhteen nähtynä koordinaatistosta E. Kiihtyvyyden saamiseksi, tulee edellinen yhtälö derivoida ajan suhteen toisen kerran molemmin puolin. Tällöin saadaan

$$(\ddot{\boldsymbol{r}}_{ob})_i = (\ddot{\boldsymbol{r}}_{oe})_i + \left.\frac{\mathrm{d}(\dot{\boldsymbol{r}}_{eb})_e}{\mathrm{d}t}\right|_i + \left.\frac{\mathrm{d}\boldsymbol{\omega}_e \times \boldsymbol{r}_{eb}}{\mathrm{d}t}\right|_i, \tag{5.4}$$

joka edelleen kaavaa 4.14 hyödyntäen muuntuu muotoon

$$(\ddot{\boldsymbol{r}}_{ob})_i = (\ddot{\boldsymbol{r}}_{oe})_i + (\ddot{\boldsymbol{r}}_{eb})_e + \boldsymbol{\omega}_e \times (\dot{\boldsymbol{r}}_{eb})_e + (\dot{\boldsymbol{\omega}}_e)_i \times \boldsymbol{r}_{eb} + \boldsymbol{\omega}_e \times (\dot{\boldsymbol{r}}_{eb})_i, \tag{5.5}$$

jossa $(\ddot{\boldsymbol{r}}_{oe})_i$ kuvaa maahan liitetyn koordinaatiston sijainnin kiihtyvyyyttä inertiaalikoordinaatiston suhteen, $(\ddot{\boldsymbol{r}}_{eb})_e$ kuvaa kappalekoordinaatiston sijainnin kiihtyvyyttä maahan liitetyn koordinaatiston suhteen ja termi $(\dot{\boldsymbol{\omega}}_e)_i$ kuvaa maan pyörimisnopeuden muutosta inertiaalikoordinaatiston suhteen. Kaavan viimeinen termi voidaan vielä purkaa kaavan 4.14 avulla jolloin saadaan kaava

$$(\ddot{\boldsymbol{r}}_{ob})_i = (\ddot{\boldsymbol{r}}_{oe})_i + (\ddot{\boldsymbol{r}}_{eb})_e + \boldsymbol{\omega}_e \times (\dot{\boldsymbol{r}}_{eb})_e + (\dot{\boldsymbol{\omega}}_e)_i \times \boldsymbol{r}_{eb} + \boldsymbol{\omega}_e \times (\dot{\boldsymbol{r}}_{eb})_e + \boldsymbol{\omega}_e \times (\boldsymbol{\omega}_e \times \boldsymbol{r}_{eb}), \tag{5.6}$$

joka kokonaisuudessaan kuvaa kappalekoordinaatiston kiihtyvyyttä inertiaalikoordinaatiston suhteen. Kaava sievenee edelleen muotoon

$$(\ddot{\boldsymbol{r}}_{ob})_i = (\ddot{\boldsymbol{r}}_{eb})_e + 2(\boldsymbol{\omega}_e \times (\dot{\boldsymbol{r}}_{eb})_e) + \boldsymbol{\omega}_e \times (\boldsymbol{\omega}_e \times \boldsymbol{r}_{eb}) \tag{5.7}$$

jos termejä yhdistellään, oletetaan että inertiaalikoordinaatiston sekä maahan kiinnitetyn koordinaatiston origot ovat aina samassa pisteessä ja oletetaan maapallon pyörimisen kiihtyvyys $(\dot{\boldsymbol{\omega}}_e)_i$ mitättömän pieneksi.

Edellä olevassa kaavassa termi $(\ddot{\boldsymbol{r}}_{eb})_e$ kuvaa siis kappalekoordinaatiston sijainnin kiihtyvyyttä maahan liitetyn koordinaatiston suhteen. Usein kappalekoordinaatiston kiihtyvyydet ja nopeudet on kuitenkin luontevaa kuvata kappalekoordinaatiston suhteen määriteltyinä, jonka vuoksi on hyödyllistä muuntaa kaava 5.7 kaavan 4.14 avulla muotoon

$$(\ddot{\boldsymbol{r}}_{ob})_i = (\ddot{\boldsymbol{r}}_{eb})_b + \boldsymbol{\omega}_b \times (\dot{\boldsymbol{r}}_{eb})_b + 2(\boldsymbol{\omega}_e \times (\dot{\boldsymbol{r}}_{eb})_b) + \boldsymbol{\omega}_e \times (\boldsymbol{\omega}_e \times \boldsymbol{r}_{eb}), \tag{5.8}$$

jossa $(\dot{\boldsymbol{r}}_{eb})_b$ kuvaa siis kappalekoordinaatiston sijainnin nopeusvektoria kappalekoordinaatistosta nähtynä ja $(\ddot{\boldsymbol{r}}_{eb})_b$ kuvaa kappalekoordinaatiston sijainnin kiihtyvyysvektoria kappalekoordinaatistosta nähtynä. Kaavassa esiintyvää termiä $2(\boldsymbol{\omega}_e \times (\dot{\boldsymbol{r}}_{eb})_b)$ kutsutaan *Coriolis*-kiihtyvyydeksi ja se kaareuttaa pyörivän maapallon ilmakehässä liikkuvan kappaleen lentorataa joko oikealle tai vasemmalle riippuen kappaleen lentosuunnasta. Termiä $\boldsymbol{\omega}_e \times (\boldsymbol{\omega}_e \times \boldsymbol{r}_{eb})$ kutsutaan keskihakuiskiihtyvyydeksi ja se pyrkii työntämään maapallon ilmakehässä liikkuvaa kappaletta kauemmaksi maapallon keskipisteestä. (Stevens & Lewis 1992, s. 18–23)

5.2 Voimien aiheuttama kiihtyvyys

Newtonin toisen lain mukaan kappaleeseen vaikuttava voima aiheuttaa kappaleen liikkeeseen muutoksen. Työn ohjusmallin tapauksessa vaikuttavat voimat ovat ohjuksen työntövoima, ohitse virtaavasta ilmasta aiheutuvat voimat sekä painovoima. Koska esimerkiksi ohjuksen työntövoima on ohjuksesta nähden aina ohjuksen kärkeä kohti, on luontevaa ilmaista Newtonin toisen lain voimat sekä nopeudet kappalekoordinaatiston suhteen ilmaistuna. Newtonin toinen laki on kuitenkin voimassa vain kun kiihtyvyydet ilmaistaan inertiaalikoordinaatiston suhteen kaavan

$$\boldsymbol{F} = m(\ddot{\boldsymbol{r}}_{ob})_i \tag{5.9}$$

mukaisesti, jossa termi $\boldsymbol{F}$ kuvaa kaikkia kappaleeseen vaikuttavia voimia, termi m kuvaa kappaleen massaa ja termi $(\ddot{\boldsymbol{r}}_{ob})_i$ kuvaa kappaleen kiihtyvyyttä inertiaalikoordinaatiston suhteen. (Stevens & Lewis 1992, s. 18–22)

Edellisessä kohdassa johdettiin kaava 5.8, joka ilmaisi kappalekoordinaatiston kiihtyvyyden inertiaalikoordinaatiston suhteen. Kun tämä kaava sijoitetaan kaavaan 5.9 termin $(\ddot{\boldsymbol{r}})_i$ paikalle, saadaan

$$\boldsymbol{F} = m\Big((\ddot{\boldsymbol{r}}_{eb})_b + \boldsymbol{\omega}_b \times (\dot{\boldsymbol{r}}_{eb})_b + 2(\boldsymbol{\omega}_e \times (\dot{\boldsymbol{r}}_{eb})_b) + \boldsymbol{\omega}_e \times (\boldsymbol{\omega}_e \times \boldsymbol{r}_{eb})\Big), \tag{5.10}$$

ja kun kaavasta ratkaistaan kappalekoordinaatiston kiihtyvyys $(\ddot{\boldsymbol{r}}_{eb})_b$ saadaan

$$(\ddot{\boldsymbol{r}}_{eb})_b = \frac{\boldsymbol{F}}{m} - \boldsymbol{\omega}_b \times (\dot{\boldsymbol{r}}_{eb})_b - 2(\boldsymbol{\omega}_e \times (\dot{\boldsymbol{r}}_{eb})_b) - \boldsymbol{\omega}_e \times (\boldsymbol{\omega}_e \times \boldsymbol{r}_{eb}), \tag{5.11}$$

jonka avulla voidaan laskea kappalekoordinaatiston suhteen määriteltyjen voimien vaikutus kappaleen kiihtyvyyteen (Stevens & Lewis 1992, s. 18–22).

5.3 Ohjuksen pyörimismäärä ja inertiamatriisi

Jos ohjus on pyörivässä liikkeessä, voidaan ohjukseen kuuluvan differentiaalisen pienen, pyörimispisteestä etäisyydellä $\boldsymbol{r}$ sijaitsevan, massan dm pyörimismäärä määrittää kaavalla

$$\mathrm{d}\boldsymbol{L} = \boldsymbol{r} \times \boldsymbol{v}\mathrm{d}m, \tag{5.12}$$

jossa d$\boldsymbol{L}$ kuvaa differentiaalisen pienen massan pyörimismäärää, $\boldsymbol{r}$ kuvaa differentiaalisen massapisteen dm etäisyyttä pyörimispisteestä ja $\boldsymbol{v}$ kuvaa massapisteen nopeutta. Koska massapiste on kiinni jäykässä kappaleessa, on sen nopeus $\boldsymbol{v}$ ainoastaan pyörimisnopeutta pyörimispisteen ympäri. Tällöin pyörimisnopeus voidaan saada

massapisteen etäisyyden ja kappaleen pyörimisnopeuden $\boldsymbol{\omega}_b$ ristitulona kaavan

$$\boldsymbol{v} = \boldsymbol{\omega}_b \times \boldsymbol{r} \tag{5.13}$$

mukaisesti. Kun tämä kaava sijoitetaan aiempaan kaavaan 5.12 saadaan

$$\mathrm{d}\boldsymbol{L} = \boldsymbol{r} \times (\boldsymbol{\omega}_b \times \boldsymbol{r})\mathrm{d}m, \tag{5.14}$$

josta edelleen integroimalla koko kappaleen ylitse ja ratkaisemalla ristitulot hyödyntäen vektorikolmituloa[1] saadaan kaava

$$\boldsymbol{L} = \int (\boldsymbol{r} \cdot \boldsymbol{r})\boldsymbol{\omega}_b - (\boldsymbol{r} \cdot \boldsymbol{\omega}_b)\boldsymbol{r}\mathrm{d}m, \tag{5.15}$$

joka kuvaa koko kappaleen pyörimismäärää. Jos kappale on koordinaatistossa, jonka kantavektorit ovat $\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z$ ja jossa $\boldsymbol{r} = [x\ y\ z]^T$ sekä $\boldsymbol{\omega}_b = [\omega_x\ \omega_y\ \omega_z]^T$ voidaan edellinen kaava esittää muodossa

$$\begin{aligned} L_x\boldsymbol{e}_x + L_y\boldsymbol{e}_y + L_z\boldsymbol{e}_z = \int \Big(&(\omega_x\boldsymbol{e}_x + \omega_y\boldsymbol{e}_y + \omega_z\boldsymbol{e}_z)(x^2 + y^2 + z^2) \\ &- (x\boldsymbol{e}_x + y\boldsymbol{e}_y + z\boldsymbol{e}_z)(\omega_x x + \omega_y y + \omega_z z)\Big)\mathrm{d}m, \end{aligned} \tag{5.16}$$

jossa osa kantavektoreista on pistetulon johdosta supistunut pois.[2] Kaavassa termit L_x, L_y, L_z kuvaavat massapisteen pyörimismäärää eri kantavektoreiden suhteen, termit $\omega_x, \omega_y, \omega_z$ kuvaavat massapisteen kulmanopeuksia kantavektoreiden ympäri ja termit x, y, z kuvaavat massapisteen sijaintia suhteessa pisteeseen, jonka ympäri massapiste pyörii. (Stevens & Lewis 1992, s. 28–33)

Laskemalla kaava 5.16 auki saadaan siitä muodostettua kaavat

$$L_x = \int (y^2 + z^2)\omega_x\mathrm{d}m - \int (xy)\omega_y\mathrm{d}m - \int (xz)\omega_z\mathrm{d}m, \tag{5.17}$$

$$L_y = -\int (yx)\omega_x\mathrm{d}m + \int (x^2 + z^2)\omega_y\mathrm{d}m - \int (yz)\omega_z\mathrm{d}m, \tag{5.18}$$

$$L_z = -\int (zx)\omega_x\mathrm{d}m - \int (zy)\omega_y\mathrm{d}m + \int (x^2 + y^2)\omega_z\mathrm{d}m, \tag{5.19}$$

joista jokainen kuvaa kappaleella olevaa pyörimismäärää eri akseleiden suhteen. Kun

[1] $\boldsymbol{a} \times (\boldsymbol{b} \times \boldsymbol{c}) = (\boldsymbol{a} \cdot \boldsymbol{c})\boldsymbol{b} - (\boldsymbol{a} \cdot \boldsymbol{b})\boldsymbol{c}$

[2] $\boldsymbol{e}_x \cdot \boldsymbol{e}_x = 1, \boldsymbol{e}_y \cdot \boldsymbol{e}_y = 1, \boldsymbol{e}_z \cdot \boldsymbol{e}_z = 1$

kaavat 5.17 – 5.19 esitetään matriisimuodossa saadaan

$$\begin{bmatrix} L_x \\ L_y \\ L_z \end{bmatrix} = \begin{bmatrix} \int(y^2+z^2)\mathrm{d}m & -\int(xy)\mathrm{d}m & -\int(xz)\mathrm{d}m \\ -\int(yx)\mathrm{d}m & \int(x^2+z^2)\mathrm{d}m & -\int(yz)\mathrm{d}m \\ -\int(zx)\mathrm{d}m & -\int(zy)\mathrm{d}m & \int(x^2+y^2)\mathrm{d}m \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \tag{5.20}$$

jossa ensimmäinen matriisi lausekkeen oikealla puolella esittää kappaleen inertiamatriisia. Kun inertiamatriisin integraalilauseet korvataan muuttujilla voidaan kaava edelleen muuttaa muotoon

$$\boldsymbol{L} = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \boldsymbol{I}\boldsymbol{\omega}_b, \tag{5.21}$$

jossa $\boldsymbol{I}$ on kappaleen *inertiamatriisi*. Inertiamatriisin diagonaaliset elementit I_{xx}, I_{yy}, I_{zz} ovat kappaleen *hitausmomentit* ja muut elementit ovat kappaleen *hitaustuloja* (Stevens & Lewis 1992, s. 28–33).

5.4 Momenttien aiheuttama kulmakiihtyvyys

Koska kappaleen pyörimismäärä on seurausta kappaleeseen kohdistuneiden momenttien vaikutuksesta, voidaan kappaleeseen kohdstuva momentti määrittää ottamalla aikaderivaatta kappaleen liikemäärästä kaavan

$$\boldsymbol{\tau} = (\dot{\boldsymbol{L}})_i \tag{5.22}$$

mukaisesti. Jos kappaleen pyörimismäärä $\boldsymbol{L}$ laskettaisiin inertiaalikoordinaatiston suhteen, muuttuisivat inertiamatriisin arvot jatkuvasti kun kappaleen asento muuttuu, jonka seurauksena inertiamatriisin muutos tulisi ottaa huomioon pyörimismäärää derivoitaessa. Tämän vuoksi on järkevää määrittää pyörimismäärä kappaleeseen liitetyn koordinaatiston suhteen, jolloin inertiamatriisi säilyy jatkuvasti vakiona eikä sen derivaattaa tarvitse huomioida momenttilaskuissa. Pyörimismäärä voidaan jälleen muuntaa kaavan 4.14 avulla kappalekoordinaatiston suhteen määritellyksi, jolloin kaavasta 5.22 saadaan

$$(\dot{\boldsymbol{L}})_i = (\dot{\boldsymbol{L}})_b + \boldsymbol{\omega}_b \times \boldsymbol{L}, \tag{5.23}$$

jossa $(\dot{\boldsymbol{L}})_i$ kuvaa pyörimismäärän derivaattaa inertiaalikoordinaatiston suhteen, $(\dot{\boldsymbol{L}})_b$ kuvaa pyörimismäärän derivaattaa kappalekoordinaatiston suhteen ja $\boldsymbol{\omega}_b$ on kappaleen pyörimisnopeus (Stevens & Lewis 1992, s. 28–33).

Kun edelliseen kaavaan sijoitetaan kaava 5.21 sekä kaava 5.22 saadaan

$$\boldsymbol{\tau} = \boldsymbol{I}(\dot{\boldsymbol{\omega}}_b)_b + (\dot{\boldsymbol{I}})_b\boldsymbol{\omega}_b + \boldsymbol{\omega}_b \times \boldsymbol{L}, \tag{5.24}$$

joka edelleen sievenee muotoon

$$\boldsymbol{\tau} = \boldsymbol{I}(\dot{\boldsymbol{\omega}}_b)_b + \boldsymbol{\omega}_b \times (\boldsymbol{I}\boldsymbol{\omega}_b), \tag{5.25}$$

sillä inertiamatriisi on kappalekoordinaatiston suhteen vakio. Kun kaavasta ratkaistaan[3] kulmakiihtyvyydet saadaan

$$(\dot{\boldsymbol{\omega}}_b)_b = \boldsymbol{I}^{-1}\Big(\boldsymbol{\tau} - \boldsymbol{\omega}_b \times (\boldsymbol{I}\boldsymbol{\omega}_b)\Big), \tag{5.26}$$

jonka avulla voidaan ratkaista kappaleeseen kohdistettujen vääntöjen seureuksena aiheutuvat kulmakiihtyvyydet (Stevens & Lewis 1992, s. 28–33). Inertiamatriisin arvot kuvaavat siis suhdetta kappaleeseen kohdistetun väännön ja tulokseksi saadun kulmakiihtyvyyden välillä. Mitä suurempi arvo inertiamatriisin elementissä on, sitä suuremman väännön jokin haluttu kulmakiihtyvyys tarvitsee.

Mille tahansa jäykälle kappaleelle on olemassa sellaiset koordinaatiston akselit, joiden suhteen määritetyn inertiamatriisin hitaustulot menevät nollaan ja inertiamatriisista muodostuu diagonaalinen. Tällaisten akseleiden määrittämää koordinaatistoa, jossa inertiamatriisi on diagonaalinen, kutsutaan kappaleen *pääakselikoordinaatistoksi*. (Stevens & Lewis 1992, s. 28–33) Jos edeltävät yhtälöt kappaleen kiihtyvyydelle esitetään pääakselikoordinaatiston suhteen, yksinkertaistuvat ne huomattavasti. Esimerkiksi kaava 5.26 muuntautuu poistuvien hitaustulojen ansiosta muotoon

$$(\dot{\omega_x})_p = \frac{\tau_x + \omega_y\omega_z(I_{yy} - I_{zz})}{I_{xx}}, \tag{5.27}$$

$$(\dot{\omega_y})_p = \frac{\tau_y + \omega_z\omega_x(I_{zz} - I_{xx})}{I_{yy}}, \tag{5.28}$$

$$(\dot{\omega_z})_p = \frac{\tau_z + \omega_x\omega_y(I_{xx} - I_{yy})}{I_{zz}}, \tag{5.29}$$

jolloin kappaleen kulmakiihtyvyydet pääakselikoordinaatiston suhteen voidaan hel-

[3] $\boldsymbol{I}^{-1}\boldsymbol{I} = \boldsymbol{1}$

posti laskea.

Aina ei kuitenkaan ole tarpeen löytää kappaleen pääakselikoordinaatistoa, sillä jos kappaleella on symmetrisiä ominaisuuksia kumoutuu osa inertiamatriisin hitaustuloista. Esimerkiksi lentokoneilla x–z -taso on usein symmetrinen, josta johtuen jokaiselle tulolle $x_i z_j$ ja $y_i x_j$ on olemassa yhtä suuri mutta vastakkaissuuntainen tulo. Tällöin vain hitaustulo I_{xz} on jotain muuta kuin nolla. (Stevens & Lewis 1992, s. 28–33) Koska ohjukset ovat sylinterimäisiä rakenteita, on niiden x–y -taso myös symmetriataso. Tämän vuoksi ohjuksen hitaustulo I_{xz} menee nollaksi, ja ohjuksen kappalekoordinaatisto on samalla sen pääakselikoordinaatisto.

5.5 Aerodynamiikka

Kun ohjus liikkuu ilmakehässä, muodostuu sitä ympäröivään ilmaan paine- sekä nopeuseroja. Paine- ja nopeuserojen seurauksena ohjukseen kohdistuu *aerodynaamisia voimia* ja *aerodynaamisia momentteja*. Aerodynaamiset voimat ja momentit riippuvat useasta eri muuttujasta, kuten esimerkiksi ohjuksen muodosta, ilmavirtauksen nopeudesta, ilmavirtauksen tiheydestä ja ohjuksen kohtauskulmasta. (Siouris 2004, s. 53–54) Aerodynaamisilla voimilla on keskeinen vaikutus ohjuksen lentoon sillä ohjus muuntelee lentorataansa, ja siten myös hakeutuu kohteeseensa aerodynaamisten voimien avulla (Strickland 2011, s. 30–34). Koska aerodynaamisten voimien suuruus riippuu monesta tekijästä, on niiden summittainen arviointi haastavaa. Tämän vuoksi seuraavissa alakohdissa esitellään kuinka ilmakehä vaikuttaa ohjuksen lentoon, jotta aerodynaamisia ominaisuuksia voidaan huomioida ohjusmallissa karkealla tasolla.

5.5.1 Aerodynaamiset voimat ja momentit

Ilmassa kulkevaan ohjukseen vaikuttavat aerodynaamiset voimat voidaan jakaa nosteeseen, ilmanvastukseen sekä sivusuuntaisiin voimiin. *Noste* on aerodynaaminen voima, joka on kohtisuorassa ohjuksen ohitse virtaavan ilman nopeusvektorin kanssa. Nostevoimat muodostuvat ohjuksen siivekkeiden ylä- ja alapinnoilla olevien paine-erojen seurauksena. *Ilmanvastus* aiheutuu ohjuksen ympärille muodostuvista paine-eroista sekä ohjuksen pinnan kitkavoimista. Ilmanvastus on samansuuntainen ohjuksen ohitse virtaavan ilmavirran kanssa. *Sivusuuntaiset voimat* syntyvät vastaavasti kuin nostevoimat, mutta niiden suunta on kohtisuorassa sekä nostevoimien että ilmanvastuksesta aiheutuvien voimien kanssa. (Siouris 2004, s. 55–56)

Kaikkien edellämainittujen aerodynaamisten voimien suuruus voidaan laskea kaa-

voilla

$$F_l = \frac{1}{2}\rho {v_w}^2 A C_l, \qquad F_d = \frac{1}{2}\rho {v_w}^2 A C_d, \qquad F_s = \frac{1}{2}\rho {v_w}^2 A C_s, \tag{5.30}$$

jossa F_l on nostevoima, F_d on ilmanvastuksesta aiheutuva voima, F_s on sivuttaissuuntainen voima, ρ on ilman tiheys ja v_w on ohjuksen ohitse virtaavan ilman nopeusvektorin pituus $\|\boldsymbol{v}_w\|$. A on vertailtava pinta-ala, jonka suhteen aerodynaamiset kertoimet C_l, C_d ja C_s on määritelty. Aerodynaamisten kertoimien suuruus riippuu monesta eri tekijästä esimerkiksi ohjuksen muodosta, kohtaamiskulmasta vastaantulevan ilman kanssa ja ohjuksen nopeudesta äänen nopeuden suhteen, eli *machluvusta*. Yhtälöillä 5.30 lasketut voimat kuvaavat aerodynaamisia voimia tuulikoordinaatison akseleiden suuntaisesti. Tällöin aerodynaamiseksi voimavektoriksi saadaan

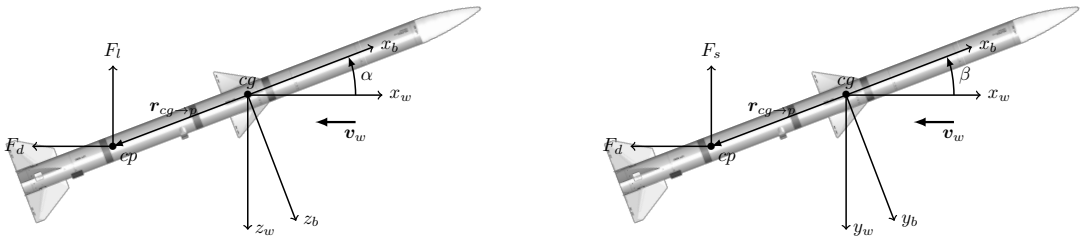
$$\boldsymbol{F}_a = \begin{bmatrix} -F_d \\ F_s \\ F_l \end{bmatrix}, \tag{5.31}$$

sillä F_d kuvaa ilmanvastusta joka on vastakkaisuuntainen tuulikoordinaatiston x-akselin suhteen, F_s kuvaa sivuttaisia voimia eli voimia tuulikoordinaatiston y-akselin suhteen ja F_l kuvaa nosteesta aiheutuvia voimia eli voimia tuulikoordinaatiston z-akselin suhteen.

Ilman aiheuttamat aerodynaamiset voimat vaikuttavat eri suuruisina ohjuksen eri pinnoilla. Voimat voidaan kuitenkin summata yhteen ja määrittää ohjuksen rungosta sellainen piste, johon kohdistettuna summattu voima tuottaa saman tuloksen kuin kaikkien pintojen voimat yhteensä. Tällaistä pistettä, johon summattu voimavektori vaikuttaa, kutsutaan *painekeskipisteeksi*. Painekeskipiste ei yleensä ole samassa kohdassa kuin ohjuksen massakeskipiste, jonka ympäri vapaasti liikkuva ohjus pyörii, joten ohjukseen aiheutuu aerodynaamisista voimista myös aerodynaamisia momentteja. (Siouris 2004, s. 53–54)

Kuvassa 5.2 on esitettynä ohjukseen liitetty tuulikoordinaatisto, massakeskipiste *cg* (center of gravity), painekeskipiste *cp* (center of pressure) ja tuulen nopeuden vektori $\boldsymbol{v}_w$. Vasemmanpuoleisessa kuvassa on lisäksi hyökkäyskulma α ja painekeskipisteeseen on liitetty aerodynaamisen voimavektorin nostekomponentti F_l sekä ilmanvastuskomponentti F_d. Tuulikoordinaatiston ja samalla kappalekoordinaatiston y-akseli osoittaa kohti lukijaa. Kun hyökkäyskulma kasvaa kuvan osoittaman nuolen suuntaisesti, kasvaa myös nosteen F_l suuruus vastakkaiseen suuntaan kuin tuulikoordinaatiston z-akseli. Oikeanpuoleisessa kuvassa on esitettynä sivuluisukulma β ja painekeskipisteeseen on liitetty sivusuuntaisten voimien komponentti F_s sekä ilmanvastuskomponentti F_d. Nyt tuulikoordinaatiston z-akseli osoittaa kohti lukijaa

ja sivuluisukulman kasvaessa nuolen osoittamaan suuntaan sivusuuntaisten voimien komponentti kasvaa vastakkaiseen suuntaan kuin tuulikoordinaatiston y-akseli.



Kuva 5.2 *Vasemmanpuoleisessa kuvassa hyökkäyskulma sekä aerodynaamisen voimavektorin noste komponentti. Oikeanpuoleisessa kuvassa sivuluisukulma sekä aerodynaamisen voimavektorin sivusuuntaisten voimien komponentti (Siouris 2004, s. 58–59).*

Koska aerodynaaminen voimavektori vaikuttaa painekeskipisteen kautta, joka ei välttämättä sijaitse täsmälleen kappaleen massakeskipisteessä, aiheuttuu aerodynaamisista voimista momentteja kappaleeseen. Jos painekeskipisteen etäisyys massakeskipisteestä on $\boldsymbol{r}_{cg\to p}$ saadaan kaavalla

$$\boldsymbol{\tau}_a = \boldsymbol{F}_a \times \boldsymbol{r}_{cg\to p}, \tag{5.32}$$

kappaleeseen aiheutuvien momenttien suuruus.

5.5.2 Aerodynaamiset kertoimet ja RASAero

Jotta jollekin kappaleelle voidaan määrittää tarkasti kaavassa 5.30 esiintyvät aerodynaamiset kertoimet C_l, C_d ja C_s, tulee sille tai sen pienoismallille suorittaa useita kokeita tuulitunnelin avulla. Kertoimet voidaan määrittää myös numeerisen virtausdynamiikan avulla, mutta nämä menetelmät ovat monimutkaisia, ja ne vaativat tietokoneelta paljon suorituskykyä. Teorian ja laajan tuulitunneleista kerätyn datan avulla on myös kehitetty menetelmiä, joiden avulla voidaan arvioida jonkin ohjuksen ominaisuuksia, jos tuulitunnelin tai numeerisen virtausdynamiikan käyttäminen ei ole mahdollista. Eräs yleinen teoreettista ja mitattua tietoa hyödyntävä ohjelmisto aerodynaamisten ominaisuuksien arviointiin on Yhdysvaltojen Ilmavoimien tuottama Missile DATCOM (Strickland 2011, s. 173 – 176). Missile DATCOM -tuote on kuitenkin saatavissa vain Yhdysvalloissa ja sen jakaminen Yhdysvaltojen ulkopuolelle on kiellettyä.

Toinen teoriaa ja tuulitunnelidataa hyödyntävä ohjelmisto on *RASAero*. RASAero on Rogers Aerosciencen tuottama ohjelmisto, jonka avulla voidaan arvioida rakettien aerodynaamisia ominaisuuksia. Ohjelman avulla voi esimerkiksi tuottaa raketin aerodynaamiset kertoimet eri mach-luvuilla ja hyökkäyskulmilla. Myös painekeskipisteen etäisyys massakeskipisteestä voidaan määrittää hyökkäyskulman ja mach-luvun funktiona. RASAeron tuottamia tuloksia on vertailtu NACA:n (National Advisory Committee for Aeronautics) ja NASA:n (National Aeronautics and Space Administration) tuulitunneleista saatuun dataan ja tulokset ovat vastanneet hyvin toisiaan. Ohjelmaa voidaan myös käyttää laukaistun raketin lentoradan ennakoimiseen. (Rogers & Cooper 2011) RASAero on suunniteltu rakettien simulointiin, joten esimerkiksi ohjusten hakeutumista ei ohjelmalla voi ennustaa. Ohjuksen ja raketin muodot ovat kuitenkin lähellä toisiaan, jonka vuoksi RASAero:lla määritetyt aerodynaamiset ominaisuudet raketeille toimivat hyvinä arvioina myös ohjuksien aerodynaamisille ominaisuuksille. Ohjelmalla ei voi simuloida raketteja joilla on takasiivekkeiden lisäksi myös etusiivekkeet. Tällaisille rakenteille saadaan kuitenkin tämän työn tarpeeseen riittävällä tarkkuudella toimivia arvioita, kun etusiivekkeet jätetään huomioimatta.

5.5.3 Ilmakehän ominaisuudet

Ilmakehän ominaisuudet muuttuvat jatkuvasti esimerkiksi ilmankosteuden ja muiden ilman epätasaisuuksien vuoksi. Yleisesti kuitenkin ilmakehä maapallon ympärillä ohenee mitä korkeammalle maanpinnasta noustaan. Suuremmilla korkeuksilla laskee myös sekä ilmanpaine että äänennopeus.

Tässä työssä hyödynnetään *U.S. Standard Atmosphere* -ilmakehämallia, joka kuvaa ilmakehää idaalisena ja tasaisena ilmamassana. Malli koostuu eri ilmakehän kerroksille muodostetuista yhtälöistä, joiden avulla voidaan ratkaista esimerkiksi ilman tiheys, äänennopeus sekä lämpötila eri korkeuksilla. (NOAA, NASA & USAF 1976, s. 1–7)

5.6 Muut ohjukseen vaikuttavat voimat

Oikeaan ohjukseen vaikuttaa sen lennon aikana voimia ja momentteja useasta eri lähteestä. Tässä työssä kuitenkin huomioidaan, edellä mainittujen aerodynaamisten voimien lisäksi, vain painovoima, ohjausmomentit ja moottorin tuottama työntövoima. Tämän lisäksi voimille sekä momenteille tehdään yksinkertaistuksia, joiden avulla yhtälöistä saadaan suorituskykyisempiä.

Maapallo vetää kappaleita puoleensa voimalla, joka riippuu kappaleen massasta. Todellisuudessa painovoiman kappaleeseen aiheuttama kiihtyvyys vaihtelee riippuen sijainnista maapallon pinnalla, mutta tässä työssä painovoiman suuruuden g ajatellaan olevan kaikkialla $9.807 m/s^2$. Lisäksi painovoiman suunnan oletetaan aina osoittavan kappaleen keskipisteestä maapallon keskipisteeseen. Tällöin ohjukseen kohdistuvalle painovoimalle $\boldsymbol{F}_g$ voidaan muodostaa yhtälö

$$\boldsymbol{F}_g = -g\frac{\boldsymbol{r}_{eb}}{\|\boldsymbol{r}_{eb}\|}m, \tag{5.33}$$

jossa $\boldsymbol{r}_{eb}$ on ohjuksen paikkavektori maan keskipisteestä ja m on ohjuksen massa.

Tässä työssä oletetaan, että moottorin tuottama työntövoima kohdistuu suoraan ohjuksen massakeskipisteeseen kappalekoordinaatiston x-akselin suuntaisesti, eikä siten muodosta kappaleeseen momentteja. Lisäksi oletetaan, että moottori ei kuluta polttoainetta, jonka väheneminen muuttaisi ohjuksen massaa. Tällöin ohjuksen työntövoima noudattaa kaavaa

$$\boldsymbol{F}_t = \begin{bmatrix} T_x \\ 0 \\ 0 \end{bmatrix}, \tag{5.34}$$

jossa T_x on ohjuksen työntövoima kappalekoordinaatiston x-akselin suuntaisesti.

Oikean ohjuksen lentorataa muunnetaan aerodynaamisten voimien avulla hyödyntäen ohjukseen kiinnitettyjä ohjaussiivekkeitä. Ohjaussiivekkeiden asennnon muutos aiheuttaa ohjukseen voimia sekä momentteja, jotka vääntävät ohjuksen johonkin haluttuun kulmaan ilman kulkusuuntaa vastaan. Tässä työssä kuitenkin oletetaan, että ohjuksen ohjausjärjestelmä voi suoraan antaa tarvittavat momentit ohjuksen rungolle ja siivekkeiden asentojen muutokset jätetään huomioimatta. Tällöin ohjausjärjestelmän tuottamia momentteja voidaan ilmaista kaavalla

$$\boldsymbol{\tau}_c = \begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix}, \tag{5.35}$$

jossa C_x, C_y ja C_z kuvaavat tuotettuja momentteja kappalekoordinaatiston eri akseleiden suhteen. Todellisuudessa ohjuksen ohjausjärjestelmän antamien ohjauskäskyjen ja suoritettujen ohjausten välissä on viivettä johtuen esimerkiksi ohjuksen hydrauliikasta, mutta tässä työssä ohjausmomenttien oletetaan välittyvän välittömästi ohjuksen runkoon.

5.7 Simulaattorin ohjusmallin fysiikka

Työn lopussa esitettävä ohjussimulaattori noudattaa luvussa esitettyjä yhtälöitä ohjusten lentoratojen mallinnuksessa. Kohdan 5.2 liikeyhtälöiden ja kohdan 5.4 momenttiyhtälöiden avulla simulaattorissa määritetään ohjuksen sijainnin sekä asennon muuttuminen simulaation eri vaiheissa. Lisäksi kohdassa 5.5 esitettyjä aerodynaamisia voimia hyödynnetään ohjuksen lentoradan muuttamisessa ja hakeutumisalgoritmin tarvitsemien kiihtyvyyksien tuottamisessa. Kohdassa 5.6 esitettyjä yhtälöitä painovoimalle, moottorin työntövoimalle ja ohjausjärjestelmän tuottamille momenteille käytetään ohjusmallissa tuottamaan mallinnetuun ohjukseen muodostuvat voimat sekä momentit.

5.7.1 Ohjusmallin dynamiikka

Alakohdassa 3.5.1 rajattiin ohjusmalli simuloimaan symmetrisiä ohjuksia. Käytännössä ohjusmalli olettaa ohjusten olevan sylinterimäisiä, jolloin kohdassa 5.3 esitetystä inertiamatriisista muodostuu ohjuksille kaavan

$$\boldsymbol{I} = \begin{bmatrix} \frac{1}{2}mr^2 & 0 & 0 \\ 0 & \frac{1}{12}mh^2 + \frac{1}{4}mr^2 & 0 \\ 0 & 0 & \frac{1}{12}mh^2 + \frac{1}{4}mr^2 \end{bmatrix} \tag{5.36}$$

mukainen, jossa m on koko ohjuksen massa, r on ohjuksen rungon halkaisija ja h on ohjuksen pituus. Koska sylinterimäisen kappaleen inertiamatriisi on diagonaalinen, ohjuksen kulmakiihtyvyyksien laskentaan voidaan käyttää momenttiyhtälöitä 5.27, 5.28 ja 5.29.

Ohjuksen lentäessä ilmakehässä, siihen muodostuvat aerodynaamiset momentit pyrkivät vakauttamaan ohjuksen lentoa. Ohjusmallissa kuitenkin jätetään aerodynaamiset momentit huomioimatta ja oletetaan, että ohjuksen ohjausjärjestelmä kykenee sekä tasapainoittamaan ohjuksen lennon että tuottamaan tarpeeksi suuria ohjausmomentteja, jotta ohjus saadaan käännettyä tarvittavaan asentoon.

Maapallon pyörimisestä aiheutuvilla Coriolis kiihtyvyydellä ja keskipakoiskiihtyvyydellä on pieni vaikutus ilmataistelu- ja ilmatorjuntaohjuksen lentorataan, sillä ohjuksien lentämä matka on usein melko lyhyt ja ohjuksen ohjausvoimista aiheutuu ohjukseen huomattavasti suurempia kiihtyvyyksiä. Tästä johtuen simulaatiossa käytetyssä ohjusmallissa maapallon pyöriminen jätetään huomioimatta, jolloin $\boldsymbol{\omega}_e = 0$ ja liikeyhtälöissä esiintyvät Coriolis kiihtyvyys sekä keskipakoiskiihtyvyys supistuvat

pois. Tällöin kaava 5.11 sievenee muotoon

$$(\ddot{\boldsymbol{r}}_{eb})_b = \frac{\boldsymbol{F}}{m} - \boldsymbol{\omega}_b \times (\dot{\boldsymbol{r}}_{eb})_b. \tag{5.37}$$

Pyörimisnopeuden poistumisen vuoksi maahan liitetty ECI koordinaatisto ja ECEF koordinaatisto ovat aina samassa kulmassa toisiinsa nähden, jonka vuoksi ne voidaan ajatella täysin toisiaan vastaaviksi koordinaatistoiksi.

Ohjusmallissa käytettävät liike- sekä momenttiyhtälöt on esitettynä taulukossa 5.1 ja mallissa huomioitavat voimat sekä momentit on esitettynä taulukossa 5.2.

Taulukko 5.1 *Simulaatiossa käytetyt liike- ja momenttiyhtälöt.*

Nimi	Yhtälö
Liikeyhtälö	$(\ddot{\boldsymbol{r}}_{eb})_b = \frac{\boldsymbol{F}}{m} - \boldsymbol{\omega}_b \times (\dot{\boldsymbol{r}}_{eb})_b$ *Yhtälön avulla voidaan laskea voimien aiheuttamat muutokset kappaleen liikkeeseen, kun maapallon pyörimistä ei huomioida.*
Momenttiyhtälöt	$(\dot{\omega_x})_b = \frac{\tau_x + \omega_y\omega_z(I_{yy} - I_{zz})}{I_{xx}}$ $(\dot{\omega_y})_b = \frac{\tau_y + \omega_z\omega_x(I_{zz} - I_{xx})}{I_{yy}}$ $(\dot{\omega_z})_b = \frac{\tau_z + \omega_x\omega_y(I_{xx} - I_{yy})}{I_{zz}}$ *Yhtälöiden avulla voidaan laskea kappaleeseen muodostuvat kulmakiihtyvyydet kappaleeseen kohdistuvien momenttien seurauksena kun inertiamatriisi on diagonaalinen.*

5.7.2 Ohjusmallin aerodynamiikka

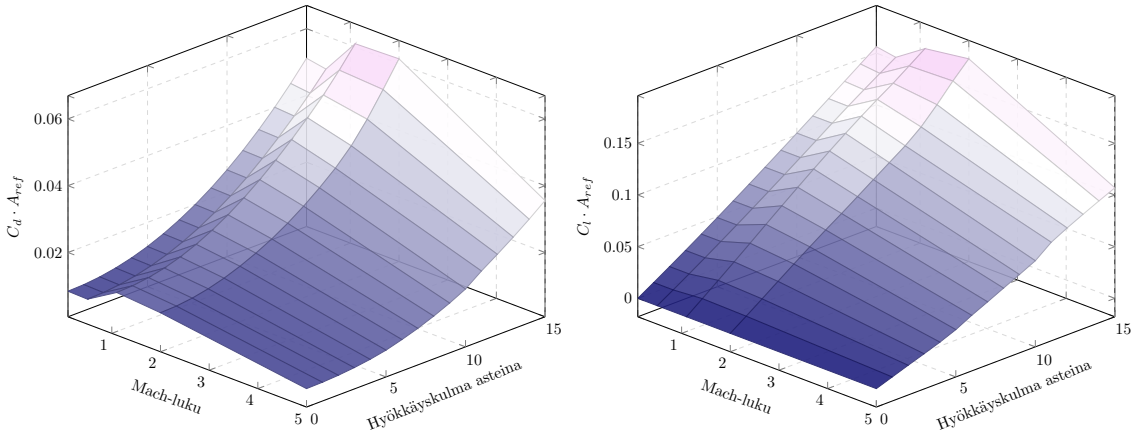
Simulaattorin ohjusmallissa hyödynnetään, alakohdassa 5.5.2 esitellyn RASAero:n avulla tuotettuja aerodynaamisia kertoimia. Kertoimet tuotetaan mach-luvun ja kohtauskulman funktioina, kohdassa 3.5 esitetylle lyhyen matkan sekä keskipitkän matkan ohjukselle. Tuotetut aerodynaamiset kertoimet lyhyen matkan ohjukselle on

Taulukko 5.2 *Ohjuksissa käytetty inertiamatriisi sekä ohjuksissa huomioidut voimat ja momentit.*

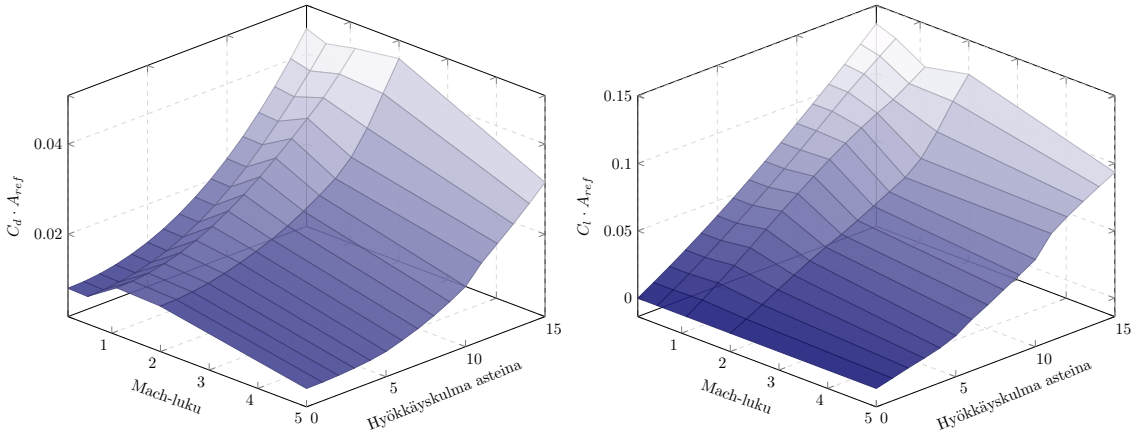
Nimi	Yhtälö
Inertiamatriisi	$\boldsymbol{I} = \begin{bmatrix} \frac{1}{2}mr^2 & 0 & 0 \\ 0 & \frac{1}{12}mh^2 + \frac{1}{4}mr^2 & 0 \\ 0 & 0 & \frac{1}{12}mh^2 + \frac{1}{4}mr^2 \end{bmatrix}$
Aerodynaamiset voimat	$F_l = \frac{1}{2}\rho {v_w}^2 AC_l, \quad F_d = \frac{1}{2}\rho {v_w}^2 AC_d, \quad F_s = \frac{1}{2}\rho {v_w}^2 AC_s$ $\boldsymbol{F}_a = \begin{bmatrix} F_d & F_s & F_l \end{bmatrix}^T$ *(ilmaistu tuulikoordinaatistossa)* *Yhtälöiden avulla voidaan määrittää ilmakehän ohjukseen aiheuttamat aerodynaamiset voimat. Ilmantiheys ρ riippuu ohjuksen korkeudesta U.S. Standard Atmosphere-mallin mukaisesti. Referenssipinta-ala A ja kertoimet tuotetaan RASAero:n avulla.*
Painovoima	$\boldsymbol{F}_g = -g\frac{\boldsymbol{r}_{eb}}{\|\boldsymbol{r}_{eb}\|}m$ *Yhtälön avulla saadaan kappaleeseen vaikuttava painovoima, joka osoittaa aina maapallon keskipisteeseen.*
Ohjausmomentit	$\boldsymbol{\tau}_c = \begin{bmatrix} C_x & C_y & C_z \end{bmatrix}^T$ *(ilmaistu kappalekoordinaatistossa)* *Ohjuksen ohjausjärjestelmän tuottamat ohjausmomentit.*
Työntövoima	$\boldsymbol{F}_t = \begin{bmatrix} T_x & 0 & 0 \end{bmatrix}^T$ *(ilmaistu kappalekoordinaatistossa)* *Ohjuksen moottorin tuottama työntövoima.*

esitettynä kuvassa 5.3 ja keskipitkän matkan ohjukselle kuvassa 5.4. Kuvissa tummansiniset värit ovat lähellä nollaa ja mitä suuremmaksi kuvien arvot kasvavat sitä vaaleampi sekä punertavampi pinta kuvan graafissa on. Molemman kuvan aerodynaamiset kertoimet on lisäksi kerrottu RASAerolta saadulla referenssipinta-alalla, jotta kuvaajia voi vertailla keskenään.

Kuvista voidaan havaita, että lyhyen matkan ohjuksen ilmanvastuskerroin sekä



Kuva 5.3 *RASAeron tuottamat aerodynaamiset kertoimet lyhyen matkan ohjukselle mach-luvun ja hyökkäyskulman funktiona. Vasen kuva esittää ilmanvastuskerrointa C_d ja oikea kuva nostekerrointa C_l.*



Kuva 5.4 *RASAeron tuottamat aerodynaamiset kertoimet keskipitkän matkan ohjukselle mach-luvun ja hyökkäyskulman funktiona. Vasen kuva esittää ilmanvastuskerrointa C_d ja oikea kuva nostekerrointa C_l.*

nostekerroin kasvaa nopeammin kuin keskipitkän matkan ohjuksella, kun ohjuksen hyökkäyskulma kasvaa. Tästä voidaan päätellä, että lyhyen matkan ohjus saa tuotettua keskpitkänmatkan ohjusta suurempia nostevoimia, jonka vuoksi se kykenee suorittamaan jyrkempiä ohjausliikkeitä. Toisaalta ohjausliikkeitä tehdessään, ilmanvastus hidastaa lyhyen matkan ohjuksen liikettä enemmän kuin keskipitkän matkan ohjuksen liikettä, joka lyhentää lyhyen matkan ohjuksen kantamaa. Simulaattorin käyttämässä ohjusmallissa RASAero:n tuottama data luetaan kaksiulotteisiin säiliöihin, jotka interpoloivan bilineaarisella interpoloinnilla asetettuja arvoja. Säiliöiltä ulostulona saatavat arvot vastaavat edeltävissä kuvissa esitettyjen pintojen pisteitä.

6. OHJUSSIMULAATTORIN TOTEUTUS

Ohjussimulaattorin vastuulla on vastaanottaa muiden simulaattoreiden lähettämiä viestejä ohjuksen laukaisutapahtumista ja tuottaa viesteissä saapuneiden tietojen avulla ohjusolioita. Simulaattori myös kuuntelee simulaatiossa olevien kohteiden tilatietoja, jotta ohjusolioille voidaan kertoa tietoa olioista, joihin niiden tulee hakeutua. Ohjussimulaattori lisäksi julkaisee ohjuksen tilatietoja muille simulaattoreille ja ilmoittaa niille ohjuksen räjähdyksestä. Räjähdyksen aiheuttamien tuhojen tarkastelu ja kohteiden tuhoaminen jätetään muiden simulaatioon osallistuvien simulaattoreiden vastuulle.

Tässä luvussa esitellään aiemissa luvuissa kuvatun ohjusmallin toteutus. Luvussa myös esitellään ohjusmallia hyödyntävän simulaattorin toteutus sekä kuvataan kuinka se hyödyntää ohjusmallia ohjusten simuloinnissa ja kuinka se liittyy kohdassa 2.4 esitettyyn valmiiseen järjestelmään. Lisäksi luvussa esitellään teknologiat, joita ohjussimulaattorin toteutuksessa hyödynnetään.

6.1 Hyödynnetyt teknologiat

Toteutuksessa hyödynnettävä ohjelmistokehys asettaa tiettyjä vaatimuksia ohjussimulaattorin teknologioille. Simulaattorin ohjelmointikielen täytyy vastata ohjelmistokehyksen kieltä, jotta sen tarjoamia kantaluokkia ja muita ominaisuuksia voidaan hyödyntää. Lisäksi tuotetun ohjelmiston tulee käyttää tuoterungon hyödyntämää teknologiaa sisäiseen viestinvälitykseen.

6.1.1 Ohjelmointikieli ja käytetyt kirjastot

Toteutuksen ohjelmointikielenä toimii *Java*. Java on yleiskäyttöinen, rinnakkainen ja oliopohjainen kieli. Kieli on staattisesti tyypitetty ja hoitaa olioiden varaaman muistin vapauttamisen roskienkeruun avulla. Java-koodi käännetään käännösaikana tavukoodiksi, joka ajetaan Javan virtuaalikoneessa. Virtuaalikone suorittaa tavukoodin komentojen perusteella varsinaisia suoritusalustan tarjoamia käskyjä. (Gosling, Joy, Steele & Bracha 2013) Tavukoodi ja virtuaalikone tekevät Javasta kielenä alustariippumattoman ja tuotettu ohjelma toimii millä vain alustalla, jos alustalle löytyy

toteutus Javan virtuaalikoneesta.

Toteutuksessa hyödynnetään Javalle tehtyä *Apache Commons Math* -kirjastoa. Apache Commons Math-kirjasto on Apache Software Foundationin tuottama kirjasto, jonka avulla voidaan suorittaa useita matemaattisia operaatioita (The Apache Software Foundation 2015). Simulaattorin toteutuksessa kirjastosta hyödynnetään etenkin sen `Vector3D`-luokkien tarjoamia vektorilaskennan operaatioita. Ohjelmassa 6.1 on Apache Commons Math-kirjastoa hyödyntävä Java-algoritmi alakohdassa 3.4.3 esitetystä suhteellisesta navigoinnista.

```
1  public static Vector3D proportionalNavigation(double N,
2                              // Suuntaviiva
3                              Vector3D LOSLine,
4                              // Ohjuksen nopeus
5                              Vector3D missileVelocity,
6                              // Kohteen nopeus
7                              Vector3D targetVelocity) {
8
9      // Lasketaan ohjuksen ja kohteen välinen nopeus
10     Vector3D Vr = targetVelocity.subtract(missileVelocity);
11
12     // Lasketaan suuntaviivan kulmanopeus
13     Vector3D w1 = Vector3D.crossProduct(LOSLine, Vr);
14     Vector3D w2 = w1.scalarMultiply(1 /
15                          Vector3D.dotProduct(LOSLine, LOSLine));
16
17     // Palautetaan vaaditut kiihtyvyydet
18     return Vector3D.crossProduct(w2, missileVelocity)
19                         .scalarMultiply(N);
20 }
```

Ohjelma 6.1 *Suhteellisen navigoinnin suorittava algoritmi toteutettuna hyödyntäen Apache Commons Math -kirjastoa.*

Kirjastosta hyödynnetään myös sen `Rotation`-luokkaa, jonka avulla voidaan suorittaa alakohdissa 4.4.1 ja 4.4.2 määriteltyjä koordinaatistorotaatioita helposti. Esimerkiksi ohjelmassa:

```
Rotation bodyToWind = new Rotation(new Vector3D(1.0,0.0,0.0),
                                   velVectorInBodyCoordinates);
bodyToWind.applyTo(testVector);
```

luodaan ensin rotaatio-olio kappalekoordinaatiston x-akselin sekä kappalekoordinaatiston suhteen esitetyn nopeusvektorin avulla, ja sen jälkeen muunnetaan vektorin `testVector`-esitys kappalekoordinaatistosta tuulikoordinaatistoon. `Rotation`-

luokka hyödyntää alakohdassa 4.4.2 esiteltyjä kvaternioita sisäisenä toteutuksenaan ja luokan instansseilta voi pyytää rotaatiota vastaavat Tait-Bryan kulmat.

Kirjastosta käytetään simulaattorin toteutuksessa myös sen differentiaaliyhtälöiden ratkaisuun tarkoitettuja, numeerisia menetelmiä hyödyntäviä, luokkia taulukossa 5.1 esiteltyjen liikeyhtälöiden ja momenttiyhtälöiden ratkaisemiseen.

6.1.2 Sisäinen viestinvälitys ja muunnokset HLA:han

Kohdassa 2.4 esitellyn, ohjelmistokehyksenä toimineen, järjestelmän sisäisenä viestinvälitysmenetelmänä toimii *Java Message Service (JMS)*. JMS on Sun Microsystems:in (nykyään Oracle) Javalle määrittelemä rajapinta, joka kuvailee menetelmät ohjelmistokomponenttien väliseen viestinvälitykseen. Se pyrkii yhdistämään yleisimmät viestinvälitysjärjestelmien tarjoamat operaatiot. JMS:n avulla järjestelmän eri kokonaisuudet voivat lähettää ja vastaanottaa viestejä, joko johonkin määritettyyn viestijonoon tai julkaisija-tilaaja tyylisesti, jossa julkaisijat ja tilaajaat eivät tunne toisiaan. (Sun Microsystems, Inc 2002)

Järjestelmän sisäiset simulaattorit kommunikoivat ohjussimulaattorille JMS:än lähetettyjen viestien avulla ja vastaavasti ohjussimulaattori lähettää JMS:än tietoja ohjuksen tilasta sekä ohjuksen räjähdyksestä. Kuten kohdassa 2.4 mainittiin, simulaattoreita sisältäneet järjestelmät kommunikoivat toisten vastaavien järjestelmien kanssa HLA:n RTI:n avulla ja muunnos sisäisestä viestinvälityksestä HLA:n RTI:hin tehdään HLA välittäjän avulla. HLA-välittäjä huolehtii JMS-viestien vastaanotosta ja niiden lähettämisestä sekä HLA:han tehtyjen muutoksien kuuntelemisesta. Välittäjä hyödyntää työn ohjussimulaattoria varten toteutettua kahta luokkaa, joista toisen tehtävänä on muuntaa ohjuksiin liittyviä JMS-viestejä HLA:han sopiviksi ja toisen tehtävänä muuntaa HLA:n muutoksia JMS-viesteiksi. Kahden luokan toteutuksessa hyödynnetään ohjelmistokehyksen tarjoamia luokkia, jotka yksinkertaistivat HLA:n läpi tapahtuvaa kommunikointia.

6.1.3 Simulaattorin koostaminen ja järjestelmään liittyminen

Simulaattorin toimintaan osallistuvat komponentit sidotaan toisiinsa Spring Framework:in tarjoamien `ApplicationContext`-kontekstien avulla. Spring Framework on laaja ohjelmistokehys, joka sisältää paljon erilaisiin tarkoituksiin käytettyjä ominaisuuksia ja valmiita palveluita. Se tarjoaa esimerkiksi toteutuksia tietokantojen käsittelyyn ja käyttäjän autentikointiin. Spring mahdollistaa vain osan sen tarjoamien ominaisuuksien hyödyntämisestä ja ohjussimulaattorin toteutuksessa hyödynnetään Springistä vain sen tarjoamia kontekseja. Springin konteksteissa voidaan määrittää

kuinka simulaattorin oliot tulee rakentaa, konfiguroida ja yhdistää muihin olioihin. (Johnson et al. 2015) Kontekstissa määritellään myös esimerkiksi kuinka järjestelmän sisäiseen viestinvälitykseen liitytään ja tarvittavat oliot JMS:n käyttöön liitetään kontekstissa simulaattorin rakentajaparametreihin. Kontekstit kuvataan Extensible Markup Language (XML)-muotoisten konfiguraatiotiedostojen avulla, josta on esitettynä yksinkertainen esimerkki ohjelmassa 6.2. Esimerkkikonfiguraatio sisältää `bean`-nimisiä lohkoja, jotka vastaavat käytännössä ohjelman olioita. Konfiguraatiossa muodostetaan ohjussimulaattori-olio, jolle annetaan rakentajaparametrinä JMS-viestinvälitykseen käytetty olio ja asetetaan `name`-muuttujaan arvoksi teksti “Ohjussimulaattori.”.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4    xsi:schemaLocation="http://www.springframework.org/schema/beans
5     http://www.springframework.org/schema/beans/spring-beans.xsd">
6
7     <bean id="simulator" class="fi.tty.di.MissileSimulator">
8         <constructor-arg ref="jmsTemplate"/>
9         <property name="name" value="Ohjussimulaattori."/>
10    </bean>
11    <!-- Muiden beanien määrittelyjä, kuten jmsTemplate -->
12 </beans>
```

Ohjelma 6.2 *Esimerkki konfiguraatiotiedosto, joka määrittelee Spring kontekstin.*

Simulaattorin ohjelmakooditiedostot käännetään Java luokiksi ja lopuksi kaikki simulaattorin tiedostot paketoidaan *OSGi*:a tukeviksi *JAR-tiedostoiksi (Java Archive)*. OSGi on OSGi Alliance:n määrittelemä standardi, jonka tarkoituksena on lisätä Java:n modulaarisuutta. OSGi:ssa yksittäistä modulaarista yksikköä kutsutaan *bundle*:ksi. Bundle on JAR-tiedosto, joka sisältää toiminnallisuutta tarjoavia tiedostoja sekä bundlen sisältöä kuvaavan tiedoston. Sisältöä kuvaavan tiedoston avulla määritetään esimerkiksi mitä luokkia bundlesta annetaan muiden bundlejen käyttöön ja mitä luokkia tulee olla olemassa, jotta bundlen voi ottaa käyttöön. OSGi:n tarkoituksen on sallia ohjelman dynaaminen koostaminen komponenteista, joilla ei ole aiempaa tietoa toisistaan. (OSGi Alliance 2014)

Järjestelmä lukee automaattisesti ohjussimulaattorin bundlejen sisällä olevat Spring-kontekstit kun bundlet ladataan. Tällöin konteksteissa kuvatut oliot, kuten esimerkiksi simulaattorit ja JMS:n käytössä tarvitsemat oliot, instantioidaan ja liitetään toisiin olioihin, jotka niitä tarvitsevat. Simulaattori asettuu kuuntelemaan JMS:ltä tulevia viestejä ja se käynnistyy kun se vastaanottaa viestin, joka käskee

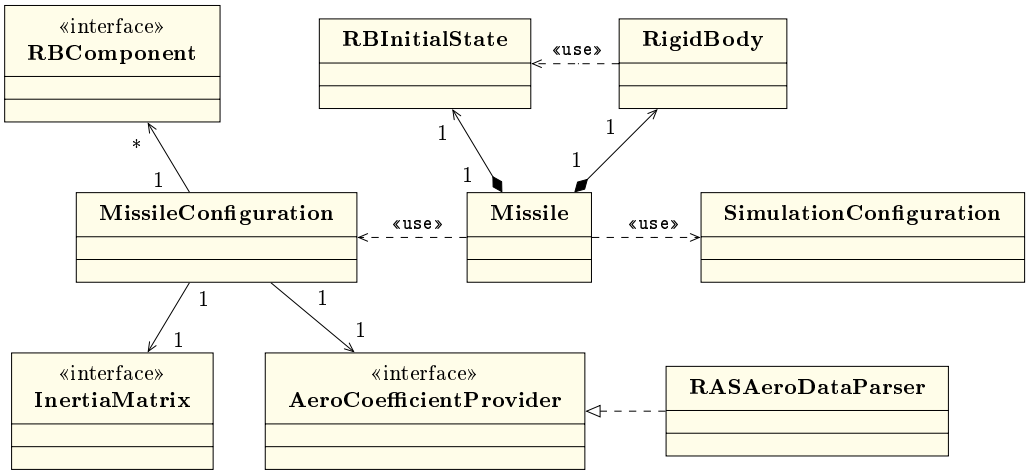
simulaattoreiden käynnistyä.

6.2 Ohjusmallin rakenne

Simulaattorin ohjusmalli noudattaa kohdissa 3.5, 4.6 ja 5.7 ohjuksille määriteltyjä ominaisuuksia. Seuraavissa alakohdissa esitetään ohjusmallin toiminnan ja rakenteen kannalta tärkeimpiä luokkia sekä niiden välisiä riippuvuuksia.

6.2.1 Mallin yleiskuvaus

Ohjusmallin keskeisimpänä luokkana toimii `Missile`-luokka, sillä sen vastuulla on ohjusolion rakentaminen annettujen alkuparametrien mukaisesti sekä simulaattorin pyytämän simulaatioaskeleen välittäminen ohjuksen simulaatiosta vastaaville olioille. Kuvassa 6.1 on esitettynä `Missile`-olion rakentajaparametreinä saamat oliot sekä oliot, jotka se itse instantioi. `Missile`-olio saa alkuparametrinä ohjuksen ominaisuudet määrittelevän `MissileConfiguration`-olion sekä yleistä tietoa simulaatiosta sisältävän `SimulationConfiguration`-olion. `Missile`-olio saa näiden lisäksi myös tiedon ohjuksen laukaisupaikasta, asennosta, alkunopeudesta sekä osoittimen simulaatiossa oleviin kohteisiin ja tunnisteen, joka yksilöi ohjukselle asetetun kohteen.



Kuva 6.1 *Ohjusmallin keskeisimmät luokat.*

Ohjuksen rakentajaparametrinä saama `MissileConfiguration`-olio sisältää esimerkiksi `InertiaMatrix`-rajapinnan toteuttavan, taulukon 5.2 inertiamatriisin mukaisen olion sekä muita ohjuksen staattisia tietoja kuten massan ja ohjuksen mitat. Olio sisältää myös ohjuksen aerodynaamiset ominaisuudet määrittelevän, `AeroCoefficientProvider`-rajapinnan toteuttavan, `RASAeroDataParser`-olion,

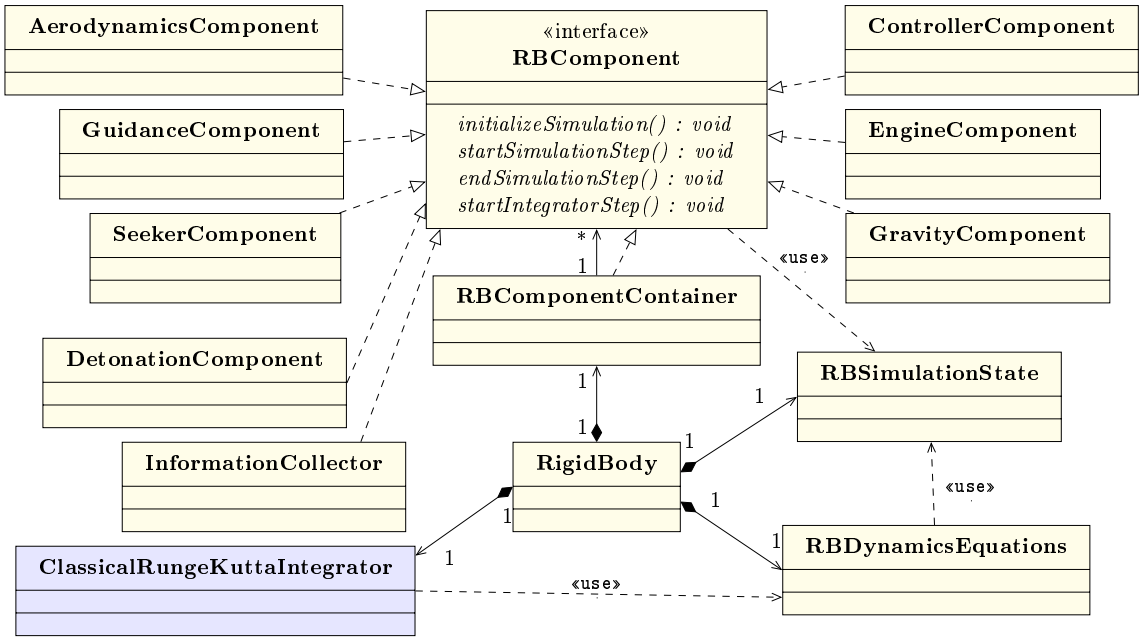
jonka tehtävänä on lukea alakohdassa 5.5.2 esitellyn RASAero:n tuottamaa dataa ja lisätä se ohjelman tietorakenteisiin. Tietorakenteena `RASAeroDataParser`-oliossa hyödynnetään työtä varten toteutettua kaksiulotteista taulukkoa, joka automaattisesti interpoloi bilineaarisella interpoloinnilla asetettujen arvojen välisiä arvoja. Taulukoiden sisällöt työn ohjuksille vastaavat alakohdassa 5.7.2 esitettyjä kuvaajia. `MissileConfiguration`-olio lisäksi sisältää joukon ohjuksen toiminnallisuuksia määritteleviä `RBComponent`-rajapinnan toteuttavia olioita. Tarkempi kuvaus `RBComponent`-komponenteista on esitettynä seuraavassa alakohdassa.

Alkuparametrien avulla `Missile`-olio luo jäykän kapaleen alkutilan määrittelevän `RBInitialState`-instanssin, jota `Missile`-olio hyödyntää `RigidBody`-olion luomisessa. Lisäksi ohjusolio lisää `MissileConfiguration`-oliossa määritellyt `RBComponent`-rajapinnan toteuttavat oliot `RigidBody`-oliolle.

6.2.2 Jäykkä kappale ja toimintaa laajentavat komponentit

`Missile`-olion luoma `RigidBody`-olio instantioi mallin dynamiikat määrittelevän `RBDynamicsEquations`-olion ja dynaamisten yhtälöiden ratkaisussa käytetyn `ClassicalRungeKuttaIntegrator`-olion. `ClassicalRungeKuttaIntegrator` on osa Apache Commons Math -kirjastoa, kuten myös rajapinta, jonka `RBDynamicsEquations`-luokka toteuttaa. `RigidBody`-olio antaa integroitavat yhtälöt integraattorille sekä ylläpitää jäykän kappaleen simulaation tilaa simulaatioaskelten välissä. Sen vastuulla on myös dynamiikan yhtälöiden alustaminen esimerkiksi oikeaan kulmaan tai oikeaan alkunopeuteen. `RigidBody`-olio säilöö jäykän kappaleen toiminnallisuutta laajentavat komponentit `RBComponentContainer`-säiliöön, joka komponenttien tavoin toteuttaa `RBComponent`-rajapinnan. `RigidBody`-olio kutsuu `RBComponentContainer`-säiliön metodeja simulaation alustuksessa, ennen simulaatioaskeleen suoritusta ja simulaatioaskeleen suorituksen jälkeen. Säiliö välittää metodikutsut yksitellen sisältämilleen komponenteille, jonka jälkeen suoritus palaa takaisin `RigidBody`-oliolle. `RigidBody`-oliossa instantioidaan myös jäykän kappaleen simuloinnissa hyödynnetty `RBSimulationState`-olio, joka sisältää simulointiin liittyviä tilatietoja. Kuvassa 6.2 on esitettynä jäykän kappaleen mallinnuksesta vastaavat luokat sekä toiminnallisuudet toteuttavat komponentit. Kuvassa on esitettynä keltaisella värillä tässä työssä toteutetut osuudet ja sinisellä värillä aiemmin toteutetut osuudet, joita hyödynnetään ohjusmallin toiminnassa.

Taulukossa 5.1 esitetyt dynamiikan yhtälöt sisältämä `RBDynamicsEquations`-olio vastaa ohjuksen liikkeen ja asennon muutoksien laskemisesta ohjukseen kohdistuneiden momenttien sekä voimien seuraukse-



Kuva 6.2 *Ohjuksen toiminnallisuuden määrittelevät komponentit.*

na. `ClassicalRungeKuttaIntegrator` kutsuu yhtälöitä laskiessaan `RBDynamicsEquations`-olion integrointiin käytettyä metodia simulaatioaskelta pienemmin väliajoin. Integroinnin suorittavassa metodissa kutsutaan lisäksi `RBComponentContainer`-säiliön metodia ennen jokaisen integrointiaskeleen suorittamista. Jokaisella komponentilla on mahdollisuus lisätä parametrinä annettuun `RBSimulationState`-olioon voimia sekä momentteja. Parametrinä välitetään myös simulaation yleistä tietoa sisältävä `SimulationConfiguration`-olio. Kun `RBComponentContainer`-säiliön integraatioaskeleella tehty metodikutsu palaa `RBDynamicsEquations`-olion suoritukseen, hyödyntää `RBDynamicsEquations`-olio `RBSimulationState`-olioon komponenttien lisäämiä voimia sekä momentteja kiihtyvyyksien ja kulmakiihtyvyyksien laskemiseen. `RBComponent`-rajapintaa laajentavat komponentit vastaavat kohdassa 3.5 esiteltyjen ohjuksen toiminnallisuuksien ja taulukossa 5.2 esitettyjen voimien toteuttamisesta. Toteuttamalla erilaisia ominaisuuksia tarjoavia `RBComponent`-rajapintaa laajentavia luokkia, voisi `RigidBody`-luokan avulla simuloida myös esimerkiksi lentokoneita.

`EngineComponent`-luokasta tuotettujen olioiden vastuulla on suorittaa ohjuksen moottorin toiminta. Käytännössä oliot lisäävät parametrinä saatuun `RBSimulationState`-olioon ohjuksen x-akselin suuntaisen voiman, jos simulaatioaika on pienempi kuin moottorin käynnissäoloaika. Ohjelmassa 6.3 on esitettynä ohjuksen työntövoiman määrittelevä komponentti.

```
1  public class EngineComponent implements RBComponent {
2      private double start;
3      private double end;
4      private double force;
5      // Rakentajaparametreinä moottorin käynnissäoloaika ja voima
6      public EngineComponent(double start, double end,
7                             double force) {
8          this.start = start;
9          this.end = end;
10         this.force = force;
11     }
12     // Kutsutaan kun simulaatio aloitetaan
13     public void initializeSimulation(double t,
14                                      RBSimulationState state) {}
15     // Kutsutaan ennen numeerista integrointia
16     public void startSimulationStep(double t,
17                                      RBSimulationState state) {}
18     // Kutsutaan numeerisen integroinnin jälkeen
19     public void endSimulationStep(double t,
20                                      RBSimulationState state) {}
21     // Kutsutaan jokaisella integraattorin askeleella
22     public void startIntegratorStep(double t,
23                                      RBSimulationState state) {}
24         if (t >= this.start && t < this.end) {
25             Vector3D thrustForce = new Vector3D(force, 0.0, 0.0);
26             // Lisää tiedon moottorin tuottamasta voimasta
27             state.addToVector(StateVector.FORCE, thrustForce,
28                         BasePresentation.BODY);
29         }
30     }
31 }
```

Ohjelma 6.3 *Ohjuksen työntövoiman määrittelevä komponentti jäykälle kappaleelle.*

`GravityComponent`-luokan oliot tuottavat ohjukseen maapallon aiheuttaman painovoiman. `AerodynamicsComponent`-oliot vastaavat ohjuksen aerodynaamisten voimien tuottamisesta. Aerodynamiikat tuottava komponentti hyödyntää toiminnassaan `AeroCoefficientProvider`-rajapinnan toteuttavaa luokkaa. `SeekerComponent`-oliot vastaavat ohjuksen hakupään toteutuksesta ja ne lisäävät `RBSimulationState`-olioon tietoja kohteesta, kuten sen sijainnin ja nopeuden. Komponentti lisää tilaolioon tietoja myös esimerkiksi ohjuksen ja kohteen välisestä etäisyydestä. `GuidanceComponent`-oliot kaivavat hakupään `RBSimulationState`-olioon lisäämiä tietoja ja laskee niiden sekä suhteellisen

navigoinnin toteuttavan algoritmin avulla tarvittavat kiihtyvyydet, joilla ohjus osuu kohteeseensa. Kiihtyvyydet lisätään `RBSimulationState`-olioon, josta ohjuksen ohjausjärjestelmän toteutuksesta vastaavat `ControllerComponent`-oliot hakevat ne. `ControllerComponent`-oliot tuottavat ohjausmomentteja, joiden avulla ohjus käännetään kulmaan, jossa aerodynaamisten nostevoimien aiheuttamat kiihtyvyydet vastaavat mahdollisimman hyvin ohjauslain tarvitsemia kiihtyvyyksiä. `DetonationComponent`-oliot toteuttavat ohjuksen sytyttimen ja ne hakevat tilaoliosta tietoja ohjuksen sekä kohteen välisestä etäisyydestä. Kun etäisyys on riittävän pieni, eikä se enää pienene, lisää `DetonationComponent`-olio `RBSimulationState`-olioon tiedon ohjuksen räjähdyksestä. `InformationCollector`-oliot keräävät tietoja `RBSimulationState`-oliosta, jonka lisäksi ne tarjoavat julkisen synkronoidun rajapinnan tiedosta kiinnostuneille osapuolille.

6.2.3 Jäykän kappaleen simuloinnin tilaolio

Jäykkä kappale ja integroidut yhtälöt välittävät jokaisessa komponenteille tehdyssä metodikutsussa `RBSimulationState`-olion, joka sisältää tietoja kappaleen simulaation tilasta. Olio sisältää vektoreita, koordinaatistomuunnoksia sekä yksittäisiä parametrejä. Ennen jokaista integrointiaskelta, `RBDynamicsEquations`-olion yhtälöissä lisätään tilaolioon tiedot kappaleen sijainnista ECEF-koordinaateissa sekä kappaleen nopeus ja kulmanopeus kappalekoordinaateissa. Yhtälöissä lisätään tilaolioon myös Apache Math kirjaston `Rotation`-olio, jonka avulla on mahdollista tehdä vektoreille koordinaatistomuunnoksia ECEF-koordinaattien ja kappalekoordinaattien välillä.

`RBSimulationState`-olio abstrahoi eri koordinaatistojen suhteen määriteltyjen vektoreiden muunnokset rajapintansa taakse. Komponentit voivat esimerkiksi summata jo valmiiksi tilaolion sisällä olevaan voimavektoriin eri koordinaatistojen suhteen määriteltyjä voimia ohjelman

```
state.addToVector(StateVector.FORCE, thrustForce,
                  BasePresentation.BODY);
state.addToVector(StateVector.FORCE, gravityForce,
                  BasePresentation.ECEF);
```

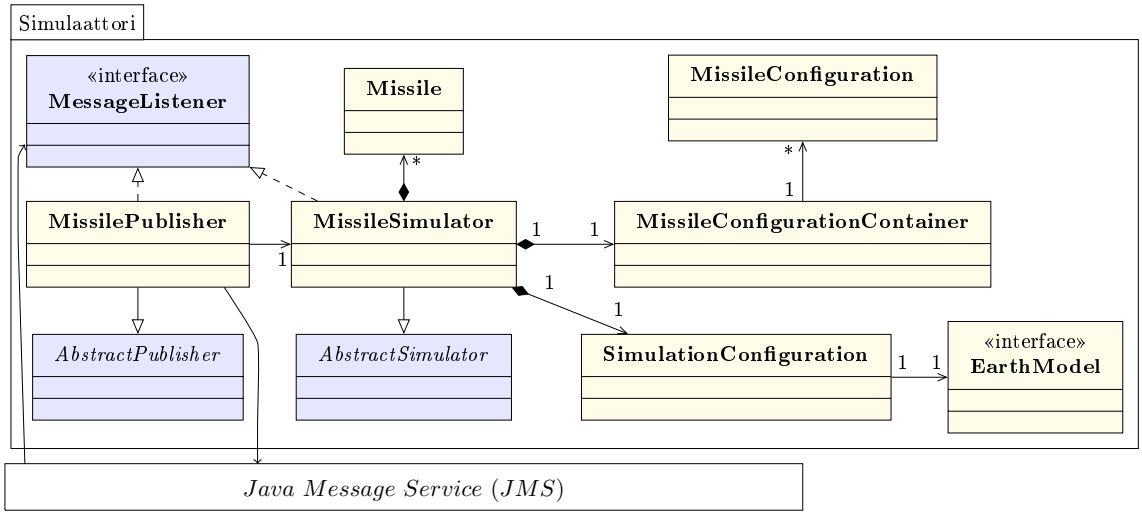
mukaisesti, jonka jälkeen jokin toinen tai sama komponentti voi hakea tilaoliosta saman vektorin tiedot haluamansa koordinaatiston suhteen määriteltynä ohjelman

```
state.getVector(StateVector.FORCE,
                BasePresentation.BODY);
```

mukaisesti. Ainoana vaatimuksena tilaolion toiminnalle on, että tilaoliolle tulee olla lisättynä ECEF-koordinaatiston ja halutun toisen koordinaatiston välisen muunnoksen suorittava `Rotation`-olio ennen kuin vektoreita voi lisätä toisen koordinaatiston suhteen määriteltyinä. Tilaolio myös varastoi tiedot haetuista, eri koordinaatistojen suhteen määritellyistä, vektoreista, jotta samoja koordinaatistomuunnoksia ei tarvitse tehdä useaan kertaan.

6.3 Simulaattorin rakenne ja suorituksen kulku

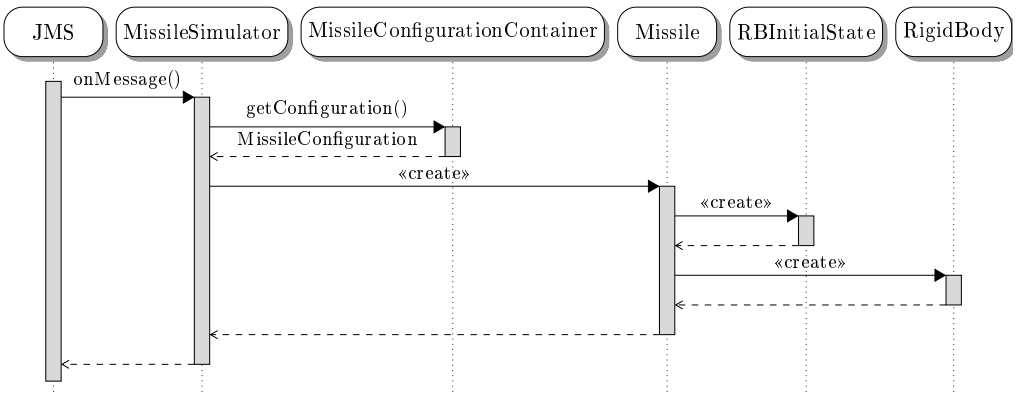
Työn simulaattori toimii osana kohdassa 2.4 kuvattua ympäristöä. Kuvassa 6.3 on esitettynä simulaattorin keskeisimpiä luokkia sekä kuinka ne liittyvät ohjusta mallintavaan `Missile`-luokkaan. Kuvassa on esitettynä työssä toteutetut osuudet keltaisella värillä ja aiemmin toteutetut osuudet sinisella värillä.



Kuva 6.3 *Ohjusten simulointiin osallistuvia luokkia.*

Kuvassa esitetty `MissileSimulator`-olio vastaa JMS:n kautta tulevien viestien, kuten ohjuksen laukaisukäskyn tai lentokoneen tilatiedon käsittelystä. Viestien avulla luokka tuottaa edellisessä kohdassa esitettyjä ohjusmallin toteuttavia olioita `Missile`-luokasta. `MissileSimulator` lisäksi sisältää tiedot simulaatiossa olevista lentokoneista, jotka se välittää eteenpäin ohjuksille hakeutumisen suorittamista varten. Kuvassa 6.4 on esitettynä ohjusmallin luomisessa tapahtuvat operaatiot. Ohjuksen luonti aloitetaan kun `MissileSimulator`-olio saa metodikutsun JMS:stä `onMessage()`-metodiin. Metodikutsun parametrinä tulee viestiolio, joka sisältää tietoja ohjuksen laukaisusta. Viesti sisältää esimerkiksi laukaisusijainnin, alkunopeuden sekä tietoja laukaistun ohjuksen tyypistä. Viestin tyyppitietojen avul-

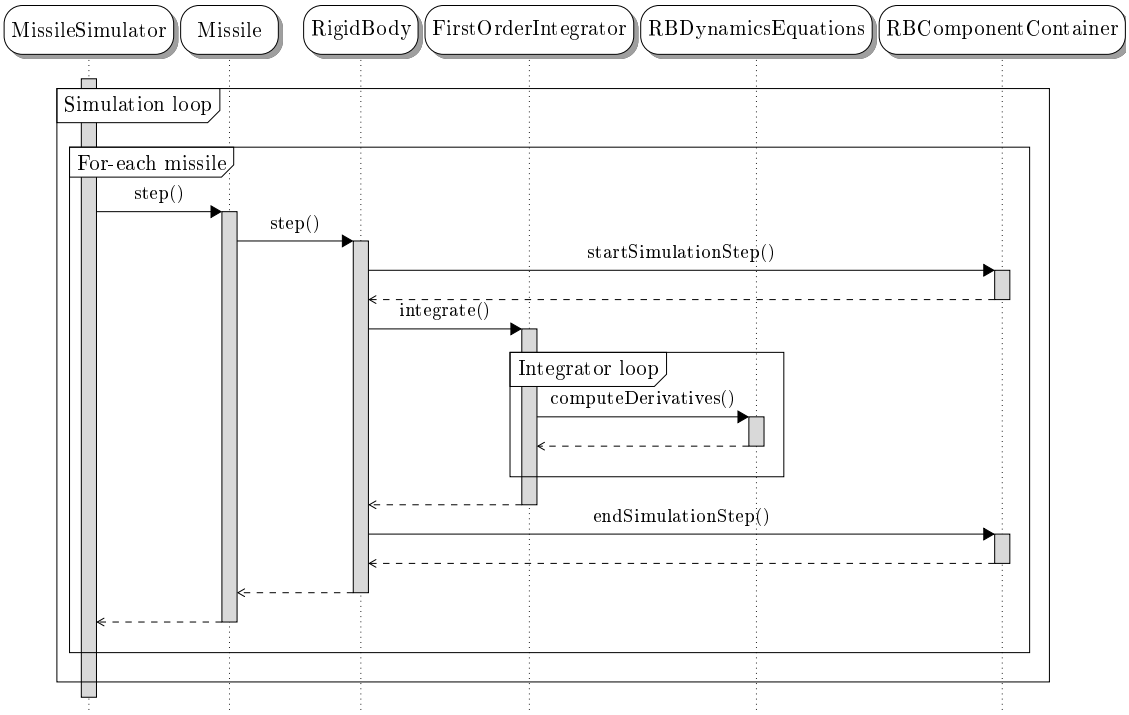
la MissileSimulator-olio hakee erilaisille ohjuksille määriteltyjä konfiguraatioita sisältävästä MissileConfigurationContainer-säiliöstä oikean ohjuskonfiguraation MissileConfiguration-olioina. Tämän jälkeen MissileSimulator-olio instantioi Missile-olion, joka edelleen luo jäykän kappaleen simulointiin tarvittavat oliot.



Kuva 6.4 *Ohjuksen luonnin sekvenssikaavio.*

MissileSimulator-luokan instanssi vastaa myös varsinaisen simulaatiosilmukan suorittamisesta ja käskee tietyin aikavälein Missile-olioiden simuloida tilaansa jonkin kiinteän aikavälin verran eteenpäin. Kuvassa 6.5 on esitettynä simulaatiosilmukan suorituksessa tapahtuvat operaatiot. MissileSimulator-olion simulaatiosilmukassa kutsutaan tietyin aikavälein jokaisen simulaatiossa olevan Missile-olion step()-metodia. Metodikutsussa annetaan parametrinä aika, joka on kulunut edellisestä simulaatiokierroksesta. Missile-olio välittää metodikutsun edelleen RigidBody-olion step()- metodille, joka aloittaa jäykän kappaleen simuloinnin. RigidBody-olio kutsuu step()-metodin alussa RBComponentContainer-säiliön startSimulationStep()-metodia, joka edelleen kutsuu samaa metodia sisältämileen komponenteille. Komponentit voivat muokata RBSimulationState-olion sisältöä ennen jäykän kappaleen simuloinnin aloitusta. Vastaavaa endSimulationStep()-metodia kutsutaan, kun simulaatioaskel päättyy. RigidBody-olio kutsuu step()-metodissa Apache Math:in FirstOrderIntegrator-rajapintaa laajentavaa numeerista integraattoria, joka sisäisessä integraatiosilmukassaan kutsuu tietyin integraatioaskelin RBDynamicsEquations-luokan computeDerivates()-metodia.

Metodin computeDerivates() sisällä asetetaan RBSimulationState-oliolle alkuarvoja, jonka jälkeen kutsutaan komponenttisäiliön startIntegratorStep()-metodia. Säiliö välittää metodikutsun sisältämilleen komponenteille, jotka muuttavat halutessaan RBSimulationState-olion sisältöä. Kun metodikutsu palaa takaisin RBDynamicsEquations-olion sisään, suoritetaan dynamiikan yhtälöiden laske-

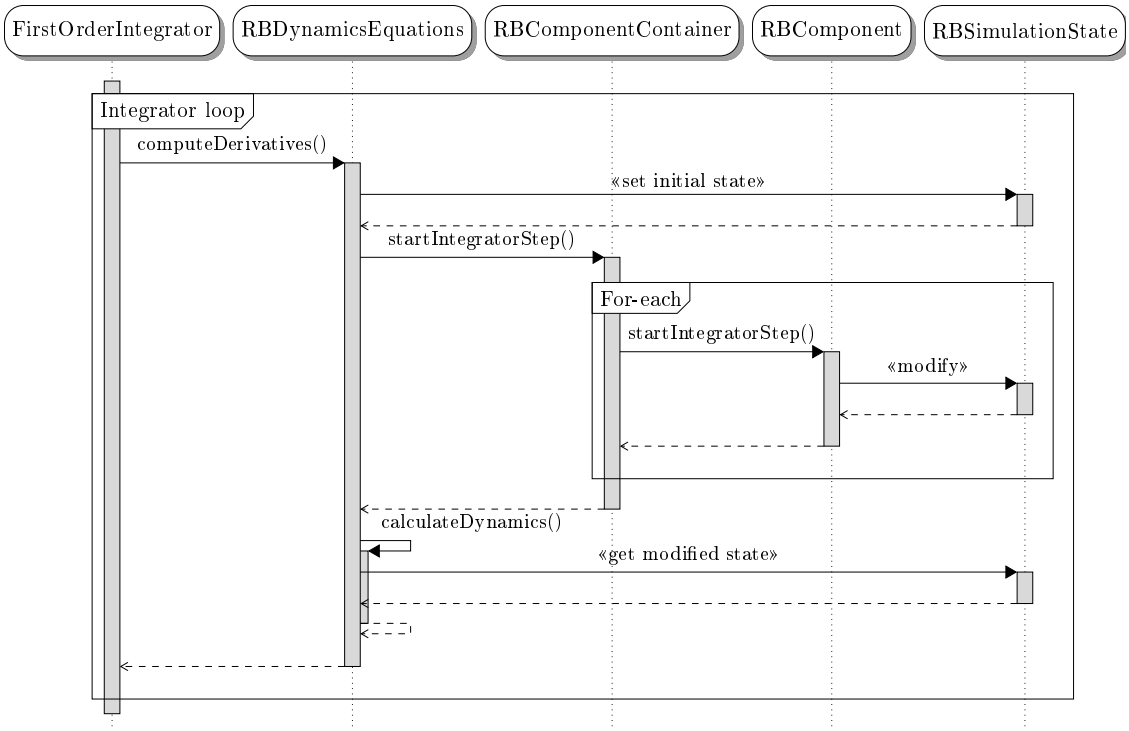


Kuva 6.5 *Ohjuksen simuloinnin sekvenssikaavio.*

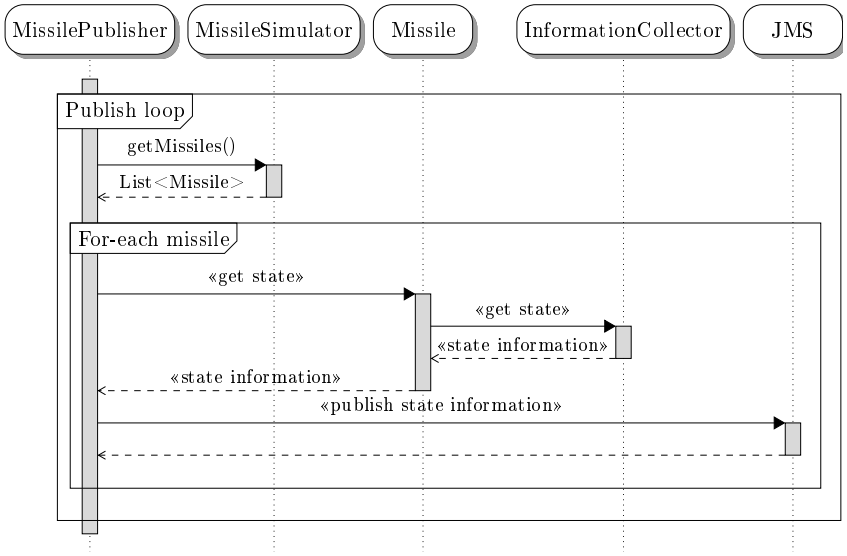
minen `RBSimulationState`-olion sisältämien tietojen avulla. Kuvassa 6.6 on esitettynä `computeDerivates()`-metodissa suoritettavat operaatiot.

`MissilePublisher`-luokan instanssi vastaa simuloitujen ohjusten tietojen lukemisesta ja niiden julkaisemisesta JMS:n. Kuvassa 6.7 on esitettynä ohjusten tietojen julkaisussa suoritettavat toiminnot. `MissilePublisher` toimii eri säikeessä kuin `MissileSimulator` ja julkaisee tietoja ohjuksista jonkin määritetyn ajan välein. Julkaisu tapahtuu julkaisusilmukassa, joka hakee simulaattorilta ohjusoliot ja pyytää jokaiselta ohjukselta tietoja sen nykyisestä tilastaa. Tilatiedot pyydetään `Missile`-olion metodien avulla, jotka edelleen hyödyntävät `InformationCollector`-olion tarjoamia synkronoituja metodeja tietoon, jota se on kerännyt `RBSimulationState`-oliosta. Lopuksi `MissilePublisher`-olio rakentaa ohjuksesta kerätyistä tiedoista viestin, jonka se julkaisee JMS:ään.

Simulaattori sisältää myös `SimulationConfiguration`-luokan, josta luodussa instanssissa määritellään ohjussimulaatiolle yleisiä tietoja kuten maapallon ominaisuudet `EarthModel`-rajapinnan avulla. `EarthModel`-rajapinnan toteuttaman luokan ominaisuudet vastaavat kohdassa 5.5.3 määriteltyjä ilmakehän ominaisuuksia ja kohdassa 4.2 määriteltyjä maapallon muotoja. Rajapinnan toteuttava luokka myös tarjoaa kohdissa 4.4.3 sekä 4.4.4 määritellyt operaatiot ECEF-koordinaattien ja geodeettisten koordinaattien välisiin muunnoksiin. Kuvissa näkyvä `MessageListener`-



Kuva 6.6 *Jäykän kappaleen simuloinnin sekvenssikaavio.*



Kuva 6.7 *Ohjusten tietojen julkaisun sekvenssikaavio.*

rajapinta on JMS:n tarjoama rajapinta, jonka avulla JMS välittää vastaanotetut viestit `MissileSimulator`- ja `MissilePublisher`-luokalle. `AbstractPublisher`- ja `AbstractSimulator`-kantaluokat ovat ohjelmistokehyksen tarjoamia luokkia, jotka hoitavat simulaattorin ajotilaan ja simulaattorin saavutettavuuteen liittyviä viestinvälityksiä.

7. OHJUSSIMULAATTORIN ARVIOINTI

Toteutetun ohjussimulaattorin toimintaa arvioitiin mittaamalla ohjusmallin suorituskykyä sekä simuloitujen ohjusten kykyä osua niille asetettuihin kohteisiin. Tämän lisäksi ohjusmallin simuloimien ohjusten lennon oikeellisuutta arvioitiin havainnoimalla tuotettuja lentoratoja visuaalisesti. Luvussa esitellään mittauksista saadut tulokset, jonka jälkeen saatuja tuloksia sekä koko ohjusmallin toteutusta ja sen soveltuvuutta käyttökohteeseen arvioidaan. Luvun lopussa myös esitellään ehdotuksia, joiden avulla toteutettua ohjusmallia sekä simulaattoria voisi jatkokehittää.

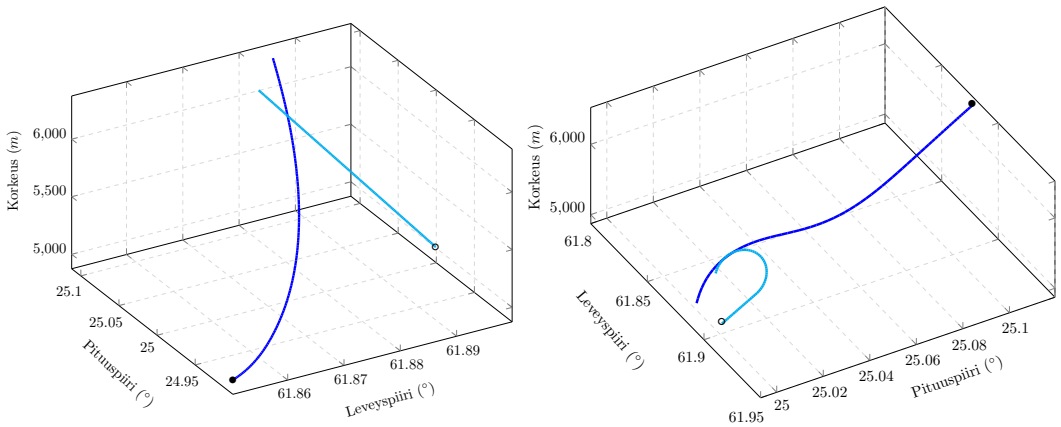
7.1 Mittaukset ja niistä saadut tulokset

Mittaukset suoritettiin työssä esitetyn lyhyen matkan ohjuksen avulla, hyödyntäen neljää erilaista skenaariota. Käytetyt skenaariot on esitettynä kuvissa 7.1 ja 7.2. Ensimmäistä skenaariota hyödynnetään simulaation osana toimivalle simulaattorille tehdyissä mittauksissa sekä suorituskykymittauksissa. Ohjusmallin mittauksessa hyödynnetään kaikkia neljää skenaariota.

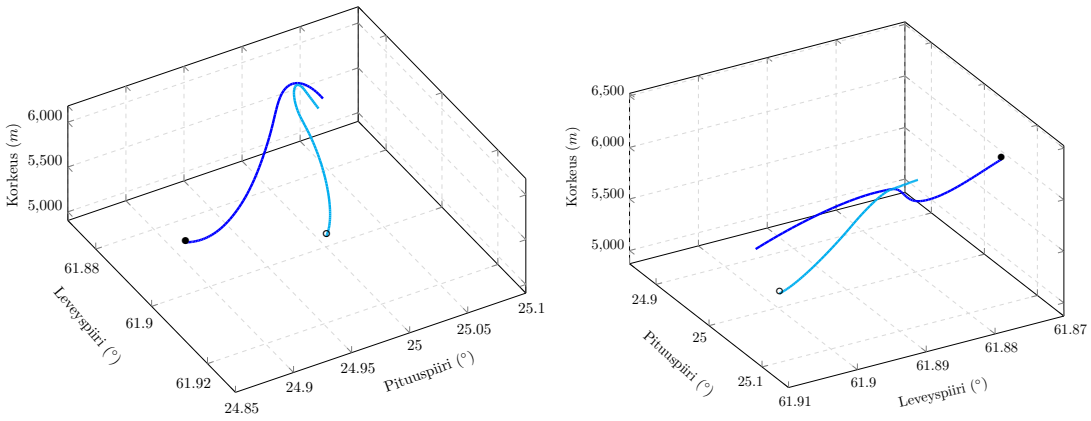
Kuvan 7.1 vasemmanpuoleisessa 1. skenaariossa kohde aloitti liikkumisen leveyspiiriltä 61.90°, pituuspiiriltä 25.00° ja korkeudelta $5000m$. Kohde liikkui tasaisesti $400m/s$ ECEF-koordinaatiston x-akselin suuntaisesti. Ohjus laukaistiin leveyspiiriltä 61.85°, pituuspiiriltä 24.90° ja korkeudelta $5000m$. Ohjus suunnattiin ennen laukaisuaan kohdetta kohti. Kuvan oikeanpuoleisessa 2. skenaariossa kohde aloitti liikkeensä samasta pisteestä ja samalla nopeudella kuin ensimmäisessä skenaariossa. Kun simulaatiota oli kulunut $2.5s$ aloitti kohde nopeusvektorinsa kääntämisen ECEF-koordinaatiston z-akselin suuntaisesti $-40°$ sekunnissa ja lopetti nopeusvektorin kääntämisen kun $10s$ oli kulunut. Ohjus kohdistettiin jälleen kohteeseen ja se laukaistiin leveyspiiriltä 61.88°, pituuspiiriltä 25.12° sekä korkeudelta $6386.62m$.

Kuvan 7.2 vasemmanpuoleisessa 3. skenaariossa kohde aloitti jälleen liikkeen samasta pisteestä ja samalla nopeudella kuin edeltävissä skenaarioissa. Kun simulaatiota oli kulunut $0.5s$ aloitti kohde nopeusvektorinsa kääntämisen ECEF-koordinaatiston z-akselin suuntaisesti $-20°$ sekunnissa ja lopetti nopeusvektorin kääntämisen kun $5s$ oli kulunut. Kohde aloitti nopeusvektorin kääntämisen uudelleen ECEF-koordinaatiston z-akselin suuntaisesti 70° sekuntivauhdilla kun simulaatiota

oli kulunut 5.5s ja lopetti nopeusvektorin kääntämisen kun 7.8s oli kulunut. Ohjus laukaistiin leveyspiiriltä 61.90°, pituuspiiriltä 24.89° ja korkeudelta 5285m. Jälleen ohjus suunnattiin kohti kohdetta. Kuvan oikeanpuoleisessa 4. skenaariossa kohde aloitti liikkeen samoilla alkuarvoilla kuin edeltävissä skenaarioissa. Kun simulaatiota oli kulunut 0.5s aloitti kohde nopeusvektorinsa kääntämisen ECEF-koordinaatiston z-akselin suuntaisesti $-20°$ sekunnissa. Kun 4s oli kulunut, kohde vaihtoi kääntymisen suuntaa ja käänsi nopeusvektoria 40° sekuntivauhdilla saman akselin ympäri. Kun simulaatiota oli kulunut 6.0s lopetti kohde nopeusvektorin kääntämisen. Ohjus laukaistiin kohdetta kohti leveyspiiriltä 61.88°, pituuspiiriltä 25.12° sekä korkeudelta 6386.62m.



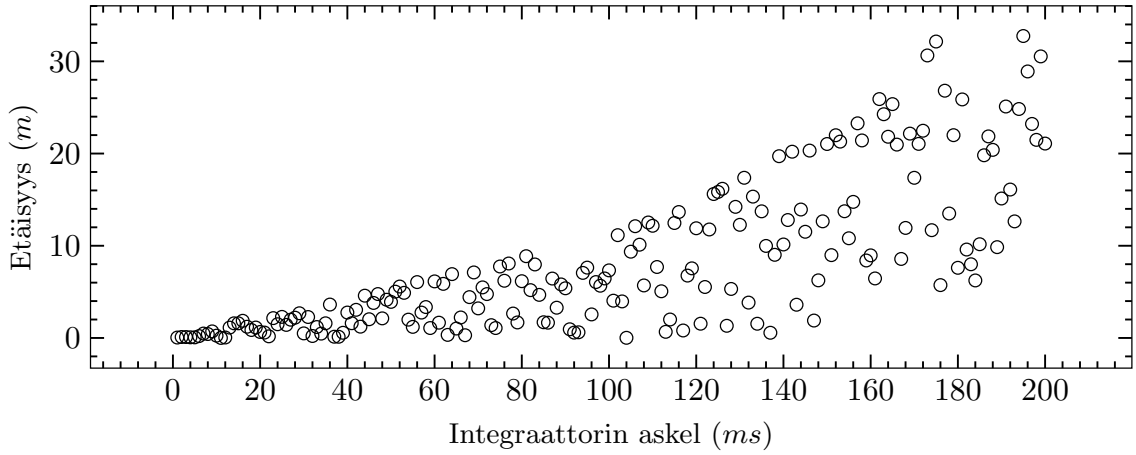
Kuva 7.1 *Mittauksissa käytetty ensimmäinen skenaario vasemmalla ja toinen skenaario oikealla.*



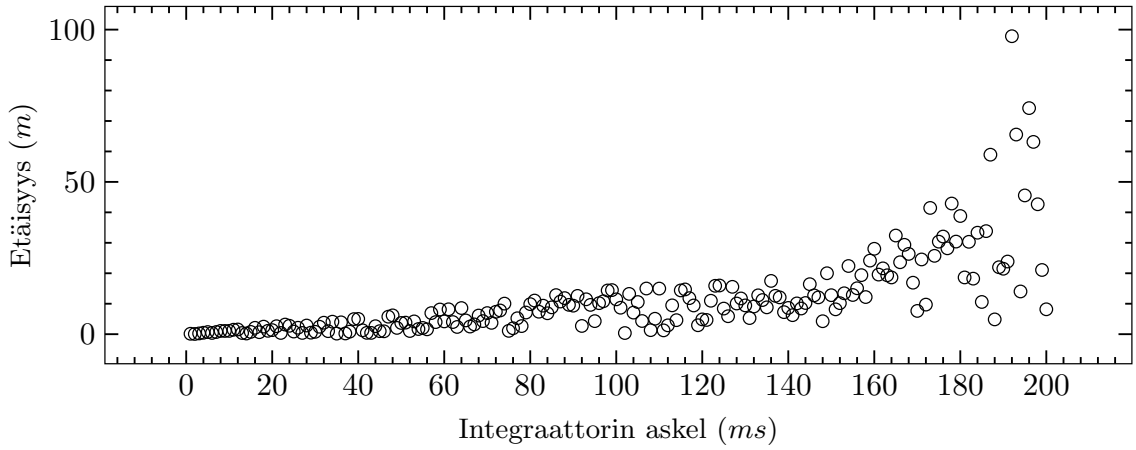
Kuva 7.2 *Mittauksissa käytetty kolmas skenaario vasemmalla ja neljäs skenaario oikealla.*

Ohjusmallin toiminnan mittauksessa hyödynnettiin kaikkia edellä mainittuja skenaarioita. Jäykän kappaleen simuloinnista vastanneen integraattorin integraatioaskelta muunneltiin ja pienimmät etäisyydet, joihin ohjus kohteestaan pääsi, otettiin

ylös. Kohteen sijaintia päivitettiin jokaisessa integraatioaskeleessa ennen varsinaisen hakeutumisen suorittamista. Saadut tulokset eri skenaarioille on esitettyinä kuvissa 7.3, 7.4, 7.5 ja 7.6.



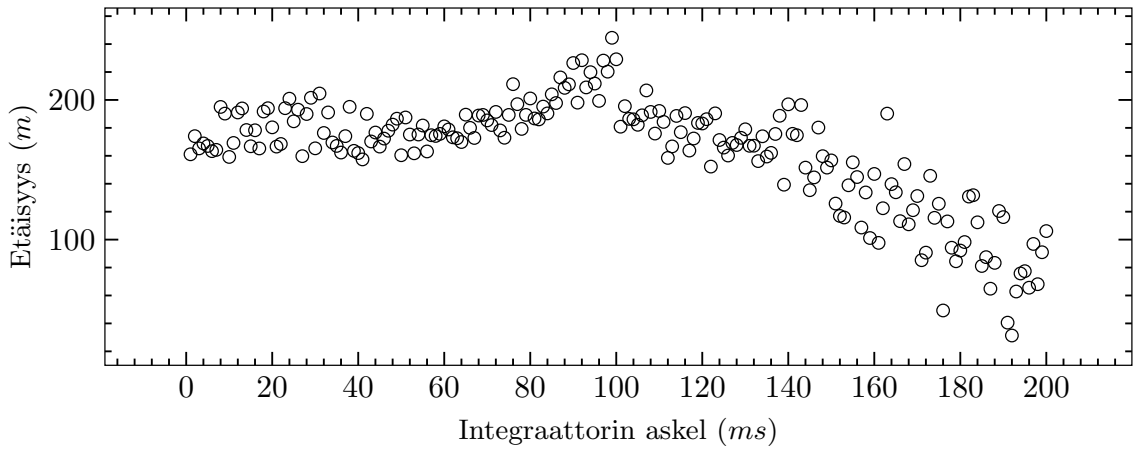
Kuva 7.3 *Ohjusmallin osumistarkkuus ensimmäisessä skenaariossa eriliaisilla integraatioaskeleilla.*



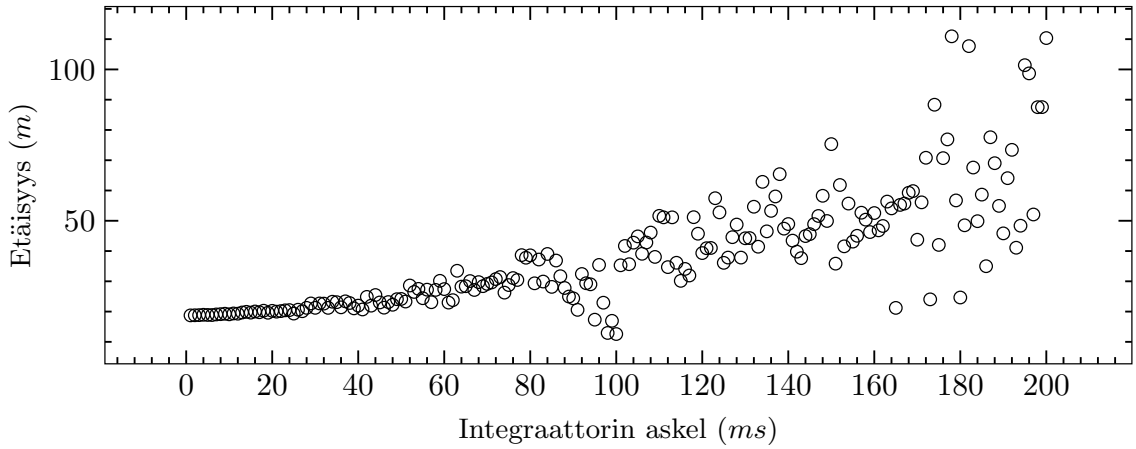
Kuva 7.4 *Ohjusmallin osumistarkkuus toisessa skenaariossa eriliaisilla integraatioaskeleilla.*

Ohjusmallin suorituskykyä mitattiin ajamalla simulaatiosilmukkaa 250ms välein. Integraattorin askelen suuruutta ja simuloitujen ohjusmallien lukumäärää muunneltiin ja kulunut aika otettiin ylös. Suorituskykymittausten tulokset on esitettynä kuvassa 7.7. Mallin suorituskykyä mitattiin tietokoneella, jossa oli 16Gt keskusmuistia ja prosessorina oli 3.7GHz kellotaajuudella toiminut i5-3470.

Ohjusmallia käyttävän simulaattorin toimintaa mitattiin myös osana kokonaista simulaatiota, jossa ohjussimulaattori vastaanotti kohteiden tilapäivitykset toisilta

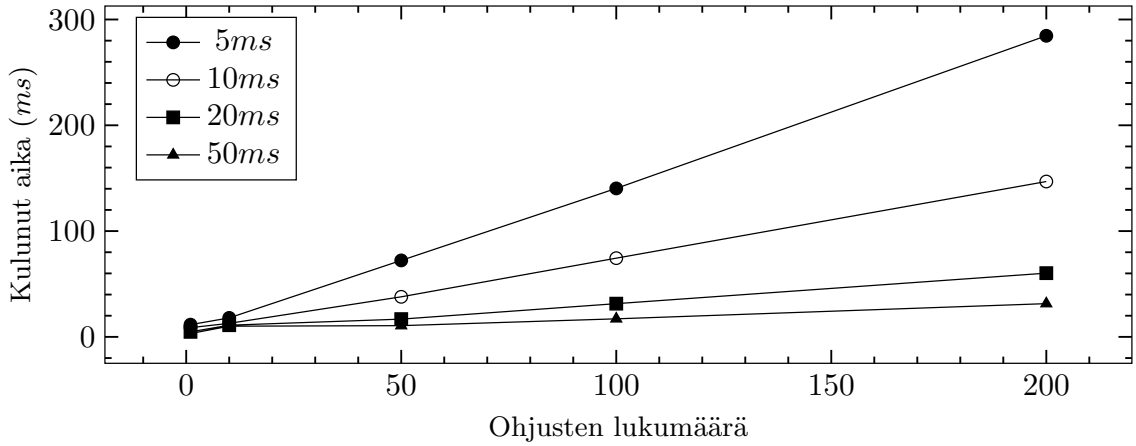


Kuva 7.5 *Ohjusmallin osumistarkkuus kolmannessa skenaariossa eriliaisilla integraatioaskeleilla.*

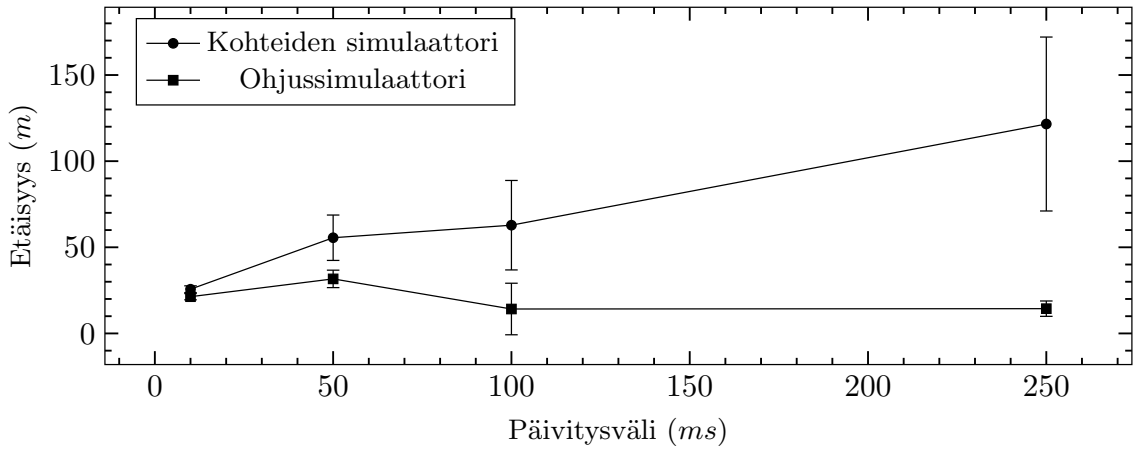


Kuva 7.6 *Ohjusmallin osumistarkkuus neljännessä skenaariossa eriliaisilla integraatioaskeleilla.*

simulaattoreilta. Toinen simulaattori päivitti kohteita ja julkaisi niiden tietoja tasaisin väliajoin. Kohteita päivittäneen simulaattorin päivitysaikoja muunneltiin mittauksen aikana, jonka lisäksi myös ohjussimulaattorin päivitys- ja julkaisuaikavälejä muunneltiin. Kaikki päivityksiin sekä julkaisuun liittyneet väliajat asetettiin aina samoiksi sekä ohjussimulaattorissa että kohteiden päivittäjässä. Asetetulla väliajalla suoritettiin 20 mittausajoa edellä mainitulla ensimmäisellä skenaariolla. Testeissä mitattiin sekä kohteita simuloivan, ja niiden tuhoutumisesta vastanneen, simulaattorin näkemää etäisyyttä ohjuksen räjähdyspisteeseen että ohjussimulaattorin näkemää etäisyyttä räjähdykseen. Saatujen etäisyyksien keskiarvot ja niiden keskihajonnat on esitettynä kuvassa 7.8. Tämän lisäksi ohjussimulaattorin ja kohteita päivittäneen simulaattorin näkemien räjähdysetäisyyksien erotuksien keskiarvot ja keskihajonnat on esitettynä kuvassa 7.9.



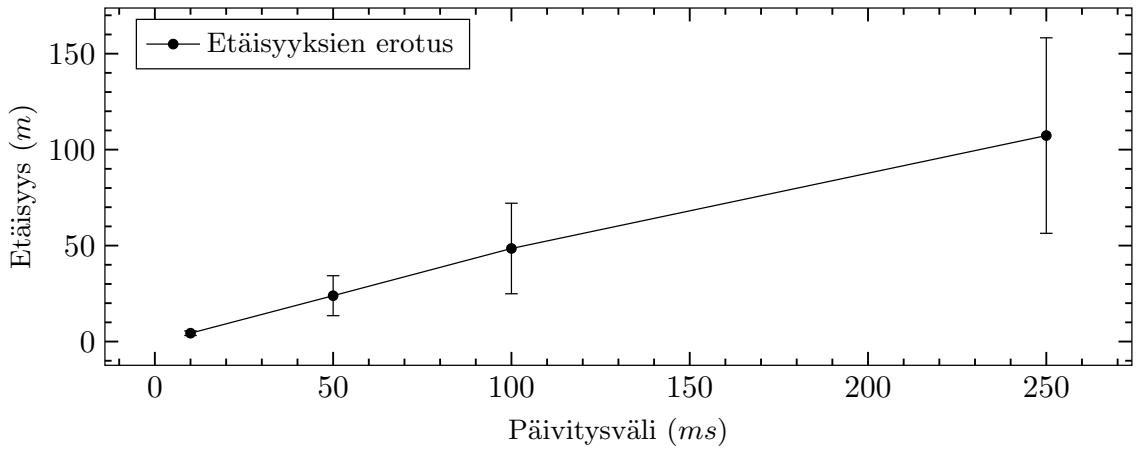
Kuva 7.7 *Suorituskykymittausten tulokset.*



Kuva 7.8 *Ohjussimulaattorin ja kohteita simuloivan simulaattorin näkemien etäisyyksien keskiarvot sekä keskihajonnat.*

7.2 Mittaustulosten ja toteutuksen arviointi

Ohjusmallille suoritetuista mittauksista voidaan havaita, että integraattorissa käytetyn integraatioaskeleen kasvaessa ohjus ei räjähdä yhtä tasaisesti jonkin tietyn etäisyyden päässä kohteestaan. Etenkin ensimmäisessä, toisessa ja neljännessä skenaariossa suuremmilla integraatioaskeleilla ohjuksen saavuttamaan etäisyyteen aiheutuu runsaasti vaihtelua. Kolmannessa skenaariossa ohjus näyttäisi pääsevän lähemmäksi kohdettaan mitä suuremmaksi integraatioaskel kasvaa. Todellisuudessa suuremmilla integraatioaskeleilla ohjus saavuttaa kohteen jo hieman ennen kuin se aloittaa väistöliikkeensä, jonka vuoksi ohjus osuu kohteeseen paremmin. Merkittävää kuitenkin on, että suuremmilla integraatioaskeleilla myös kolmannessa skenaariossa lähin saavutettu etäisyys heittelee enemmän kuin pienemmillä integraatioaskeleilla. Ohjusmalli näyttää saavan melko tasaisia tuloksia kaikissa skenaarioissa kun inte-



Kuva 7.9 *Ohjussimulaattorin ja kohteita simuloivan simulaattorin näkemien etäisyyksien erotuksen keskiarvo sekä keskihajonta.*

graatioaskel on $50ms$ tai sen alle.

Ohjusmallin suorituskyky vastaa hyvin aiemmin esitettyjä vaatimuksia mallin suorituskyvylle. Mittaustuloksista voidaan nähdä, että esimerkiksi $5ms$ integraatioaskeleella ja 100:lla ohjuksella simulaattori käyttää 250 millisekunnin simuloimiseen noin 140 millisekuntia aikaa. Tällöin muille simulaattorin toiminnoille jää vielä 110 millisekuntia aikaa suorittaa niiden tarvitsemat operaatiot. Kun ohjusmallilla simuloidaan 200 ohjusta $5ms$ integraatioaskeleilla, käyttää simulaattori mittausten mukaan ohjusten simulointiin aikaa $284ms$. Tällöin simulaattori käynnistäisi uuden simulointisilmukan välittömästi edellisen loputtua eikä se kykenisi pysymään simulaatiossa mukana vaan jätättäisi noin $30ms$ jokaisen simulaatiosilmukan aikana.

Vaikka simulaation osana ollut simulaattori näyttää mittaustuloksien perusteella osuvan omasta näkökulmastaan sitä tarkemmin mitä suuremmaksi päivitysvälit kasvavat, ei osumistarkkuus todellisuudessa muutu paremmaksi. Ohjussimulaattorin näkemä harha johtuu siitä, että kohteet pysyvät pidempään paikoillaan ennen kuin niiden seuraava tilapäivitys saapuu. Tällöin ohjussimulaattori hakeutuu paikallaan pysyvään kohteeseen, kunnes se jälleen hyppää toiseen paikkaan uuden tilapäivityksen johdosta. Muiden simulaattoreiden näkemä räjähdyksen etäisyys kasvaa mittaustulosten mukaan huomattavasti kun päivitys- ja julkaisuajat kasvavat. Kasvaneet ajat myös lisäävät räjähdysetäisyyksien hajontaa. Simulaation kannalta tärkeää olisi, että kaikki simulaatioon osallistuvat jäsenet näkisivät tapahtumat samankaltaisina, jotta ne voisivat muodostaa yhteisen kuvan simulaation tilasta. Kuvan 7.9 tuloksista voidaan nähdä, että simulaattoreiden päivitysväleistä, kuten myös muistakin operaatioista, aiheutuvat viiveet vaikuttavat merkittävästi siihen millaisena eri simulaattorit näkevät simulaatiossa esiintyvät tapahtumat.

Toteutusta arvioitiin myös loppukäyttäjän toimesta, jonka mielestä toteutus soveltui hyvin käyttökohteeseen. Loppukäyttäjän näkökulmasta mallinnuksessa pitää tulla huomioiduksi laukaisutilanteiden ja siihen liittyvien geometrioiden vaikutus, eri ohjustyyppien kyvykkyydet sekä lentoaikana tehdyt väistöliikkeet, jotta ohjustenvaihdon vaikutukset eri tilanteissa ovat oikeanlaiset. Loppukäyttäjien mukaan toteutettu ohjusmallinnus tuo paljon lisää käyttötapaan soveltuvaa realismia ja mahdollistaa aiempaa tarkemman ilmataistelumallinnuksen. Taulukossa 7.1 on kuvattuna kuinka johdannossa kuvatut loppukäyttäjän esittämät vaatimukset toteutuivat.

Taulukko 7.1 *Esitettyjen vaatimusten toteutuminen.*

1. *Ohjuksen liikemäärän tulee säilyä luonnollisen mukaisesti eikä ohjus saa tehdä liian jyrkkiä käännöksiä.*

   Kuuden vapausasteen liikeyhtälöt määrittävät ohjuksen liikkeen voimien ja momenttien seurauksena. Tällöin ohjuksen inertia pyrkii vastustamaan siihen kohdistuvia muutoksia eikä liian jyrkkiä muutoksia tapahdu. Ilmanvastus pyrkii hidastamaan ohjuksen liikettä sen tehdessä korjausliikkeitä.

2. *Ohjuksen tulee lentäessään suuntautua oikein.*

   Hakeutumislaki määrittää tarvittavan kiihtyvyyden, jonka ohjuksen ohjausjärjestelmä tuottaa kääntämällä ohjuksen kulmaan, jossa aerodynaamiset voimat tuottavat vaaditut kiihtyvyydet. Tällöin ohjus kääntyy automaattisesti oikeaan asentoon.

3. *Kohteen on voitava väistää ohjus, jos se tekee onnistuneen väistöliikkeen.*

   Aerodynaamiset voimat määrittävät ohjuksen kyvyn suorittaa korjausliikkeitä. Ilmanvastuksen hidastama ohjus ei välttämättä kykene tuottamaan tarpeeksi suuria aerodynaamisia voimia, jotta se saavuttaisi kohteensa.

4. *Mallin tulee kyetä simuloimaan noin 50–100 ohjusta reaaliaikaisesti.*

   Ohjusmalli kykeni simuloimaan 100 ohjusta reaaliaikaisesti.

5. *Mallin tulee olla modulaarinen ja helposti laajennettavissa.*

   Jäykkien kappaleiden simulointiin tarkoitetun laajennettavan mallin ja koordinaatistomuunnoksia abstrahoivan tilaolion avulla kappaleiden toimintaa on helppo muokata.

6. *Mallilla tulee voida simuloida sekä lyhyen matkan ohjuksia että keskipitkän matkan ohjuksia.*

   Ohjusmalliin toteutettiin konfiguraatiot ja tarvittavat komponentit sekä lyhyen matkan ohjukselle että keskipitkän matkan ohjukselle.

Kokonaisuudessaan toteutettu simulaattori vastasi hyvin sille asetettuja vaati-

muksia. Toteutuksen modulaarisuus ja esimerkiksi muunneltava integraatioaskel helpoittavat simulaation tarkkuuden sekä tehokkuuden muuntelua tarpeen vaatiessa. Modulaarisuuden vuoksi jäykän kappaleen mallinnuksesta saatiin toteutettua yleiskäyttöinen, jota voidaan hyödyntää jatkossa myös muiden simulaattoreiden toteutuksessa.

7.3 Jatkokehitysideoita

Toteutettua ohjusmallia ja sitä hyödyntävää simulaattoria voisi jatkokehittää usealla tavalla. Kehitystyö voisi kohdistua oikean ohjuksen toimintaan tehtyihin rajoituksiin ja ideaalisiksi oletettuihin komponentteihin. Esimerkiksi ohjuksen hakupäässä voisi huomioida hakupään kääntymisnopeuden sekä hakupään maksimikulmat, joista se voi havaita kohteensa. Hakupää voisi myös huomioida kohteesta lähtevän säteilyn tyypin sekä mahdolliset kohteen tuottamat häirinnät. Myös esimerkiksi ohjusmallissa käytetyn säätimen voisi vaihtaa johonkin toiseen säätimeen ja ohjausjärjestelmä voisi lisäksi huomioida eri komponenteissa ilmeneviä viiveitä. Näiden kehitysten lisäksi ohjusmalliin voisi myös toteuttaa muita hakeutumislakeja suhteellisen navigoinnin lisäksi.

Ohjusten dynamiikassa merkittävin jatkokehityksen kohde olisi ohjuksen moottorin käyttämän polttoaineen vaikutus ohjuksen massaan. Työn mallissa massa oletettiin vakioksi, mutta pienemmällä massalla ohjuksesta tulisi ketterämpi lennon loppuvaiheessa. Myös ohjusten inertiamatriisia voisi tarkentaa huomioimaan esimerkiksi ohjuksen siivekkeet. Ohjusten aerodynamiikkaa voisi jatkokehittää tuottamalla aerodynaamiset kertoimet ohjelmistolla, joka tukisi takasiivekkeiden lisäksi myös ohjuksen etusiivekkeitä. Myös ohjausiivekkeiden aiheuttamien ohjausvoimien riippuvuutta ohjuksen nopeudesta voisi ottaa toteutuksessa paremmin huomioon. Työn ohjusmallissa myös jätettiin aerodynaamiset momentit huomioimatta, jotka kuitenkin voisi jatkokehittäessä huomioida. Aerodynaamiset momentit tulisivat tarpeen jos ohjusmallilla haluttaisiin toteuttaa esimerkiksi liitäviä kappaleita, jotka kääntyvät itsestään ilmasta aiheutuvien voimien avulla, eivätkä hyödyntäisi kääntymisessä ohjausjärjestelmän tuottamia momentteja.

Toteutettua simulaattoria voisi jatkokehittää toteuttamalla eri ohjustyypeille parammat konfigurointimahdollisuudet. Työssä toteutetussa simulaattorissa konfigurointi hoidettiin erilaisia ohjuskonfiguraatioita sisältävän luokan avulla, mutta parempi menetelmä olisi esimerkiksi XML-muotoinen kuvaus, joka määrittelisi ohjusmallissa simuloitavat ohjukset.

Ohjusten tarkkuutta voisi parantaa pienentämällä ohjussimulaattorin ja muiden

simulaatioon osallistuvien simulaattoreien pävitys- ja julkaisuaikavälejä. Tämä kuitenkin kasvattaisi viestinvälitykseen kohdistuvaa kuormaa. Toinen menetelmä ohjusten tarkkuuden parantamiseen olisivat *Dead Reckoning* -algoritmit, joiden avulla ohjukset voisivat ennakoida kohteen sijaintia. Dead Reckoning -algoritmit ovat yhtälöitä, jotka ennakoivat kohteen tilaa, kuten sijaintia tai asentoa, kohteen tietojen, kuten nopeuden tai kulmanopeuden, avulla. Kaikki simulaation simulaattorit hyödyntäisivät samoja algoritmejä kohteiden tilojen ennakoinnissa, jolloin kaikkien simulaattoreiden näkemä kuva simulaation tilasta yhtenäistyisi (Fujimoto 2000, s. 204–209). Tällöin myös ohjusten osumatarkkuus paranisi ja simulaattoreiden päivitysvälejä ei tarvitsisi pienentää yhtä paljon kuin ilman Dead Reckoning -algoritmejä.

Ohjussimulaattoria ei myöskään erityisemmin optimoitu työtä tehdessä, sillä sen toiminta ei aiheuttanut suorituskykyongelmia. Simulaattorin suorituskykyä voisi kuitenkin todennäköisesti parantaa esimerkiksi kierrättämällä laskuissa käytettyjä olioita ja välttämällä turhien `Rotation`-olioiden luontia. Myös esimerkiksi ohjusten simuloinnin siirtäminen useamman säikeen vastuulle nopeuttaisi simulaation suoritusta.

8. YHTEENVETO

Simulaatiot ovat hyödyllisiä työkaluja sotilashenkilöstön koulutuksessa ja niiden avulla voidaan korvata osia sotaharjoituksista. Simulaatioiden avulla voidaan esimerkiksi vähentää harjoituksissa käytettyjen ammusten määrää, jolloin harjoitusten kustannuksia saadaan madallettua. Tässä työssä suunniteltiin ja toteutettiin ilmatorjunta- ja ilmataisteluohjuksia simuloiva simulaattori, joka toimi osana useamman koneen vastuulle jaettua, sotilashenkilöstön koulutuksessa hyödynnettyä hajautettua simulaatiota.

Työn alussa tarkasteltiin ohjuksen toimintaa sekä hajautettujen simulaatioiden ominaisuuksia, jonka lisäksi esiteltiin nykyisiä ratkaisuja hajautettujen simulaatioiden toteutukseen. Esiteltyjen tietojen avulla muodostettiin yksinkertainen kuvaus oikean ohjuksen toiminnasta, jota hyödynnettiin ohjuksen noudattamien kuuden vapausasteen liikeyhtälöiden sekä voimayhtälöiden muodostamisessa. Muodostettujen yhtälöiden ja yksinkertaistetun ohjuskuvauksen avulla toteutettiin ohjusmalli, jota työssä toteutettu ohjussimulaattori hyödynsi toiminnassaan.

Työssä oli tarkoituksena toteuttaa ohjusmallinnus tarkkuudella, joka soveltuisi reaaliaikaiseen simulointiin. Karkean mallinnuksen suorittavan simulaattorin toteuttaminen olisi ollut yksinkertaista, mutta sen tuottamien ohjusten realistisuus olisi jäänyt vähäiseksi. Erittäin tarkasti ohjuksen toimintaa mallintava simulaattori olisi tuottanut hyvin tarkan ohjuskuvauksen, mutta sen vaatima suoritusteho olisi estänyt mallinnuksen käytön reaaliaikaisessa simulaatiossa. Toteutuksessa tuli saavuttaa sopiva tasapaino mallinnuksen realistisuuden sekä sen vaatiman suoritustehon välillä. Tässä työssä onnistuttiin saavuttamaan sopivan realistinen kuvaus ohjuksen toiminnasta, joka soveltui erittäin hyvin sille tarkoitettuun käyttökohteeseen. Toteutettu ohjussimulaattori otettiin käyttöön osaksi loppukäyttäjän käyttämää tuotetta, jossa se korvasi aiemman ohjuksien mallinnuksesta vastanneen toteutuksen.

Toteutettu ohjusmalli oli myös suorituskyvyltään parempi, kuin mitä johdannossa esitetyissä vaatimuksissa kuvattiin. Mallin toteutuksessa kuitenkin tehtiin useita oletuksia, jonka vuoksi mallinnettujen ohjusten ominaisuuksia voisi edelleen tarkentaa. Esimerkiksi ohjuksen hakupään rajoitteet ja hakeutumisen kohteena olevan säteilyn huomioiminen jäivät toteutetusta ohjusmallista pois, jonka lisäksi ohjaus- ja

hakeutumisjärjestelmien toiminta oletettiin hyvin yksinkertaiseksi. Jos ohjussimulaattorin käyttötarkoitus tulevaisuudessa muuttuu, voi uusien, tarkemmin oikean ohjuksen toimintaa huomioivien komponenttien toteuttaminen tulla aiheelliseksi.

Simulaattoreiden käyttö sotilashenkilöstön koulutuksessa tulee todennäköisesti myös tulevaisuudessa pysymään merkittävässä roolissa. Työssä toteutettu ohjusmalli sekä sen osana toiminut jäykkien kappaleiden mallinnus antavatkin hyvän pohjan tuleville simulaattoreille jos kasvavat simulaatiotarpeet vaativat esimerkiksi tarkempaa lentokoneiden mallinnusta.

LÄHTEET

Boer, C. A. 2005. *Distributed Simulation in Industry*. Rotterdam, Erasmus University Rotterdam. 287 s.

Fujimoto, R. M. 2000. *Parallel and distributed simulation systems*. New York, Wiley Interscience. 300 s.

Gosling, J., Joy, B., Steele, G. & Bracha, G. 2013. *The Java Language Specification, Java SE 7 Edition*. Addison-Wesley Professional. 662 s.

Guelman, M., Idan, M. & Golan, O. 1995. Three-dimensional minimum energy guidance. *IEEE Transactions on Aerospace and Electronic Systems*. Vol. 31(2). pp. 835–841.

IEEE Std 1516-2010. 2010. IEEE Standard for Modeling and Simulation (M and S) High Level Architecture (HLA) - Framework and Rules. *IEEE Std 1516-2010 (Revision of IEEE Std 1516-2000)*. 26 s.

IEEE Std 1516.1-2000. 2001. IEEE Standard for Modeling and Simulation (M and S) High Level Architecture (HLA) - Federate Interface Specification. *IEEE Std 1516.1-2000*. 474 s.

IEEE Std 1516.2-2000. 2001. IEEE Standard for Modeling and Simulation (M and S) High Level Architecture (HLA) - Object Model Template (OMT) Specification. *IEEE Std 1516.2-2000*. 135 s.

Jia, W. & Zhou, W. 2005. *Distributed Network Systems*. Vol. 15 of *Network Theory and Applications*. Boston, Springer US. 531 s.

Johnson, R. et al. 2015. Spring Framework Reference Documentation. Verkkosivu. [viitattu 05.10.2015] Saatavissa: `http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/`.

Koskimies, K. & Mikkonen, T. 2005. *Ohjelmistoarkkitehtuurit*. Talentum Media Oy. 250 s.

MASA Group. 2015. MASA SWORD is a complete wargame solution with automated forces for high-level training and analysis.. Verkkosivu. [viitattu 17.10.2015] Saatavissa: `http://masa-group.biz/products/sword/`.

Möller, B., Dubois, A., Leydour, P. L. & Verhage, R. 2014. RPR FOM 2.0: A Federation Object Model for Defense Simulations. Proceedings of 2014 Fall Simulation Interoperability Workshop, 14F-SIW-039. Simulation Interoperability Standards Organization. Orlando, Florida, USA, September 8–12, 2014. New York, Curran Associates, Inc. pp. 233–248.

NOAA, NASA & USAF. 1976. *U.S. Standard Atmosphere.* Washington D.C., U.S. Governement Printing Office. 243 s.

Noureldin, A., Karamat, T. B. & Georgy, J. 2013. *Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration.* Springer-Verlag Berlin Heidelberg. 324 s.

Olson, D. 1996. Converting Earth-centered, Earth-fixed coordinates to geodetic coordinates. *IEEE Transactions on Aerospace and Electronic Systems.* Vol. 32(1), pp. 473–476.

OSGi Alliance. 2014. OSGi Core Release 6 Specification. Verkkosivu. [viitattu 05.10.2015] Saatavissa: `https://osgi.org/download/r6/osgi.core-6.0.0.pdf`.

Rogers, C. E. & Cooper, D. 2011. Rogers Aeroscience RASAero Aerodynamic Analysis and Flight Simulation Program, User Manual Version 1.0.2.0. Verkkosivu. [viitattu 11.10.2015] Saatavissa: `http://rasaero.com/dloads/RASAero_Users_Manual.pdf`.

Ross, P. 2012. Comparison of High Level Architecture Run-Time Infrastructure Wire Protocols—Part One. SimTecT 2012 Conference Proceedings. Adelaide Convention Centre, Adelaide, Australia, June 18–21, 2012.

Schaub, H. & Junkins, J. L. 2003. *Analytical Mechanics of Space Systems.* Reston, Virginia, USA, American Institute of Aeronautics and Astronautics, Inc. 818 s.

Siouris, G. M. 2004. *Missile guidance and control systems.* New York, USA, Springer-Verlag New York, Inc. 681 s.

Stevens, B. L. & Lewis, F. L. 1992. *Aircraft control and simulation.* John Wiley and Sons, Inc. 640 s.

Strickland, J. S. 2011. *Missile flight simulation.* Lulu, Inc. 534 s.

Sun Microsystems, Inc. 2002. Java Message Service Specification, Version: 1.1. Verkkosivu. [viitattu 25.10.2015] Saatavissa: http://www.oracle.com/technetwork/java/docs-136352.html.

The Apache Software Foundation. 2015. Apache Commons Math. Verkkosivu. [viitattu 03.09.2015] Saatavissa: http://commons.apache.org/proper/commons-math/.

Tolk, A. 2012. *Engineering Principles of Combat Modeling and Distributed Simulation*. Hoboken, New Jersey, USA, John Wiley and Sons, Inc. 932 s.

US Army Missile Command. 1995. *Missile Flight Simulation, Part One, Surface-to-Air Missiles*. US Army Missile Command. Redstone Arsenal. MIL-HDBK-1211. 254 s.

US DoD. 2014. Program Acquisition Cost by Weapon System. Verkkosivu. [viitattu 17.10.2015] Saatavissa: http://comptroller.defense.gov/Portals/45/documents/defbudget/fy2015/fy2015_Weapons.pdf.

VT MÄK. 2015. VR-Forces: Computer Generated Forces and Simulator Development. Verkkosivu. [viitattu 17.10.2015] Saatavissa: http://www.mak.com/products/simulate/vr-forces/.

Wikimedia. 2009. AIM-120 Advanced Medium-Range Air-to-Air Missile. Verkkosivu. [viitattu 04.09.2015] Saatavissa: https://commons.wikimedia.org/wiki/User:Rocket000/SVGs/Rockets#/media/File:AIM-120A_AMRAAM_scheme.svg. Käyttäjältä: F l a n k e r.

Wittenburg, J. 2008. *Dynamics of Multibody Systems*. 2 edn. Springer-Verlag Berlin Heidelberg. 232 s.

Zipfel, P. H. 2007. *Modeling and Simulation of Aerospace Vehicle Dynamics*. 2 edn. Reston, Virginia, USA, American Institute of Aeronautics and Astronautics, Inc. 607 s.