Tampereen yliopisto

Veini Lehkonen

NOPEUSPROFIILIN GENEROINTI LIIKKUVALLE ROBOTILLE

Informaatioteknologian ja viestinnän tiedekunta
Diplomityö
Toukokuu 2021

TIIVISTELMÄ

Veini Lehkonen: Nopeusprofiilin generointi liikkuvalle robotille
Diplomityö
Tampereen yliopisto
Sulautetut järjestelmät
Toukokuu 2021

Nopeusprofiili on kuvaus esineen nopeudesta suhteessa aikaan tai esineen tekemään matkaan. Automaattisesti liikkuva robotti voi käyttää nopeusprofiilia nopeutensa asetusarvoina ja ohjata näin omaa liikettään. Tässä työssä esitellään algoritmi, joka laatii nopeusprofiilin reitin nopeusrajojen ja robotin kiihtyvyys- ja nykäysrajoitusten perusteella. Nopeusprofiilin tuottava algoritmi eli nopeusprofiiligeneraattori on osa liikkuvan robotin laajempaa automaatio-ohjelmistoa. Ohjelmiston nykyinen generaattori on monimutkainen ja siitä on löydetty hankalasti jäljitettäviä virheitä. Se halutaan korvata uudella toteutuksella, joka on helpompi ymmärtää ja ylläpitää.

Nopeusprofiilin suunnitteluun tarvitaan tietoja reitistä ja robotin ominaisuuksista. Reitin ominaisuudet voidaan yksinkertaistaa nopeusrajoiksi, joita robotin tulee noudattaa. Robotin ominaisuudet voidaan yksinkertaistaa kiihtyvyys- ja nykäysrajoituksiksi. Esimerkiksi mitä raskaampi robotti on, sitä pidempi aika siltä kuluu kiihdyttämiseen tai hidastamiseen eli nopeutensa muuttamiseen, jos sen moottorin teho on sama. Nykäysrajoitukset rajoittavat kiihtyvyyden muuttumista; jos nykäystä ei rajoiteta, robotin liikkuvat osat voivat iskeytyä toisiaan vasten ja aiheuttaa kulumista.

Automaatio-ohjelmiston nykyinen nopeusprofiiligeneraattori tuottaa ja esittää nopeusprofiilin analyyttisesti eli laskukaavoina. Profiilin voi esittää myös näytejonona, ja sen voi tuottaa yksi näyte kerrallaan. Tässä työssä esitellään erilaisia nopeusprofiilin tuottavia algoritmeja, joita arvioidaan niiden ominaisuuksien perusteella. Yksi niistä valitaan toteutettavaksi ja analysoitavaksi.

Uusi nopeusprofiiligeneraattori kirjoitetaan Haskellilla ja se esittää nopeusprofiilin näytejonona. Uusi algoritmi valitaan pääasiassa sen yksinkertaisuuden vuoksi: se tuottaa profiilin nostamalla näytteiden nopeuksia vähitellen, kunnes ne ovat mahdollisimman suuria ja noudattavat kaikkia rajoituksia. Profiili laaditaan siis iteroimalla: Alussa se tekee profiilin, jonka kaikki nopeudet ovat nollia. Sitten se käy profiilin näyte kerrallaan läpi monta kertaa ja yrittää nostaa nopeuksia niin, että ne vielä noudattavat nopeusrajoja ja kiihtyvyys- ja nykäysrajoituksia. Koska profiilia muokataan aikatasossa, nopeusrajat siirtyvät, kun profiilin nopeus muuttuu. Näin ollen näytteiden nopeuksia joudutaan joskus myös vähentämään.

Uutta nopeusprofiiligeneraattoria testataan erilaisilla rajoituksilla. Sen tuottamat profiilit esitetään kuvaajissa, joista sen toimintaa voi arvioida. Algoritmi tuottaa suhteellisen järkevän näköisiä profiileja. Se nostaa yksittäisen näytteen nopeuden niin suureksi kuin mahdollista, vaikka seuraavien näytteiden nopeuksia joudutaan vähentämään. Tämä toimintaperiaate aiheuttaa profiiliin aaltoilua, joka loppuu, kun algoritmin annetaan vain joko nostaa tai vähentää nopeuksia samalla kierroksella. Nopeusrajojen siirtyminen voi aiheuttaa tilanteen, jossa jokin näyte haluaa nostaa nopeuttaan, vaikka sen pitäisi vähentää sitä, koska osa sen viereisistä näytteistä rikkoo omia nopeusrajojaan. Kun algoritmin annetaan vähentää nopeuksia, vaikka se aiheuttaisikin lievää rajoitusten rikkomista, ongelma korjaantuu. Rikkomukset profiilissa korjaantuvat myöhemmillä iteraatioilla. Nopeusrajojen siirtymisen takia varsinkin reitin loppupuolen nopeudet ovat pitkään jatkuvassa muutoksessa. Ongelma on lievempi, jos reitillä on vähemmän nopeusrajoja.

Uusi nopeusprofiiligeneraattori pystyy tuottamaan käyttökelpoisia nopeusprofiileja, mutta se tarvitsee vielä vähän jatkokehitystä. Se pitää integroida laajempaan automaatio-ohjelmistoon, ja pitää varmistaa, että se kykenee saamaan profiilin valmiiksi. Profiilin valmistumiseen tarvittavien iteraatioden määrää olisi hyvä vähentää.

Avainsanat: nopeusprofiilin suunnittelu, nopeusprofiili, liikkuva robotti, robotin automaatio-ohjelmisto, nopeuden asetusarvot, nopeusprofiiligeneraattori, nopeusprofiilin tuottava algoritmi

ABSTRACT

Veini Lehkonen: Velocity Profile Generation for a Mobile Robot
Master of Science Thesis
Tampere University
Embedded Systems
May 2021

A velocity profile presents velocity of an object as a function of time or distance travelled by the object. A mobile robot can control its movement by using a velocity profile as its velocity setpoints. This thesis is about a new velocity profile generation algorithm that uses restrictions on acceleration and jerk and a velocity limit function to compute a velocity profile. The new velocity profile generator will replace the existing generator of robot control software. The existing generator is complicated and contains errors that are difficult to trace, so the goal is to make a simpler implementation that is easier to maintain.

The velocity profile generator needs information about the robot and its route to make a velocity profile. A velocity limit function is a simplification of the properties of a route. Acceleration and jerk restrictions are a simplification of the robot's properties. For example, if the output of its motor is the same, a heavier robot needs a longer time to accelerate or decelerate. Restricting jerk is important in order to reduce wear and impacts between any moving parts of the robot.

The robot control software's current velocity profile generator creates and presents velocity profiles analytically. A profile can also be made a sample at a time and comprise a sample sequence. This thesis introduces several different algorithms for velocity profile generation and compares them. One of them is chosen for implementation and analysis.

The new velocity profile generator is written with Haskell and presents velocity profiles as sample sequences. The new algorithm generates a velocity profile by simply iterating through the profile's samples and gradually increasing the velocities until they are as high as possible while still adhering to the restrictions on acceleration and jerk. At start, the generator makes a zero velocity profile. Then it goes through the profile one sample at a time multiple times and tries to increase the velocities without breaking any limitations. Since the profile is handled as a function of time, the places of the velocity limits change as the profile's velocities change. Therefore, the generator must also decrease the velocities of the samples sometimes.

Testing of the new velocity profile generator is conducted by giving it several different limitations. The velocity profiles it creates are presented as graphs from which its functionality can be evaluated. The algorithm produces mostly sensible profiles. It increases the velocity of any given sample as high as possible, even if later samples would need to decrease their velocities. This functionality causes oscillations in the profile. To remove them, the algorithm is modified so that on any given iteration, it will only either increase or decrease the velocities. The changing of the velocity limits can cause a sample to be stuck in a place where it wants to increase its velocity but instead it should further decrease its velocity because nearby samples are breaking their velocity limits. To remove this problem, the algorithm is modified to allow to decrease velocities even if doing so causes some limitations to be broken. Further iterations of the profile will fix the broken limitations. The changing of the velocity limits also causes the later end of the profile to constantly change. Having less velocity limits on the route mitigates this problem.

The new velocity profile generator can produce usable velocity profiles but it does need some further development. It needs to be integrated into the robot control software. It needs a verification method to check that it actually completes the wanted profile. It would be beneficial to reduce the amount of iterations needed to complete a profile.

Keywords: velocity planning, velocity profile, mobile robot, robot control software, velocity setpoints, velocity profile generation, velocity profile generation algorithm

ALKUSANAT

Tämä diplomityö on tehty Tampereen yliopistolle syksyn 2019 – kevään 2021 aikana.

Kiitän Atostek oy:tä diplomityöpaikasta ja työpaikasta, jossa sain tutustua suureen ja vaativaan ohjelmistoon ja osallistua sen kehittämiseen. Kiitän myös Tampereen yliopistoa (silloista Tampereen teknillistä yliopistoa) ikuisesta opinto-oikeudesta, jonka ansiosta sain tämän työn tehtyä. Lopuksi kiitän vielä vanhempiani, jotka tukivat ja auttoivat minua niin kauan kuin sitä tarvitsin.

Tampereella, 6. toukokuuta 2021

Veini Lehkonen

SISÄLLYSLUETTELO

1 Johdanto 1
2 Nopeusprofiilin suunnittelu 3
2.1 Nopeusprofiilin käyttötarkoitus 3
2.2 Reitti 4
2.3 Nopeuden rajoitukset 5
2.4 Nopeusprofiilin hienousasteet 6
2.4.1 Robotin fyysiset rajoitteet ja nykäys 8
2.4.2 Liian karkean nopeusprofiilin ongelmat 9
2.5 Nopeusprofiilin esitysmuotoja 13
2.6 Nopeusprofiilin generointimenetelmiä 13
2.6.1 Laskukaavat 13
2.6.2 Diskreetit pisteet 16
3 Nopeusprofiiligeneraattorin uudelleensuunnittelu 18
3.1 Korvattava nopeusprofiiligeneraattori 18
3.1.1 Ongelmat 19
3.2 Uusi nopeusprofiiligeneraattori 20
3.2.1 Polynomialgoritmi 22
3.2.2 Kuminauha-algoritmi 22
3.2.3 Osakäyräalgoritmi 24
3.2.4 Optimistinen kaksisuuntainen algoritmi 29
3.2.5 Ryömintäalgoritmi 31
3.2.6 Vakiokiihtyvyysalgoritmi 36
3.3 Valittu algoritmi 36
4 Uusi nopeusprofiiligeneraattori 37
4.1 Vaatimukset 37
4.2 Tekniset ratkaisut 38
4.2.1 Ohjelmointikieli 38
4.2.2 Rajapinta 38
4.2.3 Diskreetit askeleet 39
4.3 Toiminnallisuus 39
4.3.1 Esimerkki 39
4.4 Testaus 45
4.4.1 Yksinkertainen nopeusprofiili 46
4.4.2 Nopeampi yksinkertainen nopeusprofiili 47
4.4.3 Kuoppia nopeusprofiilissa 48
4.4.4 Matala nopeusraja 49

4.4.5 Pitkä nopeusprofiili satunnaisilla nopeusrajoilla 50
4.4.6 Iterointi 53
4.4.7 Testauksen tulokset 56
4.5 Muutokset algoritmiin 56
4.5.1 Lopputulokset 57
4.6 Jatkokehitys 61
4.6.1 Välttämättömät 62
4.6.2 Tärkeät 62
4.6.3 Hyödylliset 63
5 Yhteenveto 64
Lähteet 66

KUVALUETTELO

1.1 Esimerkki liikkuvasta robotista. 2

2.1 Esimerkki robotin reitistä. 4
2.2 Reitin geometria esitettynä xy-koordinaatistossa. 5
2.3 Esimerkki nopeusrajoista etenemän funktiona. 6
2.4 Esimerkki nopeusprofiilista, jossa kiihtyvyys on paloittain vakio. 7
2.5 Esimerkki nopeusprofiilista, jossa nykäys on paloittain vakio. 7
2.6 Esimerkki kuvan 2.5 nopeusprofiilista etenemän funktiona. 8
2.7 Yksinkertainen robotin malli. 9
2.8 Robotin ongelmia. 10
2.9 Ideaalinen nopeusprofiili. 10
2.10 Toteutunut nopeusprofiili. 11
2.11 Nopeusprofiilin yksinkertainen korjausyritys. 11
2.12 Korjausyritys aiheuttaa värähtelyä hidastuvuudessa. 12
2.13 Korjausyritys aiheuttaa värähtelyä nopeudessa. 12
2.14 Esimerkki oikean robotin liikkeestä. 13
2.15 Polynomit välillä $[t_0, t_3]$. 16
2.16 Pisteen laskeminen. 16

3.1 Nykyisen generaattorin etsimät polynomien leikkauskohdat. 19
3.2 Nykyisen generaattorin tuottamat polynomit. 19
3.3 Vuokaavio kuminauha-algoritmista. 23
3.4 Kuminauha-algoritmin toimintaperiaate. 24
3.5 Osakäyrät 1–7. 25
3.6 Vuokaavio osakäyräalgoritmista. 26
3.7 Nopeusprofiilin aloitus- ja lopetuspiste ja alin nopeusraja. 27
3.8 Alinta nopeusrajaa noudattavan nopeusprofiilin osakäyrät. 27
3.9 Seuraavaksi alimmat rajat ja pisteet, joiden välillä nopeutta voi vielä nostaa. 27
3.10 Seuraaviin nopeusrajoihin yltävät osakäyrät. 28
3.11 Aloitus- ja lopetuspiste, joiden välillä profiilia pitää korjata. 28
3.12 Kelvollinen nopeusprofiilin korjaus. 28
3.13 Vasemmalta oikealle laadittu optimistinen profiili. 30
3.14 Oikealta vasemmalle laadittu optimistinen profiili. 30
3.15 Optimististen profiilien yhdistelmä. 30
3.16 Tasoitettu optimistinen profiili. 31
3.17 Ryömintäalgoritmi osuu nopeusrajaan. 32
3.18 Ryömintäalgoritmi tasoittaa profiilin nopeusrajaan. 32

3.19 Ryömintäalgoritmi osuu seuraavaan nopeusrajaan. 32
3.20 Ryömintäalgoritmi muokkaa profiilia yltääkseen seuraavaan nopeusrajaan. 33
3.21 Ryömintäalgoritmi tasoittaa profiilin seuraavaan nopeusrajaan. 33
3.22 Ryömintäalgoritmi jatkaa ja osuu seuraavaan nopeusrajaan. 34
3.23 Ryömintäalgoritmi tasoittaa kiihtyvyyden muutoksen. 34
3.24 Ryömintäalgoritmi etenee reitin loppuun. 34
3.25 Ryömintäalgoritmi korjaa profiilia reitin lopussa. 35
3.26 Ryömintäalgoritmi tasoittaa profiilin reitin lopussa. 35

4.1 Lähtötilanne: nopeus on nolla. 40
4.2 Ensimmäinen iteraatio: ensimmäinen piste on kiinnitetty, muita voi nostaa. . 40
4.3 Toinen iteraatio: toinen piste on kiinnitetty, seuraavia voi nostaa. 40
4.4 Kolmas iteraatio: kolmannen pisteen nosto aiheuttaa nostojen ketjureaktion. 41
4.5 Nopeuden muutokset etenevät kasvavana mäkenä. 42
4.6 Aallon synty. 43
4.7 Nopeuden muutokset etenevät aaltoina. 44
4.8 Yksinkertainen nopeusprofiili. 46
4.9 Nopeampi yksinkertainen nopeusprofiili. 47
4.10 Kuoppia nopeusprofiilissa. 48
4.11 Nopeusprofiili, jolla on hetkellisesti hyvin matala nopeusraja. 49
4.12 Nopeusprofiili ei noudata matalaa nopeusrajaa välillä [5,4 s, 5,8 s]. 50
4.13 Pitkä reitti ja 10 nopeusrajaa. 51
4.14 Pitkä reitti ja 100 nopeusrajaa. 52
4.15 Iteraatio 41: viimeinen nopeusraja on saavutettu. 53
4.16 Iteraatio 55: profiilissa on aaltoja ja loppuosa rikkoo nopeusrajoja. 54
4.17 Iteraatio 90: myöhemmät aallot rikkovat nopeusrajoja. 54
4.18 Iteraatio 120: viimeisen nopeusrajan takana on yksi aalto. 55
4.19 Iteraatio 180: valmis nopeusprofiili. 55
4.20 Iteraatio 37: profiili saavuttaa reitin lopun. 57
4.21 Iteraatio 60: profiili on tasainen, loppupuolen nopeudet vähenevät. 58
4.22 Iteraatio 110: profiili alkaa asettua nopeusrajoihin. 58
4.23 Iteraatio 160: profiilin loppupuolen nopeudet jatkavat vähenemistä. 59
4.24 Iteraatio 200: profiili on lähes valmis. 59
4.25 Iteraatio 210: versio valmiista profiilista. 60
4.26 Iteraatio 214: toinen versio valmiista profiilista. 60
4.27 Nopeusprofiililla on nollasta poikkeava alku- ja loppunopeus. 61

LYHENTEET JA MERKINNÄT

a	kiihtyvyys (lat. acceleratio)
C++	Moniparadigmainen, vahvasti tyypitetty ja dynaamisen muistinhallinnan salliva ohjelmointikieli.
Google Protocol Buffers	Googlen kehittämä tapa sarjallistaa rakenteellista dataa (lyh. Protobuf).
Haskell	Funktionaalinen, vahvasti ja staattisesti tyypitetty ohjelmointikieli.
j	nykäys (engl. jerk)
s	pituus (lat. spatium)
SI-järjestelmä	Kansainvälinen mittayksikköjärjestelmä (ransk. Système international d'unités).
t	aika (lat. tempus)
v	nopeus (lat. velocitas)

1 JOHDANTO

Kun robotti liikkuu ennalta määrättyä *reittiä* pitkin, sille voidaan suunnitella etukäteen *nopeusprofiili*. Etukäteen suunniteltu nopeusprofiili kuvaa, millä nopeudella robotin tulisi kulkea missäkin kohdassa reittiä.

Tässä työssä suunnitellaan ja testataan algoritmi, joka tuottaa nopeusprofiilin, joka noudattaa reitin *nopeusrajoja* ja muita mahdollisia ennalta tiedettyjä rajoituksia. Nopeusprofiilia käytetään tasaisella pinnalla liikkuvan robotin nopeuden säätöarvoina. Nopeusprofiilin tuottavaa ohjelmamoduulia kutsutaan *nopeusprofiiligeneraattoriksi*.

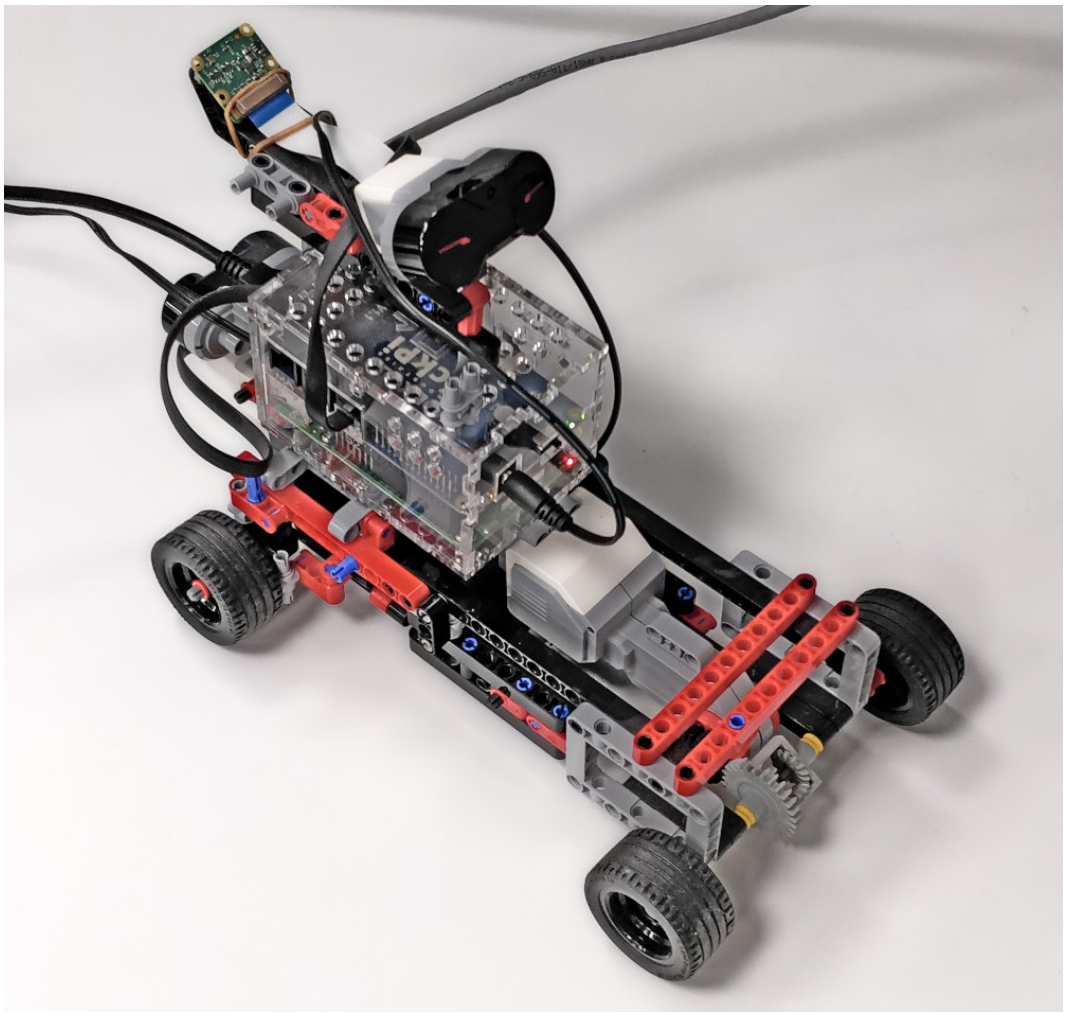
Nopeusprofiiligeneraattori on osa liikkuvan robotin laajempaa automaatio-ohjelmistoa. Automaatio-ohjelmiston nykyinen nopeusprofiiligeneraattori halutaan korvata uudella toteutuksella, joka on helpompi ymmärtää ja ylläpitää. Uusi toteutus on automaatio-ohjelmistosta erillinen kokonaisuus.

Tämän työn tekemisessä käytetään konstruktiivista menetelmää. Lähtökohtana eli ongelmana on nopeusprofiilin laatiminen. Pohjaymmärrys ongelman ratkaisemiseen saadaan olemassa olevasta nopeusprofiiligeneraattorista, tunnetuista robotin liikkumisominaisuuksista ja perinteisestä mekaniikasta. Seuraava askel on laatia ratkaisu. Tässä tapauksessa ratkaisun ensimmäinen vaihe on ideoida erilaisia nopeusprofiilin laadintamenetelmiä ja vertailla niitä. Toinen vaihe on valita niistä lupaavin toteutettavaksi ja toteuttaa se. Sitten ratkaisun toiminta käydään läpi testaamalla, ja sitä muokataan testauksen tulosten perusteella. Lopuksi tarkastellaan sen soveltuvuutta käyttötarkoitukseensa ja mitä jatkokehitystarpeita siinä on.

Luvussa 2 (Nopeusprofiilin suunnittelu, alkaen sivulta 3) käydään läpi, mihin nopeusprofiilia voi käyttää, mitä sen tekemiseen tarvitaan, millainen nopeusprofiili voi olla, miten sen voi esittää ja miten sen voi tuottaa.

Luku 3 (Nopeusprofiiligeneraattorin uudelleensuunnittelu, alkaen sivulta 18) esittelee korvattavan nopeusprofiiligeneraattorin ja sen ongelmat. Koska nopeusprofiiligeneraattori halutaan korvata uudella, seuraavaksi käydään läpi uusia nopeusprofiilin tuottavia algoritmiehdotuksia. Ehdotuksista valitaan yksi, joka toteutetaan.

Luvussa 4 (Uusi nopeusprofiiligeneraattori, alkaen sivulta 37) selvitetään uuden nopeusprofiiligeneraattorin vaatimukset ja tekniset ratkaisut. Sen jälkeen käydään läpi generaattorin toiminta ja testaus. Viimeiseksi selvitetään uuden nopeusprofiiligeneraattorin jatkokehitystarpeet.



Kuva 1.1. *Esimerkki liikkuvasta robotista.*

Kuvassa 1.1 on tuotekehityksessä käytettävä robotin pienoismalli, joka on esimerkki siitä, millainen liikkuva robotti voisi olla. Robotin perusosia ovat esimerkiksi pyörät, moottorit, jarrut, ohjaustietokone ja erilaiset ympäristöä ja robotin osien tilaa havainnoivat mittalaitteet. Robotti ohjaa liikettään kääntämällä pyöriään. Robotin ja sen osien perusominaisuuksia ovat esimerkiksi robotin massa, painopisteen korkeus, renkaiden ominaisuudet ja lukumäärä ja moottorien ja jarrujen teho.

2 NOPEUSPROFIILIN SUUNNITTELU

Nopeusprofiili [1] [2] (engl. velocity profile tai speed profile) on kuvaus robotin (tai minkä tahansa liikkuvan esineen) nopeudesta suhteessa aikaan tai robotin tekemään matkaan (eli *etenemään*) eli nopeus ajan tai etenemän funktiona. Se voi olla kuvaus jo toteutuneesta liikkeestä tai etukäteen tehty suunnitelma. Etukäteen suunniteltu nopeusprofiili määrää kullekin etenemälle siinä kohdassa tavoitellun nopeuden.

Tässä luvussa käydään läpi nopeusprofiiliin liittyviä termejä ja käsitteitä ja miksi ja miten se suunnitellaan.

2.1 Nopeusprofiilin käyttötarkoitus

Robotin käyttötarkoituksesta riippuu, voiko ja kannattaako nopeusprofiilia tehdä. Jotta nopeusprofiili voidaan suunnitella etukäteen, tarvitaan tietoa robotin reitistä ja reittiin ja robottiin kohdistuvista rajoituksista. Vaikka nopeusprofiili voitaisiinkin suunnitella etukäteen, se ei aina ole robotin liikkumisen kannalta välttämätön.

Toteutunut eli jälkikäteen mittausarvoista laadittu nopeusprofiili voi olla hyödyllinen robotin liikkeen analysoinnin kannalta. Jos nopeusprofiili on tehty myös etukäteen, voidaan verrata suunniteltua ja toteutunutta liikettä.

Tässä työssä etukäteen tehty nopeusprofiili on välttämätön, sillä sitä käytetään robotin liikkumisnopeuden säätämiseen. Robotin laajempi automaatio-ohjelmisto käyttää nopeusprofiilin nopeusarvoja asetusarvoina, jotka lähetetään eteenpäin moottoria ohjaavalle alijärjestelmälle. Nopeusprofiilissa on otettava huomioon robotin ja reitin rajoitukset niin tarkasti, että sen avulla voidaan selvittää halutulla tarkkuudella matkaan käytetty aika ja robotin tuleva paikka.

Jos jotain odottamatonta tapahtuu, nopeusprofiilia ei kuitenkaan käytetä, vaan robotin automaatio-ohjelmiston erillinen turvajärjestelmä reagoi. Yleensä robotti pysähtyy hätätilanteessa mahdollisimman nopeasti.

Jatkossa kaikkien erikseen määrittelemättömien nopeusprofiilien oletetaan olevan etukäteen suunniteltuja.

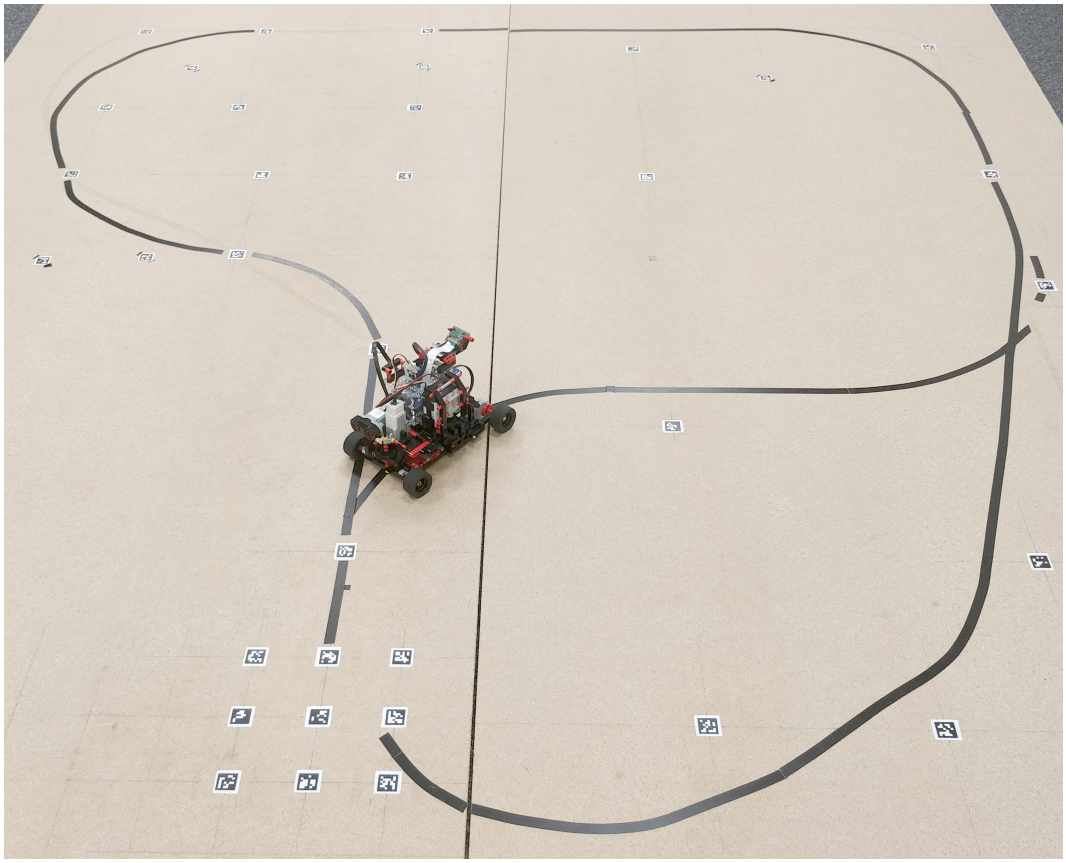
Nopeusprofiilin avulla voidaan optimoida robotin liikettä ja matkaan käytettyä aikaa, kun sitä käytetään nopeuden asetusarvoina. Jos nopeusprofiilia ei ole, mutta tiedetään, että reitissä on suoria osuuksia ja muutama tiukka käännös, robotin pysyväksi tai ainakin

reittikohtaiseksi kulkunopeudeksi on asetettava tarpeeksi alhainen nopeus, jotta se pysyy pystyssä ja reitillään käännöksissäkin ja ohjaus ehtii kääntää pyöriä käännösten mukaan. Tällöin robotin nopeus suorilla osuuksilla on pienempi kuin se voisi olla. Jos reitille tehdään nopeusprofiili, jossa nopeus on rajoitettu alhaiseksi vain käännösten kohdalla, robotti voi käyttää sitä asetusarvoina ja liikkua nopeammin suorilla osuuksilla.

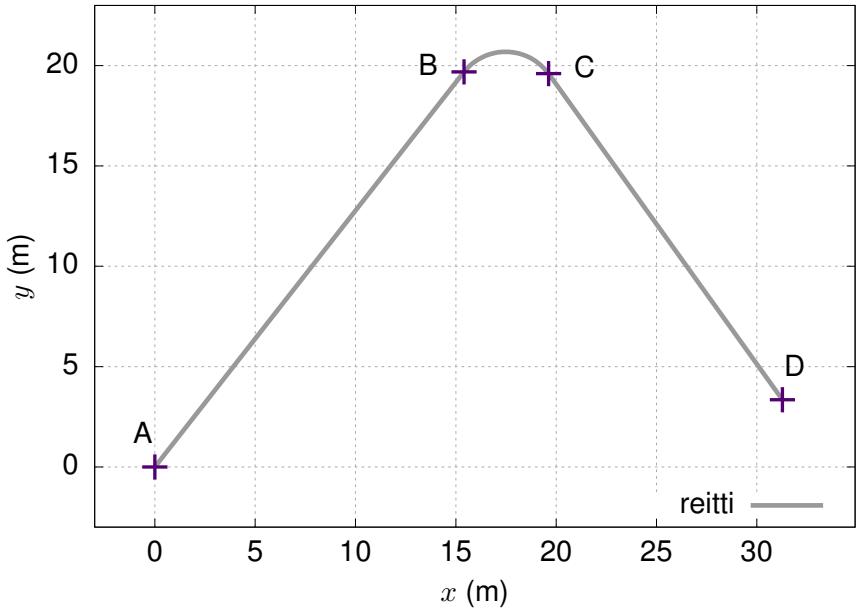
Jos nopeusprofiili noudattaa annettuja nopeusrajoja ja muita rajoituksia, sitä kutsutaan tässä työssä *lailliseksi*. Nopeusprofiilin laillisuus voidaan tarkistaa esimerkiksi erillisellä ohjelmalla tai funktiolla, joka vertaa tehdyn profiilin arvoja annettuihin rajoituksiin. Lisäksi profiilien laillisuutta ja järkevyyttä voi arvioida silmämääräisesti piirtämällä niistä kuvaajia. Ensimmäiset kuvat nopeusprofiileista löytyvät aliluvusta 2.4 Nopeusprofiilin hienousasteet, mm. kuva 2.5 (s. 7).

2.2 Reitti

Nopeusprofiilin laatimisen lähtökohtana on, että robotti liikkuu ennalta määrättyä reittiä pitkin. Reitin *geometria* on käyrä, jota robotin tulee seurata. Kuvassa 2.1 on esimerkki robotin reitistä, joka on merkitty alustaan mustalla teipillä. Reittiin kuuluu geometrian lisäksi muitakin osia. Etenemä kuvaa robotin sijaintia tietyssä reitin pisteessä, eli se on geometrian alkupisteestä käyrää pitkin mitattu etäisyys. Robotti voi tehdä käännöksiä ja kulkea suoraan. Reitin tai sen geometrian suunnittelu ei kuulu tämän työn piiriin.



Kuva 2.1. *Esimerkki robotin reitistä.*



Kuva 2.2. *Reitin geometria esitettynä xy-koordinaatistossa.*

Kuva 2.2 esittää yksinkertaista reitin geometriaa xy-koordinaatistossa.

- Robotti lähtee liikkeelle pisteestä A (0,0 m, 0,0 m), jossa sen etenemä on 0,0 m.
- Robotti etenee 25 m suoraan pisteeseen B.
- Robotti etenee 5,0 m kääntyessään oikealle pisteeseen C.
- Robotti etenee 20 m suoraan pisteeseen D.
- Robotti pysähtyy pisteeseen D, jolloin sen etenemä on 50 m.

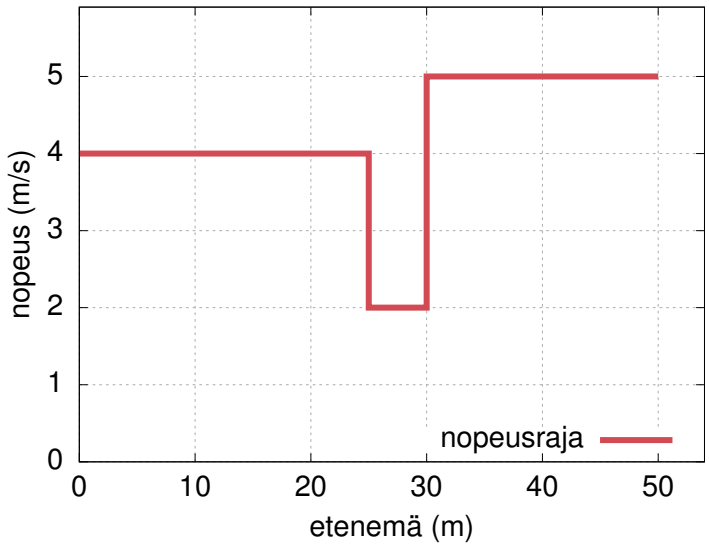
Alusta, jolla robotti liikkuu, on tasainen, eli korkeudessa ei tapahdu muutoksia. Tällöin voidaan käyttää 2-ulotteista geometriaa.

2.3 Nopeuden rajoitukset

Reitille on määritelty ennalta nopeusrajat, joita robotin tulee noudattaa. Rajoitukset voivat perustua ainakin kolmeen eri asiaan: liikkumisalueen ominaisuuksiin, reitin ominaisuuksiin ja robotin ominaisuuksiin.

Liikkumisalueen ominaisuuksia ovat esimerkiksi alustan tasaisuus ja ympäristön muut olosuhteet. Reitin ominaisuuksia ovat suorat ja käännökset. Robotin ominaisuuksia ovat muun muassa sen mitat, massa ja moottorin ja jarrujen teho.

Robotin laajempi automaatio-ohjelmisto käsittelee liikkumisalueen ja reitin ominaisuuksia ja tuottaa niiden perusteella *nopeusrajafunktion*. Nopeusrajafunktio esittää nopeusrajan kunkin etenemän kohdalla. Kuvassa 2.3 on esimerkki nopeusrajoista, joita voitaisiin käyttää kuvan 2.2 reitillä.



Kuva 2.3. *Esimerkki nopeusrajoista etenemän funktiona.*

- Ensimmäisen 25 m suoran aikana nopeusraja on 4,0 m/s.
- 5,0 m käännöksen aikana nopeusraja on 2,0 m/s.
- Viimeisen 20 m suoran aikana nopeusraja on 5,0 m/s.

Ympäristön olosuhteet voidaan ottaa huomioon esimerkiksi siten, että jäisellä alustalla tai tuulisella säällä nopeusrajat ovat alemmat. Käännöksissä robotin on liikuttava hitaammin kuin suorilla osuuksilla, jotta se ei kaadu tai joudu luisuun.

Robotin ominaisuuksista aiheutuvia rajoituksia käsitellään tarkemmin aliluvussa 2.4.1 (s. 8).

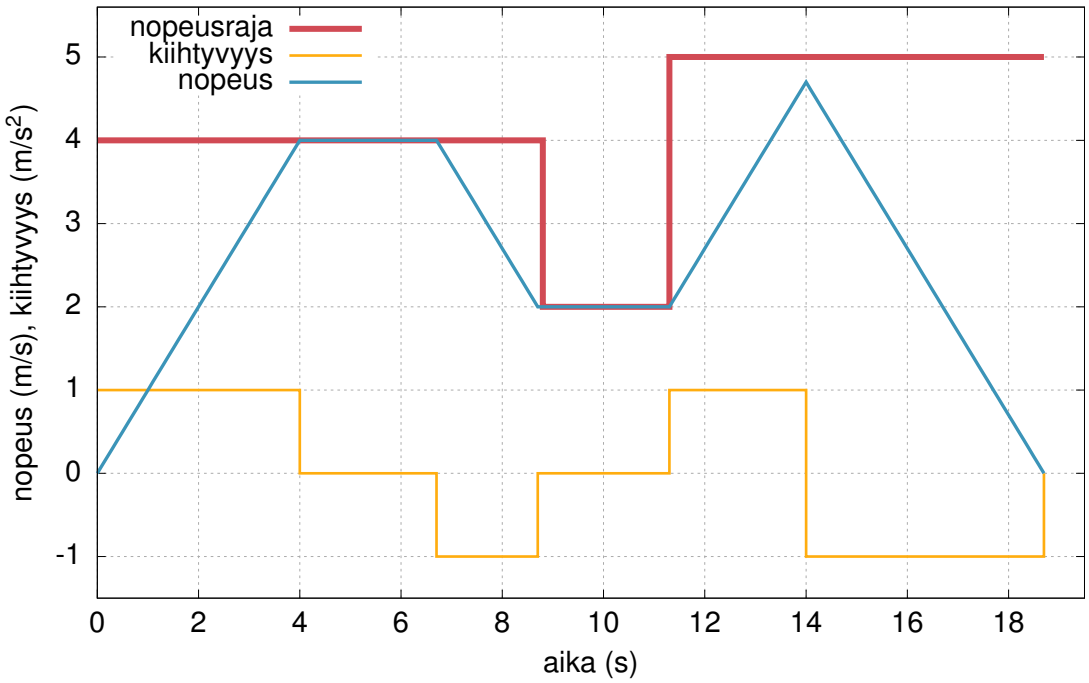
2.4 Nopeusprofiilin hienousasteet

Laitteistosta, ympäristöstä ja rajoista riippuen robotti saattaa liikkua tarpeeksi hyvin, vaikka nopeusprofiili olisi sama kuin nopeusrajafunktio.

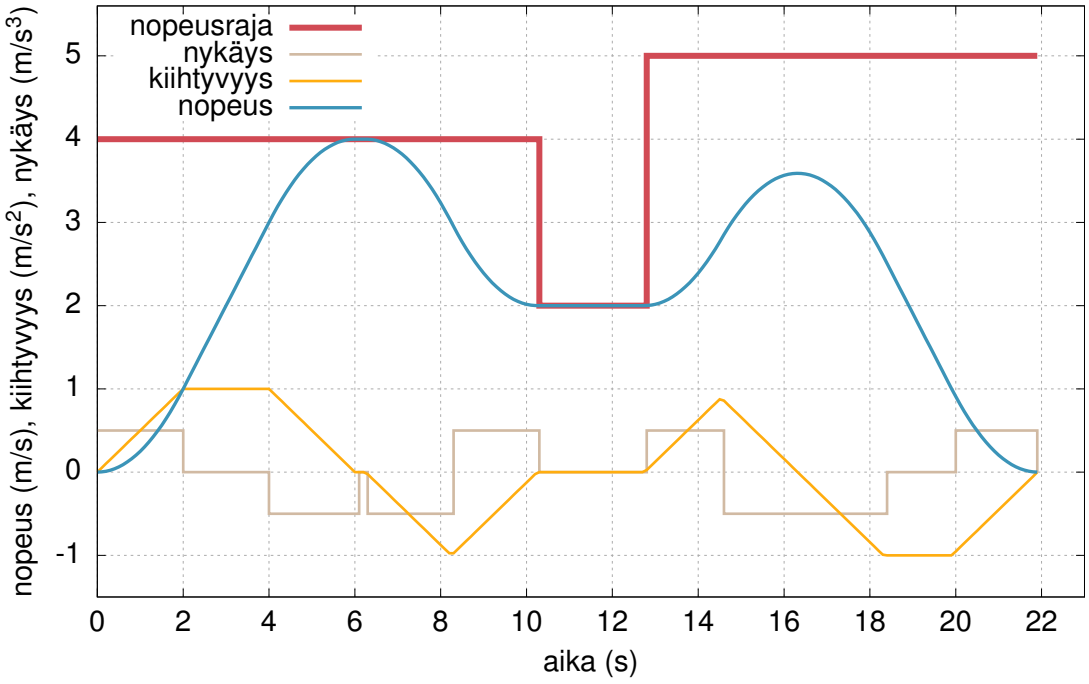
Edistyneemmässä nopeusprofiilissa mukaan otetaan nopeuden ensimmäinen derivaatta eli *kiihtyvyys*. Tällöin kiihtyvyys rajoitetaan tietylle vaihteluvälille ja se on ajan funktiona paloittain vakio. Esimerkkinä on kuva 2.4, jossa kiihtyvyys voi olla joko $-1{,}0$ m/s^2, 0,0 m/s^2 tai 1,0 m/s^2. Tällöin nopeusprofiililla on ajan funktiona lineaarisesti muuttuva tai vakiona pysyvä nopeus. Kuvan nopeusprofiili on tehty kuvan 2.3 nopeusrajojen perusteella.

Nopeusprofiilissa voidaan ottaa huomioon myös korkeampia nopeuden derivaattoja. Nopeuden toinen derivaatta *nykäys* tai *nykäisy* [3] [4] määrittää, kuinka nopeasti kiihtyvyys saa muuttua. Nopeuden kolmas derivaatta *napse* [3] [4] [5] määrittää, kuinka nopeasti nykäys saa muuttua.

Tässä työssä oletetaan, että nykäys on korkein derivaatta, joka yritetään pitää annetuissa rajoissa.

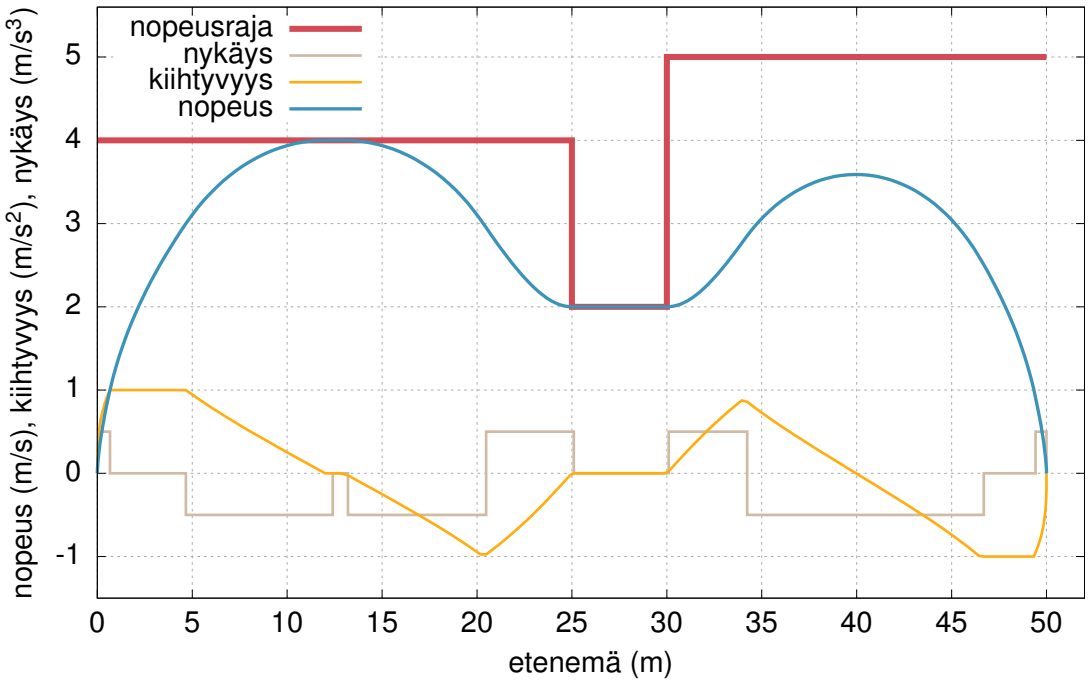


Kuva 2.4. *Esimerkki nopeusprofiilista, jossa kiihtyvyys on paloittain vakio.*



Kuva 2.5. *Esimerkki nopeusprofiilista, jossa nykäys on paloittain vakio.*

Kuvassa 2.5 on nopeusprofiili, jossa kiihtyvyys on rajoitettu välille $[-1{,}0\ \mathrm{m/s^2}, 1{,}0\ \mathrm{m/s^2}]$ ja nykäys välille $[-0{,}5\ \mathrm{m/s^3}, 0{,}5\ \mathrm{m/s^3}]$. Kuvassa 2.6 sama nopeusprofiili on esitetty etenemän funktiona.



Kuva 2.6. *Esimerkki kuvan 2.5 nopeusprofiilista etenemän funktiona.*

2.4.1 Robotin fyysiset rajoitteet ja nykäys

Nykäys lisää nopeusprofiilin generoinnin monimutkaisuutta ja olisi houkuttelevaa jättää se pois nopeusprofiilin suunnittelusta. Nopeusprofiilin on kuitenkin tarkoitus olla mahdollisimman tarkka suunnitelma robotin liikkeestä ja nopeudesta, jota robotti pystyy noudattamaan. Kun nopeusprofiili on laadittu tarpeeksi tarkasti, robotin matka-aika voidaan selvittää, se liikkuu turvallisesti halutulla nopeudella ja sillä on teoreettinen mahdollisuus päätyä oikeaan paikkaan, vaikka sen paikkaa ei voitaisikaan aina mitata tarkasti.

Robotti ei välttämättä fyysisten ominaisuuksiensa takia pysty noudattamaan nopeusprofiilia, jossa nykäystä tai jopa kiihtyvyyttä ei ole otettu huomioon (”karkea” nopeusprofiili). Esimerkiksi kiihtyvyyteen vaikuttavia ominaisuuksia ovat robotin massa, moottorin ajo- ja jarrutusvoima ja renkaiden ominaisuudet. Hyvin raskas robotti, jolla on suhteellisen heikkotehoinen moottori, joutuu käyttämään paljon aikaa nopeutensa nostamiseen.

Jos robotti taas pystyisikin noudattamaan nopeusprofiilia, jossa nykäyksen tai kiihtyvyyden on oletettu olevan ääretön, sen ei välttämättä haluta tekevän niin. Robotti, jonka painopiste on korkealla, voi alkaa heilua tai jopa kaatua, jos se kiihdyttää tai hidastaa liian nopeasti. Nykäyksen rajoittamattomat muutokset aiheuttavat mekaanista kulumista ja liikkumaan pääsevien osien iskeytymistä muihin osiin. Robotin renkaat voivat luisua tai lipsua.

Robotin kaikkien rajoittavien ominaisuuksien huomioon ottaminen ja niiden vaikutusten laskeminen nopeusprofiiliin on työlästä. Siksi ominaisuudet on yksinkertaistettu nopeuden derivaattoihin, eli robotille on määrätty raja-arvot kiihtyvyydelle, hidastuvuudelle ja nykäykselle. Myös nämä raja-arvot tiedetään ennalta; ne voidaan esimerkiksi mitata tes-

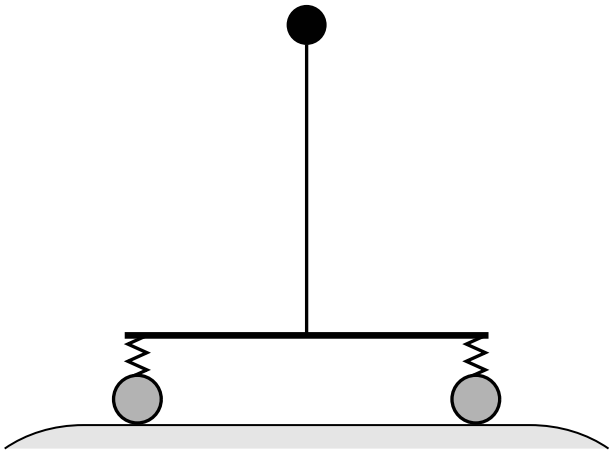
teissä.

Osa mahdollisista ongelmista johtuu myös siitä, että robotin moottorien ja jarrujen ohjausjärjestelmät eivät ole ideaalisia, eli ne eivät välttämättä pysty ottamaan tarpeeksi hyvin huomioon esimerkiksi omaa viivettään, moottorien viivettä, kiihdytystehoa tai robotin massaa. Kokemuksesta tiedetään, että ongelmat poistuvat tai lievenevät tarpeeksi, kun nopeusprofiilia tehtäessä otetaan huomioon nykäys.

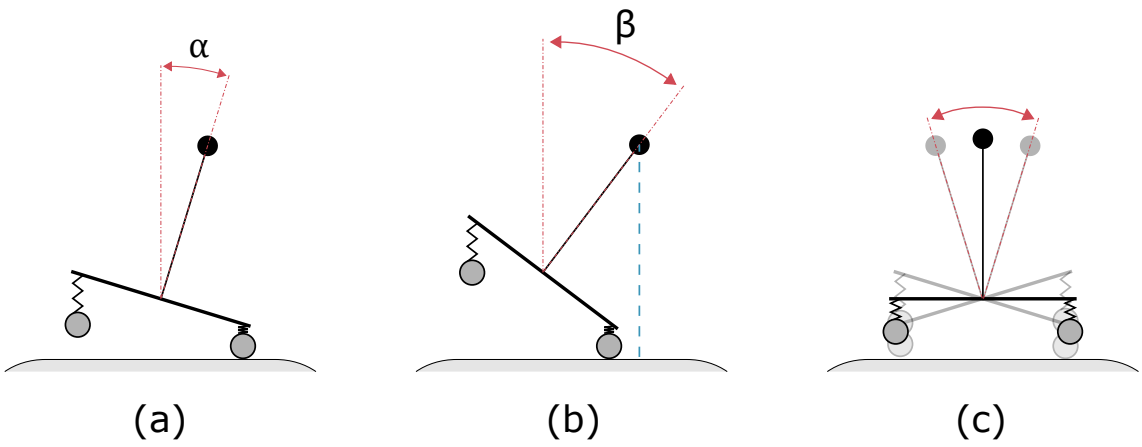
2.4.2 Liian karkean nopeusprofiilin ongelmat

Kuten edellä mainittiin, liian karkea nopeusprofiili saattaa aiheuttaa robotille erilaisia ongelmia. Ongelmat voivat johtua mm. siitä, että robotti yrittää toteuttaa profiilin mukaista liikettä, mutta ei onnistu siinä, ja jopa siitä, että robotti pystyy liikkumaan profiilin mukaan. Tässä aliluvussa havainnollistetaan ongelmia yksinkertaistettujen nopeusprofiilien ja robotin mallin avulla.

Kuvassa 2.7 on yksinkertaistettu robotin malli, jonka painopiste on korkealla (ylhäällä oleva musta ympyrä) ja jolla on kaksi rengasta tukipisteinä (harmaat ympyrät). Kuvat 2.8, 2.10, 2.11, 2.12 ja 2.13 esittävät ongelmia, joita liian karkea nopeusprofiili saattaa aiheuttaa.



Kuva 2.7. *Yksinkertainen robotin malli.*



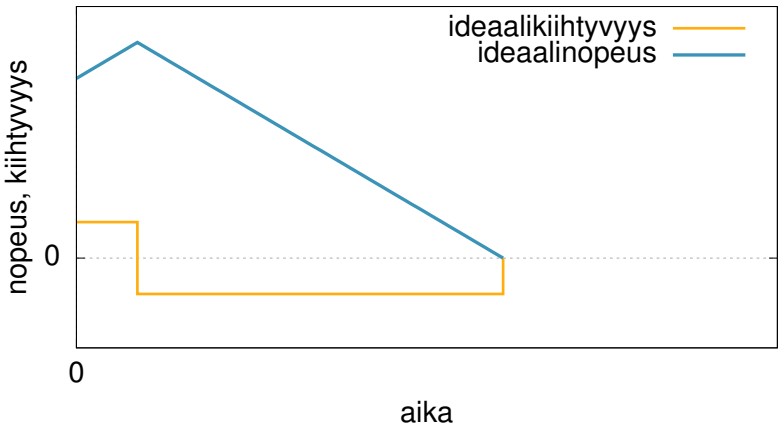
Kuva 2.8. *Robotin ongelmia.*

Kuvan 2.8 kohdassa (a) robotti on kallistunut kulmaan α, mikä ei vielä aiheuta kaatumista. Se saattaa silti vaikuttaa jarrutuksen tai kiihdytyksen tehoon tai aiheuttaa renkaiden lipsumista tai mahdollisen lastin siirtymistä. Kallistumisen voi aiheuttaa liian voimakas kiihdytys tai jarrutus.

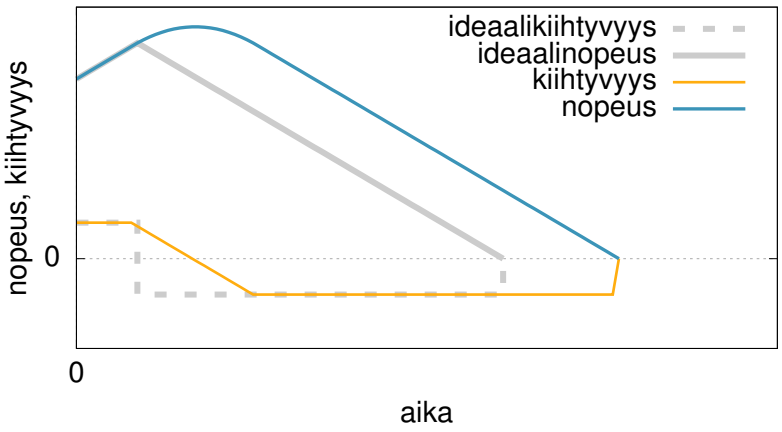
Kuvan 2.8 kohdassa (b) kallituskulma on β, joka ylittää robotin tukipisteen, joten robotti kaatuu. Kaatumisen seurauksena robotti saattaa vahingoittua tai esimerkiksi liikkua hallistemattomasti, jos osa sen renkaista on edelleen kiinni alustassa.

Kallistumisesta seuraa todennäköisesti heilumisliikettä, kuten kuvan 2.8 kohdassa (c) on esitetty. Heiluminen aiheuttaa samoja ongelmia kuin kallistuminenkin. Lisäksi se voi aiheuttaa värähtelyä kiihtyvyyteen ja nopeuteen ja mahdollisesti haitata robotin muita mittauslaitteita. Lisäksi jos robotti jatkaa kiihdyttämistä tai hidastamista, heiluminen saattaa voimistua ja johtaa robotin kaatumiseen.

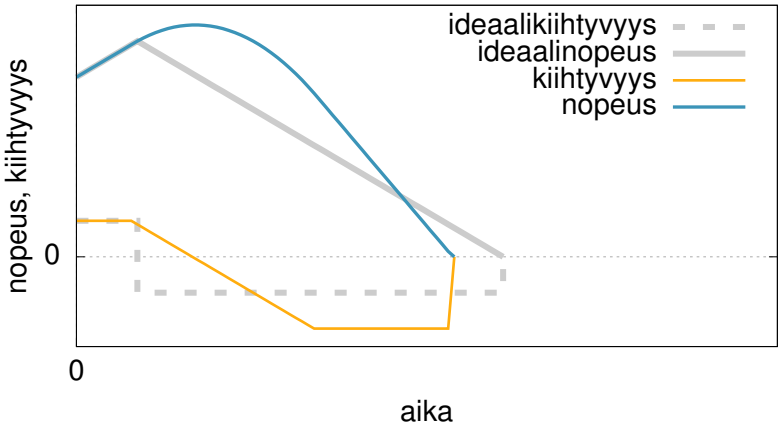
Kuvissa 2.10, 2.11, 2.12 ja 2.13 on havainnollistettu robotin ongelmia yksinkertaisten nopeusprofiilien avulla ja esitetty tapoja, joilla robotin moottorin ohjausjärjestelmä voi yrittää ratkaista niitä. Kaikissa tapauksissa säätöarvo (ideaalinen mutta käyttötilanteeseen liian karkea nopeusprofiili) on kuvassa 2.9 oleva nopeusfunktio, jossa robotilla on tietty alku-



Kuva 2.9. *Ideaalinen nopeusprofiili.*



Kuva 2.10. *Toteutunut nopeusprofiili.*



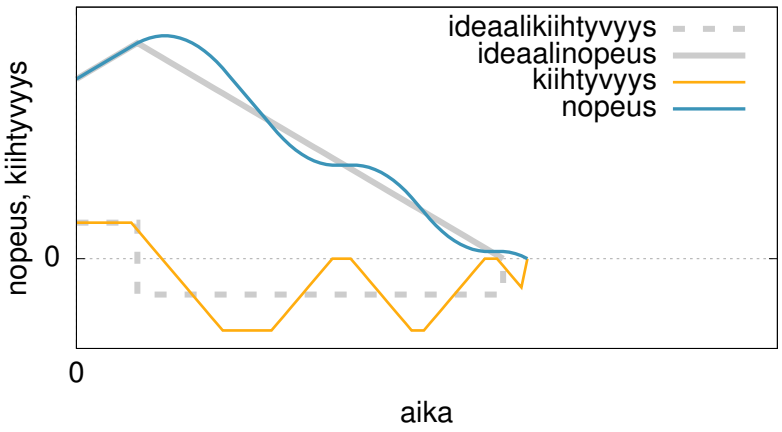
Kuva 2.11. *Nopeusprofiilin yksinkertainen korjausyritys.*

nopeus ja jonka perusteella nopeuden tulisi ensin kasvaa hieman tasaisella kiihtyvyydellä ja sitten laskea tasaisella hidastuvuudella nollaan. Tapauksissa oletetaan myös, että hidastaminen tehdään moottorilla.

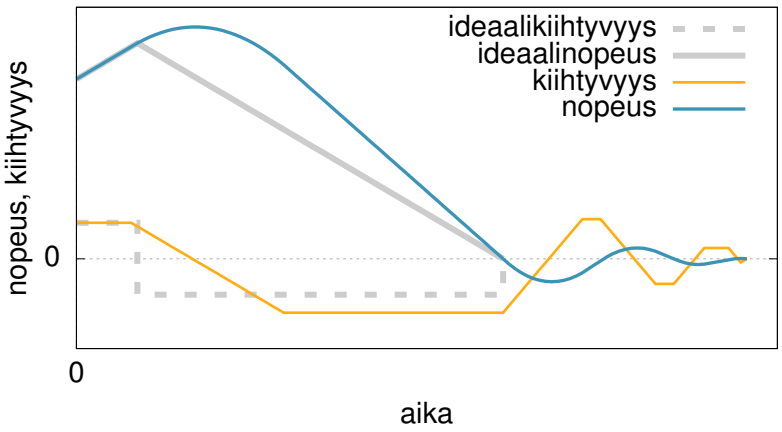
Kuvassa 2.10 on tapaus, jossa robotin moottori ei kykene toteuttamaan välitöntä kiihtyvyyden muutosta. Sen sijaan kiihtyvyys muuttuu tietyn ajan kuluessa (kuvassa tasaisella nykäyksellä), kunnes sillä on haluttu arvo. Näin ollen robotin toteutunut matka-aika on pidempi ja etenemä suurempi kuin suunniteltiin.

Kuvassa 2.11 robotti yrittää korjata tilannetta kasvattamalla hidastuvuuttaan suuremmaksi kuin nopeusprofiilissa on määritelty. Riippuen siitä, kuinka suuri hidastuvuus voi olla ja mikä oli robotin suurin nopeus ennen hidastusta, robotin toteutunut etenemä ja matka-aika todennäköisesti edelleen poikkeavat suunnitellusta suuntaan tai toiseen. Esimerkkikuvassa matka-aika on suunniteltua lyhyempi ja etenemä hieman suurempi. Ero etenemässä voidaan arvioida vertaamalla nopeuskäyrien ja koordinaattiakselien rajoittamia pinta-aloja; ideaalinopeuskäyrän alla pinta-ala on hieman pienempi (eli etenemä on pienempi) kuin toteutuneen nopeuskäyrän.

Jos robotti on kasvattanut hidastuvuuttaan kuvan 2.11 tavalla, moottorin ohjausjärjes-



Kuva 2.12. *Korjausyritys aiheuttaa värähtelyä hidastuvuudessa.*

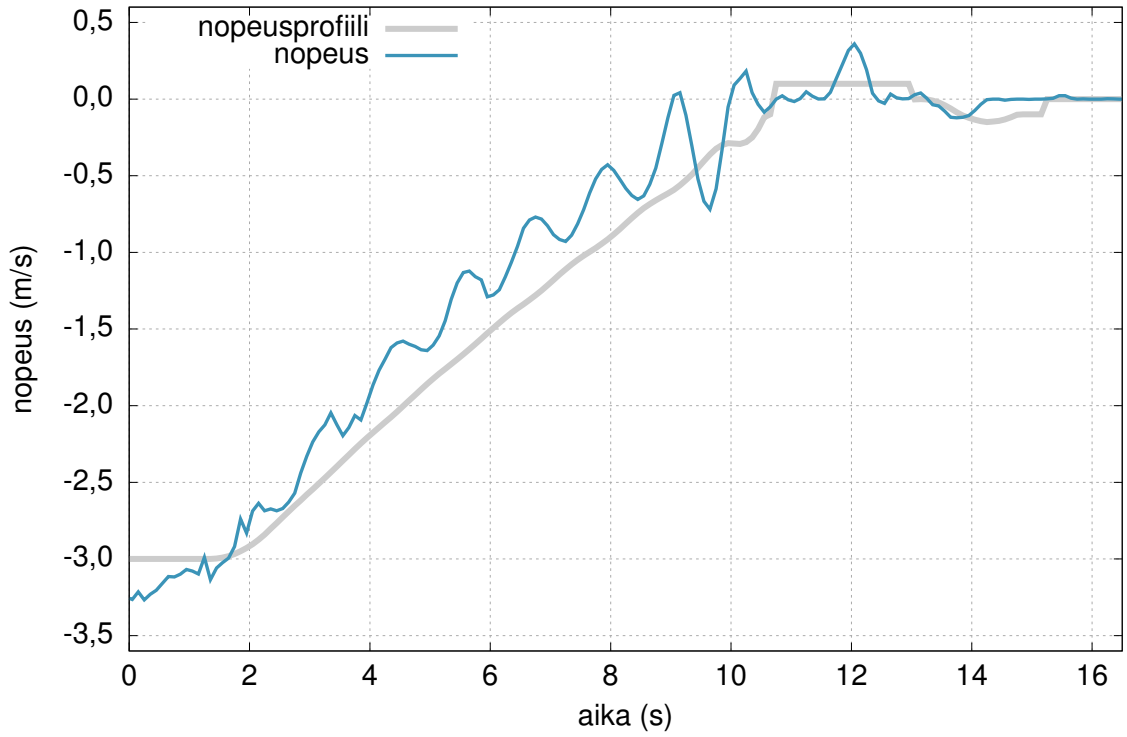


Kuva 2.13. *Korjausyritys aiheuttaa värähtelyä nopeudessa.*

telmä saattaa osata reagoida siihen, että robotti saavuttaa tavoitenopeuden, ja asettaa uuden hidastuvuuden/kiihtyvyyden. Koska kiihtyvyyden muutos ei edelleenkään tapahdu välittömästi, kiihtyvyys saattaa alkaa värähdellä, kuten kuvassa 2.12 esitetään.

Todennäköisesti robotti ei pysty toteuttamaan kuvissa 2.10 ja 2.11 tapahtuvaa yhtäkkistä pysähtymistä (kun nopeuskäyrä saavuttaa pystyakselin nollakohdan) eli kiihtyvyyden ja nopeuden asettamista nollaan. Tällöin myös nopeus saattaa alkaa värähdellä, eli robotti kulkee hetken aikaa vuorotellen taakse ja eteen, mikä näkyy kuvan 2.13 negatiivisessa nopeudessa. Jos pysähtyminen toteutettaisiin jarruilla, värähtelyä ei tapahtuisi.

Kokemuksesta tiedetään, että robotin kiihtyvyyden ja nopeuden värähtelyä tapahtuu. Kuvassa 2.14 on esimerkki oikean robotin käyttämästä nopeusprofiilista ja mitatusta nopeudesta, kun robotti jarruttaa ja lopulta pysähtyy.



Kuva 2.14. *Esimerkki oikean robotin liikkeestä.*

2.5 Nopeusprofiilin esitysmuotoja

Nopeusprofiili voidaan esittää ainakin näytteistettynä tai analyyttisesti eli laskukaavoina.

Näytteistetty nopeusprofiili on diskreettien näytteiden (joita kutsutaan tässä työssä myös *pisteiksi*) jono. Näytteet sisältävät vähintään etenemän ja nopeuden, mutta niissä voi olla myös aika ja nopeuden derivaattoja. Mikäli haluttua etenemää ei löydy jonosta, sen kohdalla oleva nopeus pitää interpoloida lähimmistä arvoista.

Laskukaava on täsmällisin keino kuvata nopeusprofiilia. Tietyn etenemän kohdalla olevan nopeuden saa yksinkertaisesti laskemalla kaavan arvon.

2.6 Nopeusprofiilin generointimenetelmiä

Nopeusprofiilin generoimisen voi toteuttaa useilla eri menetelmillä. Esitysmuoto vaikuttaa menetelmään sikäli, että jos profiili halutaan esittää kaavoina, kaavat on tuotettava. Kaavoja voi myös käyttää näytejonon tekemiseen. Mikäli kaavoja ei haluta tehdä, nopeusprofiilin voi generoida esimerkiksi laskemalla diskreettejä pisteitä.

2.6.1 Laskukaavat

Nopeusprofiilin voi esittää laskukaavoina. Tämän aliluvun esimerkissä kaavat esittävät etenemää, nopeutta, kiihtyvyyttä ja nykäystä ajan funktioina. Funktiot on määritelty pa-

loittain polynomeina. Nopeusprofiilia kuvaavien polynomien tuottamiseen käytetään seuraavia sääntöjä:

- Nykäyksen kaava on paloittain vakio. Sen arvot voivat olla joko j_{min}, j_0 (eli 0,0 m/s^3) tai j_{max}.
- Kiihtyvyyden kaava on paloittain vakio tai ensimmäisen asteen polynomi niin, että yllä olevat nykäyksen arvot ovat sen derivaattoja. Kiihtyvyys myös pysyy välillä [a_{min}, a_{max}].
- Nopeuden kaava on paloittain vakio, ensimmäisen asteen polynomi tai toisen asteen polynomi niin, että yllä olevat kiihtyvyyden arvot ovat sen derivaattoja. Lisäksi nopeus noudattaa annettuja nopeusrajoja, jotka vaihtelevat etenemän mukaan.

Jotta esimerkkitapauksen polynomeja on helpompi seurata, oletetaan, että nykäyksen ja kiihtyvyyden raja-arvot ja nopeusrajat ovat samat kuin aliluvussa 2.4 Nopeusprofiilin hienousasteet (s. 6) ja kuvassa 2.5 (s. 7).

Oletetaan, että robotti lähtee liikkeelle ajanhetkellä t_0 = 0,0 s täysin pysähdyksistä, eli nykäyksen, kiihtyvyyden, nopeuden ja etenemän alkuarvot ovat nollia. Kun kaavojen alkuarvot ovat nollia, niiden integraalien integrointivakiot ovat myös nollia. Näin ollen nykäyksen j, kiihtyvyyden a, nopeuden v ja etenemän s polynomit (kaavat 2.1, 2.2, 2.3 ja 2.4) näyttävät suhteellisen yksinkertaisilta aikavälillä [t_0, t_1]:

$$j_1(t) = j_{\text{max}} \tag{2.1}$$

$$a_1(t) = \int j_1(t)\ \mathrm{d}t = \int j_{\text{max}}\ \mathrm{d}t = j_{\text{max}} \cdot t \tag{2.2}$$

$$v_1(t) = \int a_1(t)\ \mathrm{d}t = \int j_{\text{max}} \cdot t\ \mathrm{d}t = \frac{1}{2} \cdot j_{\text{max}} \cdot t^2 \tag{2.3}$$

$$s_1(t) = \int v_1(t)\ \mathrm{d}t = \int \frac{1}{2} \cdot j_{\text{max}} \cdot t^2\ \mathrm{d}t = \frac{1}{6} \cdot j_{\text{max}} \cdot t^3 \tag{2.4}$$

Ajanhetkellä t_1 = 2,0 s kiihtyvyys saavuttaa maksimiarvonsa:

$$a_1(t_1) = a_{\text{max}} \tag{2.5}$$

Koska kiihtyvyys saavuttaa maksiminsa, nykäys asetetaan nollaksi ja muille suureille lasketaan uudet polynomit. Nyt kiihtyvyyden, nopeuden ja etenemän polynomeilla on nollasta poikkeavat alkuarvot. Aikavälillä [t_1, t_2] polynomit ovat seuraavat:

$$j_2(t) = j_0 \tag{2.6}$$

$$a_2(t) = \int j_2(t)\ \mathrm{d}t = j_0 \cdot t + c_{a_2} = c_{a_2} \tag{2.7}$$

$$v_2(t) = \int a_2(t)\ \mathrm{d}t = \int c_{a_2}\ \mathrm{d}t = c_{a_2} \cdot t + c_{v_2} \tag{2.8}$$

$$s_2(t) = \int v_2(t)\ \mathrm{d}t = \int c_{a_2} \cdot t + c_{v_2}\ \mathrm{d}t = \frac{1}{2} \cdot c_{a_2} \cdot t^2 + c_{v_2} \cdot t + c_{s_2} \tag{2.9}$$

Ajanhetkellä t_1 samojen suureiden eri aikavälien polynomeilla on oltava sama arvo:

$$a_1(t_1) = a_2(t_1) \tag{2.10}$$

$$v_1(t_1) = v_2(t_1) \tag{2.11}$$

$$s_1(t_1) = s_2(t_1) \tag{2.12}$$

Näin ollen integrointivakiot voidaan selvittää. Kiihtyvyyden integrointivakio saadaan kaavoista 2.5, 2.7 ja 2.10:

$$c_{a_2} = a_{\text{max}} \tag{2.13}$$

Nopeuden integrointivakio saadaan kaavoista 2.3, 2.8, 2.11 ja 2.13:

$$c_{v_2} = \frac{1}{2} \cdot j_{\text{max}} \cdot {t_1}^2 - a_{\text{max}} \cdot t_1 \tag{2.14}$$

Etenemän integrointivakio saadaan kaavoista 2.4, 2.9, 2.12, 2.13 ja 2.14:

$$c_{s_2} = -\frac{1}{3} \cdot j_{\text{max}} \cdot {t_1}^3 + \frac{1}{2} \cdot a_{\text{max}} \cdot {t_1}^2 \tag{2.15}$$

Ajanhetkellä t_2 = 4,0 s nykäys asetetaan vakioksi j_{min}. Aikavälillä [t_2, t_3] funktiot ovat seuraavat:

$$j_3(t) = j_{\text{min}} \tag{2.16}$$

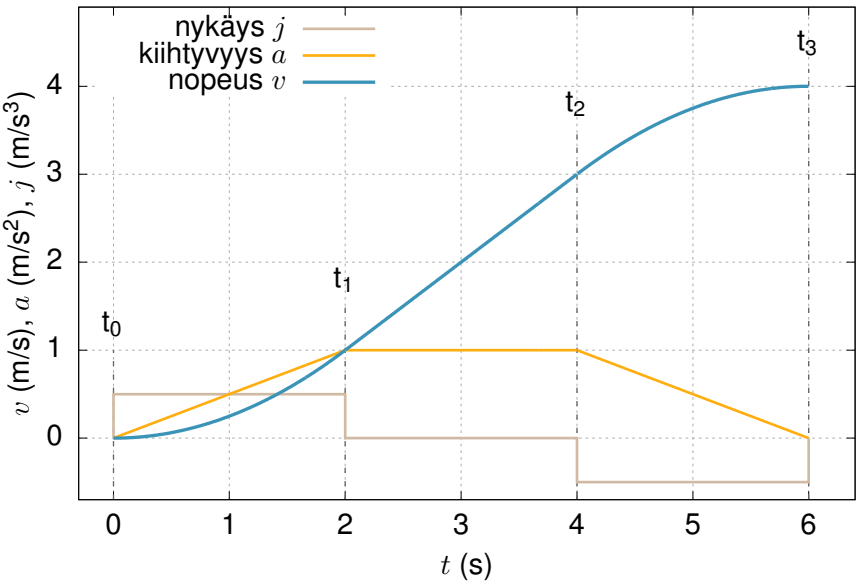
$$a_3(t) = \int j_3(t)\ \mathrm{d}t = \int j_{\text{min}}\ \mathrm{d}t = j_{\text{min}} \cdot t + c_{a_3} \tag{2.17}$$

$$v_3(t) = \int a_3(t)\ \mathrm{d}t = \int j_{\text{min}} \cdot t + c_{a_3}\ \mathrm{d}t = \frac{1}{2} j_{\text{min}} \cdot t^2 + c_{a_3} \cdot t + c_{v_3} \tag{2.18}$$

$$s_3(t) = \int v_3(t)\ \mathrm{d}t = \int \frac{1}{2} \cdot j_{\text{min}} \cdot t^2 + c_{a_3} \cdot t + c_{v_3}\ \mathrm{d}t = \frac{1}{6} j_{\text{min}} \cdot t^3 + \frac{1}{2} c_{a_3} \cdot t^2 + c_{v_3} \cdot t + c_{s_3} \tag{2.19}$$

Integrointivakiot voidaan selvittää kuten edellä. Olennaista on se, että polynomit monimutkaistuvat alkuarvojensa takia sitä enemmän, mitä useamman nykäyksen muutoksen päästä niitä halutaan tuottaa. Lisäksi edellä esitetyssä esimerkissä oletettiin, että tähdätään nopeusrajaan 4,0 m/s ajanhetkellä t_3 = 6,0 s. Nopeusrajafunktion vielä saavuttamattomat nopeusrajat vaikuttavat siihen, mikä haluttu nopeus oikeasti on; esim. kuvassa 2.5 (s. 7) nopeus ei koskaan saavuta viimeistä nopeusrajaa (5,0 m/s), koska robotin pitää alkaa hidastaa ennen sitä, jotta se voi pysähtyä oikeaan kohtaan. Vaikein osa kaavojen generointia onkin selvittää rajakohdat (t_i), jolloin nykäystä pitäisi muuttaa.

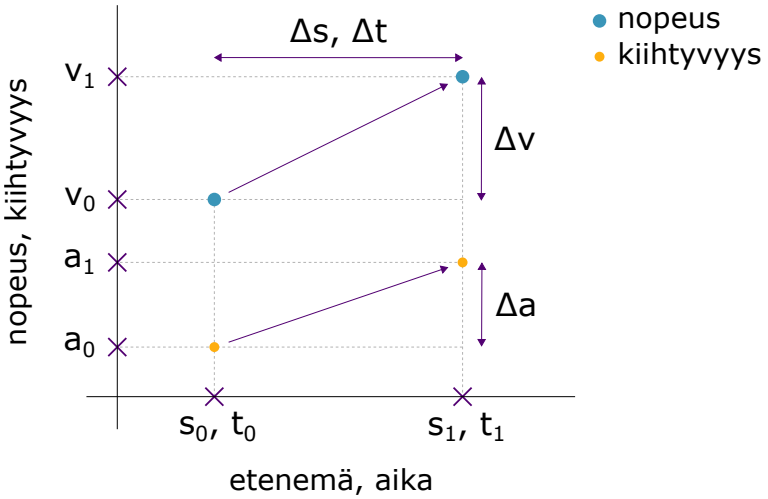
Kuvassa 2.15 on havainnollistettu nykäyksen, kiihtyvyyden ja nopeuden polynomeja aikavälillä [t_0, t_3].



Kuva 2.15. *Polynomit välillä $[t_0, t_3]$.*

2.6.2 Diskreetit pisteet

Uusi nopeusprofiilin piste voidaan laskea edellisen pisteen avulla. Kuvassa 2.16 on havainnollistettu pisteen laskemista.



Kuva 2.16. *Pisteen laskeminen.*

Oletetaan, että edellisellä pisteellä on seuraavat arvot:

etenemä s_0,
aika t_0,
nopeus v_0,
kiihtyvyys a_0 ja
nykäys j_0.

Kun aikaa kuluu Δt, muutokset arvoissa ovat seuraavat:

$$\Delta a = j_0 \cdot \Delta t,$$
$$\Delta v = a_0 \cdot \Delta t \text{ ja}$$
$$\Delta s = v_0 \cdot \Delta t.$$

Uudet arvot voidaan laskea seuraavasti (kun nykäys ei muutu):

$$a_1 = a_0 + \Delta a,$$
$$v_1 = v_0 + \Delta v,$$
$$s_1 = s_0 + \Delta s \text{ ja}$$
$$t_1 = t_0 + \Delta t.$$

Pisteitä voidaan laskea myös Δs:n avulla. Tällöin arvioidaan ensin Δt:

$$\Delta t = \Delta s / v_0,$$

minkä jälkeen muut muutokset ja uudet pisteet voidaan laskea kuten aiemmin.

3 NOPEUSPROFIILIGENERAATTORIN UUDELLEENSUUNNITTELU

Uuden nopeusprofiiligeneraattorin on tarkoitus korvata robotin laajemmassa automaatio-ohjelmistossa oleva nykyinen nopeusprofiiligeneraattori. Nykyinen generaattori toimii useimmissa tapauksissa, mutta siinä on tietyt ongelmansa. Uuden nopeusprofiiligeneraattorin algoritmiksi esitellään useita vaihtoehtoja. Algoritmien toimintaa on havainnollistettu kuvaajilla, jotka on piirretty Inkscape-ohjelmalla.

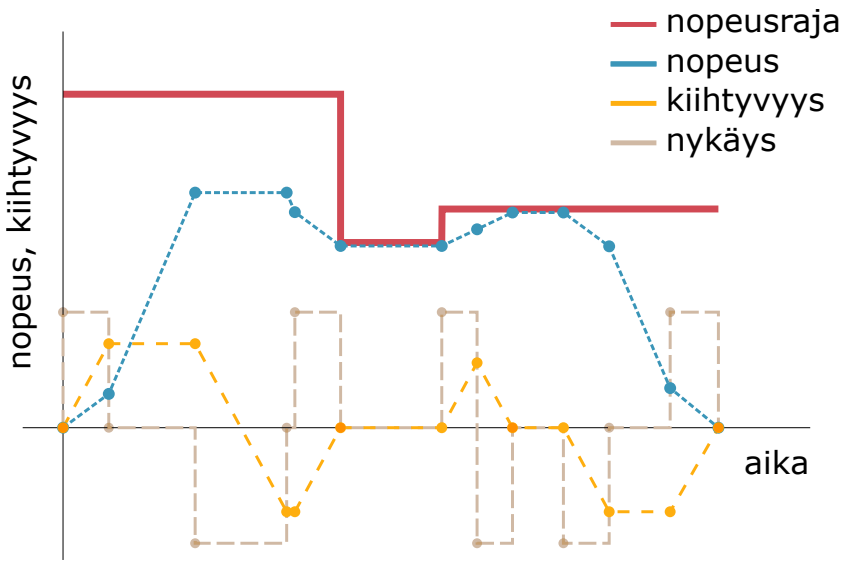
3.1 Korvattava nopeusprofiiligeneraattori

Robotin automaatio-ohjelmiston nykyisin käytössä oleva nopeusprofiiligenerattori käsittelee profiileja ajan funktioina. Etenemäfunktio on kolmannen asteen polynomi, nopeusfunktio on toisen asteen polynomi ja kiihtyvyysfunktio on ensimmäisen asteen polynomi. Nykäys on paloittain vakio.

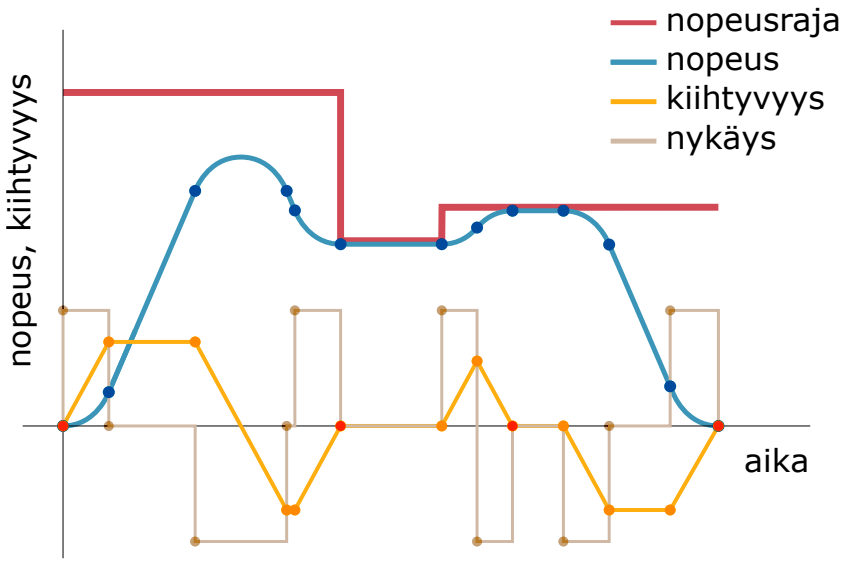
Yksinkertaistetusti sanottuna generaattori tuottaa polynomit etsimällä kohdat, joissa nykäyksen pitää muuttua, jotta nopeusprofiili pysyy nopeusrajoissa eikä riko kiihtyvyyden rajoituksia. Kuvassa 3.1 on havainnollistettu pisteillä kohtia, joiden välillä profiilin tuottamiseen käytetään erilaisia polynomeja. (Katkoviivat auttavat hahmottamaan, mikä on seuraava piste.) Kuvaan 3.2 on piirretty varsinaiset polynomit ja pisteet näyttävät edelleen eri polynomien vaihtokohdat.

Generaattorin tuottamasta profiilista voidaan selvittää robotin haluttu nopeus, kiihtyvyys ja nykäys joko ajan tai etenemän perusteella.

Generaattori on kirjoitettu C++:lla. C++ [6] on standardoitu, yleiskäyttöinen ja vahvasti tyypitetty ohjelmointikieli, joka tukee olio-ohjelmointia ja dynaamista muistinhallintaa.



Kuva 3.1. *Nykyisen generaattorin etsimät polynomien leikkauskohdat.*



Kuva 3.2. *Nykyisen generaattorin tuottamat polynomit.*

3.1.1 Ongelmat

Vaikka nykyinen nopeusprofiiligeneraattori useimmissa tapauksissa toimiikin, se ei ole täysin luotettava. Siitä on löydetty vaikeasti jäljitettäviä virheitä ja niitä uskotaan olevan piilossa vielä useita.

Nopeusprofiiligeneraattori ei ole täysin muusta ohjelmiston koodista erotettu moduuli, vaan se käyttää funktioita, olioita ja jäsenmuuttujia useammasta eri moduulista ja toisiinsa liittyvien kooditiedostojen ryppäästä. Joissain tapauksissa generaattori jopa rikkoo luokkien kapselointia eli käyttää suoraan vieraiden luokkien jäsenmuuttujia (tai päinvastoin, eli nopeusprofiiligeneraattorin kapselointia rikotaan). Näin ollen koodin tutkiminen on aikaavievää, sen ymmärtäminen vaativaa ja muokkaaminen riskialtista.

Nopeusprofiiligeneraattorin C++-kielinen koodi on sinänsä tehokas ja sopii tehtäväänsä,

mutta siinä on käytetty joitakin riskialttiita ominaisuuksia. Koodissa on esimerkiksi paljon osoittimia, joiden käytön kanssa on oltava erityisen huolellinen. Osoittimet ovat suoria viittauksia tietokoneen muistipaikkoihin, joten väärän tai vapautetun muistin lukeminen tai muuttaminen voi aiheuttaa lukuisia ongelmia tiedon korruptoitumisesta tietoturvariskeihin ja ohjelman tai jopa käyttöjärjestelmän kaatumisen. Koodia on myös voimakkaasti optimoitu suorituskykyä varten; tiettyjä koodinpätkiä ja lohkoja on esimerkiksi kopioitu suoraan tai vain pienin muutoksin useisiin eri paikkoihin, mikä vaikeuttaa ylläpidettävyyttä.

Laajemman automaatio-ohjelmiston puitteissa on katsottu, että polynomeja tuottava algoritmi on turhan monimutkainen vaatimuksiin nähden.

3.2 Uusi nopeusprofiiligeneraattori

Nopeusprofiiligeneraattorin uudelleensuunnittelun on tarkoitus yksinkertaistaa nopeusprofiilin suunnittelualgoritmia, selkeyttää koodia ja tehdä kunnollinen dokumentaatio.

Nopeusprofiiligeneraattori kirjoitetaan uudestaan niin, että uusi toteutus on muusta automaatio-ohjelmistosta erillinen kokonaisuus: se on helpompi ymmärtää omana kokonaisuutenaan, ei ole riippuvainen muista ohjelmiston osista eikä riko muiden olioiden kapselointia tai anna rikkoa omaansa. Muut tekniset ratkaisut esitetään aliluvussa 4.2 (s. 38).

Seuraavaksi esitellään ideoita erilaisista algoritmeista, joilla nopeusprofiilin voi generoida. Juhana Helovuo keksi kuminauha-algoritmin idean, Veini Lehkonen loput.

- Käytetään alkuperäistä (aliluku 3.2.1, sivulla 22): kirjoitetaan uudelleen polynomeja tuottava algoritmi.
- Kuminauha-algoritmi (aliluku 3.2.2, sivulla 22): tehdään näytteistetty nopeusprofiili, jossa nopeus on sama koko etenemälle. Sitten nopeusprofiilia muokataan useiden iteraatioiden aikana niin, että pisteiden nopeutta nostetaan tai lasketaan nykäyksen ja kiihtyvyyden rajoissa, kunnes profiili osuu nopeusrajoihin.
- Osakäyräalgoritmi (aliluku 3.2.3, sivulla 24): Ensin selvitetään alin nopeusraja ja tehdään näytteistetty nopeusprofiili nykäyksen ja kiihtyvyyden rajoissa niin, että robotin nopeus on mahdollisimman suuren osan ajasta alimman nopeusrajan mukainen. Seuraavaksi etsitään profiilin ne osat, joissa nopeus on alle nopeusrajojen ja joissa nopeutta olisi vielä mahdollista nostaa. Sitten selvitetään kullekin keskeneräiselle profiilin osalle seuraavaksi alin nopeusraja ja iteroidaan profiilien osien generointia, kunnes nopeuksia ei voi enää nostaa.
- Optimistinen kaksisuuntainen algoritmi (aliluku 3.2.4, sivulla 29): Lähdetään tuottamaan näytteistettyä nopeusprofiilia etenemän alusta niin, että kiihtyvyys nostetaan nykäyksen rajoissa maksimiinsa. Nopeuden annetaan kasvaa nopeusrajaan saakka, jolloin kiihtyvyys asetetaan välittömästi nollaksi ja nopeus vakioksi. Profiili etenee, ja jos alempi raja alkaa, nopeus asetetaan suoraan alempaan rajaan. Kun ylempi raja alkaa, kiihtyvyyttä nostetaan taas nykäyksen rajoissa. Tämä tehdään koko nopeusraja-alueelle. Sitten sama tehdään uudestaan toiseen suuntaan (mak-

simihidastuvuudella/minimikiihtyvyydellä ja miniminykäyksellä). Molemmista profiileista valitaan alemmat nopeudet uuteen profiiliin, josta tasoitetaan kaikki jäljelle jääneet kiihtyvyyden äkkimuutokset.

- Ryömintäalgoritmi (aliluku 3.2.5, sivulla 31): Lähdetään tuottamaan näytteistettyä nopeusprofiilia etenemän alusta niin, että kiihtyvyys nostetaan nykäyksen rajoissa maksimiinsa. Nopeuden annetaan kasvaa nopeusrajaan saakka. Sitten kiihtyvyyden äkkimuutos korjataan, eli osaa aiemmista nopeuksista vähennetään, kunnes profiilin tuotettu osa noudattaa rajoituksia. Sitten profiilia tuotetaan eteenpäin, kunnes se osuu seuraavaan nopeusrajaan, ja sitten profiilia korjataan taas.
- Vakiokiihtyvyysalgoritmi (aliluku 3.2.6, sivulla 36): tehdään ensin näytteistetty profiili, jossa nykäystä ei ole huomioitu (kuten kuvassa 2.4, sivulla 7), ja sitten muokataan se sopivaksi.

Osa algoritmien hyödyistä ja haitoista riippuu siitä, käsitelläänkö profiileja ajan vai etenemän funktioina. Ne ovat periaatteessa samat jokaiselle algoritmille:

- Jos profiilia käsitellään aikatasossa, nopeuksien muutokset on helppo laskea.
- Jos profiilia käsitellään etenemätasossa, tarvittava määrä pisteitä on helppo tuottaa kerralla ja kunkin pisteen kohdalla oleva nopeusraja tiedetään aina.
- Aikatasossa nopeusrajojen muutoskohtia pitää jatkuvasti laskea uudestaan tai ne pitää laskea etukäteen. Jos niitä ei laske etukäteen ja profiili rakennetaan useassa vaiheessa, profiili muuttuu koko ajan; etenkin pisteet lähempänä profiilin loppua muuttuvat jatkuvasti.
- Etenemätasossa ajan, nopeuden ja kiihtyvyyden muutokset pitää laskea uudestaan joka kerta, kun nopeus muuttuu.
- Etenemätasossa hyvin matalia nopeuksia pitää käsitellä varoen, sillä hitaasti etenevän profiilin aika-askel voi kasvaa liian suureksi. Jos robotti on pysähdyksissä, se voi olla jopa ääretön.

Seuraavissa aliluvuissa käydään tarkemmin läpi kunkin ehdokasalgoritmin ominaisuuksia. Esimerkkikuvissa esitetään nopeusfunktiot ja nopeusrajat yksinkertaistettuna ja viitteellisenä etenemän tai ajan funktiona.

3.2.1 Polynomialgoritmi

On mahdollista kirjoittaa uudestaan polynomeja tuottava algoritmi. Vaikka koodi selkenisi ja yksinkertaistuisi uudelleenkirjoituksessa, polynomien tuottaminen on silti monimutkaista ja vaatii varsinkin reitin loppupuolella monien alkuarvojen huomioon ottamista.

Polynomialgoritmin hyödyt:

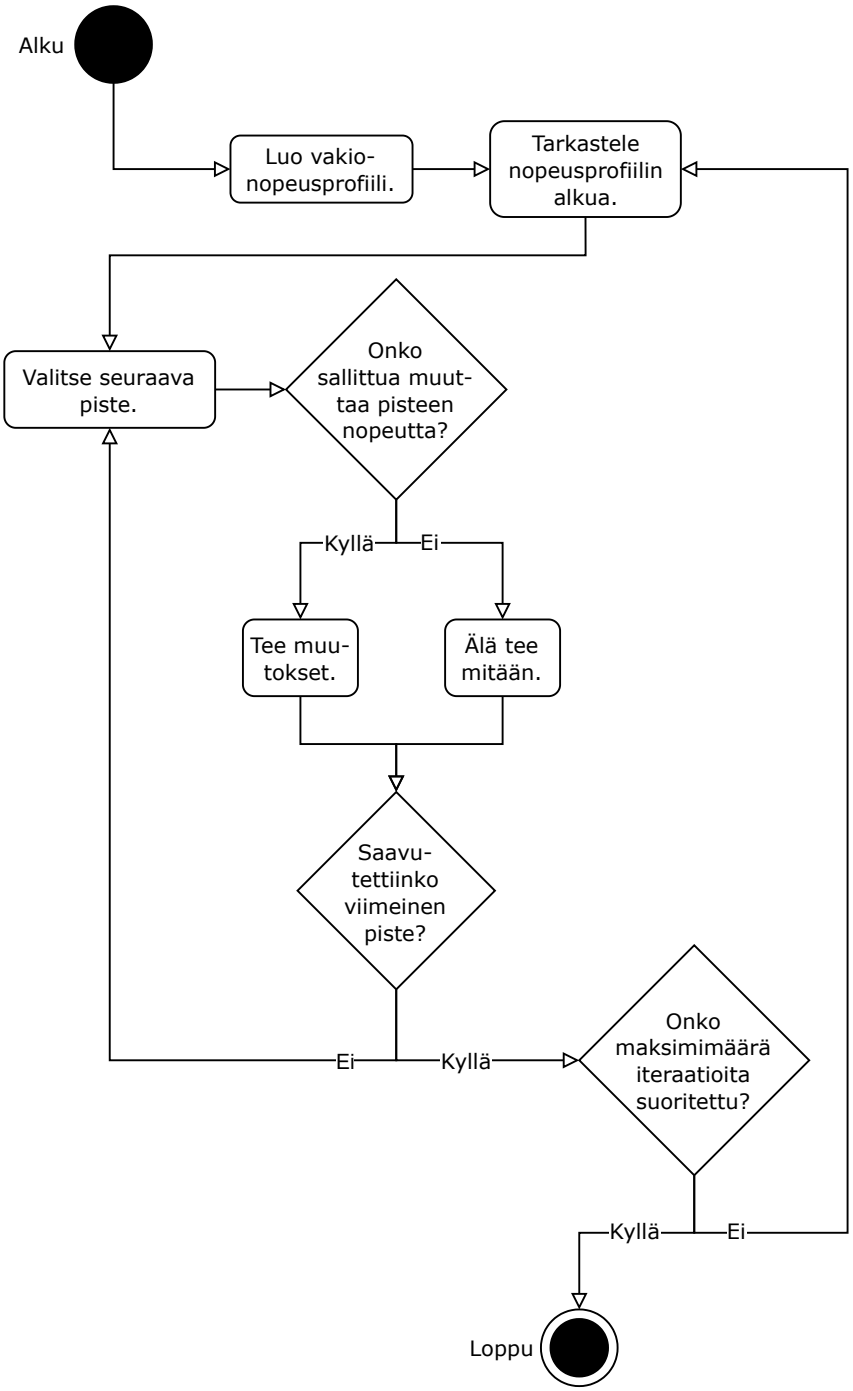
- Nykyisen generaattorin perusteella tiedetään, että polynomien avulla saadaan toimivia nopeusprofiileja.
- Polynomien avulla tehty nopeusprofiili on hyvin täsmällinen.
- Uuden generaattorin tuottamaa profiilia on helppo verrata aikaisempaan.

Polynomialgoritmin haitat:

- Polynomien tuottaminen on monimutkaista.
- Nykyisessä algoritmissa olevat ongelmat (tai osa niistä) saattavat liittyä nimenomaan polynomeihin, jolloin ongelmista ei pääsisi eroon.
- Algoritmi olisi edelleen turhan monimutkainen vaatimuksiin nähden.

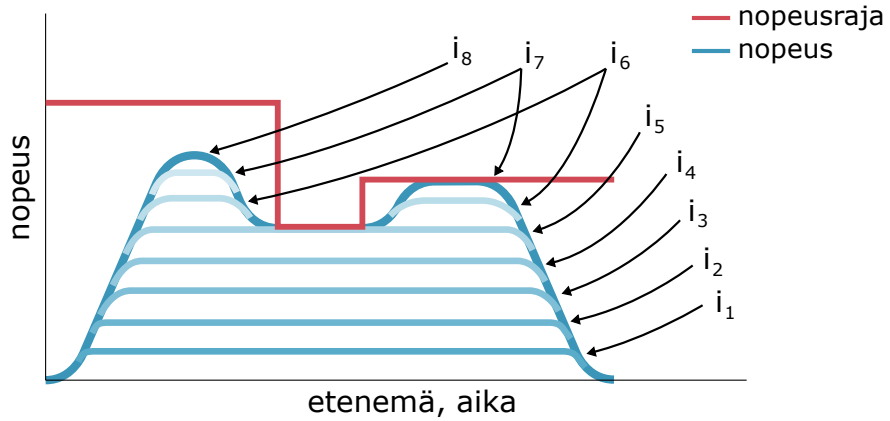
3.2.2 Kuminauha-algoritmi

Kuminauha-algoritmin idea on, että näytteistetyn nopeusprofiilin pisteiden nopeuksia nostetaan vähitellen korkeammalle. Alussa näytteistetty nopeusprofiili on hyvin hidas, jopa nolla. Pisteiden nopeuksia nostetaan yksi kerrallaan niin, että nodeuden muutos pysyy kiihtyvyyden rajoissa, ja kiihtyvyys taas pysyy nykäyksen rajoissa. Profiilin pisteet käydään läpi useita kertoja ja joka kerta nopeuksia yritetään nostaa. Algoritmin toteutuksesta riippuen nopeuksia saatetaan joutua myös vähentämään. Kun nopeuksia ei enää pysty nostamaan, profiili on valmis. Algoritmin nimi tulee siitä, että profiili venyy nopeusrajoja kohti hieman kuminauhan tapaan.



Kuva 3.3. *Vuokaavio kuminauha-algoritmista.*

Kuvassa 3.3 on vuokaavio kuminauha-algoritmin toimintaperiaatteesta. Alussa oletetaan, että robotin laajempi automaatio-ohjelmisto on lähettänyt nopeusrajafunktion nopeusprofiiligeneraattorille ja että se on kelvollinen.



Kuva 3.4. *Kuminauha-algoritmin toimintaperiaate.*

Kuvassa 3.4 on havainnollistettu nopeusprofiilin asteittaista nostamista. Lähtökohtana on nollanopeusprofiili, eli profiilin kaikkien pisteiden nopeus on nolla. Ensimmäisessä iteraatiossa (i_1) tehdään hyvin hidas profiili. Iteraatioiden i_2–i_5 aikana nopeus saadaan nostettua alimpaan nopeusrajaan. Iteraatiot i_6 ja i_7 saavat nopeusprofiilin saavuttamaan reitin loppuosan nopeusrajan ja viimeisellä iteraatiolla (i_8) saavutetaan reitin alkuosan ylin nopeus, jota ei voi enää nostaa rikkomatta kiihtyvyyden tai nykäyksen rajoja.

Kuminauha-algoritmin hyödyt:

- Toimintaperiaate on yksinkertainen.
- Ei vaadi erikoistapausten käsittelyä.

Kuminauha-algoritmin haitat:

- Iterointi vie aikaa ja suoritustehoa.
- Iterointi saattaa jumittua generoimaan esimerkiksi kahta tai jopa useampaa hieman erilaista profiiliversiota, joista kaikki ovat melkein kelvollisia ja jotka korjaavat toistensa ongelmat, mutta jotka jatkuvasti tuottavat samat ongelmat uudestaan.
- Jos profiilia käsitellään aikatasossa, sen matka-aikaa pitää alussa rajoittaa. Muuten se tuottaa turhaan suuria määriä hitaita pisteitä, joista suurinta osaa ei käytetä.

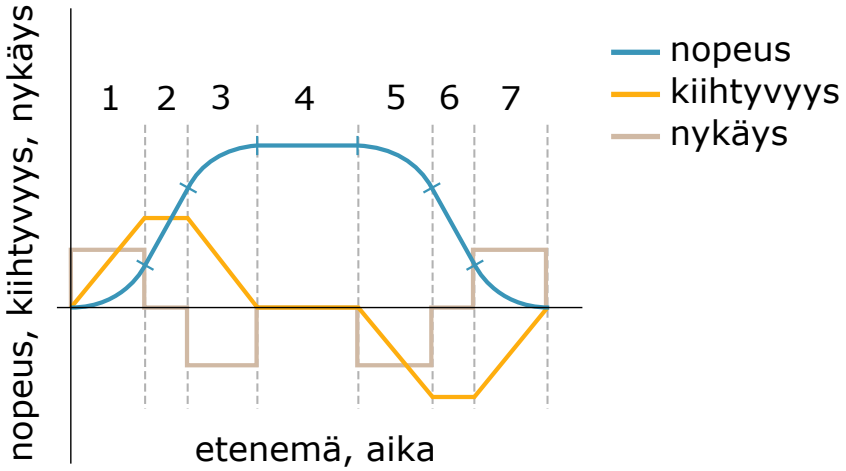
3.2.3 Osakäyräalgoritmi

Osakäyräalgoritmissa nopeusprofiili muodostetaan osakäyristä, joita on seitsemän erilaista:

1. Positiivinen vakionykäys, kasvava kiihtyvyys ja kasvava nopeus.
2. Positiivinen vakiokiihtyvyys ja kasvava nopeus.
3. Negatiivinen nykäys, vähenevä kiihtyvyys ja kasvava nopeus.
4. Vakionopeus.
5. Negatiivinen nykäys, vähenevä kiihtyvyys ja vähenevä nopeus.

6. Negatiivinen kiihtyvyys (hidastuvuus) ja vähenevä nopeus.
7. Positiivinen vakionykäys, kasvava kiihtyvyys ja vähenevä nopeus.

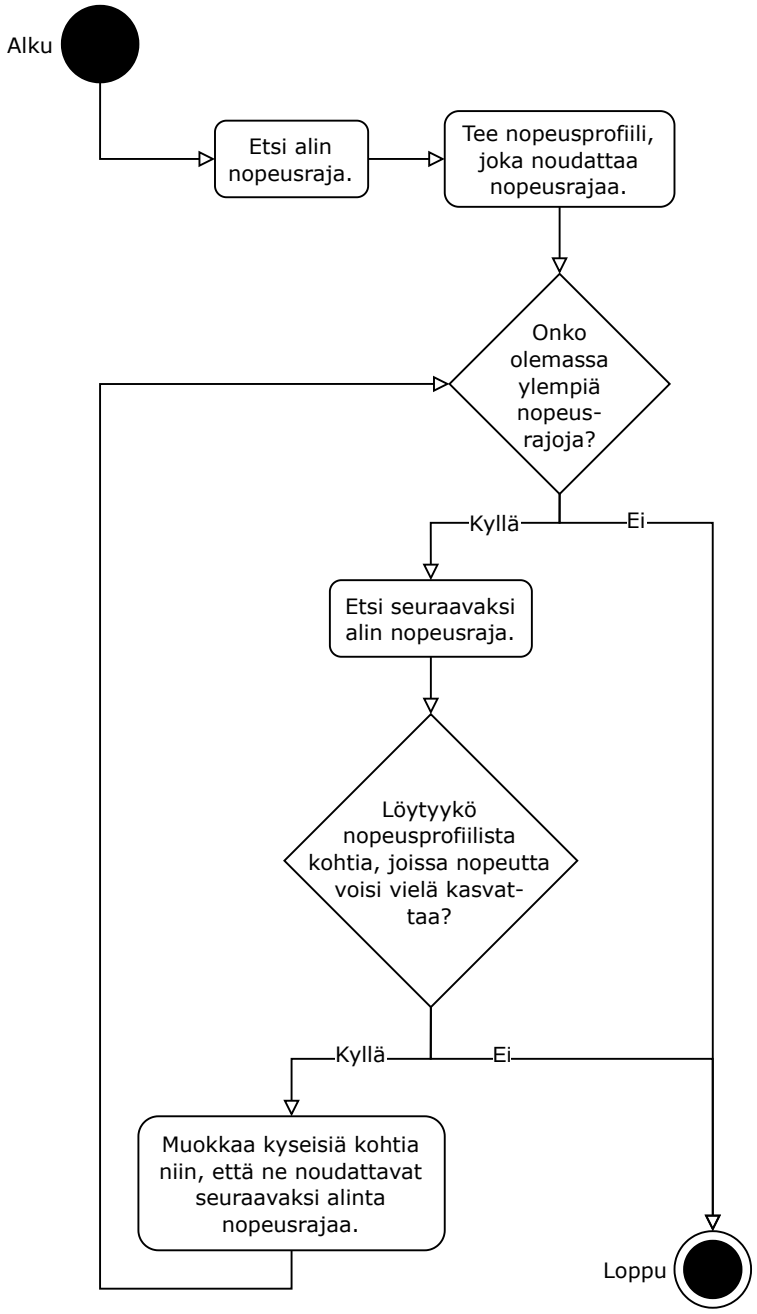
Osakäyrät on esitelty kuvassa 3.5. Osakäyriä 1, 3, 5 ja 7 pitää tarvittaessa lyhentää, jotta ne saadaan sovitettua nopeusrajoihin.



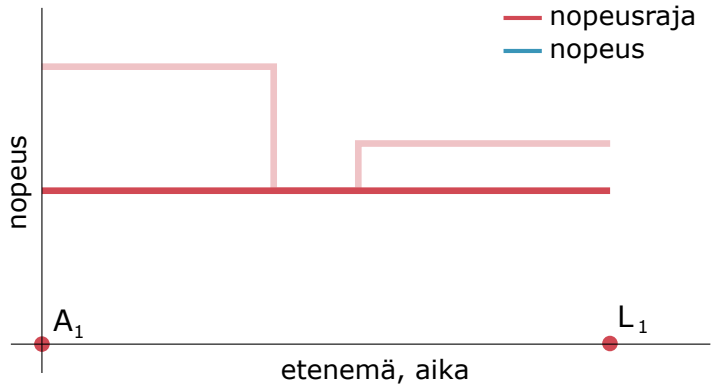
Kuva 3.5. *Osakäyrät 1–7.*

Tämän osakäyräalgoritmin kaltainen idea on esitelty myös artikkelissa [2], jossa käsitellään syvemmin, miten nykäyksen muutoskohdat löydetään.

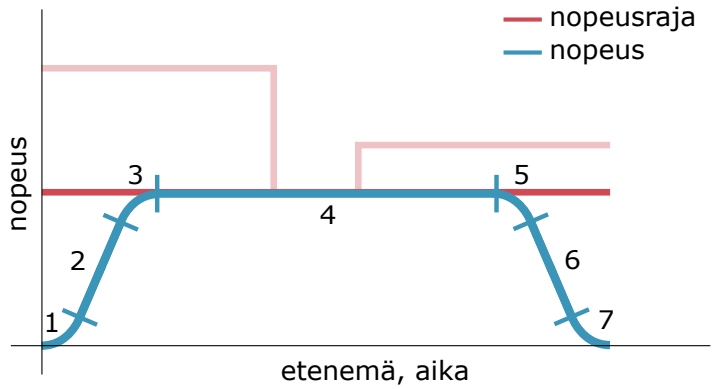
Kuvassa 3.6 on vuokaavio osakäyräalgoritmin toimintaperiaatteesta.



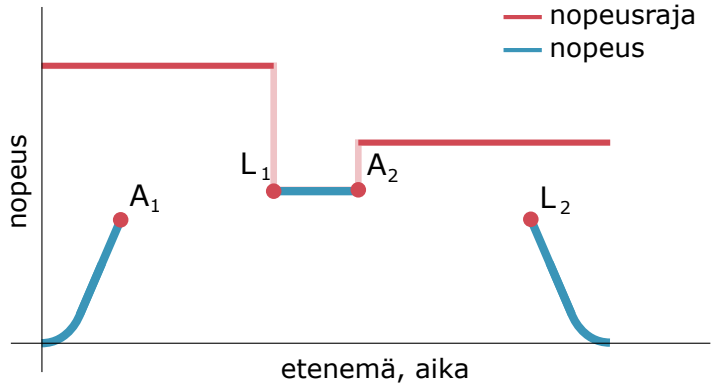
Kuva 3.6. *Vuokaavio osakäyräalgoritmista.*



Kuva 3.7. *Nopeusprofiilin aloitus- ja lopetuspiste ja alin nopeusraja.*



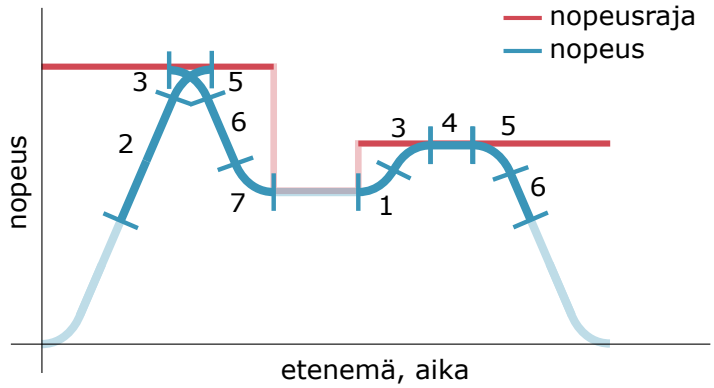
Kuva 3.8. *Alinta nopeusrajaa noudattavan nopeusprofiilin osakäyrät.*



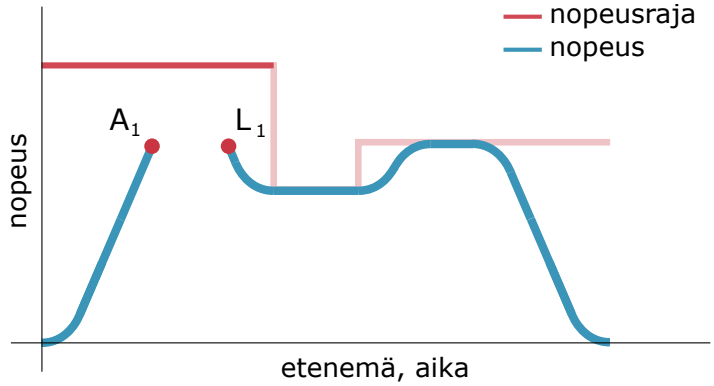
Kuva 3.9. *Seuraavaksi alimmat rajat ja pisteet, joiden välillä nopeutta voi vielä nostaa.*

Osakäyräalgoritmi rakentaa nopeusprofiilin seuraavissa vaiheissa:

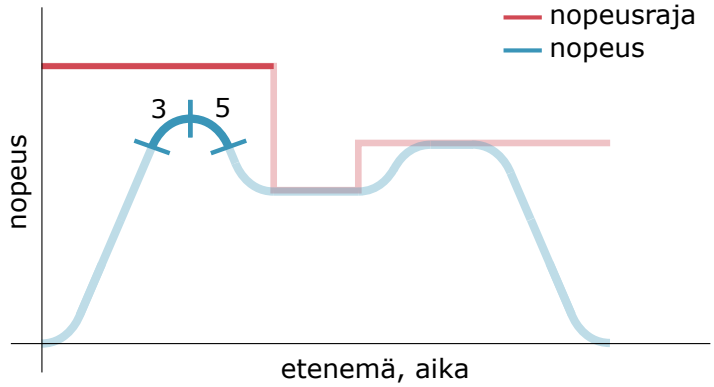
1. Etsitään nopeusprofiilin aloitus- ja lopetuspiste (A_1 ja L_1) ja niiden välinen alin nopeusraja (kuva 3.7).
2. Tehdään aloitus- ja lopetuspisteparin välille nopeusprofiili, joka nousee kiihtyvyyden ja nykäyksen rajoissa löydettyyn nopeusrajaan asti (kuva 3.8). Käytetyt osakäyrät on merkitty kuvaan.
3. Etsitään ne aloitus- ja lopetuspisteparit ((A_1, L_1) ja (A_2, L_2)), joiden välillä nopeutta voi vielä nostaa, ja niiden väliset seuraavaksi alimmat nopeusrajat (kuva 3.9).



Kuva 3.10. *Seuraaviin nopeusrajoihin yltävät osakäyrät.*



Kuva 3.11. *Aloitus- ja lopetuspiste, joiden välillä profiilia pitää korjata.*



Kuva 3.12. *Kelvollinen nopeusprofiilin korjaus.*

4. Tehdään aloitus- ja lopetuspisteparien välille nopeusprofiili, joka nousee löydettyihin rajoihin asti (kuva 3.10). Profiilista huomataan, että reitin alkuosan profiili ei ole kelvollinen: käyrät 3 ja 5 risteävät toisensa eli nopeus ei voi saavuttaa annettua nopeusrajaa. Siispä profiilia pitää korjata.
5. Etsitään aloitus- ja lopetuspisteet (A_1 ja L_1), joiden välille voidaan rakentaa kelvollinen nopeusprofiili (kuva 3.11).
6. Rakennetaan profiilin korjaus (kuva 3.12).

Osakäyräalgoritmin hyödyt:

- Alussa tehtyjä hitaampia profiileja voi käyttää, vaikka nopeampien profiilien tuottaminen epäonnistuisi.
- Osakäyriä on aikatasossa vain 7 erilaista.
- Osakäyrät on helppo sijoitella etenemätasossa.
- Osakäyrillä voidaan saada aikaan polynomeja vastaava tarkkuus.

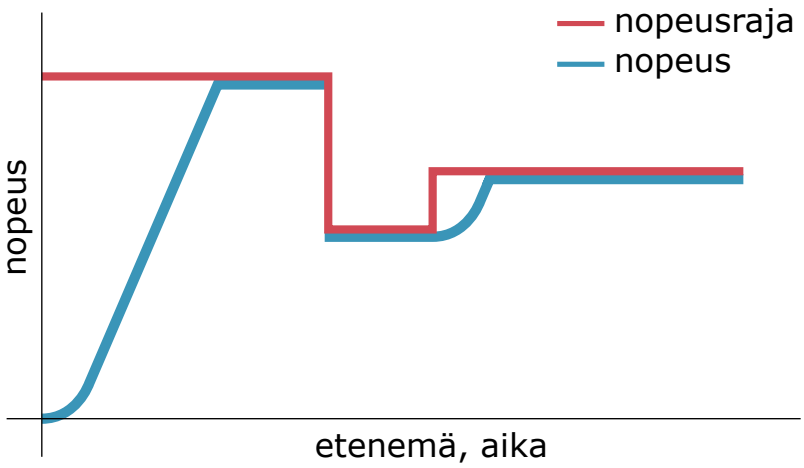
Osakäyräalgoritmin haitat:

- Algoritmi muistuttaa polynomialgoritmia, eli se on jokseenkin monimutkainen.
- Erityistapaukset on käsiteltävä erikseen, kuten esimerkiksi silloin, kun nopeuden alkuarvo on suurempi kuin matalin nopeusraja ja kiihtyvyyden alkuarvo ei ole nolla.
- Profiilista ei välttämättä tule optimaalisinta mahdollista, erityisesti tapauksessa, jossa nopeusraja muuttuu osakäyrien 3 tai 5 kohdalla – ellei sitä käsitellä erikoistapauksena.
- Etenemätasossa osakäyriä ei ole vain 7 erilaista, vaan käyrien 2, 3, 5 ja 6 (ks. kuva 3.5) muoto riippuu siitä, millä nopeudella robotti kulkee.
- Aikatasossa osakäyrien sijoittelu ei ole yksinkertaista; se joudutaan tekemään profiilin loppuosalle uudelleen aina, kun on onnistuneesti laskettu uusi nopeampi profiilin osa.

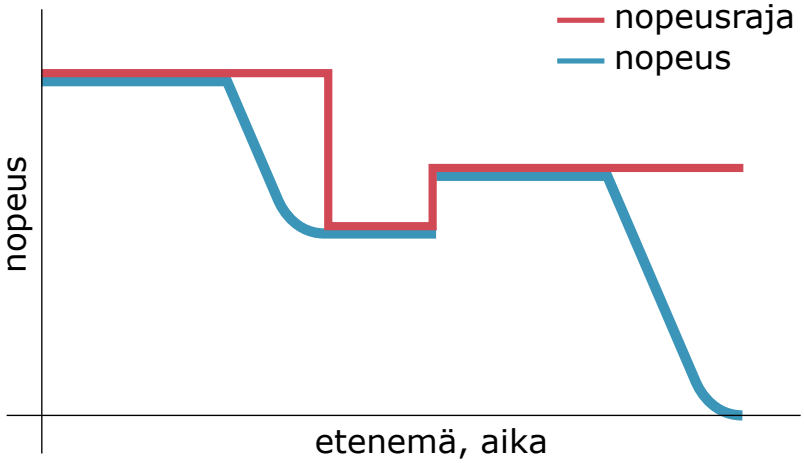
3.2.4 Optimistinen kaksisuuntainen algoritmi

Optimistinen kaksisuuntainen algoritmi generoi nopeusprofiilin tuottamalla ”optimistisen” nopeusprofiilin kaksi kertaa molempiin suuntiin.

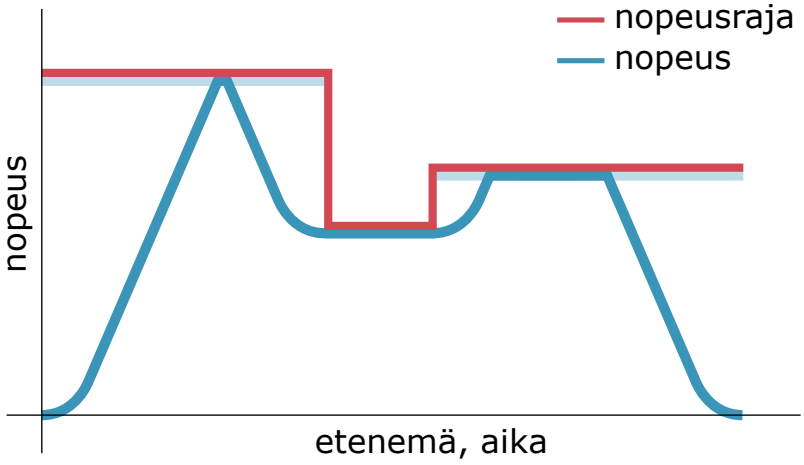
1. Nopeusprofiilin generointi aloitetaan reitin alusta niin, että nopeus nousee kiihtyvyyden ja nykäyksen rajoissa vastaan tulevaan nopeusrajaan asti.
2. Jatketaan nopeusprofiilin generoimista, kunnes seuraava nopeusraja tulee vastaan. Jos raja on pienempi, nopeus asetetaan välittömästi rajan mukaiseksi. Jos se on suurempi, nopeutta nostetaan kohdan 1 mukaisesti.
3. Jatketaan generointia, kunnes reitin loppu tulee vastaan. Tuotettua nopeusprofiilia on havainnollistettu kuvassa 3.13.
4. Generoidaan vastaavasti toinen nopeusprofiili mutta aloittaen reitin lopusta ja edeten reitin alkuun. Tähän suuntaan generoidessa pitää ottaa huomioon, että kiihtyvyyden maksimi- ja minimiarvot ovat vastakkaiset. Tuotettua nopeusprofiilia on havainnollistettu kuvassa 3.14.
5. Kun molemmat profiilit ovat valmiit, yhdistetään ne valitsemalla niistä kunkin etenemän kohdalla oleva alempi nopeus. Tuotettua nopeusprofiilia on havainnollistettu kuvassa 3.15.



Kuva 3.13. *Vasemmalta oikealle laadittu optimistinen profiili.*

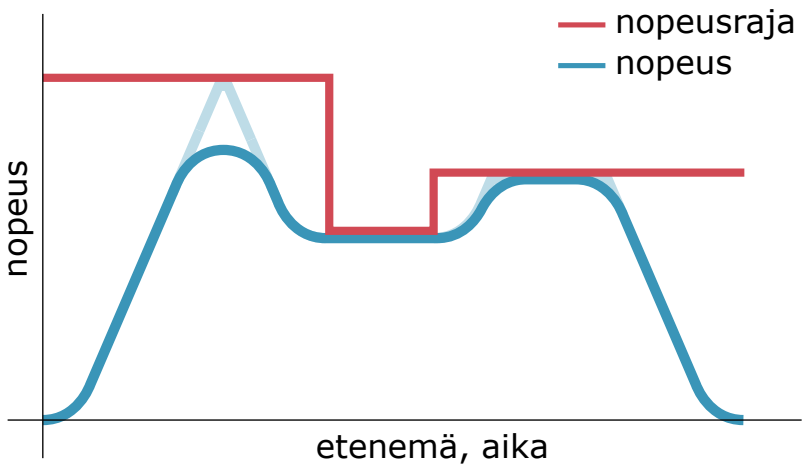


Kuva 3.14. *Oikealta vasemmalle laadittu optimistinen profiili.*



Kuva 3.15. *Optimististen profiilien yhdistelmä.*

6. Lopuksi profiili pitää vielä tasoittaa eli korjata mahdolliset siinä esiintyvät kiihtyvyyden äkkimuutokset, esimerkiksi samaan tapaan kuin osakäyräalgoritmin kuvissa 3.11 ja 3.12 on esitetty, jolloin tuloksena on kuvan 3.16 kaltainen käyrä.



Kuva 3.16. *Tasoitettu optimistinen profiili.*

Optimistisen kaksisuuntaisen algoritmin hyödyt:

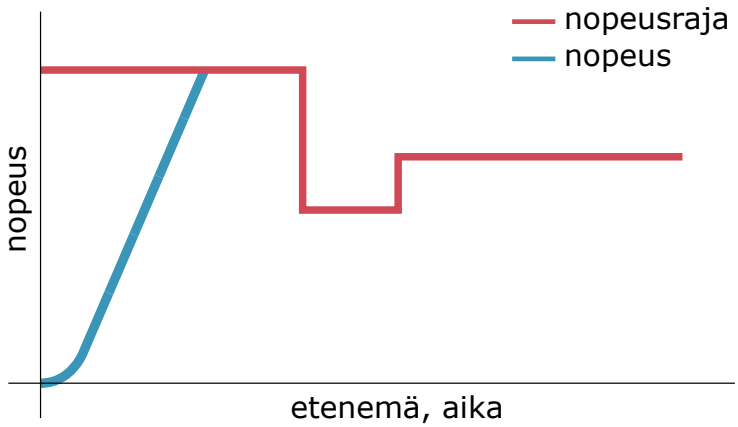
- Tasoitusta vaille oleva nopeusprofiili (kuva 3.15) on helppo rakentaa: ”yksisuuntaisten” nopeusprofiilien rakentaminen on yksinkertaista aikatasossa ja niiden yhdistäminen on yksinkertaista etenemätasossa.

Optimistisen kaksisuuntaisen algortimin haitat:

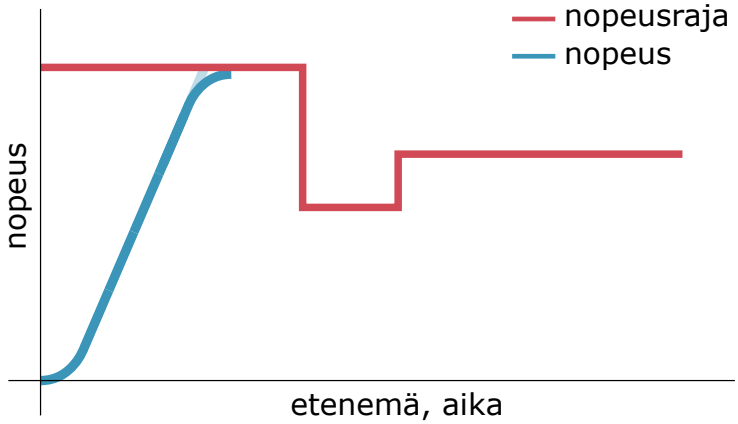
- Tasoitus saattaa vaatia paljonkin jälkimuokkausta ja laskemista.
- Jotkut erikoistapaukset saattavat aiheuttaa sen, että profiili ei ole kelvollinen (esim. porrasmaisesti muuttuva nopeusrajafunktio), jolloin erikoistapausten käsittelyä tarvitaan vielä enemmän.

3.2.5 Ryömintäalgoritmi

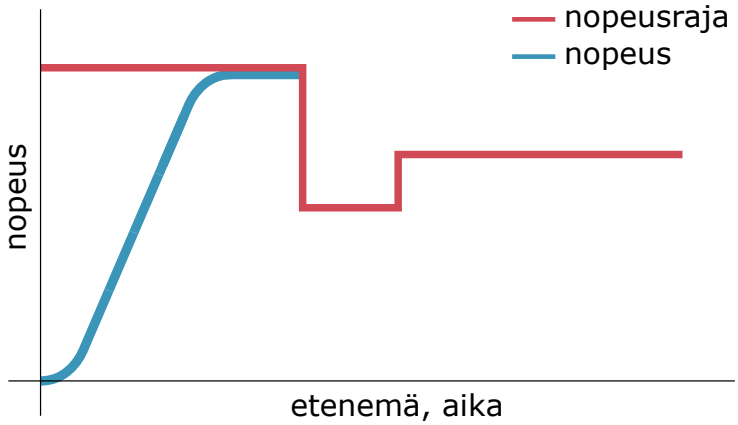
Ryömintäalgoritmi tuottaa aluksi nopeusprofiilia samaan tyyliin kuin optimistinen kaksisuuntainen algoritmi: se tuottaa nykäyksen ja kiihtyvyyden rajoissa olevia pisteitä, kunnes se osuu nopeusrajaan. Sitten se tasoittaa nopeudet niin, että nopeusrajaan osuminen on myös rajoitusten puitteissa. Profiilin tuottaminen etenee niin, että profiilia tuotetaan ”optimistisesti”, mutta aina kun seuraava raja kohdataan, profiili korjataan ennen kuin tuottaminen jatkuu. Algoritmi siis tavallaan ryömii eteenpäin ja välillä ryömii takaisin korjatakseen profiilia.



Kuva 3.17. *Ryömintäalgoritmi osuu nopeusrajaan.*

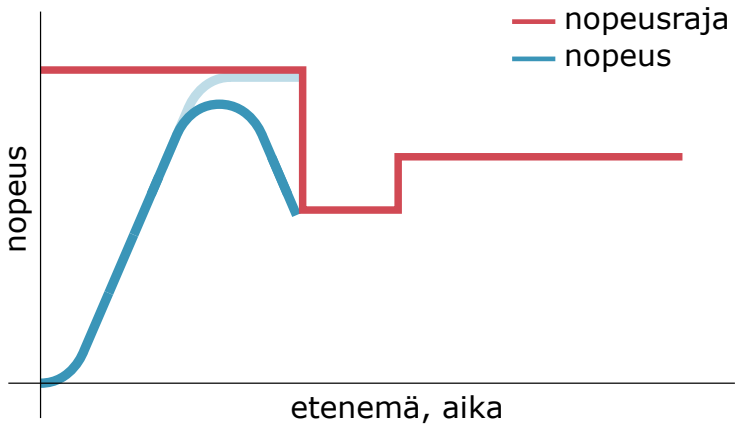


Kuva 3.18. *Ryömintäalgoritmi tasoittaa profiilin nopeusrajaan.*

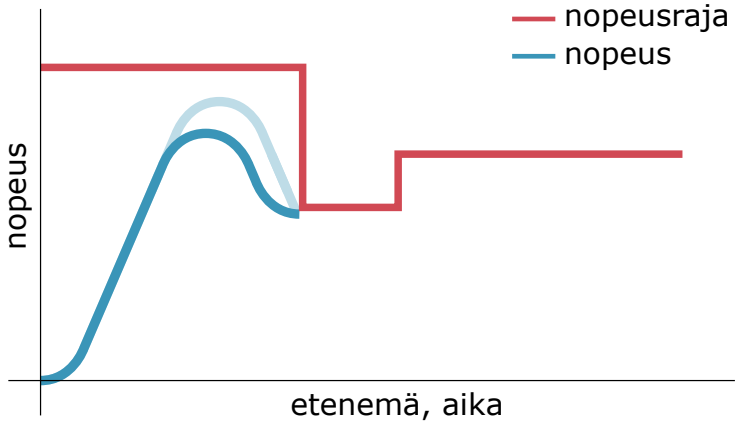


Kuva 3.19. *Ryömintäalgoritmi osuu seuraavaan nopeusrajaan.*

1. Nopeusprofiilin generointi aloitetaan reitin alusta niin, että nopeus nousee kiihtyvyyden ja nykäyksen rajoissa vastaan tulevaan nopeusrajaan asti (kuva 3.17).
2. Profiili korjataan niin, että se ei riko nykäyksen ja kiihtyvyyden rajoituksia osuessaan nopeusrajaan (kuva 3.18).
3. Generointia jatketaan seuraavaan nopeusrajaan (kuva 3.19).

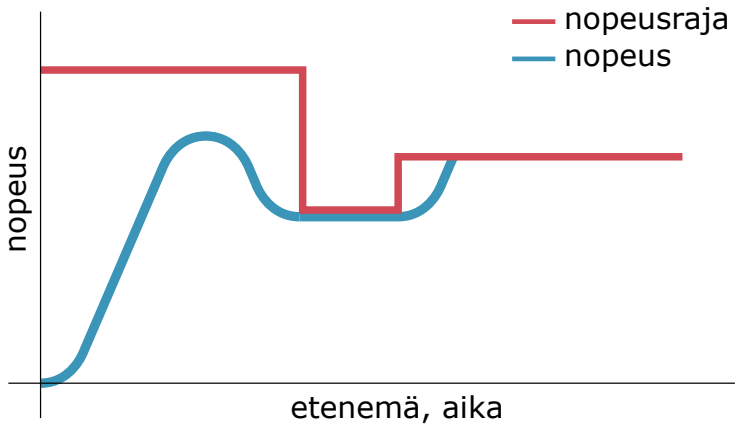


Kuva 3.20. *Ryömintäalgoritmi muokkaa profiilia yltääkseen seuraavaan nopeusrajaan.*

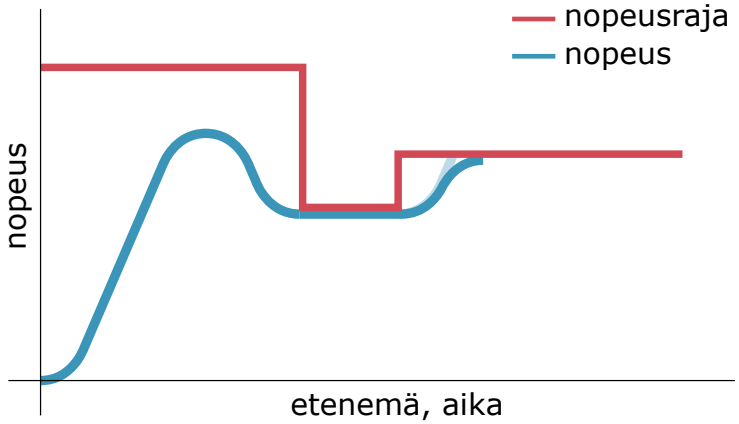


Kuva 3.21. *Ryömintäalgoritmi tasoittaa profiilin seuraavaan nopeusrajaan.*

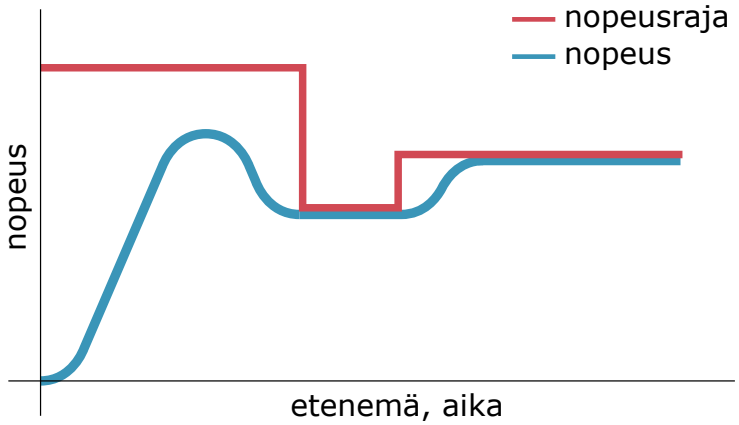
4. Seuraava nopeusraja on matalammalla kuin nykyinen. Kuten kuvasta 3.20 nähdään, profiilia pitää korjata pidemmältä matkalta, jotta se yltää rajaan rajoituksien puitteissa. Tässä kohtaa profiili ohittaisi nopeusrajan maksimihidastuvuudella. Korjattu profiili ei enää yllä aiempaan nopeusrajaan.
5. Kuvassa 3.21 profiilin hidastuvuus tasoitetaan matalampaan nopeusrajaan, koska muuten profiili menisi matalamman nopeusrajan alle ja joutuisi kiihdyttämään uudestaan, jotta se yltäisi kyseiseen nopeusrajaan.



Kuva 3.22. *Ryömintäalgoritmi jatkaa ja osuu seuraavaan nopeusrajaan.*

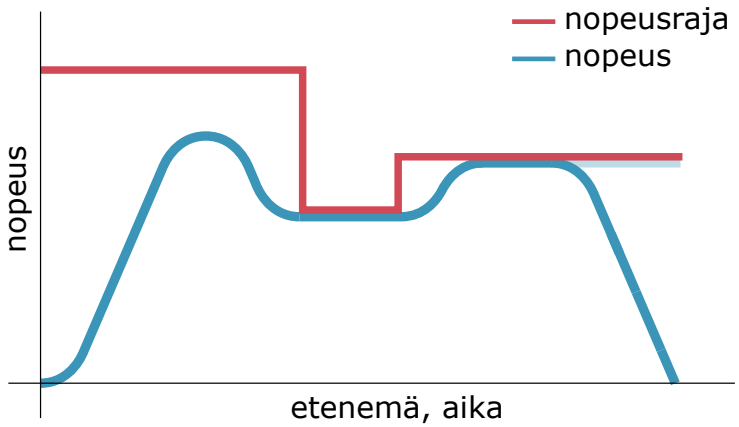


Kuva 3.23. *Ryömintäalgoritmi tasoittaa kiihtyvyyden muutoksen.*

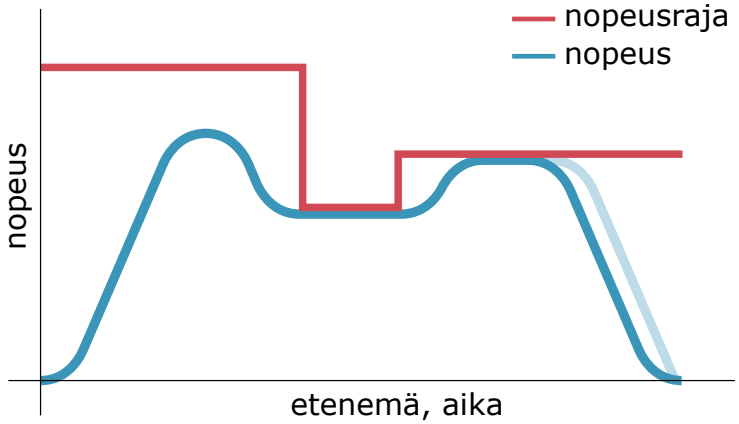


Kuva 3.24. *Ryömintäalgoritmi etenee reitin loppuun.*

6. Profiilin generoimista jatketaan optimistiseen tyyliin. Profiili ohittaa matalamman nopeusrajan ja osuu seuraavaan ylempään nopeusrajaan (kuva 3.22).
7. Profiili tasoitetaan nopeusrajaan (kuva 3.23).
8. Generointia jatketaan reitin loppuun (kuva 3.24).



Kuva 3.25. *Ryömintäalgoritmi korjaa profiilia reitin lopussa.*



Kuva 3.26. *Ryömintäalgoritmi tasoittaa profiilin reitin lopussa.*

9. Profiilia korjataan niin, että sen nopeus on reitin lopussa 0 m/s (kuva 3.25). Se rikkoo vielä nykäyksen ja kiihtyvyyden rajoituksia.
10. Lopuksi profiilin loppupää tasoitetaan nopeusrajaan niin, että se ei riko nykäyksen rajoituksia (kuva 3.26).

Ryömintäalgoritmin hyödyt:

- Profiilin generoiminen eteenpäin on helppoa.
- Profiilin korjaaminen tehdään yksinkertaisissa askelissa.
- Joitakin profiilin aiempia korjausaskelia voi käyttää uusina nopeusrajoina (esim. kuva 3.20), jolloin kaikkia korjauksessa tarvittavia arvoja ei tarvitse hakea tai laskea erikseen.

Ryömintäalgoritmin haitat:

- Profiilia pitää korjata useampaan kertaan.
- Nopeuksien vähentäminen eli profiilin korjaaminen saattaa vaatia paljon laskemista.
- Nopeuksien vähentäminen on erilaista kuin niiden nostaminen, eli algoritmilla pitäisi olla kaksi erilaista toimintamoodia.

3.2.6 Vakiokiihtyvyysalgoritmi

Vakiokiihtyvyyteen perustuvassa algoritmissa tehdään ensin nopeusprofiili, jossa kiihtyvyys on paloittain vakio, kuten kuvassa 2.4 (s. 7). Sen jälkeen profiilin pisteiden nopeuksia vähennetään, kunnes ne noudattavat nykäyksenkin rajoituksia.

Vakiokiihtyvyysalgoritmin hyödyt:

- Vakiokiihtyvyysprofiili on yksinkertainen rakentaa.
- Tilanteesta riippuen robotti saattaa liikkua tarpeeksi hyvin vakiokiihtyvyysprofiililla.

Vakiokiihtyvyysalgoritmin haitat:

- Nopeuksien vähentäminen saattaa vaatia paljonkin jälkimuokkausta ja laskemista.
- Menetelmä, jolla nopeuksia vähennetään, olisi oikeastaan oma algoritminsa. Käytännössä vakiokiihtyvyysprofiili toimisi ehkä parhaiten kuminauha-algoritmin lähtöarvoina.

3.3 Valittu algoritmi

Uuden nopeusprofiiligeneraattorin algoritmiksi valittiin kuminauha-algoritmi, jonka toimintaperiaate on kuvattu aliluvussa 3.2.2 (s. 22). Algoritmi valittiin yksinkertaisen toimintaperiaatteensa takia. Koska se vain kasvattaa tai vähentää profiilin pisteiden nopeutta, se teoriassa pystyy tekemään toimivan profiilin minkälaisessa tilanteessa vain ja erikoistapauksille ei tarvita erillistä tarkistusta tai käsittelyä. Näin ollen erikoistapauksiin liittyviä ongelmia tai puutteita ei jää piiloon generaattorin toteutukseen.

Kuminauha-algoritmin haittapuolet ovat myös hallittavissa: Nykyiset tietokoneet ovat tarpeeksi tehokkaita selvitäkseen iteroimisesta tarpeeksi nopeasti. Iterointiin voi toteuttaa muutosten tarkastelijan, joka lopettaa suorituksen, kun pisteiden nopeuksia ei enää pysty nostamaan. Iterointien määrä on myös helppo rajoittaa tiettyyn maksimiin, eli suoritus ei jää ikuiseen silmukkaan, vaikka algoritmi juuttuisikin jostain syystä tuottamaan muutamaa hädin tuskin erilaista profiilia.

Kun profiilia käsitellään aikatasossa, sen muuttuminen nopeammaksi aiheuttaa sen, että profiilin loppupään pisteet ovat jatkuvassa muutoksessa. Ongelman katsottiin olevan tarpeeksi lievä, sillä usein nopeusrajojen muutoksia ei ole reitillä niin paljoa, että siitä olisi merkittävää haittaa. Jos rajojen muutoksia on enemmän, ne voi tarvittaessa jakaa useampaan erään.

4 UUSI NOPEUSPROFIILIGENERAATTORI

Tässä luvussa kuvataan tarkemmin uuden nopeusprofiiligeneraattorin ominaisuuksia ja toimintaa. Ensin selostetaan sen vaatimukset, tekniset ratkaisut ja teoreettinen toimintaperiaate. Sen toimintaa käydään läpi useiden testitapausten avulla, joita havainnollistetaan Gnuplot-ohjelmalla tehdyillä kuvaajilla. Testien perusteella algoritmia muokataan paremmaksi.

4.1 Vaatimukset

Nopeusprofiiligeneraattorin pitää pystyä tuottamaan kaikille nopeusrajafunktioille edes jonkinlainen nopeusprofiili, jolla robotti voi kulkea reitin loppuun.

Nopeusprofiiligeneraattorilla ei ole kriittisiä reaaliaikavaatimuksia, sillä nopeusprofiili laaditaan aina etukäteen niin, että robotti voi ajaa etukäteen määrätyn matkan sallitulla reitillä ja pysähtyä turvallisesti. Uusia nopeusprofiileja generoidaan sen jälkeen hyvissä ajoin etukäteen, kun robotti on liikkumassa. Jos uusi nopeusprofiili on jostain syystä laiton, robotti käyttää vanhaa profiilia ja lopulta pysähtyy, jos se ei saa tehtyä kunnollista profiilia.

Vaikka reaaliaikavaatimuksia ei ole, generaattorin pitää tuottaa profiili kohtuullisessa ajassa. Lisäksi se ei saa käyttää profiilin tekemiseen kaikkea suoritinaikaa, sillä samassa järjestelmässä toimivalla laajemmalla automaatio-ohjelmistolla on muutakin tekemistä. Järjellinen aika riippuu robotin käyttötarkoituksesta, reitistä, laajemmasta automaatio-ohjelmistosta ja siitä, kuinka usein nopeusprofiileja pitää tuottaa. Tämän työn tapauksessa on riittävää, jos nopeusprofiili saadaan generoitua noin sekunnissa. Helpoimmillaan vaatimus voidaan toteuttaa rajoittamalla nopeuksien nostamiseen käytettyjen iterointien määrää niin, että etukäteen asetettua aikarajoitusta ei ylitetä. Aika-askelen pituuden kasvattaminen vähentää generointiin käytettyä aikaa, mutta profiilin tarkkuus kärsii hieman.

Generaattorin pitää pystyä tuottamaan profiileja reiteille, joiden pituus on muutamasta kymmenestä metristä pariin kilometriin. Pidemmät reitit suunnitellaan yleensä osissa. Nopeudeksi oletetaan maksimissaan 10 m/s, ja kiihtyvyyden ja nykäyksen rajoitukset ovat samaa kokoluokkaa. Reiteillä voi olla useita nopeusrajojen muutoksia, joista generaattorin pitää suoriutua.

Reittejä ja nopeusprofiileja pitää pystyä ketjuttamaan niin, että toinen reitti jatkaa siitä mihin ensimmäinen jäi. Näin ollen generaattorin pitää pystyä luomaan profiileja, joissa robotin nopeus tai kiihtyvyys on reitin alussa tai lopussa nollasta poikkeava.

Kuten aliluvussa 2.1 Nopeusprofiilin käyttötarkoitus (s. 3) mainitaan, nopeusprofiiligeneraattori ei ota huomioon tai käsittele yllättäviä tai hätätilanteita. Jos robotti havaitsee ennalta määrätyllä reitillään esteen, se ei noudata nopeusprofiilia vaan esimerkiksi pysähtyy mahdollisimman nopeasti (riippuen laajemmasta automaatio-ohjelmistosta ja robotin ohjausjärjestelmistä).

4.2 Tekniset ratkaisut

Seuraavissa aliluvuissa esitellään nopeusprofiiligeneraattorin uudelleenkirjoituksessa käytettyjä teknisiä valintoja.

4.2.1 Ohjelmointikieli

Toteutuskieleksi on valittu Haskell. Haskell [7] on vahvasti tyypitetty lähes täysin puhtaasti funktionaalinen ohjelmointikieli, joka auttaa koodin pitämisessä modulaarisena, selkeänä, yksinkertaisena ja turvallisena. Funktionaalisuus tarkoittaa sitä, että ohjelmassa ei ole sivuvaikutuksia, eli funktion (aliohjelman) paluuarvo riippuu vain sille annetuista parametreista eikä ohjelman sisäisestä tilasta. Oletusarvoisesti Haskell-ohjelmat evaluoidaan laiskasti, joten mitään turhaa ei lasketa. Koska Haskellissa ei käytetä osoittimia, niistä mahdollisesti aiheutuvia ongelmia ei ole.

Haskelliin on toteutettu Dimensional-paketti [8], joka auttaa käsittelemään mittayksiköllisiä arvoja. Sen avulla voidaan helpommin varmistua operaatioiden laillisuudesta, koska se suorittaa yksikkötarkastelun käännösaikaisesti: voidaan esimerkiksi määritellä SI-järjestelmän mukaiset nopeus- ja aikamuuttujat, ja jos niille lasketaan tulo, sen dimensio on etäisyys.

4.2.2 Rajapinta

Uusi nopeusprofiiligeneraattori on laajemmasta automaatio-ohjelmistosta selkeästi erillinen kokonaisuus. Laajempi automaatio-ohjelmisto lähettää nopeusprofiiligeneraattorille viestin, joka sisältää ainakin nopeusrajafunktion ja robotin kiihtyvyys- ja nykäysrajoitteet. Nopeusprofiiligeneraattori lähettää takaisin viestin, joka sisältää rajoituksia noudattavan näytteistetyn nopeusprofiilin, jonka jokaisessa pisteessä on arvot ajalle, etenemälle, nopeudelle, kiihtyvyydelle ja nykäykselle.

Nopeusprofiiligeneraattori kommunikoi laajemman automaatio-ohjelmiston kanssa käyttäen Google Protocol Buffersin (lyh. Protobuf) avulla luotuja viestejä. Protobuf [9] on työkalu, jolla toteutetaan rakenteellisen datan sarjallistaminen ja purkaminen (engl. data serialisation ja deserialisation). Se on toteutettu usealle eri ohjelmointikielelle, mm. C++:lle ja Haskellille. Toteutukset ovat ovat yhteensopivia keskenään, joten kun viestejä lähetetään eri moduulien, ohjelmien, laitteiden yms. välillä, datan muuntamista muodosta toiseen ei tarvitse tehdä käsin kullekin järjestelmän osapuolelle.

4.2.3 Diskreetit askeleet

Algoritmin toteutuksessa käytetään vakiokokoisia askeleita niin, että eri pisteiden ajan ja nopeuden arvot eroavat toisistaan tietyn askeleen monikerran verran. Myös kiihtyvyyden ja nykäyksen muutosaskeleet ovat vakiokokoisia. Näin ollen on yksinkertaista laskea muutokset nopeudessa. Näytejono voidaan rakentaa niin, että sen indeksinä toimii aika-askelen monikerta, jolloin sitä voi indeksoida kokonaisluvuilla.

Aika-askel ja nopeuden muutosaskel valittiin 2:n potensseiksi, sillä tietokone pystyy operoimaan niillä tekemättä pyöristysvirheitä. Esimerkiksi kymmenkantaista lukua 0,2 ei voi esittää tarkasti binäärijärjestelmässä ($0{,}2_{10} = 0{,}001100110011..._2 = 0{,}333..._{16}$). Ohjelman kääntäjä voi myös mahdollisesti optimoida 2:n potenssien operaatioita nopeammiksi (esim. bittisiirto vasemmalle tai oikealle).

4.3 Toiminnallisuus

Kuten aliluvussa 3.3 (s. 36) todetaan, uudeksi algoritmiksi valittiin kuminauha-algoritmi, jonka teoreettinen toimintaperiaate on kuvattu aliluvussa 3.2.2 (s. 22). Tässä aliluvussa kuvataan tarkemmin toteutetun algoritmin toiminnallisuutta ja eroja teoreettiseen periaatteeseen.

Aluksi uusi nopeusprofiiligeneraattori luo nopeusprofiilin, jonka matka-aika on 50 s ja jonka kaikki nopeusarvot ovat nollia. Nopeusprofiilia käsitellään aikatasossa ja sen aika-askeleeksi on valittu 0,125 s, eli pisteet ovat 0,125 sekunnin päässä toisistaan. 0,125 s on 2:n potenssi (2^{-3}) ja tarpeeksi pieni tarkkaa profiilia varten.

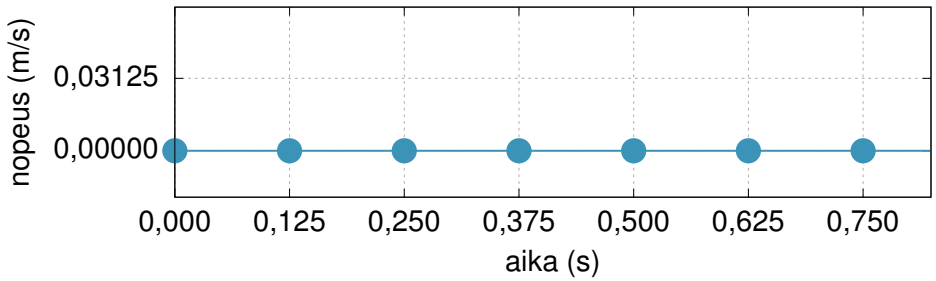
Seuraavaksi algoritmi alkaa lisätä pisteiden nopeuksia. Myös nopeuden lisääminen tehdään diskreeteissä askelissa. Nopeusaskeleeksi on valittu 0,03125 s (2^{-5}).

Algoritmi käyttää viiden profiilipisteen kokoista tarkasteluikkunaa selvittääkseen ne pisteet, joiden nopeutta voi lisätä. Ikkunan keskimmäinen piste tarkastaa ensin nopeusrajoista, pitäisikö sen lisätä vai vähentää nopeuttaan vai olla muuttamatta sitä. Jos sen pitäisi muuttaa nopeuttaan, se kysyy kahdelta edelliseltä ja kahdelta seuraavalta pisteeltä, onko niin sopivaa tehdä. Muilla pisteillä on kolme vastausvaihtoehtoa:

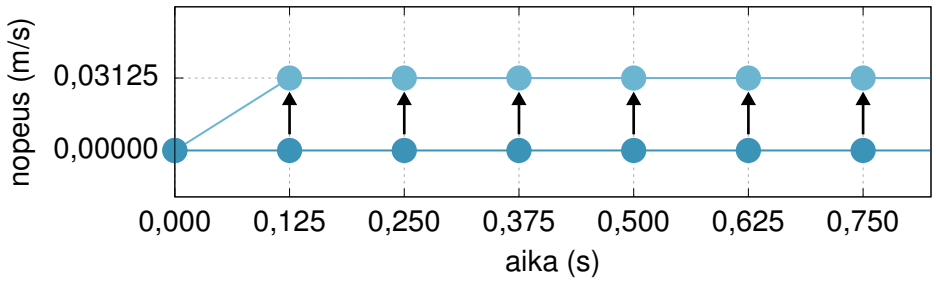
1. Ei.
2. Kyllä.
3. Kyllä, jos piste voi itse lisätä/vähentää nopeuttaan. Sitten piste kysyy muilta ympärillään olevilta pisteiltä, onko sen sopivaa lisätä/vähentää nopeuttaan.

4.3.1 Esimerkki

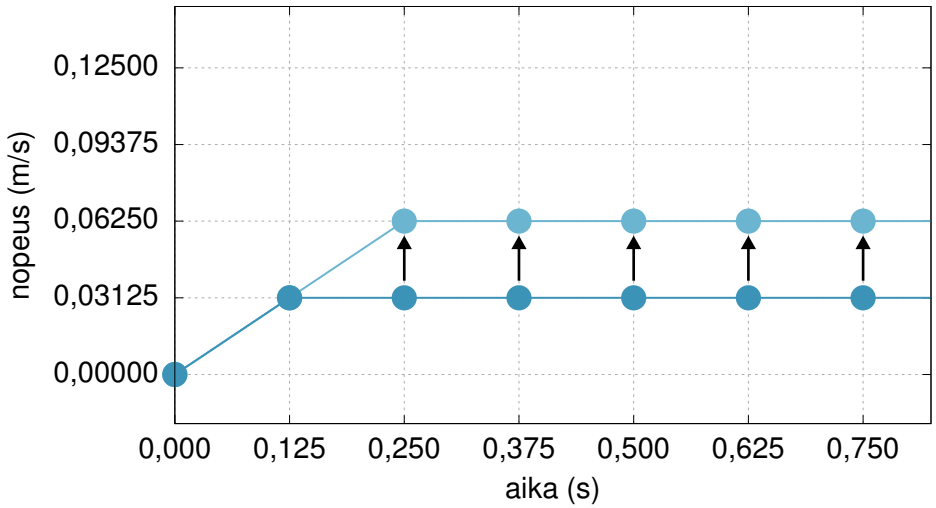
Kuvissa 4.1, 4.2, 4.3 ja 4.4 on havainnollistettu pisteiden nopeuden lisäämistä reitin alussa ensimmäisten iteraatioiden aikana. Nopeusraja on niin korkealla, että se ei esimerkeissä rajoita profiilin pisteitä.



Kuva 4.1. *Lähtötilanne: nopeus on nolla.*



Kuva 4.2. *Ensimmäinen iteraatio: ensimmäinen piste on kiinnitetty, muita voi nostaa.*

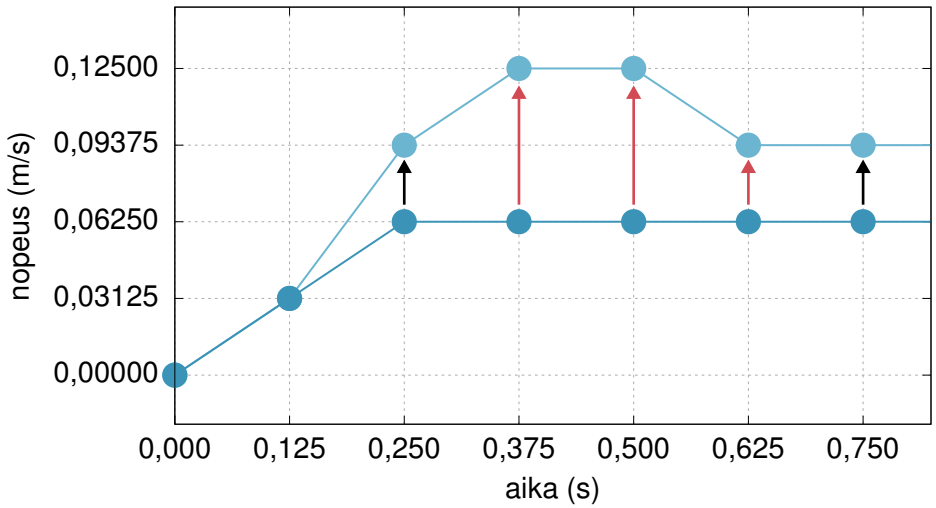


Kuva 4.3. *Toinen iteraatio: toinen piste on kiinnitetty, seuraavia voi nostaa.*

Kuvassa 4.1 on alkutilanne, jossa profiilin kaikkien pisteiden nopeus on nolla. Pisteistä on kuvattu seitsemän ensimmäistä.

Kuvassa 4.2 on ensimmäinen iteraatio: Ensimmäisen pisteen (t = 0,000 s) nopeus pysyy nollana, koska robotti on alussa pysähtynyt. Toisen ja myöhempien pisteiden nopeutta voidaan lisätä.

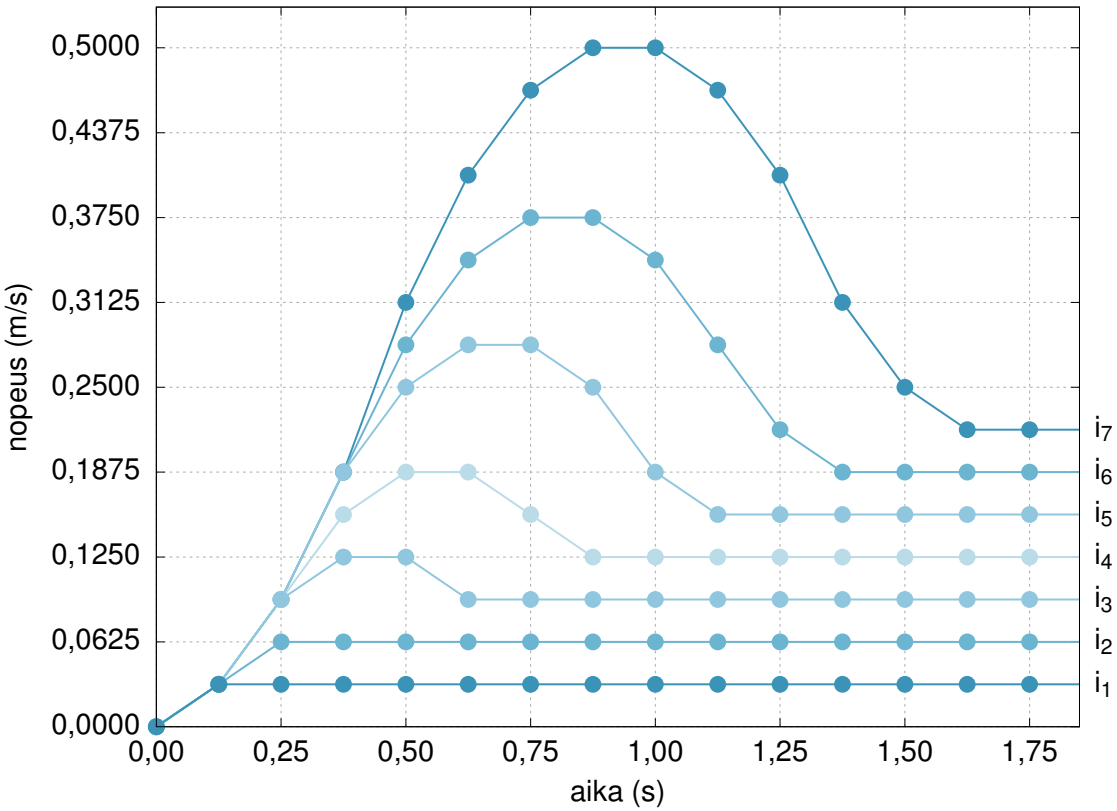
Kuvassa 4.3 on toinen iteraatio: Nyt toisen pisteen (t = 0,125 s) nopeutta ei voida lisätä, sillä jos niin tehtäisiin, se rikkoisi kiihtyvyyden tai nykäyksen rajoitteita. Kolmannen ja sen jälkeen tulevien pisteiden nopeutta on edelleen mahdollista lisätä.



Kuva 4.4. *Kolmas iteraatio: kolmannen pisteen nosto aiheuttaa nostojen ketjureaktion.*

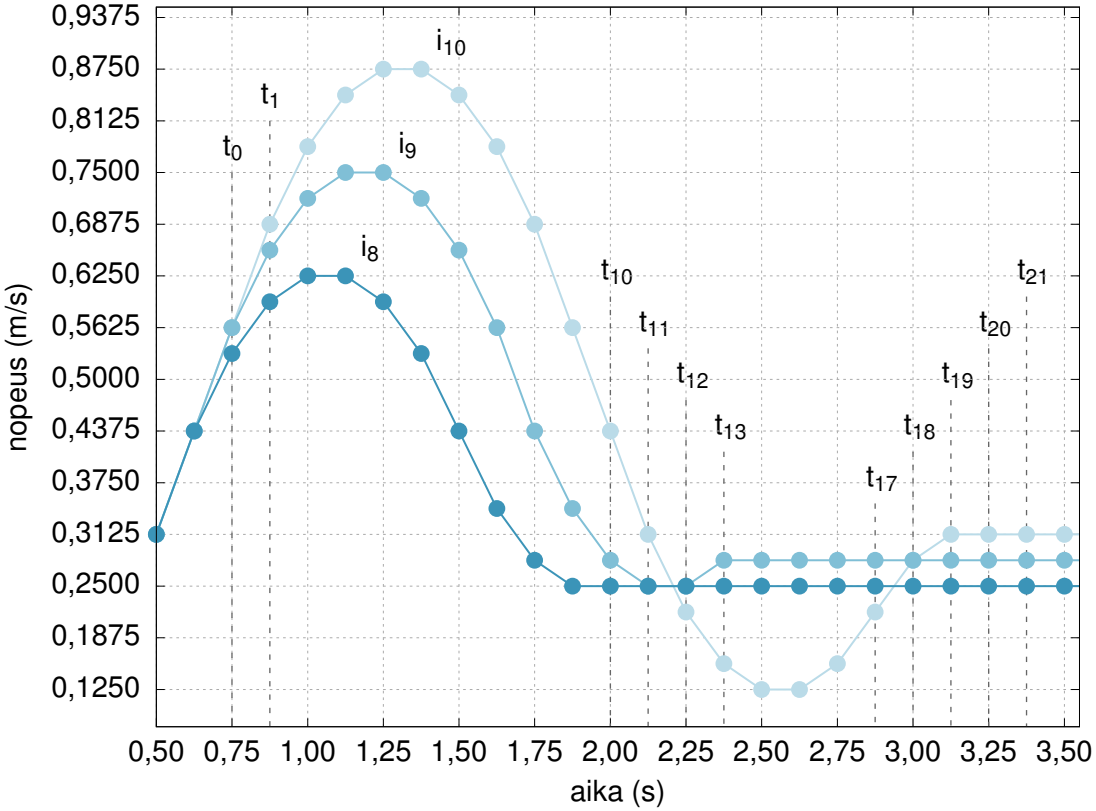
Kuvassa 4.4 on kolmas iteraatio: Kolmannen pisteen (t = 0,250 s) nopeutta voidaan lisätä, mutta jotta se ei riko kiihtyvyys- ja nykäysrajoitteita, neljännen pisteen nopeutta pitäisi lisätä kahdella askeleella. Kuvaan on merkitty kyselemisen tuottamat nostot punaisilla nuolilla mustien sijaan. Kyseleminen etenee seuraavasti:

1. Kolmas piste kysyy ensimmäiseltä ja toiselta pisteeltä, onko kolmannen sopivaa lisätä nopeuttaan. Ne vastaavat kyllä.
2. Kolmas piste kysyy neljänneltä, onko kolmannen sopivaa lisätä nopeuttaan yksi askel.
3. Neljäs piste laskee, että jos kolmas piste lisää nopeuttaan, neljännen pitäisi lisätä nopeuttaan kahdella askeleella, jotta se ei riko kiihtyvyys- ja nykäysrajoja.
4. Neljäs piste kysyy viidenneltä, onko neljännen sopivaa lisätä nopeuttaan kaksi askelta.
5. Viides piste laskee, että jos neljäs piste lisää nopeuttaan, viidennen pitäisi myös lisätä nopeuttaan kahdella askeleella, jotta se ei riko kiihtyvyys- ja nykäysrajoja.
6. Viides piste kysyy kuudennelta, onko viidennen sopivaa lisätä nopeuttaan kaksi askelta.
7. Kuudes piste laskee, että jos viides piste lisää nopeuttaan, kuudennen pitäisi lisätä nopeuttaan yhdellä askeleella, jotta se ei riko kiihtyvyys- ja nykäysrajoja.
8. Kuudes piste kysyy seitsemänneltä ja kahdeksannelta pisteeltä, onko kuudennen sopivaa lisätä nopeuttaan yksi askel. Ne vastaavat kyllä.
9. Kuudes piste tekee muutoksensa ja välittää taaksepäin tiedon siitä, että muutos on sallittu.
10. Koska pisteet kuudenteen pisteeseen asti ovat jo muuttaneet nopeuttaan tällä iteraatiolla, tarkastelu jatkuu seitsemännestä pisteestä. Se lisää nopeuttaan yhden askeleen, samoin kaikki muutkin jäljellä olevat pisteet.



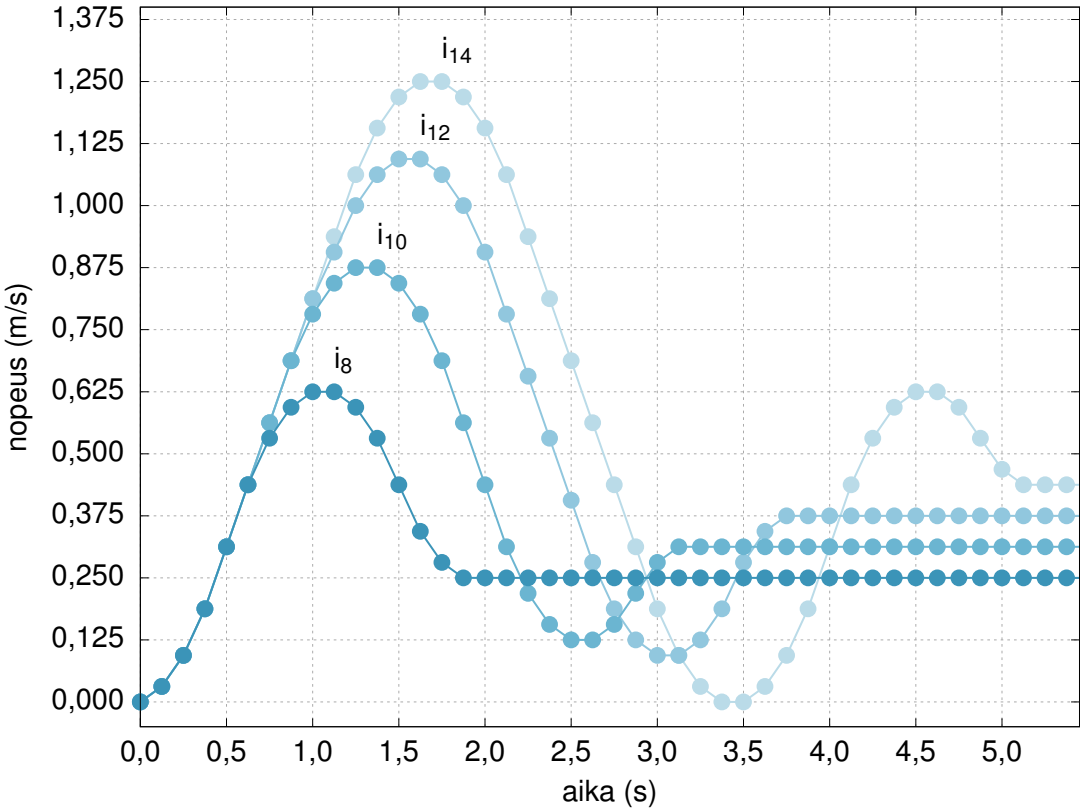
Kuva 4.5. *Nopeuden muutokset etenevät kasvavana mäkenä.*

Kuvassa 4.5 on seitsemän ensimmäistä iteraatiota. Pisteiden nopeuden lisäykset lähtevät etenemään kasvavana mäkenä. Algoritmi muuttaa kunkin pisteen nopeutta kerran yhden iteraation aikana.



Kuva 4.6. *Aallon synty.*

Kuvassa 4.6 ovat iteraatiot 8, 9 ja 10. Niistä näkyy, miten algoritmi tietyssä tilanteessa jopa vähentää pisteiden nopeuksia. Iteraation 9 alku on samanlainen kuin iteraatiolla 8. Ajanhetkellä t_0 iteraation 8 piste voi lisätä nopeuttaan yhdellä askeleella. Kyseleminen etenee ajanhetkeen t_{12}. Pisteet kohdissa t_{11} ja t_{12} vastaavat, että niiden ei tarvitse lisätä nopeuttaan, jotta nykäys- ja kiihtyvyysrajoitukset pysyvät laillisina. Näin ollen pisteet aikavälillä $[t_0, t_{10}]$ toteuttavat muutoksensa ja tarkastelu jatkuu pisteestä ajanhetkessä t_{11}. Piste kohdassa t_{11} kysyy edelliseltä pisteeltä, onko sen sopivaa lisätä nopeuttaan. Piste kohdassa t_{10} vastaa, että ei ole, sillä jos niin tehtäisiin, aikavälillä $[t_{10}, t_{11}]$ rikottaisiin kiihtyvyyden tai nykäyksen rajoituksia. Pisteelle kohdassa t_{12} käy samoin. Piste kohdassa t_{13} pystyy lisämään nopeuttaan yhden askeleen vaikuttamatta muihin, samoin pisteet sen jälkeen. Iteraatiolla 10 nopeuden väheneminen näkyy entistä selvemmin. Kun iteraation 9 piste kohdassa t_1 lisää nopeuttaan yhden askeleen, piste kohdassa t_{12} joutuu vähentämään nopeuttaan, jotta piste kohdassa t_{11} olisi laillinen. Tästä muutoksesta seuraa, että myös pisteiden aikavälillä $[t_{13}, t_{17}]$ on vähennettävä nopeuksiaan. Pisteen kohdassa t_{18} ei tarvitse muuttaa nopeuttaan, mutta pisteiden kohdissa t_{19} ja t_{20} pitää lisätä nopeuttaan yhdellä askeleella. Muutoksista muodostuu aalto, ja tarkastelu jatkuu tyypilliseen tapaan pisteestä kohdassa t_{21}, joka lisää nopeuttaan yhdellä askeleella. Nopeuden vähenemistä tapahtuu siis silloin, kun kiihtyvyyden ja nykäyksen rajoitukset eivät muuten sallisi jonkin aiemman pisteen nopeuden lisäämistä.



Kuva 4.7. *Nopeuden muutokset etenevät aaltoina.*

Kuvassa 4.7 ovat iteraatiot 8, 10, 12 ja 14. Käyristä huomataan, että myöhemmillä iteraatioilla profiilin alun pisteiden nopeuksia lisätään edelleen siitäkin huolimatta, että myöhempien pisteiden nopeuksia joudutaan vähentämään tai että niiden nopeus vähenee nollaan asti. Algoritmi siis pyrkii kasvattamaan nopeuden mahdollisimman suureksi ja pitämään sen mahdollisimman suurena mahdollisimman pitkään, mikä aiheuttaa nopeusprofiilin aaltoilua.

4.4 Testaus

Uudessa nopeusprofiiligeneraattorissa on toiminnallisia parametreja. Se laskee parametrien perusteella nopeusprofiilin eli tuottaa näytejonon, joka sisältää kullekin pisteelle indeksin ja nopeuden. Aika, etenemä, kiihtyvyys ja nykäys lasketaan erikseen indeksin, aika-askelen ja nopeuden perusteella. Näytejonon perusteella voidaan tuottaa esimerkiksi tiedosto, jossa on arvot jokaiselle halutulle suureelle, ja datan voi esittää kuvaajana. Tässä aliluvussa algoritmin toiminta ja nopeusprofiilin laillisuus tarkistetaan kuvaajista.

Seuraavia parametreja voi muuttaa ilman ongelmia:

- Generaattorille voi antaa erilaiset nopeusrajat.
- Profiilin matka-aikaa voi muuttaa.
- Iterointien määrää voi muuttaa.

Seuraavat muutokset ovat mahdollisia:

- Aika-askelta voi muuttaa.
- Nopeusaskelta voi muuttaa.
- Maksimikiihtyvyyttä voi muuttaa.
- Maksiminykäystä voi muuttaa.

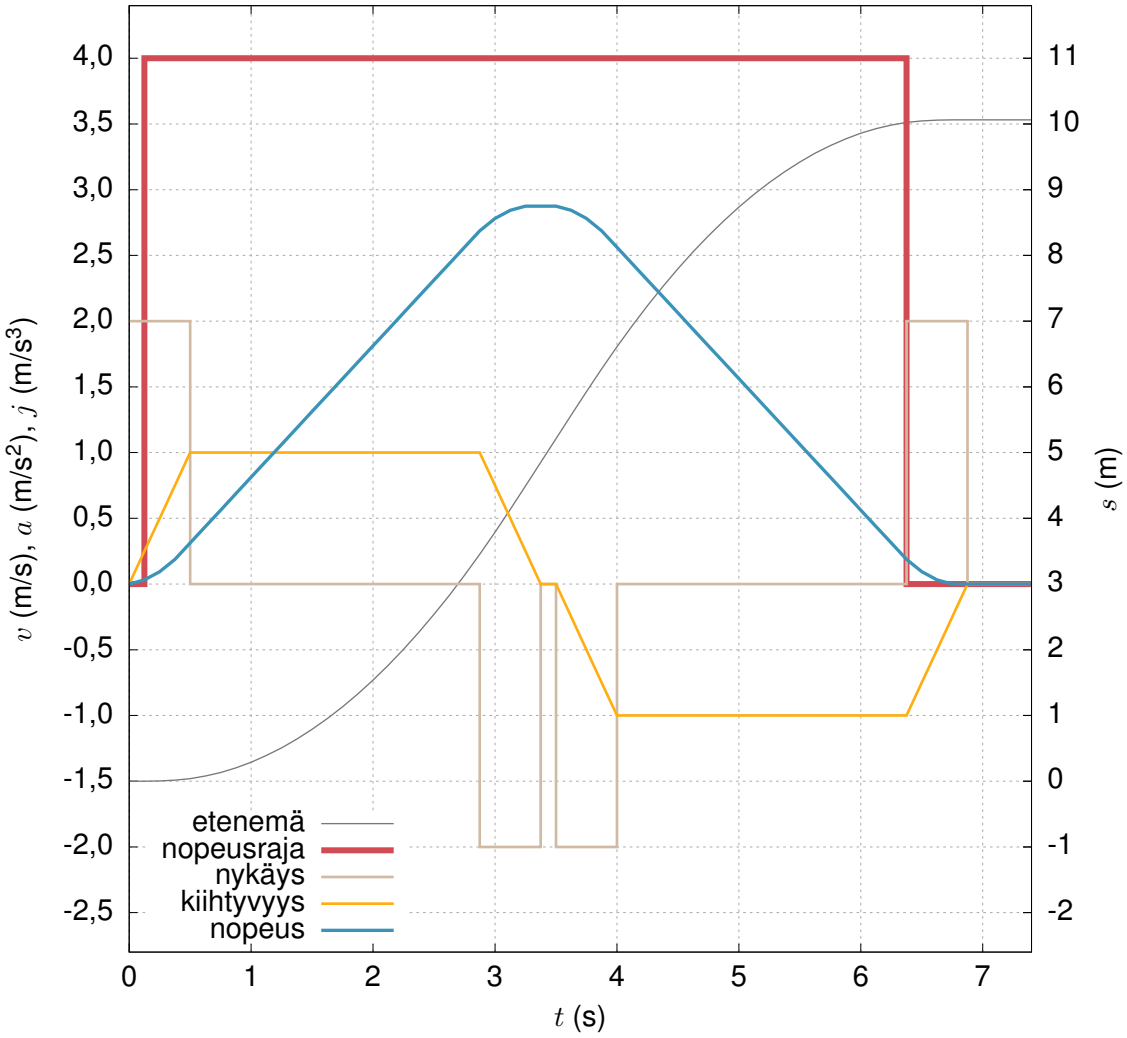
Käytännössä yo. arvot pitää valita tarkkaan, sillä jos askelten tai maksimien arvot ovat vääränlaisia, algoritmi ei kasvata profiilin pisteiden nopeuksia lainkaan. Esimerkiksi jos nykäyksen maksimi ja aika-askel ovat hyvin pieniä ja nopeusaskel suuri, nykäyksen maksimi ei riitä nostamaan nopeutta yhtä nopeusaskelta.

Seuraavissa aliluvuissa on esimerkkejä, joissa uutta nopeusprofiiligeneraattoria testataan antamalla sille syötteeksi erilaisia nopeusrajafunktioita ja osittain erilaisia nykäyksen ja kiihtyvyyden rajoituksia. Viimeisenä, aliluvussa 4.4.7 (s. 56), on yhteenveto testauksesta.

4.4.1 Yksinkertainen nopeusprofiili

Kuvassa 4.8 on esimerkki uuden generaattorin tuottamasta yksinkertaisesta nopeusprofiilista ajan (t) funktiona. Pysty-akselilla on kuvattu nopeuden (v), kiihtyvyyden (a) ja nykäyksen (j) lisäksi etenemä (s), jonka arvot on merkitty kuvaajan oikealle puolelle. Nopeusrajafunktio on 4,0 m/s etenemävälillä [0,0 m, 10 m], kiihtyvyysrajoitus on 1,0 m/s^2 ja nykäysrajoitus 2,0 m/s^3.

Algoritmi tuottaa järkevän profiilin, joka noudattaa kiihtyvyys- ja nykäysrajoituksia ja pyrkii saavuttamaan asetetun nopeusrajan. Profiili ei kuitenkaan ole täysin halutun kaltainen, sillä sitä noudattamalla robotti ei pysähtyisi reitin loppuun (s = 10 m). Sen sijaan robotti alkaisi laskea hidastuvuuttaan vasta lopun saavutettuaan ja pysähtyisi, kun etenemä on 6,3 cm yli halutun loppukohdan.

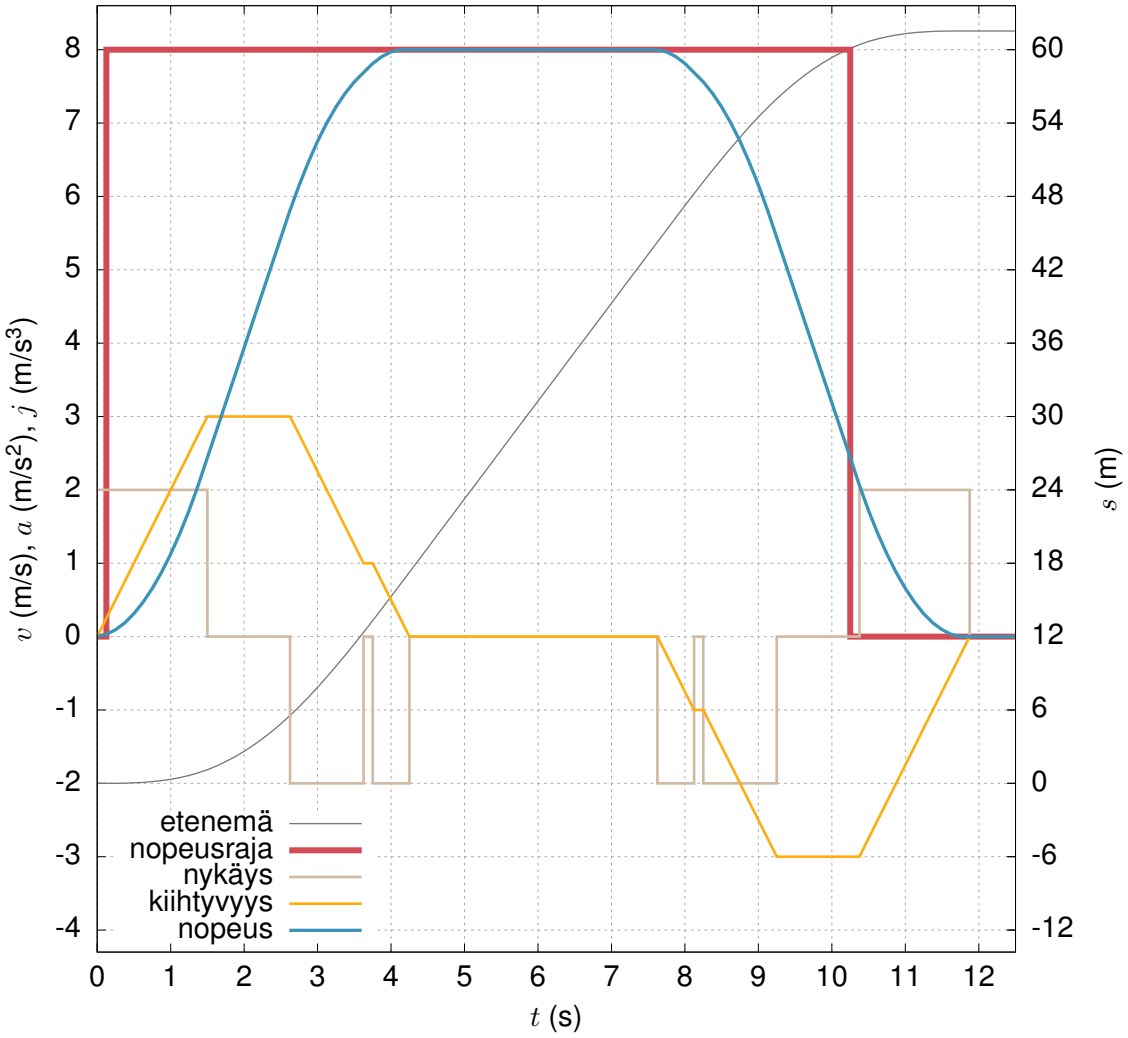


Kuva 4.8. *Yksinkertainen nopeusprofiili.*

4.4.2 Nopeampi yksinkertainen nopeusprofiili

Kuvassa 4.9 on edellisen esimerkin kaltainen tapaus, mutta nopeusraja on 8,0 m/s etenemävälillä [0,0 m, 60 m] ja kiihtyvyysrajoitus 3,0 m/s^2. Profiili on taas järkevä ja kiihtyminen hidastuu sopivasti ennen nopeusrajan saavuttamista, minkä jälkeen nopeus jatkaa nopeusrajan mukaisena, kunnes on aika hidastaa. Kuvasta näkyy, että liian pitkälle ajamisen ongelma on pahempi kuin edellisessä esimerkissä: robotilla olisi edelleen 2,4 m/s:n nopeus reitin loppupisteessä (s = 60 m) ja se pysähtyisi vasta etenemän ollessa 61,5 m.

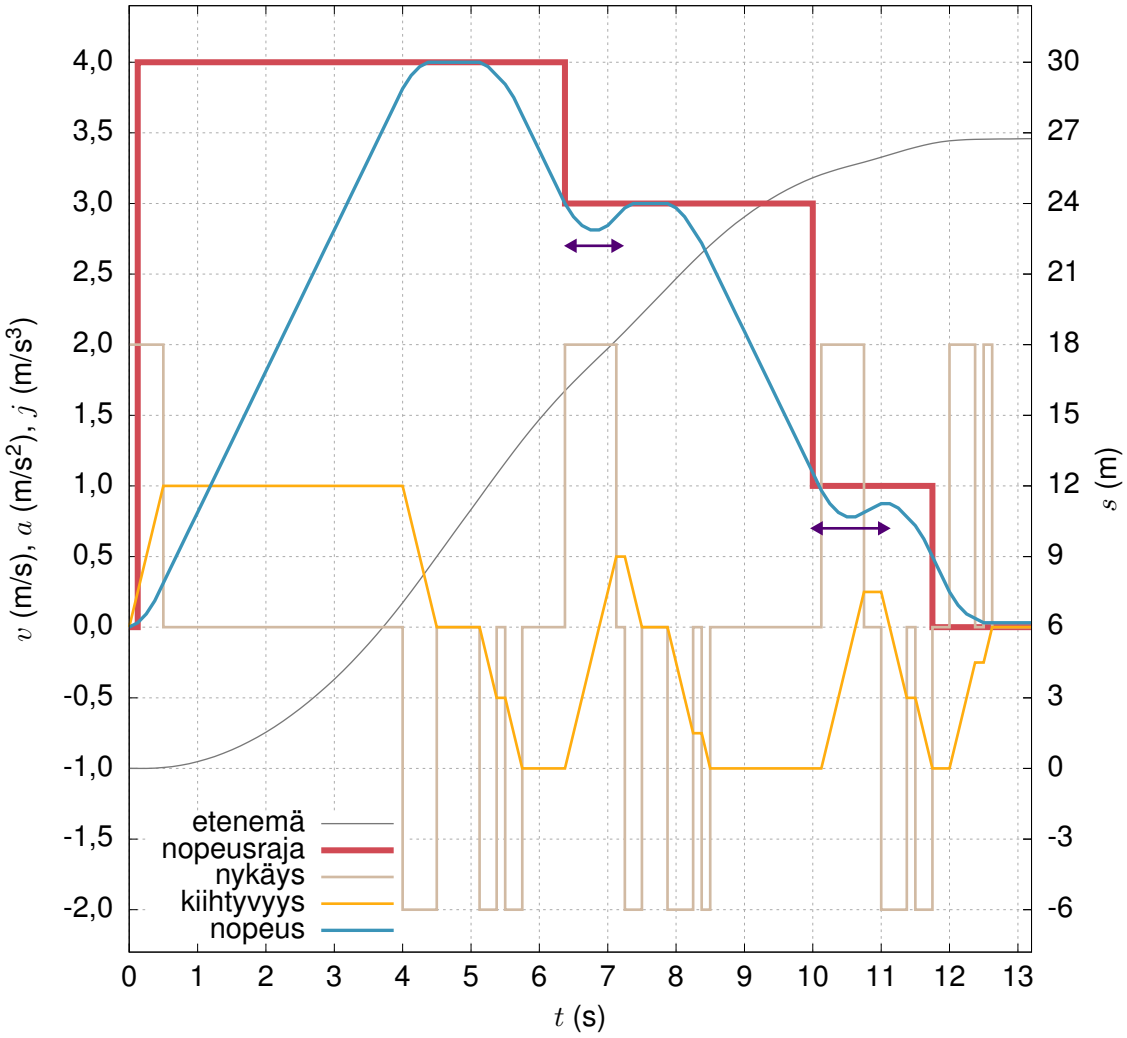
Tämän aliluvun kuvien perusteella algoritmi pitää hidastuvuuden maksimissaan reitin loppuun saakka. Vasta kun loppu on saavutettu, algoritmi asettaa nykäyksen maksimiin ja hidastuvuus vähenee nollaan.



Kuva 4.9. *Nopeampi yksinkertainen nopeusprofiili.*

4.4.3 Kuoppia nopeusprofiilissa

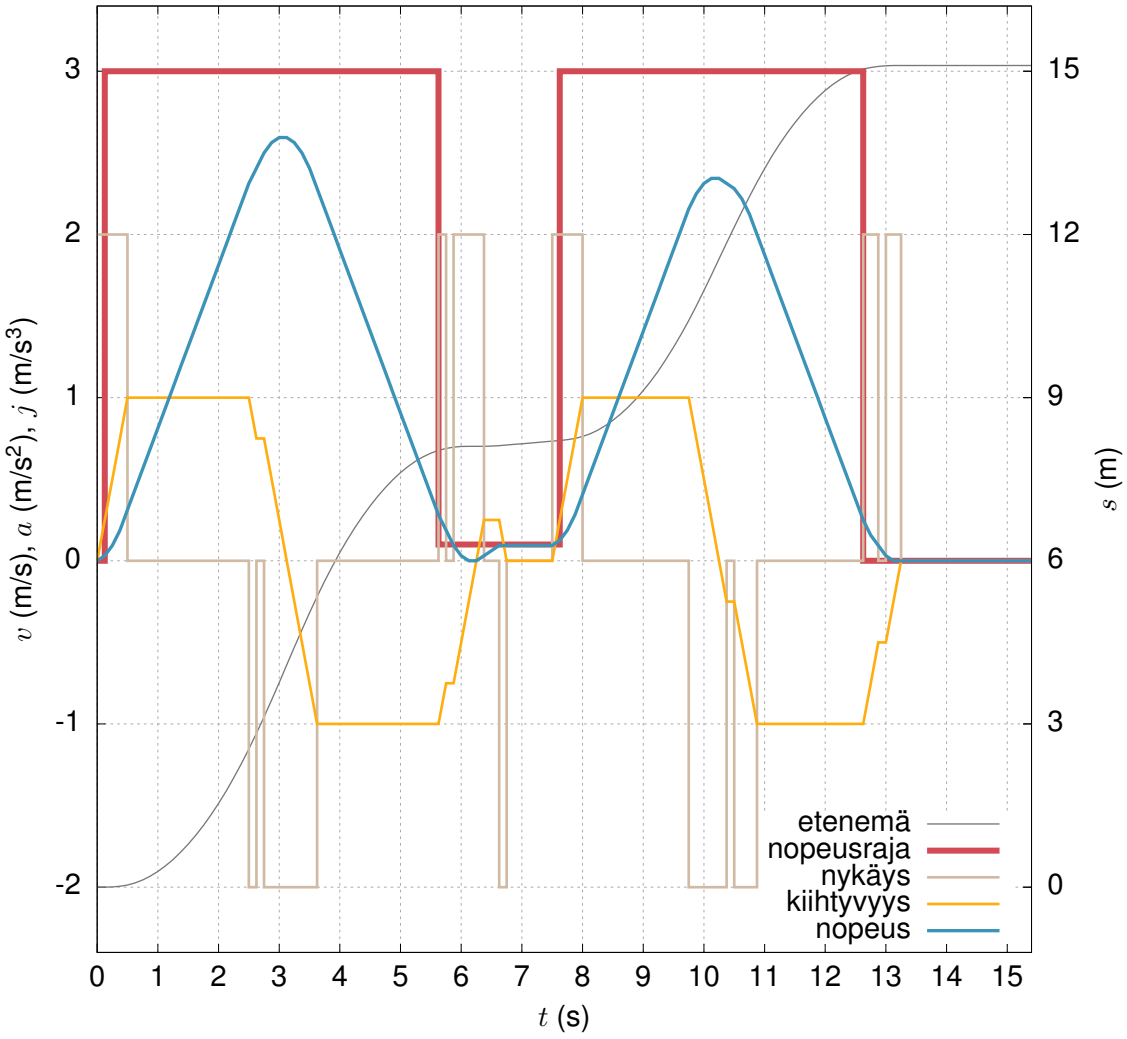
Kuvassa 4.10 on esimerkki nopeusprofiilista, joka yrittää noudattaa nopeusrajafunktiota, joka muuttuu kaksi kertaa matalammaksi reitin aikana. Alussa nopeusraja on 4,0 m/s, etenemän ollessa 16 m (ajanhetki 6,4 s) se vähenee arvoon 3,0 m/s ja etenemän ollessa 25 m (ajanhetki 10,0 s) se vähenee arvoon 1,0 m/s. Robotin pitäisi pysähtyä edettyään 26,5 m. Kuvasta näkyy, miten nopeuden pitäminen mahdollisimman suurena mahdollisimman pitkään toteutuu käytännössä: seuraava nopeusraja ohitetaan maksimihidastuvuudella rajaa hipoen, joten nopeuteen tulee pieni ”kuoppa”, kun robotti alkaa vähentää hidastuvuutta vasta rajan ohituksen jälkeen ja joutuu sitten kiihdyttämään hieman, jotta se saavuttaa alemman rajan. Kuopat esiintyvät aikaväleillä [6,4 s, 7,2 s] ja [10,0 s, 11,1 s] ja ne on merkitty kuvaan nuolilla. Myös tässä tapauksessa algoritmi tuottaa profiilin, jolla robotti kulkee liian pitkälle: etenemä 26,5 m saavutetaan ajanhetkellä 11,75 s, mutta robotti jatkaisi hidastamista ja liikkumista ajanhetkeen 12,5 s, jolloin sen etenemä on 26,73 m. Lisäksi profiilin nopeus on hetken 12,5 s jälkeen edelleen yhden nopeusaskeleen suuruinen, joten se jatkaisi hidasta liikettään.



Kuva 4.10. *Kuoppia nopeusprofiilissa.*

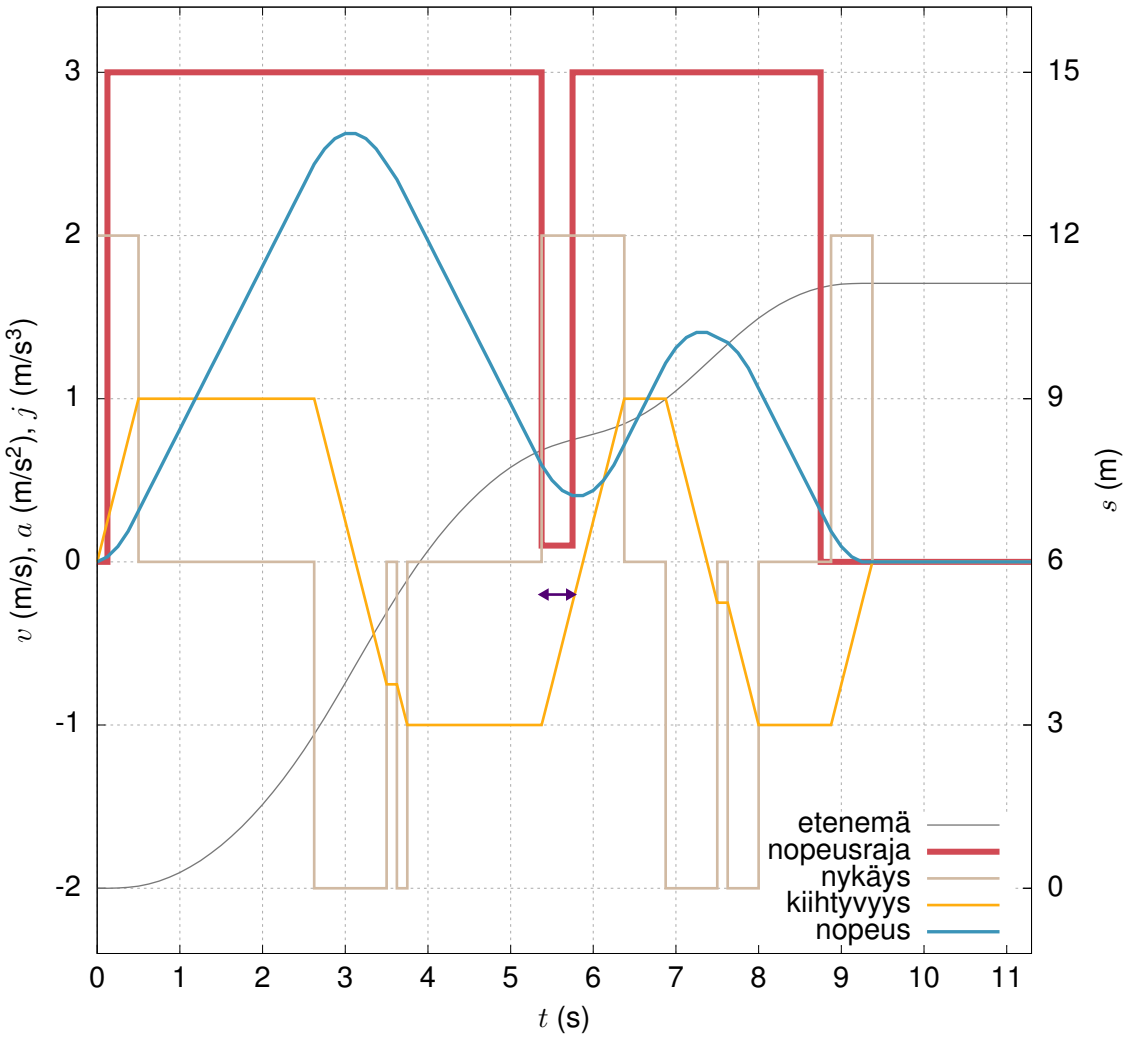
4.4.4 Matala nopeusraja

Kuvassa 4.11 on nopeusprofiili, jolla on hyvin matala nopeusraja (0,1 m/s aikavälillä [5,6 s, 7,6 s]). Profiilin nopeus vähenee ja lopulta noudattaa rajoitusta, mutta koska algoritmi yrittää pitää nopeuden mahdollisimman suurena, nopeus vähenee rajan ohituksen jälkeen nollaan ja robotin on sen jälkeen lähdettävä uudestaan liikkeelle.



Kuva 4.11. *Nopeusprofiili, jolla on hetkellisesti hyvin matala nopeusraja.*

Kuvan 4.12 nopeusprofiili on generoitu lähes samalla nopeusrajafunktiolla kuin kuvassa 4.11. Ero on, että reitin päätepiste on kohdassa s = 11 m, kun se edellisessä tapauksessa on kohdassa s = 15 m. Periaatteessa profiilista ei pitäisi tulla muodoltaan kovin erilaista edelliseen esimerkkiin verrattuna, mutta algoritmi ei silti kykene ottamaan matalaa nopeusrajaa kunnolla huomioon. Todennäköisesti se liittyy siihen, että profiilissa nopeusraja ohitetaan jo siinä vaiheessa, kun nopeutta vielä vähennetään.

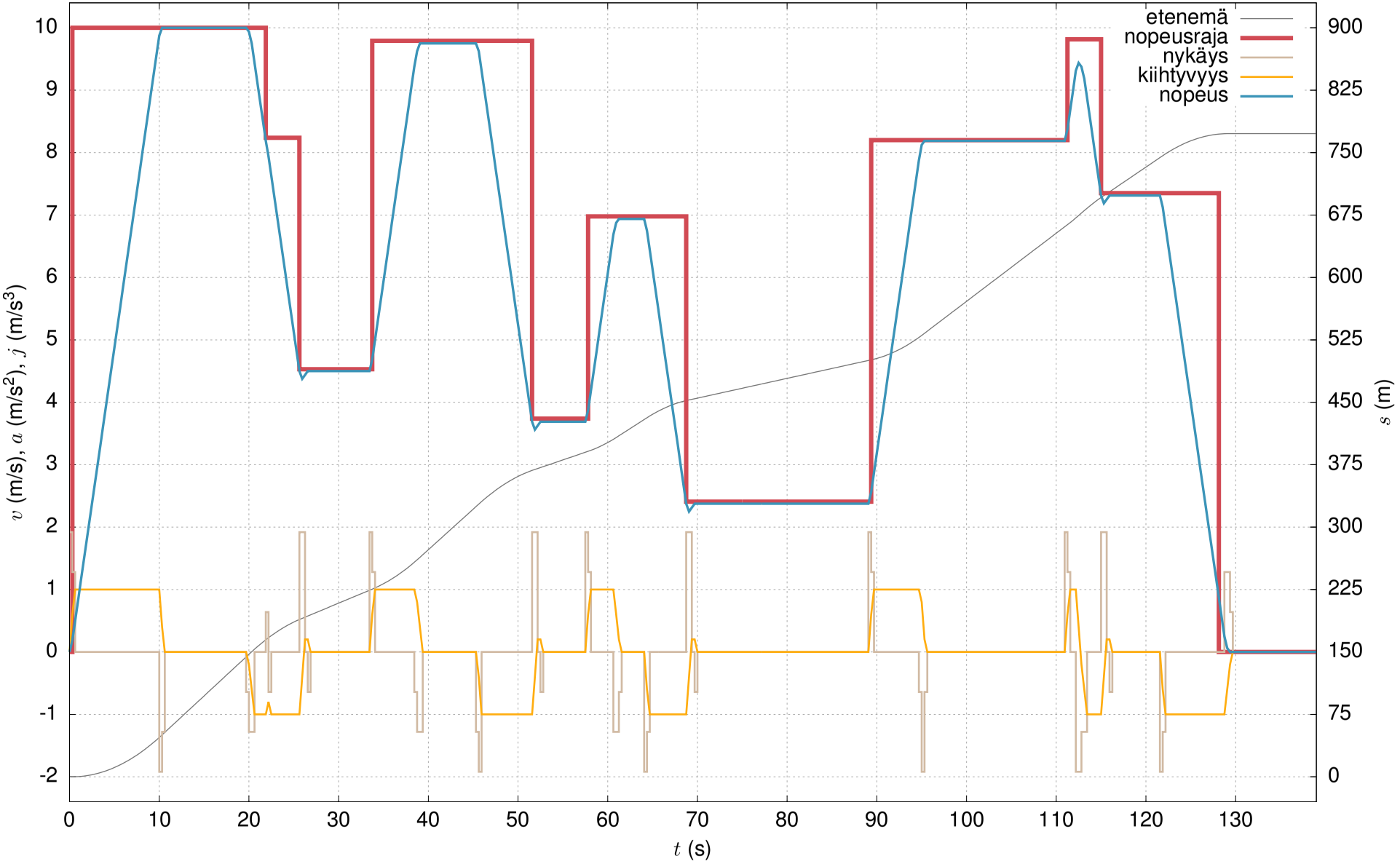


Kuva 4.12. *Nopeusprofiili ei noudata matalaa nopeusrajaa välillä [5,4 s, 5,8 s].*

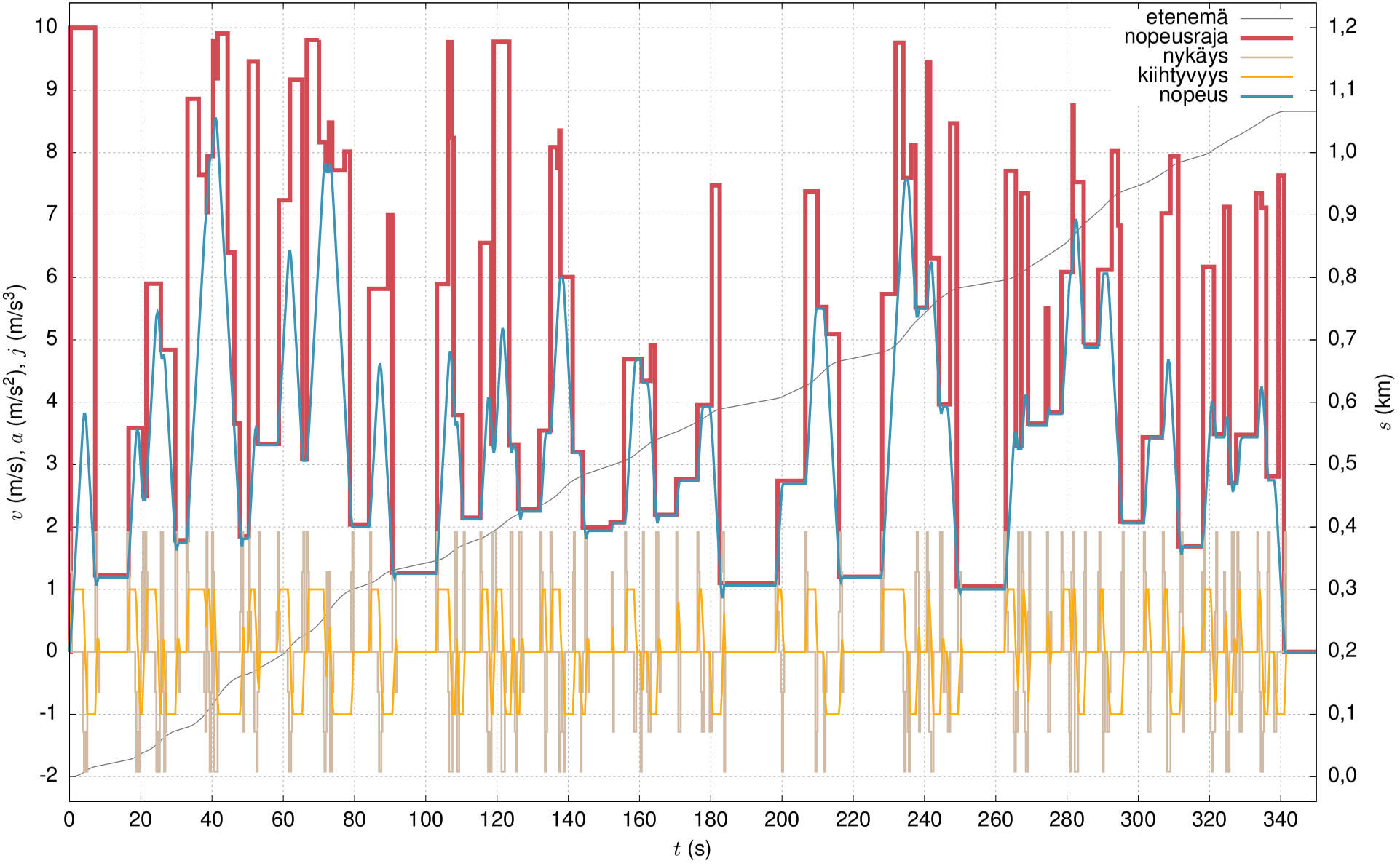
4.4.5 Pitkä nopeusprofiili satunnaisilla nopeusrajoilla

Kuvassa 4.13 on nopeusprofiili, jonka nopeusrajafunktion muodostavat 10 satunnaisesti valittua etenemää ja nopeutta. Profiilin saaminen valmiiksi vaatii noin 400 iteraatiota. Kuvassa 4.14 on nopeusprofiili, jonka nopeusrajafunktio muuttuu 100 kertaa. Profiilin saaminen valmiiksi vaatii noin 1000 iteraatiota. Molemmat profiilit ovat yhtä laillisia kuin aiemmatkin; ne rikkovat nopeusrajoja reittinsä lopussa. Vaikka oikeissa tilanteissa nopeuden muutoksia ei ole näin paljoa, algoritmi selviää niistä, jos iteraatioden määrää kasvatetaan.

Mitä pidempi reitti on ja mitä enemmän nopeusrajoja on, sitä pidempään nopeusprofiilin saaminen valmiiksi kestää. Sekä 10:n että 100:n satunnaisen etenemän ja nopeuden tapauksessa nopeusprofiilin tuottaminen kestää useita kymmeniä sekunteja. Osa ajasta kuluu tosin testikuvaajan tuottamiseen, jota ei ole erotettu profiilin tuottamisesta.



Kuva 4.13. *Pitkä reitti ja 10 nopeusrajaa.*

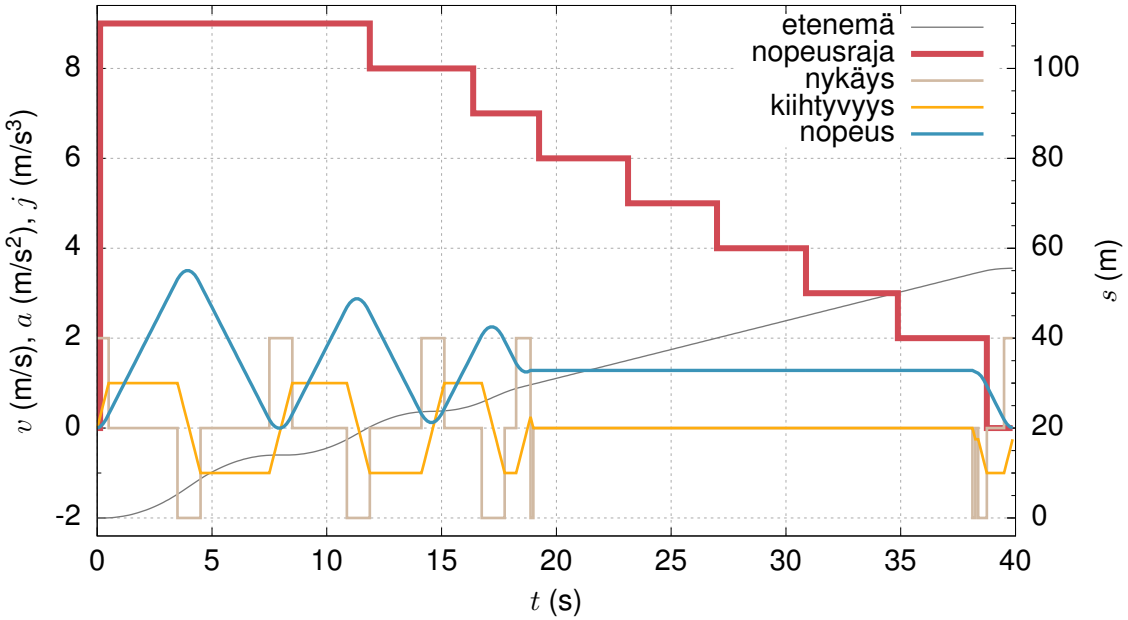


Kuva 4.14. *Pitkä reitti ja 100 nopeusrajaa.*

4.4.6 Iterointi

Kuvat 4.15, 4.16, 4.17, 4.18 ja 4.19 havainnollistavat iteroinnin merkitystä. Erityisesti kuvista 4.16 ja 4.17 voi huomata, että generaattori tuottaa tietyssä iteroinnin vaiheessa nopeusprofiileja, jotka rikkovat nopeusrajoja räikeästi. Näin tapahtuu iteroinnin keskivaiheilla, eli kun viimeinen nopeusraja on saavutettu, mutta aaltoilua ei ole saatu vielä tasoitettua.

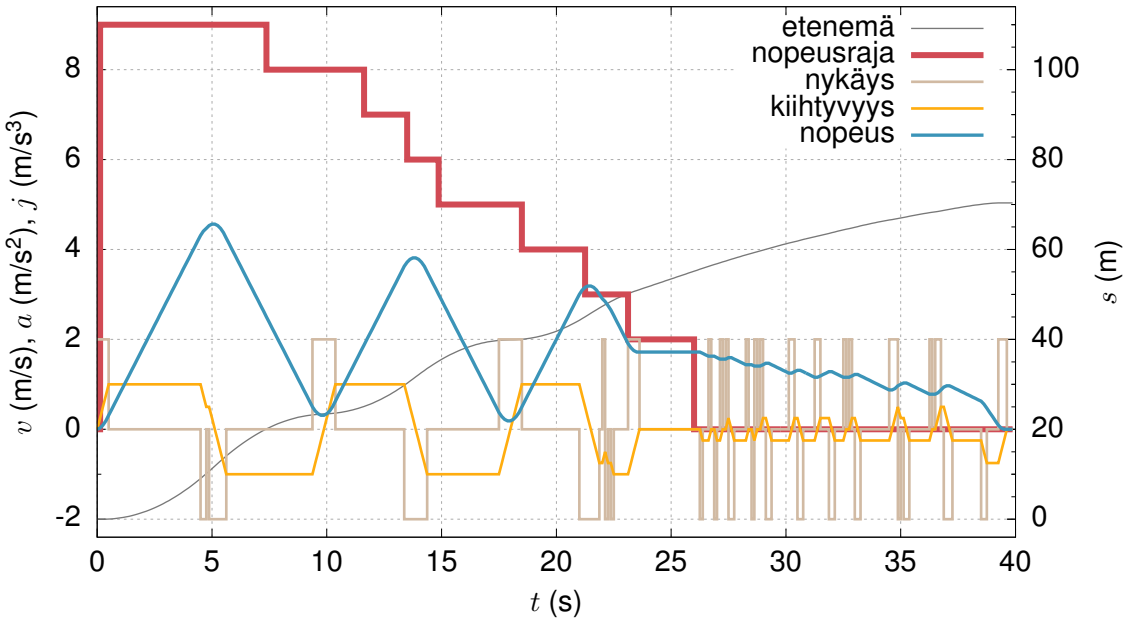
Kuvassa 4.15 on iteraatio 41, jolloin nopeusprofiili on saavuttanut viimeisen nopeusrajan. Nopeusprofiilin alku on aaltoileva. Algoritmi jatkaa profiilin alkupuolen nopeuksien lisäämistä ja alkaa vähentää loppupään nopeuksia.



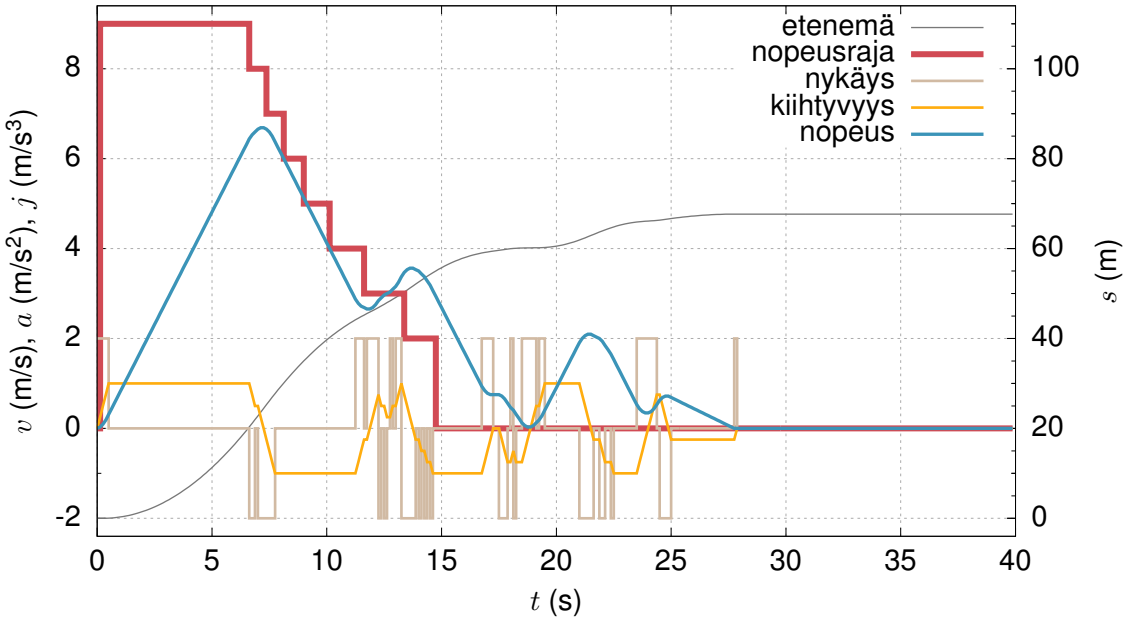
Kuva 4.15. *Iteraatio 41: viimeinen nopeusraja on saavutettu.*

Kuvassa 4.16 on iteraatio 55. Nopeusprofiilin alku on edelleen aaltoileva ja viimeinen aalto on saavuttanut nopeusrajat. Huomattava osa profiilista on jäänyt viimeisen nopeusrajan taakse. Nopeuden pitäisi olla siellä nolla, mutta algoritmi ei pysty vähentämään nopeuksia välittömästi.

Kuvassa 4.17 on iteraatio 90. Nopeusprofiilin ensimmäinen aalto on kasvanut noudattamaan nopeusrajoja lähes täysin ja suurin osa aiemmin nopeusrajojen taakse jääneestä profiilista on nollassa. Sen sijaan nopeusprofiilissa olleet kaksi myöhempää aaltoa ovat jääneet osittain viimeisen rajan taakse, joten algoritmin pitää tasoittaa nekin nollaan.



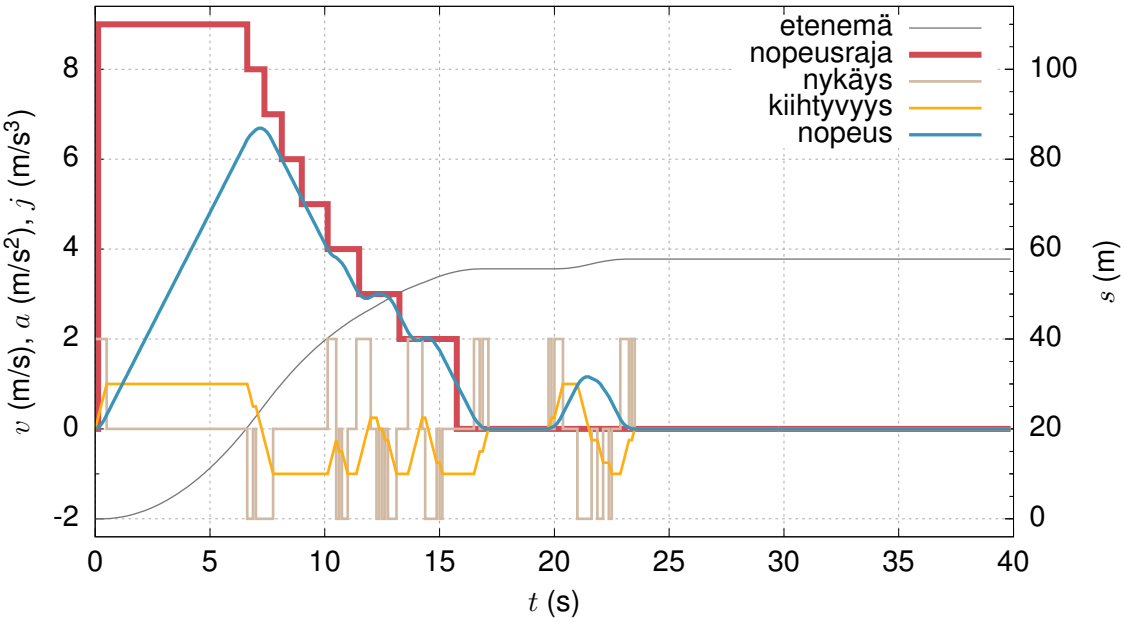
Kuva 4.16. *Iteraatio 55: profiilissa on aaltoja ja loppuosa rikkoo nopeusrajoja.*



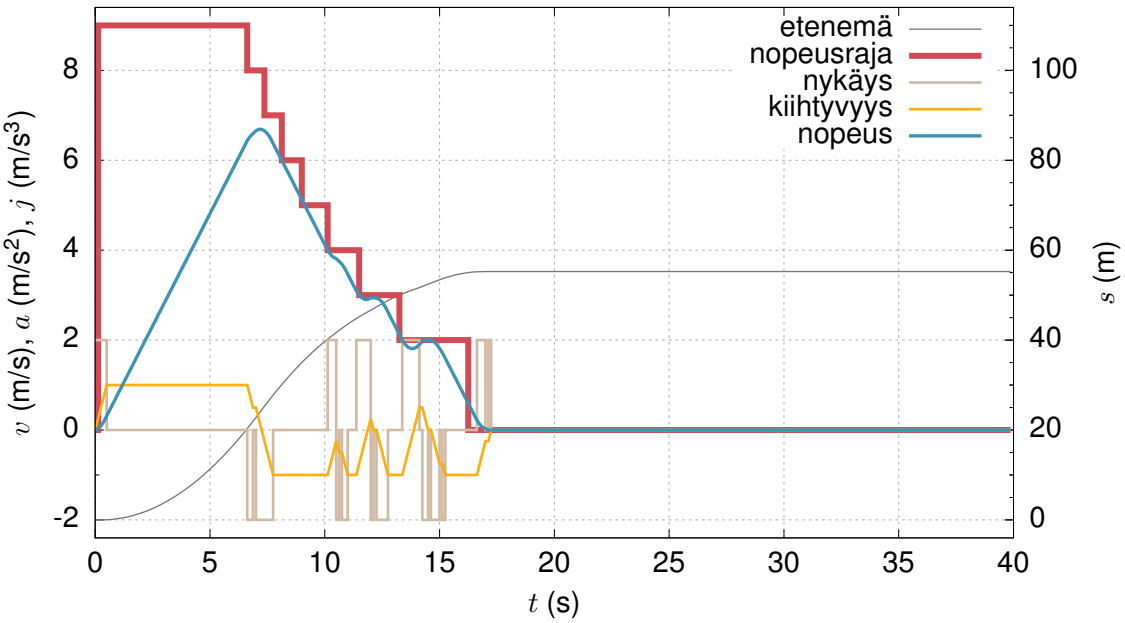
Kuva 4.17. *Iteraatio 90: myöhemmät aallot rikkovat nopeusrajoja.*

Kuvassa 4.18 on iteraatio 120. Algoritmi tasoittaa toisen aallon viimeisiin nopeusrajoihin ja pienentää kolmatta aaltoa vähitellen.

Kuvassa 4.19 on iteraatio 180. Algoritmi on saanut kolmannenkin aallon tasoitettua. Profiili on algoritmin kannalta valmis, eli se ei muutu myöhemmillä iteraatioilla.



Kuva 4.18. *Iteraatio 120: viimeisen nopeusrajan takana on yksi aalto.*



Kuva 4.19. *Iteraatio 180: valmis nopeusprofiili.*

4.4.7 Testauksen tulokset

Parametrit

Uuden nopeusprofiiligeneraattorin toiminnallisia parametreja voi muuttaa. Se ei suoriudu täysin mielivaltaisista arvoista, mutta järkevästi valituilla arvoilla se pystyy tuottamaan järkevän profiilin. Algoritmi ei osaa tehdä nopeusprofiilia, jonka alkunopeus on nollasta poikkeava.

Algoritmin toiminta

Algoritmi pystyy tuottamaan enimmäkseen järkevän näköisen ja rajoja noudattavan nopeusprofiilin. Se pyrkii kasvattamaan nopeuden mahdollisimman suureksi ja pitämään sen mahdollisimman suurena mahdollisimman pitkään. Tästä toimintatavasta johtuen nopeusprofiilissa esiintyy aaltoilua iteroinnin keskivaiheilla ja lopulliseen profiiliin jää ”kuoppia”.

Nopeusrajojen noudattaminen

Nopeusprofiili rikkoo saamaansa nopeusrajafunktiota reitin loppupäässä, eli sen mukaan liikuva robotti ei pysähtyisi aivan sinne minne pitäisi. Jos nopeusrajoissa on vain lyhyen hetken kestävä matala nopeusraja, algoritmi ei välttämättä noudata sitä.

Pitkät reitit

Nopeusprofiilin generointi pitkälle reitille ja useille nopeusrajojen muutoksille kestää liian pitkään.

Iterointi

Iterointia ei voi lopettaa automaattisesti, jos nopeusprofiili tulee valmiiksi jo ennen kuin maksimimäärä iterointeja on tehty. Ei ole tapaa selvittää, onko nopeusprofiili valmis. Jos profiili ei tule valmiiksi maksimi-iteraatiomäärässä, on mahdollista, että se rikkoo nopeusrajoja räikeästi. Näin ollen keskeneräistä profiilia ei välttämättä ole mahdollista käyttää. Iterointi saattaa jäädä tuottamaan kahta tai useampaa hieman erilaista nopeusprofiilia, kuten kuminauha-algoritmin haitoissa (s. 24) kuvataan, vaikka algoritmi ei jääkään ikuiseen silmukkaan, koska iteroinnille on asetettu maksimimäärä.

4.5 Muutokset algoritmiin

Koska algoritmi toimii jokseenkin epäoptimaalisesti ja tuottaa profiileja, jotka eivät pysähdy reitin loppuun, siihen tehtiin joitakin muutoksia.

”Kuoppien” syntyminen estetään, kun nopeuttaan nostavan pisteen ei sallita kasvattaa seuraavien pisteiden nopeuksia, eli kuvassa 4.4 (s. 41) tapahtuva nopeuksien nostaminen ei toteudu. Tällöin kolmas piste ei kasvata nopeuttaan lainkaan, vaan nopeutta nostetaan yksi askel vasta neljännen pisteen kohdalla. Aaltoilun poistava muutos on se, että yhdellä iteraatiolla ei sallita nopeuksien muutoksia eri suuntiin. Nopeuttaan kasvattavan pisteen ei siis sallita vähentää muualla profiilissa olevan pisteen nopeutta ja nopeuttaan

vähentävä piste ei saa kasvattaa muiden pisteiden nopeuksia. Käytännössä kuvassa 4.6 (s. 43) tapahtuva nopeuksien vähentäminen estetään.

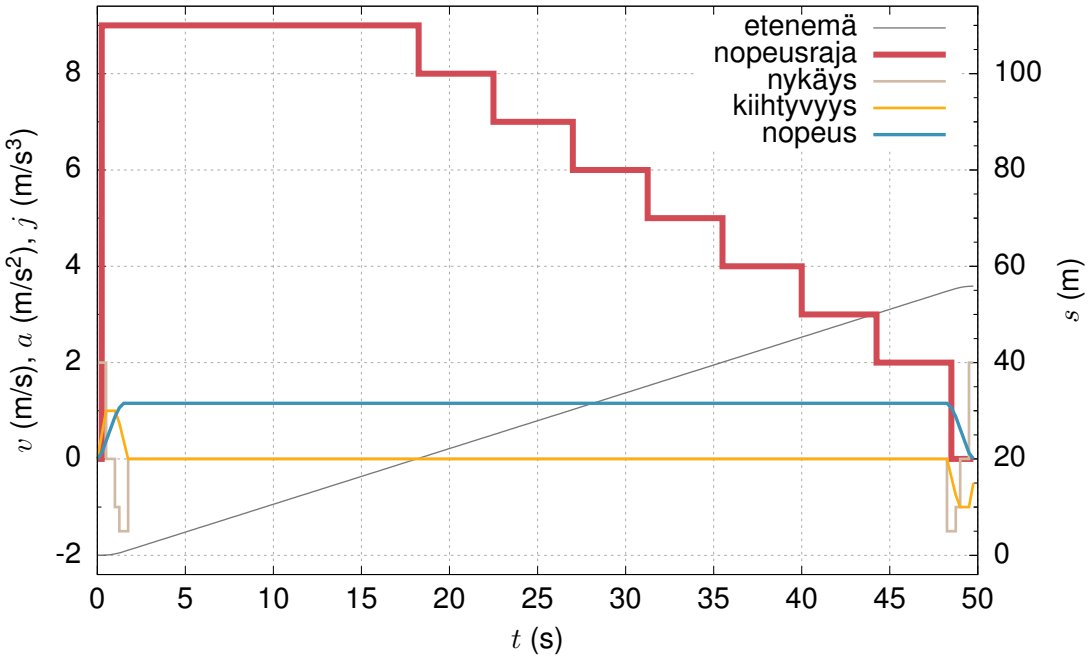
Nopeusrajojen rikkominen reitin lopussa saadaan estettyä antamalla pisteiden vähentää nopeuksiaan, vaikka siitä seuraisikin muiden rajojen rikkomista. Tämä muutos ratkaisee myös ongelman, jossa alemman nopeusrajan reunalla ja siitä askel tai pari eteenpäin olevat pisteet rikkovat lievästi nopeusrajojaan, kun profiili ohittaa matalamman nopeusrajan maksimihidastuvuudella (esimerkiksi kuvan 4.11 (s. 49) ajanhetkellä 5,5 s). Rikkomukset korjaantuvat myöhemmillä iteraatioilla.

Tehdyt muutokset kasvattavat profiilin tekemiseen tarvittavien iteraatioiden määrää, mutta suurimmassa osassa tapauksia profiili tuotetaan edelleen tarpeeksi nopeasti.

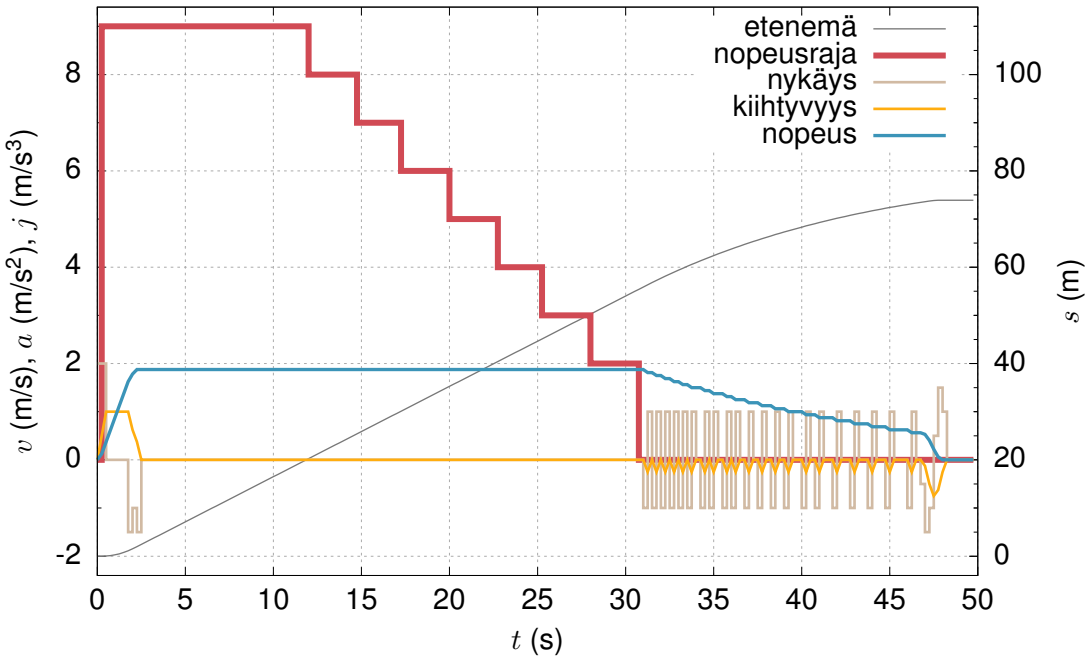
Nollasta poikkeavalla nopeudella alkava profiili saadaan laadittua, kun laajennetaan profiilin käsittelyaluetta alkupisteestä taaksepäin eli negatiivisen ajan suuntaan.

4.5.1 Lopputulokset

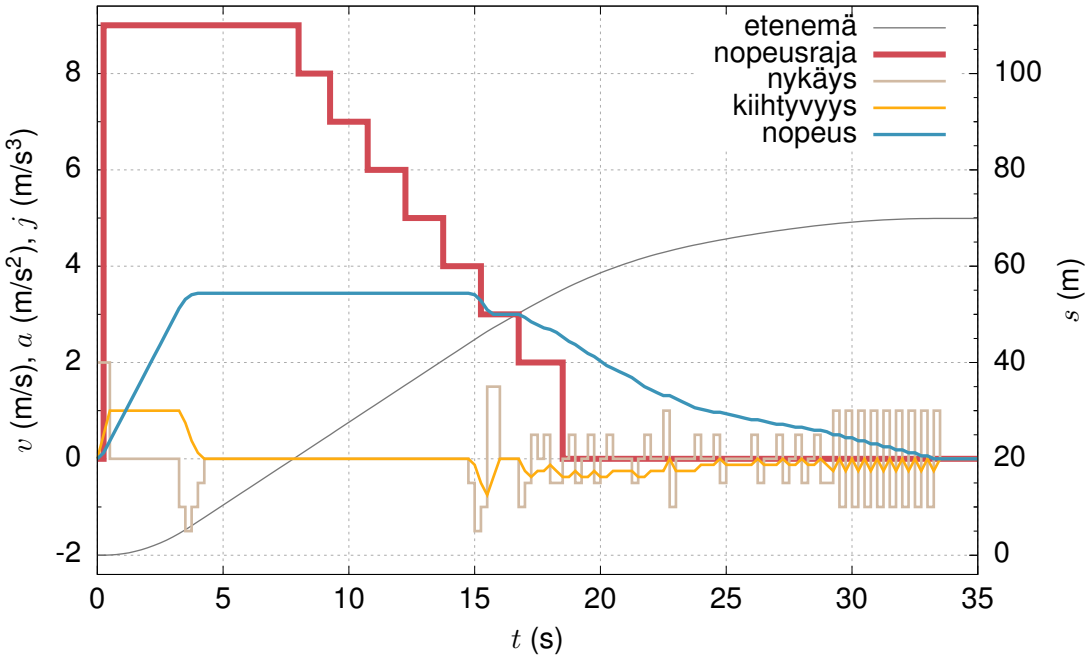
Muokatun algoritmin toiminta on esitetty kuvissa 4.20, 4.21, 4.22, 4.23, 4.24, 4.25 ja 4.26. Kuvassa 4.20 profiili saavuttaa reitin lopun. Kuten kuvasta voi nähdä, profiili on tasainen, koska aaltoilu on korjattu.



Kuva 4.20. *Iteraatio 37: profiili saavuttaa reitin lopun.*

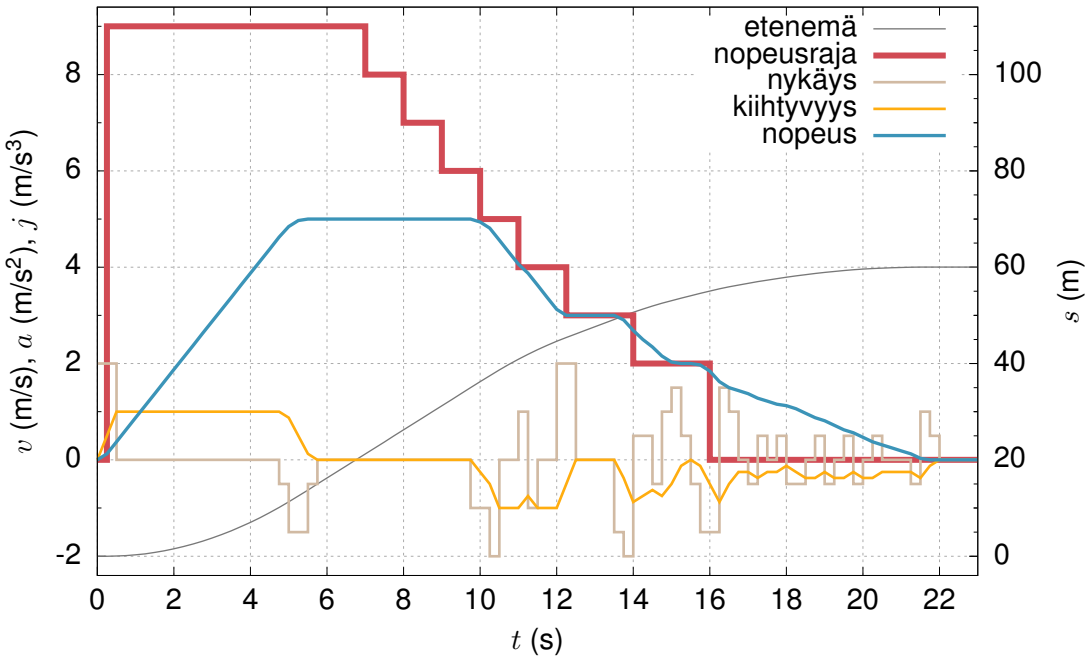


Kuva 4.21. *Iteraatio 60: profiili on tasainen, loppupuolen nopeudet vähenevät.*

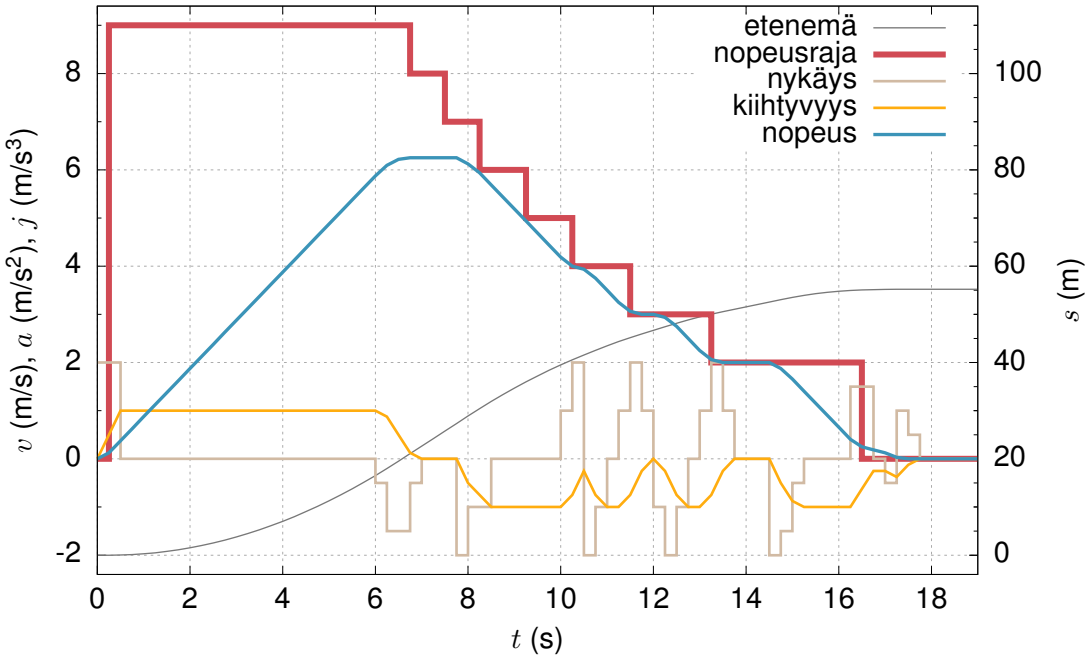


Kuva 4.22. *Iteraatio 110: profiili alkaa asettua nopeusrajoihin.*

Kuvassa 4.21 profiilin alkupuoli on edelleen tasainen. Koska alkupuolen nopeudet kasvavat joka iteraatiolla ja reitin loppu saavutetaan yhä aikaisemmin, loppupuolen nopeuksia vähennetään. Kuvassa 4.22 profiili alkaa mukailla nopeusrajoja ja loppupuolen nopeudet vähenevät edelleen.

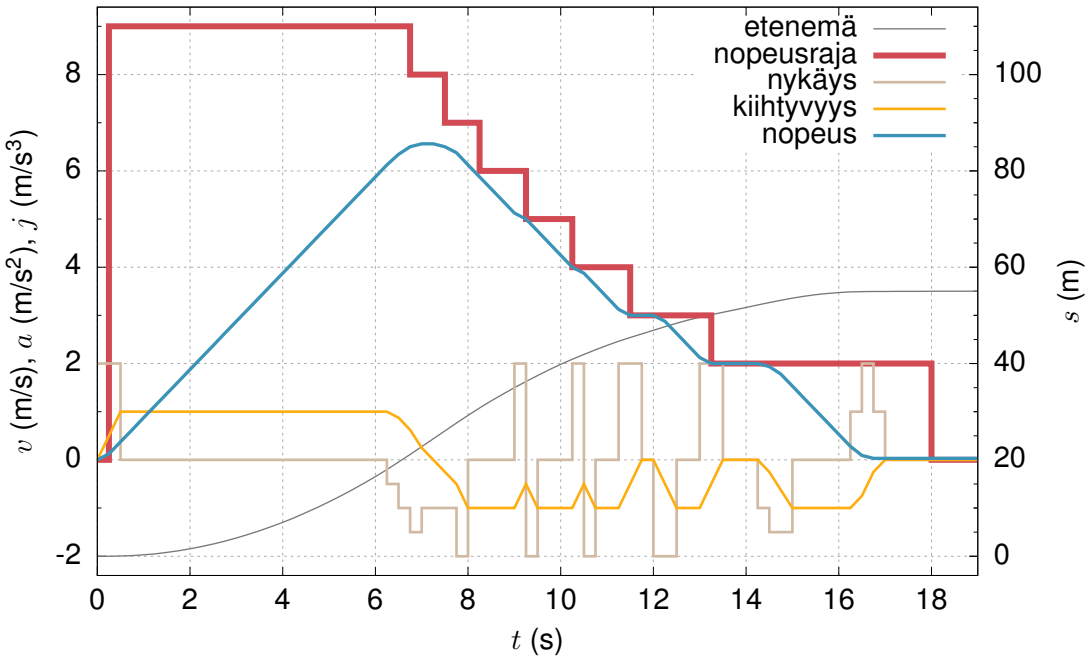


Kuva 4.23. *Iteraatio 160: profiilin loppupuolen nopeudet jatkavat vähenemistä.*

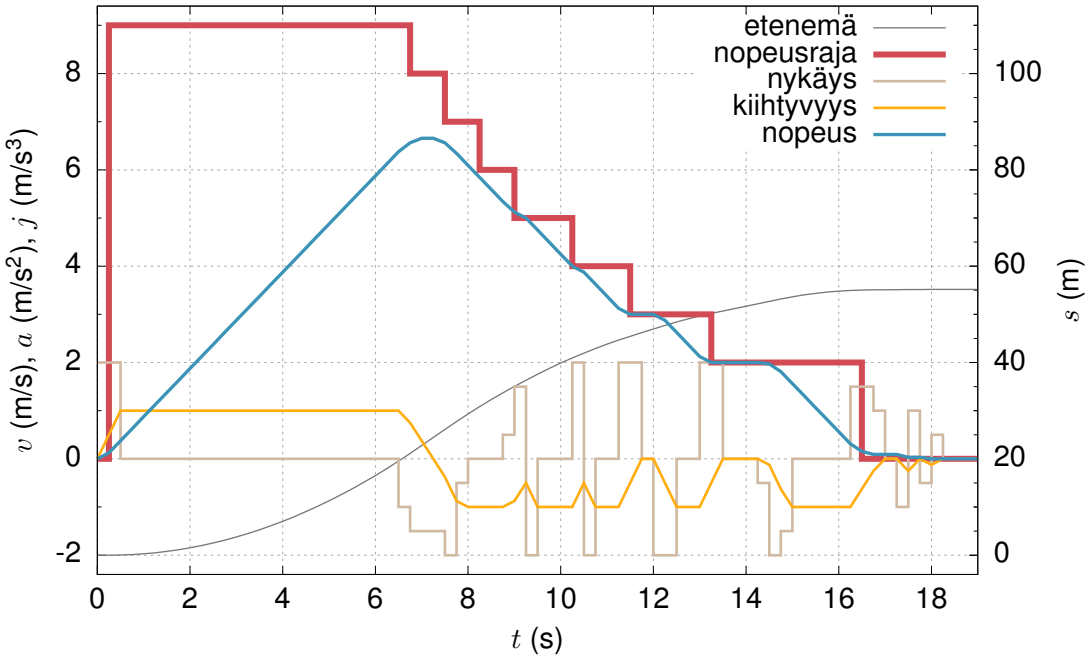


Kuva 4.24. *Iteraatio 200: profiili on lähes valmis.*

Kuvassa 4.23 profiili jatkaa mukautumista. Kuvassa 4.24 nopeusprofiili on lähes valmis. Se rikkoo lievästi nopeusrajoja reitin lopussa ja voisi kasvattaa maksiminopeuttaan vielä vähän.

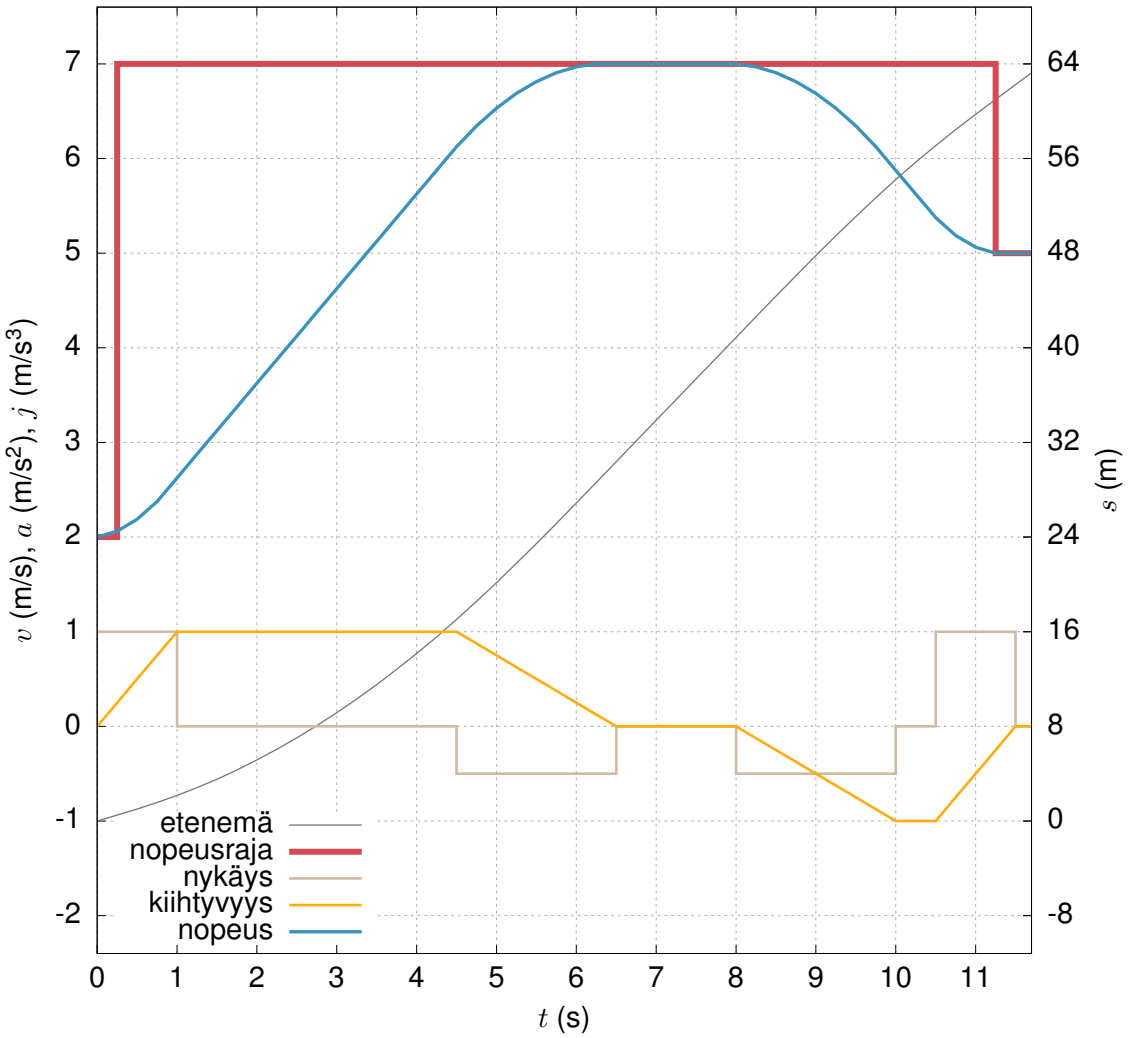


Kuva 4.25. *Iteraatio 210: versio valmiista profiilista.*



Kuva 4.26. *Iteraatio 214: toinen versio valmiista profiilista.*

Nopeusprofiili valmistuu noin iteraation 210 kohdalla. Sen jälkeen generaattori tuottaa hieman erilaisia versioita, joista kaksi on esitetty kuvissa 4.25 ja 4.26. Profiili kuvassa 4.25 noudattaa kaikkia nopeusrajoja ja sen viimeinen nopeus on hyvin matala; se kulkee mini-minopeudella viimeisen sekunnin ajan. Kuvan 4.26 profiili ei ole aivan yhtä laillinen, mutta se on hyvin lähellä ja nopeus reitin lopussa on sen verran pieni, että sen voi todennäköisesti asettaa suoraan nollaan.



Kuva 4.27. *Nopeusprofiililla on nollasta poikkeava alku- ja loppunopeus.*

Kuvassa 4.27 on nopeusprofiili, jolla on nollasta poikkeava alku- ja loppunopeus. Profiilin alkunopeus on 2 m/s, jonka algoritmi kasvattaa kiihtyvyys- ja nykäysrajojen puitteissa nopeusrajaan 7 m/s. Sitten algoritmi vähentää nopeutta, kunnes se osuu oikealla hetkellä nopeusrajaan 5 m/s. Ylimmän nopeusrajan ympärillä nykäys on vain –0,5 m/s^3, vaikka se voisi teoriassa olla –1 m/s^3. Tällainen toiminnallisuus ei kuitenkaan ole haitallista; todennäköisesti on jopa parempi, että nykäys on maltillisempi, kun nopeus on suurempi.

4.6 Jatkokehitys

Uusi nopeusprofiiligeneraattori tuottaa käyttökelpoisia profiileja mutta ei vielä täytä kaikkia sille asetettuja vaatimuksia. Jatkokehityskohdat on jaettu kolmeen luokkaan: nopeusprofiiligeneraattorin toiminnan kannalta välttämättömät, tärkeät ja hyödylliset. Ne esitellään seuraavaksi omissa aliluvuissaan.

4.6.1 Välttämättömät

Uutta generaattoria ei ole vielä täysin integroitu laajempaan automaatio-ohjelmistoon. Tällä hetkellä laajempi automaatio-ohjelmisto osaa tehdä ja lähettää nopeusrajat ja muut rajotukset sisältävän Protobuf-viestin ja vastaanottaa, tulkita ja käyttää nopeusprofiilin sisältäviä Protobuf-viestejä, kunhan yksi profiili mahtuu yhteen viestiin. Lisäksi on tehty viestinkäsittelijäohjelma, joka ottaa vastaan rajoitukset sisältävän Protobuf-viestin, muuntaa ne Haskellin Dimensional-pakettia käyttäviksi arvoiksi ja vastaavasti muuntaa Dimensional-pakettia käyttävän näytteistetyn nopeusprofiilin Protobuf-viestiksi. Viestinkäsittelijän olisi tarkoitus liittyä uuteen generaattoriin. Uusi generaattori ei kuitenkaan käytä Dimensional-pakettia eikä ota rajoituksia vastaan dynaamisesti. Integraation voi saada valmiiksi esimerkiksi muokkaamalla nykyistä viestinkäsittelijäohjelmaa niin, että se tekee viestin myös arvoista, jotka eivät käytä Dimensional-pakettia, ja muokkaamalla uutta generaattoria niin, että se littyy viestinkäsittelijään.

Jos generaattori ei ehdi saamaan nopeusprofiilia valmiiksi annetussa maksimi-iteraatiomäärässä, nopeusprofiili saattaa jäädä laittomaksi (esim. kuva 4.23 sivulla 59). Tämä ongelma lievenee tai poistuu, jos reitti jaetaan useampaan osaan, jolloin nopeusrajat eivät muutu niin usein.

Generaattorissa ei ole vielä aikarajoitusta, eli se ei lopeta ennen kuin maksimimäärä iteraatioita on tehty, vaikka profiilin tuottaminen kestäisi liian kauan. Siihen pitäisi toteuttaa jonkinlainen rajoitin.

Nopeusprofiilin tuottamiseen käyttämää aikaa ei mitattu tarkasti, koska sitä ei ole erotettu generaattorin tekemän testikuvaajan tuottamisesta. Testauksessa kävi ilmi, että mitä pidempi reitti oli, sitä kauemmin sille kesti tuottaa nopeusprofiili. Se johtuu kuitenkin osittain siitä, että pidemmällä reitillä pitää kirjoittaa suurempi määrä tietoa tiedostoon. Generaattoria pitäisi testata oikeassa ympäristössä eli tapauksessa, jossa se kommunikoi laajemman automaatio-ohjelmiston kanssa Protobuf-viestien välityksellä.

Nopeusprofiilin laillisuuden tarkistavaa erillistä tarkistusfunktiota ei ole vielä tehty.

4.6.2 Tärkeät

Koska uusi algoritmi toimii aikatasossa, reitin loppupuolella oleva nopeusprofiilin osuus on jatkuvassa muutoksessa varsinkin iteroinnin alkupuolella. Turhaa muuttelua pitäisi pystyä vähentämään, jotta pitkien ja useita nopeusrajoja sisältävien reittien profiloimiseen ei kuluisi liian kauan aikaa. Profiilin loppupään muokkaamiseen tarvittavia iteraatioita voisi ehkä vähentää asettamalla viimeisen nopeusrajan jälkeiset nopeudet suoraan nollaksi. Profiilin lopun nopeuksien asettaminen nollaksi saattaa tosin vähentää suorituskykyä, jos myöhemmillä iteraatioilla todetaan, että alkupään profiili olikin liian nopea, jolloin profiilin loppupään nopeuksia pitää kasvattaa uudestaan. Nopeusrajat voisi mahdollisesti pilkkoa jo iteroinnin alussa erillisiin osiin paikallisten minimien (matalimpien nopeusrajojen) kohdalta. On myös mahdollista, että jos aloitusprofiilina olisi nollanopeusprofiilin sijaan

esimerkiksi vakiokiihtyvyysprofiili, kuten aliluvussa 3.2.6 (s. 36) esitetään, nopeusprofiilin laatimiseen menisi vähemmän aikaa.

Turhaa nopeuksien muuttelua kannattaa vähentää tarkastelemalla, millä tavalla tietyssä kohdassa oleva nopeus muuttuu. Jos profiili on suunnilleen valmis ja jonkin pisteen nopeus vaihtelee eri iteraatioilla vain muutaman arvon välillä, nopeuden voisi asettaa suoraan kyseisistä nopeuksista pienimpään ja olla enää muuttamatta sitä. Esim. kuvan 4.25 (s. 60) profiilin voisi kiinnittää täysin välillä 0–14 s ja sitten viimeistellä profiilin viimeisen osuuden niin, että siihen tehdyt muutokset eivät vaikuta profiilin muihin osiin.

Jos nopeuden, kiihtyvyyden, nykäyksen ja ajan muutosaskelet on määritelty huolimattomasti, generaattori ei välttämättä pysty tekemään profiilia lainkaan. Muutosaskeleiden arvot pitäisi tarkistaa etukäteen tai mieluummin laskea automaattisesti generaattorin sisällä.

4.6.3 Hyödylliset

Koska suoritettavien iteraatioiden määrä annetaan generaattorille vakiona, nopeusprofiilin generointia ei voi lopettaa joustavasti, jos tarpeeksi ideaalinen profiili saadaan aikaiseksi jo aikaisemmin. Algoritmiin kannattaa siis toteuttaa tarkastelija, joka tarvittaessa pysäyttää generoinnin. Pidemmille reiteille iteraatioita tarvitaan taas enemmän, jotta nopeusprofiili saadaan edes lähelle nopeusrajoja. Iteraatioiden määrää olisi siis hyvä pystyä muuttamaan pääpiirteittäin reitin pituuden ja reitillä olevien nopeusrajojen määrän mukaan.

5 YHTEENVETO

Tämän työn alussa esiteltiin, miten liikkuvalle robotille voi suunnitella etukäteen nopeusprofiilin. Nopeusprofiili kuvaa robotin nopeutta ajan tai etenemän funktiona ja sen avulla voidaan optimoida robotin liikettä. Robotin liikkumiskyky riippuu robotin reitistä ja mm. robotin ja alustan fyysisistä ominaisuuksista, kuten robotin moottorien tehosta, sen massasta, reitin käännöksistä ja alustan liukkaudesta. Nämä ominaisuudet on yksinkertaistettu nopeusrajoiksi ja kiihtyvyyden ja nykäyksen minimi- ja maksimiarvoiksi, jotka annetaan nopeusprofiiligeneraattorille ja joiden avulla generaattori tuottaa rajoituksia noudattavan nopeusprofiilin. Nopeusprofiilin voi esittää analyyttisesti eli laskukaavoina tai diskreetteinä pisteinä eli näytejonona.

Uusi nopeusprofiiligeneraattori suunniteltiin olemassa olevaa laajempaa robotin automaatio-ohjelmistoa varten. Laajemmassa automaatio-ohjelmistossa on generaattori, mutta se on vaatimuksiin nähden turhan monimutkainen ja sen toteutus on virhealtis, joten se haluttiin korvata helpommin ymmärrettävissä ja ylläpidettävissä olevalla toteutuksella. Uuden nopeusprofiiligeneraattorin algoritmiksi esitettiin useita vaihtoehtoja, joista valittiin kuminauha-algoritmi.

Valitun algoritmin tekemä nopeusprofiili koostuu diskreeteistä pisteistä. Algoritmi tuottaa nopeusprofiilin iteroimalla, eli alussa profiilin kaikkien pisteiden nopeus on nolla, ja joka iteraatiolla nopeuksia nostetaan hieman niin, että profiili ei riko nykäyksen ja kiihtyvyyden rajoituksia. Profiilia käsitellään aikatasossa, joten nopeusrajat siirtyvät, kun profiili muuttuu nopeammaksi. Tästä syystä jotkut pisteet saattavat päätyä matalamman nopeusrajan alueelle, jolloin algoritmin on vähennettävä pisteiden nopeuksia.

Uutta nopeusprofiiligeneraattoria testattiin antamalla sille erilaisia nopeusrajafunktioita. Sen tuottamat profiilit esitettiin kuvaajissa. Algoritmi tuottaa nopeusprofiilin niin, että nopeus nostetaan mahdollisimman korkeaksi mahdollisimman pian ja pidetään mahdollisimman korkealla mahdollisimman pitkään, vaikka seuraavien pisteiden nopeuksia jouduttaisiin vähentämään. Tällainen toimintaperiaate ei ole täysin toivottu, siellä se aiheuttaa profiilin aaltoilua ja sen, että nopeusprofiilissa seuraava vastaantuleva matalampi nopeusraja ohitetaan maksimihidastuvuudella, jolloin nopeus vähenee matalampaa nopeusrajaa pienemmäksi ja nopeutta joudutaan taas erikseen lisäämään. Tämä ongelma korjattiin antamalla algoritmin vain joko kasvattaa tai vähentää pisteiden nopeuksia yhdellä kierroksella. Algoritmin tuottama nopeusprofiili rikkoo nopeusrajoja reitin lopussa ja joskus lyhyen aikaa kestävän matalan nopeusrajan aikana. Tämä ongelma korjattiin sallimalla algoritmin vähentää nopeuksia, vaikka siitä seuraisi lievää rajoitusten rikkomista,

sillä rikkomukset korjaantuvat myöhemmillä iteraatioilla.

Jos generaattorille annetut kiihtyvyyden ja nykäyksen rajat ovat järkeviä, se pystyy tuottamaan nopeusprofiileja kaikille nopeusrajafunktioille. Profiili noudattaa annettuja nopeusrajoja ja kiihtyvyyden ja nykäyksen rajoituksia, jos se tulee valmiiksi annetussa maksimi-iteraatiomäärässä. Generaattori pystyy tuottamaan nopeusprofiileja alle sekunnissa, paitsi jos reitti on hyvin pitkä ja siinä on paljon nopeusrajojen muutoksia.

Vaikka uusi nopeusprofiiligeneraattori pystyy tuottamaan käyttökelpoisia nopeusprofiileja, siinä on vielä joitain jatkokehitystarpeita. Tärkeimpiä niistä ovat generaattorin integrointi laajempaan automaatio-ohjelmistoon ja iteraatioiden määrän selvittäminen niin, että profiili tulee varmasti valmiiksi. Lisäksi olisi hyvä rajoittaa lähes valmiin profiilin nopeuksien muuttelua edestakaisin ja vähentää tarvittavien iteraatioden määrää esimerkiksi antamalla aloitusprofiiliksi vakiokiihtyvyysprofiili.

LÄHTEET

[1] Al-Sharif, L. Intermediate Elevator Kinematics and Preferred Numbers (METE III). *Lift Report* 40 (joulukuu 2014), 20–31. URL: https://www.researchgate.net/publication/275408222_Intermediate_Elevator_Kinematics_and_Preferred_Numbers_METE_III (viitattu 23. 02. 2021).

[2] García-Martínez, J. R., Rodríguez-Reséndiz, J. ja Cruz-Miguel, E. E. A New Seven-Segment Profile Algorithm for an Open Source Architecture in a Hybrid Electronic Platform. *Electronics* 8.6 (2019). ISSN: 2079-9292. DOI: 10.3390/electronics8060652. URL: https://www.mdpi.com/2079-9292/8/6/652 (viitattu 23. 02. 2021).

[3] Eager, D., Pendrill, A.-M. ja Reistad, N. Beyond velocity and acceleration: jerk, snap and higher derivatives. *European Journal of Physics* 37.6 (lokakuu 2016), 065008. DOI: 10.1088/0143-0807/37/6/065008. URL: https://doi.org/10.1088/0143-0807/37/6/065008 (viitattu 23. 02. 2021).

[4] Lahdelma, S. *Vikojen tunnuslukutaulukko*. 2008. URL: https://lahdelma.files.wordpress.com/2017/06/vikojen-tunnuslukutaulukko.pdf (viitattu 23. 02. 2021).

[5] Alberg, H. ja Pendrill, A.-M. Hastighet, acceleration, ryck – och sedan? *Nämnaren* 4 (2017), 57–58. URL: http://ncm.gu.se/pdf/namnaren/5758_17_4.pdf (viitattu 23. 02. 2021).

[6] *Standard C++*. 2020. URL: https://isocpp.org/ (viitattu 04. 12. 2020).

[7] *Haskell Language*. 2020. URL: https://www.haskell.org/ (viitattu 04. 12. 2020).

[8] *Dimensional: Statically checked physical dimensions, using Type Families and Data Kinds.* 2018. URL: https://hackage.haskell.org/package/dimensional (viitattu 22. 12. 2020).

[9] *Protocol Buffers | Google Developers*. URL: https://developers.google.com/protocol-buffers/ (viitattu 08. 04. 2021).