Katariina Kotanen

Hyperelliptisen satunnaisjakauman algoritmin kehittäminen

Diplomityö

Tekniikan ja innovaatiojohtamisen akateeminen yksikkö
Diplomityö
Energia- ja informaatiotekniikka

Vaasa 2025

VAASAN YLIOPISTO
Tekniikan ja innovaatiojohtamisen akateeminen yksikkö
Tekijä: Katariina Kotanen
Tutkielman nimi: Hyperelliptisen satunnaisjakauman algoritmin kehittäminen : Diplomityö
Tutkinto: Diplomi-insinööri
Oppiaine: Informaatiotekniikka
Työn ohjaaja: Jouni Lampinen
Valmistumisvuosi: 2025 **Sivumäärä:** **87**

TIIVISTELMÄ:
Työn aiheena on hyperelliptisen satunnaisjakauman algoritmin kehittäminen. Tavoitteena on luoda algoritmi, jolla voidaan mallintaa ja tutkia hyperelliptisen satunnaisjakauman joukkoa. Algoritmille asetetaan parametrina todennäköisyysarvo, jota säätämällä populaatio suppenee tai harvenee jakaumalla hyperelliptisellä alueella. Algoritmin tulee toimia käytännöllisellä laitteistolla. Työssä perehdytään moniulotteisiin malleihin, ja tutkimuksessa käytetään suunnittelutieteellistä menetelmää, jossa keskeisenä tavoitteena on käytännöllisen ratkaisun kehittäminen ja arviointi. Kehitetyn algoritmin on tarkoitus olla monikäyttöinen, jotta sitä voidaan soveltaa erilaisiin tilanteisiin muokkaamalla esimerkiksi alkutietoja tai muita ohjelman osia sopimaan tiettyyn käyttötarkoitukseen.

Tutkimuksen teoriaosuus kattaa ellipsien, hyperellipsien, ellipsoidien sekä hyperellipsoidien määritelmät ja ominaisuudet. Lisäksi teoriaosuudessa käsitellään euklidista geometriaa ja euklidista etäisyyttä, jotka ovat keskeinen osa algoritmin toteutusta. Suunnittelutieteellisen tutkimusmenetelmän teorian avulla luodaan suunnitelma tutkimuksen etenemiselle.

Teorian pohjalta luodaan pseudokielellä algoritmista ohjelma, joka mallinnetaan mallinnusohjelmalla. Mallinnusohjelma valitaan sen perusteella, että mallinnusohjelmassa algoritmi toimii vaivattomasti, ja sillä onnistuu myös tulosten talteenottaminen ja datan muokkaaminen. Työssä käytetään MATLAB-ohjelmistoa mallintamaan ja luomaan algoritmi. MATLAB-ohjelmistolla luodaan ja suoritetaan algoritmin perusteella ohjelma käyttäen MATLAB-ohjelman omaa ohjelmointikieltä. Tulosten saaminen visuaaliseen muotoon analysoitavaksi onnistuu käyttäen MATLAB-ohjelmistoa, ja tulokset pystytään viemään ulkoiseen tiedostomuotoon.

Tulosten analysoinnissa keskitytään erityisesti siihen, miten populaation kehitys vastaa ennalta asetettuja oletuksia. Oletusarvot ovat, että populaatio suppenee, harvenee tai pysyy vakaana riippuen määritellystä todennäköisyysparametrista. Työssä todetaan, että algoritmi tuottaa odotusten mukaisia tuloksia ja mahdollistaa satunnaisjoukon visuaalisen tarkastelun.

Algoritmia voidaan käyttää tutkittaessa hyperelliptistä satunnaisjakaumaa, ja MATLAB sopii tulosten tuottamiseen käytännössä. Työn tulokset vahvistavat algoritmin toimivuuden ja mahdollisuudet eri sovellusalueilla. Tulokset täsmäävät ennakoituja olettamuksia satunnaisjoukolle, eli populaatio suppenee, harvenee tai pysyy samana, riippuen annetusta todennäköisyysparametrista. Algoritmia voidaan kehittää eteenpäin luomaan yhä moniulotteisempia malleja. Koska kehitelty algoritmi on yleisluontoinen, sitä on mahdollista hyödyntää erilaisissa käyttötarkoituksissa. Diplomityössä käytetään MATLAB-ohjelmistoa, mutta samaan tarkoitukseen voi löytyä toinen tehokkaampi ohjelmisto.

AVAINSANAT: Algoritmit, hyperellipsi, satunnaisjoukko, MATLAB, populaatio

Sisällys

1 Johdanto 7

2 Teoriaa ellipsit, ellipsoidit, hyperellipsoidit 9

2.1 Euklidinen geometria ja etäisyydet 9

2.2 Ellipsit 13

2.2.1 Ellipsin määritelmä 14

2.2.2 Piste ellipsin sisäpuolella 14

2.2.3 Hyperellipsi 15

2.3 Ellipsoidit 15

2.3.1 Ellipsoidin määritelmä 16

2.3.2 Piste ellipsoidin sisäpuolella 16

2.4 Hyperellipsoidit 17

2.4.1 Hyperellipsoidin määritelmä 17

2.4.2 Piste hyperellipsoidin sisäpuolella 18

3 Tutkimusmenetelmä, satunnaisjakaumien taustoitus, alkuasetelma 19

3.1 Suunnittelutieteellinen tutkimusmenetelmä 19

3.1.1 Ongelman määrittely ja tavoite 21

3.1.2 Ratkaisun tavoitteet 22

3.1.3 Suunnittelu ja kehittely 23

3.1.4 Havainnollistaminen 23

3.1.5 Arviointi 24

3.1.6 Viestintä 24

3.2 Satunnaisjakaumat 25

3.2.1 Satunnaisjakauma neuroverkoissa 26

3.2.2 Geometrinen todennäköisyys 27

3.2.3 Todennäköisyyspinta 28

3.3 Alkuasetelman suunnittelu 29

3.3.1 Alkuarvot ja -tiedot 29

3.3.2 Satunnaisotanta 30

3.3.3 Algoritmiin liittyvät valinnat 31

4 Algoritmin kuvaus pseudokielellä 33
4.1 Algoritmi 33
4.2 Populaation generointi, alku 35
4.3 Valitaan satunnaisesti pisteet 36
4.4 Määritellään ellipsi tai pyörähdysellipsoidi 37
4.5 Korvataan piste 38
4.6 Toistorakenne, lopetusehto 41
5 Koeasetelma 43
5.1 Matlab esittely 43
5.2 Valmistelu 44
5.2.1 Parametrit käyttäjältä 44
5.2.2 Funktiot 45
5.2.3 Pisteen generoiminen 46
5.2.4 Hylkäämismenetelmä 47
5.3 Ellipsi-ohjelma 47
5.4 Ellipsoidi-ohjelma 48
6 Tulokset 49
6.1 Ellipsi, T=0,00 50
6.2 Ellipsi, T=0,50 53
6.3 Ellipsi, T=1,00 57
6.4 Ellipsoidi, T=0,00 60
6.5 Ellipsoidi, T=0,50 62
6.6 Ellipsoidi, T=1,00 65
6.7 Ajanmittaus, hylkäysmenetelmän tarpeellisuus 67
7 Johtopäätökset 70
Lähteet 72
Liitteet 77
Liite 1. Algoritmi 1. 77
Liite 2. Algoritmi 2. 82

Kuvat

Kuva 1. Esimerkki euklidisesta etäisyydestä. (Park & Lee, 2011, Figure 1) 11
Kuva 2. Ellipsin muodostaminen ja piirtäminen (Tatum, n.d.). 13
Kuva 3. Esimerkki diskreetistä jakaumasta. 25
Kuva 4. Esimerkki jatkuvasta jakaumasta. 26
Kuva 5. Algoritmin kulku ellipsoidille. 34
Kuva 6. Populaatio XYZ-koordinaatistossa. 35
Kuva 7. A, B ja C pisteet valittu. 36
Kuva 8. Muodon piirtäminen pisteiden perusteella. 37
Kuva 9. Pyörähdysellipsoidin piirtäminen pisteiden A, B ja C perusteella. 38
Kuva 10. Pisteen C korvaaminen uudella pisteellä. 39
Kuva 11. Yksikköpallo ja pisteet sen pinnalla. (Li ja muut, 2019.) 39
Kuva 12. Litteä ellipsi ja yksikköympyrä. 40
Kuva 13. Ellipsi, T=0,00, alkuiteraatiot. 50
Kuva 14. Ellipsi, T=0,00, iteraatio 300. 51
Kuva 15. Ellipsi, T=0,00, populaation kehittyminen. 52
Kuva 16. Ellipsi, T=0,00, populaation kehittyminen, iteraatio 5000. 53
Kuva 17. Ellipsi, T=0,50, alkuiteraatiot. 54
Kuva 18. Ellipsi, T=0,50, iteraatio 300. 55
Kuva 19. Ellipsi, T=0,50, populaation kehittyminen. 56
Kuva 20. Ellipsi, T=1,00, alkuiteraatiot. 57
Kuva 21. Ellipsi, T=1,00, iteraatio 300. 58
Kuva 22. Ellipsi, T=1,00, populaation kehittyminen. 59
Kuva 23. Ellipsoidi, T=0,00, alkuiteraatiot. 60
Kuva 24. Ellipsoidi, T=0,00, iteraatio 300. 61
Kuva 25. Ellipsoidi, T=0,00, populaation kehittyminen. 62
Kuva 26. Ellipsoidi, T=0,50, alkuiteraatiot. 63
Kuva 27. Ellipsoidi, T=0,50, iteraatio 300. 63
Kuva 28. Ellipsoidi, T=0,50, populaation kehittyminen. 64
Kuva 29. Ellipsoidi, T=1,00, alkuiteraatiot. 65

Kuva 30. Ellipsoidi, T=1,00, iteraatio 300. 66

Kuva 31. Ellipsoidi, T=1,00, populaation kehittyminen. 67

Taulukot

Taulukko 1. Suunnittelutieteellisen menetelmän osiot. (Venable ja muut, 2017, taulukko 3. Peffers ja muut, 2007, taulukko 1.) 21

Taulukko 2. Suunnittelutieteellisen tutkimuksen vaiheet tässä työssä. 24

Taulukko 3. Ellipsi, T=0,00, keskietäisyys ja varianssi. 52

Taulukko 4. Ellipsi, T=0,50, keskietäisyys ja varianssi. 56

Taulukko 5. Ellipsi, T=1,00, keskietäisyys ja varianssi. 59

Taulukko 6. Ellipsoidi, T=0,00, keskietäisyys ja varianssi. 61

Taulukko 7. Ellipsoidi, T=0,50, keskietäisyys ja varianssi. 64

Taulukko 8. Ellipsoidi, T=1,00, keskietäisyys ja varianssi. 66

Taulukko 9. Ellipsi-ohjelman ajanmittaus. 68

Taulukko 10. Ellipsoidi-ohjelman ajanmittaus. 68

Taulukko 11. Hylkäysmenetelmän käyttö. 69

Algoritmit

Algoritmi 1. Ellipsin Matlab-ohjelma. Algoritmi on luotu tekoälyä hyödyntäen yhdistämällä useaa kehotetta (OpenAI, 2024). 81

Algoritmi 2. Ellipsoidin Matlab-ohjelma. Algoritmi on luotu tekoälyä hyödyntäen yhdistämällä useaa kehotetta (OpenAI, 2024). 87

Lyhenteet

a	ensimmäinen puolipituusakseli
b	toinen puolipituusakseli
c	kolmas puolipituusakseli
HP	Hewlett-Packard
n	iteraatiokerrat
ms	millisekunti
T	todennäköisyys

1 Johdanto

Mallinnuksen avulla testataan abstraktejakin ideoita ja hypoteeseja, sekä kokeillaan uusia menetelmiä vähäisin kustannuksin. Mallinnuksella myös todetaan jo ennen projektin aloittamista sen mahdollisuuksia tai tarvittavia rajauksia. Erityisesti sähköiset mallinnuskeinot pystyvät käsittelemään kerralla suuria datamääriä ja niihin kohdistuvia toimintoja. Ajatuksellinen ymmärrys hahmottaa konseptuaalisen mallin, kun se tuodaan konkreettisempaan muotoon.

Bork & Smajevic (2021, s. 610) toteavat mallinnuksen hyödyntämisen tarjoavan merkittäviä etuja tietojärjestelmien kehityksessä, kuten moniulotteisten kokonaisuuksien hallinnassa ja analysoinnissa. Myös analysointivaihe on automatisoitavissa. Tehokas datankäsittely pienentää tarvittavan ajan määrää ja virheitä, ja analyysien avulla voidaan selvittää tietoa mallin laadusta ja parannuskohteista.

Mallinnuksen avulla luodaan yksilöllistettävä malli, jota voidaan muokata omiin tarpeisiin. Gogollan ja muiden (2021, s. 244) mukaan mallinnuksen joustavuus sallii mallinnustyökaluille säätelyvaraa, mikä helpottaa prototyyppien ja mallien kehittämistä varhaisessa vaiheessa. Lisäksi tämä tukee lähestymistapaa tilanteissa, joissa järjestelmän vaatimukset ovat epäselviä, kehittyvät ajan myötä, tai vaativat jossain määrin intuitiivista lähestymistä ongelman äärelle. Mallinnusmenetelmän joustavuus lisää luovuutta ratkaisun kehittämisessä tinkimättä verifioinnin ja validoinnin hyödyistä (Gogolla ja muut, 2021, s. 244).

Diplomityön aiheena on kehittää algoritmi hyperelliptiselle satunnaismuuttujista koostuvalle jakaumalle. Jakaumalle muodostetaan satunnainen populaatio. Luodusta joukosta valitaan kolme pistettä, ja ellipsoidin määritelmään perustuen määritellään ja muodostetaan pisteiden perusteella jakaumalle ellipsoidi. Kolmantena valittu piste korvataan satunnaisella uudella luodulla pisteellä. Todennäköisyys T kuvaa, millä todennäköisyydellä korvattavat pisteet ovat mallinnetun ellipsoidin sisäpuolella. Tuloksista selvitetään, suppeneeko vai laajeneeko joukko jakaumalla, sijaitsevatko pisteet

hyperelliptisen muodon sisä- vai ulkopuolella. Vertaamalla tuloksia ja alussa annettua todennäköisyyttä arvioidaan lopuksi, toimiiko algoritmi sille tarkoitetulla tavalla.

Tutkimusmenetelmänä käytetään suunnittelutieteellistä menetelmää löytämään sopivin ratkaisu tutkimuskysymyksen asettamiin ongelmiin. Tutkimuskysymyksenä on, millainen on algoritmi, jolla luodaan hyperelliptisessä muodossa oleva satunnaismuuttujien jakauma. Onko se mahdollinen, ja pystyykö sellaisen kehittämään rajatuilla resursseilla. Liikaa aikaa vievä ohjelma ei ole käytännöllinen, eikä ohjelma, joka vaatii valtavan prosessitehon toimiakseen. Tavoitteena ja hypoteesina on, että tällainen ratkaisu löytyy ja se pystytään toteuttamaan. Tutkimuksessa perehdytään kolmiulotteisiin ellipseihin.

Diplomityön kappaleet 2 ja 3 käsittelevät teoriaa tutkimuskysymyksen taustalla ja alkuasetelmaa, kuten elliptinen muoto ja satunnaisjakauma. Käytäntökappaleissa 4, 5 ja 6 esitellään algoritmia tarkemmin, koeasetelma sekä tulokset. Johtopäätökset ovat koottuna kappaleessa 7.

2 Teoriaa ellipsit, ellipsoidit, hyperellipsoidit

Ellipsit, ellipsoidit ja hyperellipsoidit ovat samankaltaisia muotoja, mutta eriulotteisia. Ellipsi on kaksiulotteinen, ellipsoidi on kolmiulotteinen, ja hyperellipsoidi on n-ulotteinen. Eli ellipsi voidaan piirtää tasona paperille, soikionmuotoisena kehänä. Ellipsoidi on 3D-versio tästä, jolloin muoto laajennetaan katsojaa kohti. Hyperellipsoidi on ellipsoidin erikoistapaus, jonka ulotteisuus vaihtelee. Toisin sanoen, ellipsoidi on hyperellipsoidin tasan kolmiulotteinen tapaus. Näille kaikille kappaleille on yhtenäisiä asetettuja määritelmiä ja sääntöjä.

2.1 Euklidinen geometria ja etäisyydet

Euklidinen geometria ja sen johdannaiskäsitteet ovat saaneet nimensä antiikinkreikkalaisen matemaatikon, Eukleides Aleksandrialaisen, mukaan. Eukleides Aleksandrialaisen oppia esitellään Boyerin teoksessa Tieteiden kuningatar – Matematiikan historia, osa I (1994) seitsemännessä luvussa. Eukleides on kirjoittanut matematiikkaa ja geometriaa käsittelevän Elementa-kirjan. Elementa sisältää viisi postulaattia ja viisi aksioomaa, jotka ovat loogisesti pääteltyjä oletuksia kappaleista. Oletukset liittyvät muun muassa kulmiin, suuruusvertailuihin, pisteisiin ja janoihin. Näiden lisäksi Boyer kertoo Eukleideen perehtyneen geometrian ja algebran lisäksi muihin matemaattisiin lainalaisuuksiin, mutta kaikki kirjoitukset eivät ole säilyneet nykypäivään.

Alestalo (2019, kalvo 3) kuvailee euklidista avaruutta n-vektoriavaruudeksi, jossa voidaan yleistää R^n-joukolle etäisyyksien tavanomaisia käsitteitä. Hoffman (2013, s. x, 16) toteaa saman, ja lisäksi sen, että R^n-joukon pisteitä voidaan myös kutsua vektoreiksi. Molemmat Alestalo ja Hoffman esittelevät teksteissään euklidiseen avaruuteen liittyviä aksioomia, joiden avulla voidaan määritellä vektoreille laskusääntöjä. Näiden avulla päästään perehtymään tarkemmin euklidisiin etäisyyksiin.

Euklidinen etäisyys on euklidisessa avaruudessa olevan kahden pisteen välisen matkan pituus. Etäisyydelle lasketaan itsenäisarvo, jolloin etäisyyden arvoksi tulee nolla tai positiivinen luku (Smith, 2013, s. 8). Etäisyys voidaan laskea yhden tai useamman ulottuvuuden pisteille. Yhden ulottuvuuden euklidinen etäisyys on jana kahden pisteen välillä kaavan 1 mukaisesti (Smith, 2013, s. 8).

$$d = | x_2 - x_1| \tag{1}$$

Kahden ulottuvuuden euklidinen etäisyyden laskeminen kahden pisteen välillä nojaa pythagoraan lauseeseen

$$a^2 + b^2 = c^2. \tag{2}$$

Lausetta muunnetaan niin, että selvitetään hypotenuusan pituus (Peda.net, n.d.), jolloin lause tulee muotoon

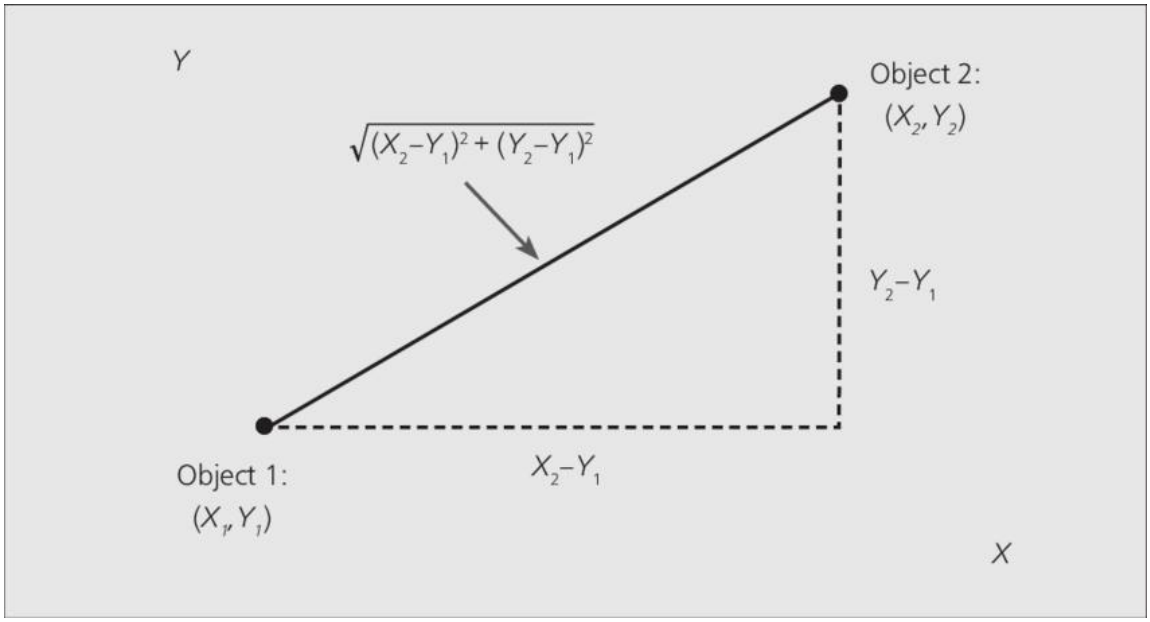
$$c = \sqrt{a^2 + b^2}. \tag{3}$$

Kun yhtälöön korvataan vastaavat etäisyydet kaksiulotteisessa avaruudessa, euklidisen etäisyyden lauseke kaksiulotteisessa avaruudessa on

$$d(x, y) = \sqrt{x^2 + y^2} \tag{4}$$

ja sama kateettien pituudet tarkennettuna

$$d(x, y) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \tag{5}$$



Kuva 1. Esimerkki euklidisesta etäisyydestä. (Park & Lee, 2011, Figure 1)

Kuvassa 1 on esimerkki euklidisesta etäisyydestä kaksiulotteisessa avaruudessa. Object 1 ja Object 2 ovat pisteet euklidisessa avaruudessa, ja kiinteä suora näiden pisteiden välillä on hahmotellun kolmion hypotenuusa eli kahden valitun pisteen euklidinen etäisyys. Kuvan 1 esimerkki on XY-koordinaatistossa. Vastaavasti kolmiulotteisessa avaruudessa, kahden pisteen euklidinen etäisyys voidaan esittää XYZ-koordinaatistossa, ja samalla logiikalla yhä moniulotteisimmille avaruudellisille pisteiden etäisyyksille karteesisessa muodossa.

Kolmiulotteisille ja suuremmille ulottuvuuksille euklidisen etäisyyden kaava voidaan esittää muodossa

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{6}$$

(Krishnan ja muut, 2011, s. 423) (Laksito ja muut, 2022, s. 387). n on ulottuvuuksien määrä. McMeekinin ja muiden (2014, s. 964) mukaan euklidinen etäisyys voi olla muihin menetelmiin, kuten city blockin tai chessboardin etäisyyteen, verrattuna työläämpi

laskea. Kuitenkin euklidisen etäisyyden lause antaa lyhyimmän ja täsmällisimmän etäisyyden kahden pisteen välille.

Epäeuklidinen avaruus ei kaikissa tapauksissa noudata samoja matemaattisia sääntöjä kuin euklidisen avaruuden säännöt. Euklidinen geometria on epäeuklidiseen verrattuna yksinkertaisempaa ja suoraviivaisempaa. Epäeuklidinen geometria ottaa huomioon muun muassa erilaiset käyrät ja niiden ominaisuudet. Epäeuklidisessa avaruudessa ja siihen liittyvissä laskutoimituksissa esiintyy enemmän epävarmuutta ja estimointia. (Lu ja muut, 2012, s. 1–4)

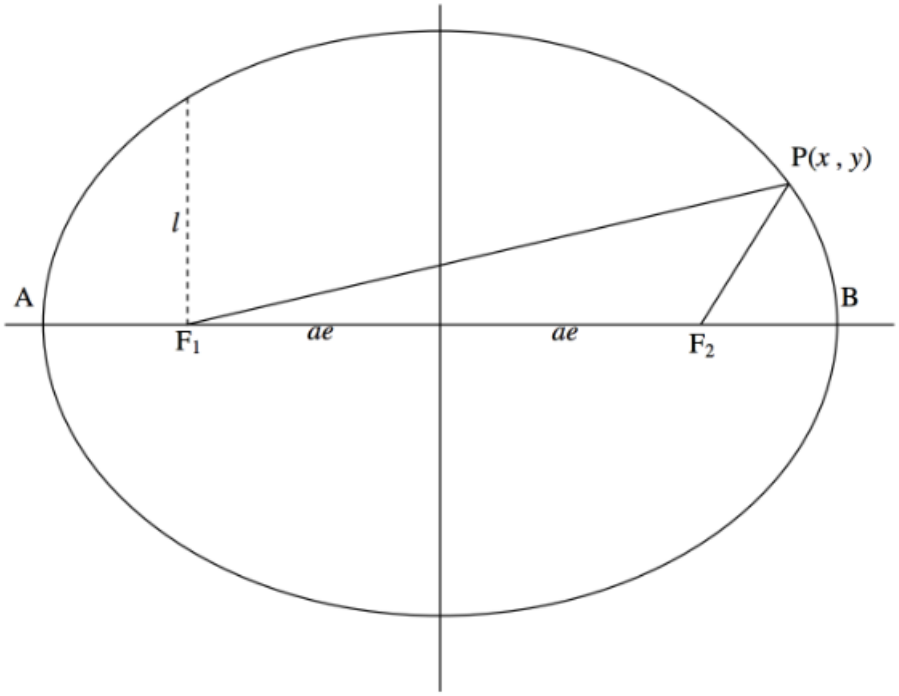
Elliptinen geometria kuuluu epäeuklidisten geometrioiden lajeihin, mutta euklidisen geometrian menetelmiä voidaan hyödyntää elliptisessä geometriassa (Voronova & Tsarena, 2022, s. 493). Elliptiseen geometriaan kuuluvat ellipsit ja ellipsoidit. Voronkova & Tsarena (2022, s. 493) esittävät tutkimuksessaan, että esimerkiksi datan klusteroinnissa ja luokittelussa käytetään epäeuklidisen avaruuden kappaleille euklidisen avaruuden menetelmiä onnistuneesti, sillä epäeuklidisen avaruuden matemaattisille menetelmille ei ole aina täsmällisiä ratkaisuja. Euklidista etäisyyttä käytetään pisteiden lyhimmän välimatkan löytämiseksi ja mittaamiseksi.

Vasile-George (2014, s. 778) toteaa, että euklidinen etäisyys sopii hyperellipsoidin luomiseen, kunhan pisteiden ryhmittymiä on enintään yksi. Tämän työn tavoitteena ei ole luoda hyperellipsoidia koko ryhmittymän perusteella, eikä ole tarvetta monelle klusterin jaottelulle, vaan luoda hyperellipsoidi ja tarkastella pistejoukkoa sen perusteella. Eli määritelmä on tavoitteen mukainen.

Elliptinen geometria ja elliptiset käyrät ovat keskeisiä useissa tietotekniikan ja matematiikan sovelluksissa ja visualisoinneissa, kuten kryptografiassa ja tietojärjestelmien analyyseissä (Pote ja muut, 2016, s. 1890) (Starodubov ja muut, 2023, s. 802). Vaikka elliptiset käyrät pohjautuvat epäeuklidiseen geometriaan, niiden soveltaminen voidaan monissa tapauksissa yksinkertaistaa käyttämällä euklidista geometriaa.

2.2 Ellipsit

Ellipsin piirtämiseksi on perinteinen menetelmä, jossa käytetään kahta naulaa, lankaa ja kynää. Langan päät kiinnitetään kahteen naulaan, ja kynällä piirretään kehä lanka jännitettynä sen rajaamalta alueelta. Alla kuvassa 2 esimerkki.



Kuva 2. Ellipsin muodostaminen ja piirtäminen (Tatum, n.d.).

Kuvassa 2 F_1 ja F_2 kuvastavat ellipsin polttopisteitä eli esimerkin nauloja, ja P(x, y) pistettä ellipsin kehällä. Polttopisteet sijaitsevat yhtä kaukana keskipisteestä. Etäisyys polttopisteistä kehällä sijaitsevaan pisteeseen on vakio, eli piirtämisesimerkin lanka. Polttopisteistä kehän pisteeseen etäisyyksien summa pysyy aina samana. Etäisyys voidaan selvittää euklidisen etäisyyden määritelmän avulla, ja yhtälöstä saadaan myös ellipsin yhtälö. (Tatum, n.d.)

Ellipsin piirtäminen onnistuu, vaikka kaikkia lähtötietoja ei olisi valmiiksi annettu, vaan ne on mahdollista päätellä muista tiedoista. Ellipsin piirtämiselle on olemassa muitakin menetelmiä, mutta naruesimerkki on yksi perinteisimmistä.

2.2.1 Ellipsin määritelmä

Kaavassa 7 on kaksiulotteisen ellipsin määritelmä.

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1 \tag{7}$$

Ylläolevassa yhtälössä a ja b tarkoittavat ellipsin puolipituusakseleita, a x-akselilla ja b y-akselilla, ja lisäksi a kuvaa suurempaa ja b pienempää ellipsin puolipituusakselia. Ellipsin eksentrisyydellä tarkoitetaan, kuinka lähellä tai kaukana polttopisteet ovat ellipsin keskipistettä. (Kivelä, n.d.)

Kun kaksi pistettä on annettu, niistä voidaan laskea pisteiden keskipiste. Kahden pisteen ja keskipisteen perusteella yhdistäen euklidisen etäisyyden ominaisuuksiin voidaan laskea suuri ja pieni puolipituusakseli.

2.2.2 Piste ellipsin sisäpuolella

Piste on ellipsin kehällä tai sen sisäpuolella, kun

$$\mathrm{P}F_1 + \mathrm{P}F_2 \leq 2a, \tag{8}$$

jossa P on piste, F on polttopiste ja a on suurempi pituusakseli. Eli jos pisteen etäisyyksien summat ellipsin polttopisteisiin on yhtä suuri tai pienempi kuin suuri pituusakseli, piste sijaitsee ellipsin kehällä tai sisäpuolella. Kun yhtälön molemmat puolet ovat tasan yhtä suuret, piste sijaitsee ellipsin kehällä, muuten ellipsin sisäpuolella.

2.2.3 Hyperellipsi

Hyperellipsi on geometrinen moniulotteinen pinta, joka on kuin ellipsin ja ellipsoidin välimalli. Verrattuna ellipsi-käsitteeseen, hyperellipsi on yleiskäsitteisempi, ja ellipsi koostuu enemmän vakioista. Hyperellipsille voidaan kuitenkin yleistää monia ellipsin määritelmiä ja hyödyntää niitä, kun hyperellipsi sovitetaan kaksiulotteiseen malliin.

$$\left(\frac{x}{a}\right)^k + \left(\frac{y}{b}\right)^m = 1 \tag{9}$$

Kaavassa 9 on hyperellipsin määritelmä (Wood Mt. Design, LLC, n.d.). Määritelmä on muuten kuin ellipsissä, mutta potenssit *k* ja *m* tekevät yhtälöstä moniulotteisemman kuin ellipsin potenssiin kaksi.

Huangin (2013) dataluokitteluun liittyvä algoritminkehittelytutkimus käyttää hyperelliptistä metodia. Tutkimuksessa tarkasteltiin tulosten tarkkuutta ja yhtenäisyyttä. Ilmeni, että jo kahden ellipsin yhdistelmää hyödyntämällä saadaan riittävä onnistumisaste luokittelussa. Hyperellipsin määritelmä ja laskutoimitukset nojaavat suuresti ellipsin ja euklidisen etäisyyden ominaisuuksiin.

2.3 Ellipsoidit

Ellipsi voidaan ajatella ellipsoidin läpileikkauksena. Ellipsi on kaksiulotteinen, ja ellipsoidi on kolmiulotteinen kappale. Ellipsin ominaisuudet ovat ellipsoidiin verrattuna samankaltaiset, mutta akseleita on yksi lisää. Akselin lisääminen lisää koordinaatistoon z-akselin, ja z-koordinaatiston lisäämisen myötä myös puolipituusakselien määrä on kolme. Kolmatta puolipituusakselia merkitään c-kirjaimella, ja sen pituus suhteessa muihin puolipituusakseleihin määrää ellipsoidin muodon.

2.3.1 Ellipsoidin määritelmä

Ellipsoidi on muoto xyz-koordinaatistossa. Kaavassa 10

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2 = 1 \tag{10}$$

on ellipsoidin määritelmä. Yhtälössä a, b ja c ovat xyz-koordinaatiston puolipituusakselit. (Han ja muut, 2019, s. 7)

Yksi ellipsoidin erikoistapaus on pyörähdysellipsoidi. Pyörähdysellipsoidi muodostuu, kun ellipsoidi käännetään sen pääakselin ympäri. Tästä seuraa, että pyörähdysellipsoidin puolipituusakselit ovat yhtä suuria lukuun ottamatta pyöräytettävän akselin suuntaista puolipituusakselia. Esimerkiksi x-akselin suunnassa pyöräytettävän hyperellipsoidin b ja c puolipituusakselit ovat yhtä suuria. Tässä tutkimuksessa käytetään ellipsoidina pyörähdysellipsoidia.

2.3.2 Piste ellipsoidin sisäpuolella

Yhtälö pisteen sijainnin tarkistamiseksi ellipsoidissa on lähes sama kuin ellipsoidin määritelmä. Kaavassa 11

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2 \leq 1 \tag{11}$$

yhtälön ollessa tasan 1, piste sijaitsee ellipsoidin kehällä. Jos tulokseksi saadaan pienempi kuin 1, piste sijaitsee ellipsoidin sisällä. (Han ja muut, 2019, s. 7)

2.4 Hyperellipsoidit

Hyperellipsoidi on ellipsoidin n-ulotteinen erikoistapaus (Kesäniemi & Virtanen, 2017, s. 63, 66). Kun ellipsoidi on kolmeulotteinen kappale, hyperellipsoidi on ellipsoidia vielä moniulotteisempi kappale.

Vasile-Georgen (2014, s. 778) mukaan euklidista etäisyyttä voidaan käyttää luomaan hyperellipsoidin muotoinen alue. Erityisesti sellaisissa tapauksissa, kun pistejoukko rajataan yhteen klusteriin.

2.4.1 Hyperellipsoidin määritelmä

Hyperellipsoidi on ellipsoidin n-ulotteinen tapaus, ja tämä näkyy vertaamalla hyperellidsoidin määritelmää kolmiulotteisen ellipsoidin määritelmään. Puolipituusakselien määrä riippuu suoraan ulottuvuuksien määrästä. Kaavassa 12

$$\left(\frac{x_1}{a_1}\right)^2 + \left(\frac{x_2}{a_2}\right)^2 + \cdots + \left(\frac{x_n}{a_n}\right)^2 = 1 \tag{12}$$

a_1-a_n kuvaavat puolipituusakseleita.

Kuten pyörähdysellipsoidi on ellipsoidin erikoistapaus, pyörähdyshyperellipsoidi on hyperellipsoidin erikoistapaus. Kun hyperellipsoidi käännetään sen pääakselin ympäri, muodostuu pyörähdyshyperellipsoidi. Pyörähdyshyperellipsoidin puolipituusakselit ovat yhtä suuria lukuun ottamatta pyöräytettävän akselin suuntaista puolipituusakselia. Pyörähdysellipsoidi säilyttää moniulotteisuutensa, mutta on määriteltävissä kolmen pisteen avulla.

2.4.2 Piste hyperellipsoidin sisäpuolella

Yhtälö pisteen sijainnin tarkistamiseksi hyperellipsoidissa perustuu hyperellipsoidin määritelmään, ja määritelmä on samankaltainen ellipsoidin vastaavan yhtälön kanssa. Kaavassa 13

$$\left(\frac{x_1}{a_1}\right)^2 + \left(\frac{x_2}{a_2}\right)^2 + \cdots + \left(\frac{x_n}{a_n}\right)^2 \leq 1 \tag{13}$$

yhtälön ollessa tasan 1, piste sijaitsee hyperellipsoidin kehällä. Jos tulokseksi saadaan pienempi kuin 1, piste sijaitsee hyperellipsoidin sisällä.

3 Tutkimusmenetelmä, satunnaisjakaumien taustoitus, alkuasetelma

Lähtökohtana on lähestyä tutkittavaa aihetta teorian pohjalta ja tehdä määritelmät, sekä miettiä keinoja asteittain saavuttaa tavoitteet. Tässä diplomityössä käytetään suunnitteluteoreettista tutkimusmenetelmää, sillä se sopii ongelmalähtöiseen ratkaisun löytämiseen (vom Brocke ja muut, 2020, s. 1).

Tutkimusongelmana on yleisluontoinen asetelma, jossa halutaan tutkia hyperelliptisesti jakautunutta satunnaismuuttujien joukkoa. Joukolle asetetaan muuteltava todennäköisyys T, kuinka todennäköisesti pisteet levittäytyvät muodon sisäpuolelle. Tutkimuksen tavoitteena on tutkia, löytyykö keinoa luoda asetelma, tarkastella joukkoa, yleistää määritelmä ellipseille, ellipsoideille ja hyperellipsoideille, sekä onnistuuko toteutus suhteellisen vähäisillä resursseilla.

3.1 Suunnittelutieteellinen tutkimusmenetelmä

Kaksi yleisintä tutkimusmenetelmää ovat laadullinen ja määrällinen tutkimusmenetelmä (Abbadia, 2023), mutta kumpikaan näistä menetelmistä ei kokonaan vastaa tämän työn tutkimusongelmalle asetettuja tavoitteita. Raisamo (2006) toteaa, että laadullisessa tutkimuksessa jätetään varaa tulkinnoille, ja se sopii enemmän ilmiöiden luokitteluun kuin numerelliseen dataan. Määrällisestä tutkimuksesta hän toteaa, että siinä tutkimusmenetelmässä käytetään jo ennalta mitattuja arvoja ja aineistoa, tai muita valmiita ratkaisuja, eli sitä ei varsinaisesti käytetä luomaan uusia keinoja.

Joten, työssä käytetään suunnittelutieteellistä tutkimusmenetelmää. Vom Brocken ja muiden (2020, s.2–3) mukaan suunnittelutieteellisen tutkimusmenetelmän avulla voidaan kehittää täysin uudenlaisia ratkaisuja ja malleja, juuri sellaisia, kuin niiden tulisi olla käyttötarkoitus huomioon ottaen. Peffersin ja muiden (2007, s. 49) mukaan

suunnittelutieteellisen tutkimusmenetelmän mukaisesti ratkaisuna luotu artefakti on olemassa varta vasten tutkimusongelmaa varten. Ratkaisun tavoite saavutetaan hyödyntämällä viitekehystä ja käsitteitä.

Suunnittelutieteellinen tutkimusmenetelmä sopii ongelmalähtöiseen tutkimukseen, ja se voisi olla myös hyvä menetelmä tähän tutkielmaan. Peffersin ja muut (2007, s. 47) mainitsevat tutkimusmenetelmän yleisluontoisesti sopivaksi tietojärjestelmällisten tutkimusten lisäksi insinöörialan ja tietotekniikan tutkimuksiin, kunhan tutkimuksessa käytetään täsmällisiä ja sopivia määrityksiä. Myös Venable ja muut (2017, s. 2) tukevat ajatusta suunnittelutieteellisen tutkimusmenetelmän sopimisesta tietotekniikan alan tutkimuksiin.

Yhdistelemällä jo olemassaolevien sekä uusien tekniikoiden ja keinojen avulla tutkitaan, saavutetaanko tutkimuksen ongelmaan ratkaisu, ja miten se onnistuu. Tämä muihin tutkimusmenetelmiin verrattuna hieman uudempi, mutta vakaalla pohjalla oleva, menetelmä tarjoaa ratkaisuja jopa luoviin ja monimuotoisiin ongelmiin (Peffers ja muut, 2007, s. 49). Edellä esiteltyjen seikkojen lisäksi Venable ja muut (2017, s. 2) esittelevät suunnittelutieteellisen menetelmän sopivan myös aiempien menetelmien paranteluun.

Taulukko 1. Suunnittelutieteellisen menetelmän osiot. (Venable ja muut, 2017, taulukko 3. Peffers ja muut, 2007, taulukko 1.)

Tutkimusvaihe	Kiinnonkohteet	Ennen seuraavaa vaihetta
1. Ongelman määrittely ja tavoite	**Ongelman määrittely, tärkeyden osoittaminen.**	**Määritelmät ja päätelmät.**
2. Ratkaisun tavoitteet	**Nykyisen tilanteen parantaminen.**	**Teoria.**
3. Suunnittelu ja kehittely	**Tuotos, artefakti.**	**Etenemissuunnitelma, mitä tehdään seuraavaksi.**
4. Havainnollistaminen	**Sopivan asiayhteyden käyttäminen, artefaktin esittely ja käyttö.**	**Mittaukset ja analyysi.**
5. Arviointi	**Tehokkuuden tarkastelu. Tarvittaessa palataan suunnittelu- ja kehitysvaiheeseen ja toistetaan.**	**Pääpiirteet ja periaatteet.**
6. Viestintä	**Julkaisut.**	

Taulukossa 1 on mukailtuna Venablen ja muiden (2017) koostama taulukko suunnittelutieteellisen tutkimuksen vaiheista. Kuusivaiheinen taulukko on tehty Peffersin ja muiden (2007) esittelemien vaiheiden pohjalta. Taulukon vaiheet ovat kokonaisuuksia, joiden yksityiskohdat tarkennetaan oman tutkimuksen tavoitteilla ja määreillä. Aloittamalla eri vaiheista saadaan eri lähestymisnäkökulmia tutkimukseen, kuten ongelma-, tavoite- tai asiakaslähtöinen näkökulma.

3.1.1 Ongelman määrittely ja tavoite

Tämän tutkimuksen ongelma on hyperelliptiseen muotoon jakautunut joukko, joiden satunnaismuuttujien jakaumaa tarkastellaan. Joukon jakauman käyttäytymistä tarkastellaan ajan edetessä ja määrettä todennäköisyys T muutettaessa. Joukon arvojen vaihteluväli tiedetään, ja todennäköisyyden arvoa pystytään säätämään. Hyperelliptinen muoto piirretään määrättyjen pisteiden avulla. Iterointikierroksilla toistuu näiden pisteiden perusteella hyperellipsin hahmottelu ja pisteen korvaaminen uudella satunnaisella pisteellä.

Datamäärä on suuri, jolloin manuaalinen ratkaiseminen olisi aikaa vievää ja tehotonta työskentelyä. Ohjelmoimalla asetelma ja satunnaisjoukon käyttäytymisen havainnollistus säästetään resursseja ja nopeutetaan prosessia. Myös erilainen testailu ja uusien versioiden kokeilu ja vaihtaminen on sujuvampaa. Lisäksi ohjelmoiminen mahdollistaa työvaiheen helpon jakamisen muiden käyttöön. Tämän työn asetelmaa kohdataan esimerkiksi neuroverkkojen yhteydessä alkuasetelmana. Mahdollisesti saavutettua ratkaisua voidaan hyödyntää muissa töissä ja ohittaa ohjelmoinnin alustus. Jakamalla ohjelma muiden käyttöön voidaan saada lisää optimoituja ratkaisuja, kun myös muut voivat muokata uusia versioita ohjelmasta.

3.1.2 Ratkaisun tavoitteet

Päätavoitteen mukaisesti tutkimuksessa pyritään löytämään ongelmaan ratkaisuksi sopiva algoritmi. Ratkaisussa on pystyttävä tarkastelemaan ja säätämään satunnaisjakauman joukkoa, ja tavoitteena on löytää käytännöllinen ratkaisu. Tämä tarkoittaen algoritmia, joka voidaan suorittaa käyttäen tavallista tietokonetta. Erikoistietokoneen hankintaan ja käyttöön ei ole mahdollisuutta tämän työn osalta, ja se kaventaisi muiden mahdollisuuksia hyödyntää algoritmia.

Ratkaisun tavoitteena on koota algoritmi, jossa todennäköisyyttä T muokkaamalla pystytään määrittelemään joukon suppeneminen, samana pysyminen tai laajeneminen. Alkuasetelmaan pitää pystyä asettamaan alussa tietyt arvot, kuten joukon arvojen vaihteluväli ja todennäköisyys T. Kokeessa käytetään iterointia, eli ratkaisussa pitää pystyä toistamaan tiettyjä vaiheita.

Ratkaisun selvittäminen aloitetaan pohjustamalla teoriaa. Tässä työssä käydään läpi euklidisten etäisyyksien taustateoriaa sekä ellipsien, hyperellipsien, ellipsoidien ja hyperellipsoidien määritelmiä kappaleessa 2. Satunnaisjakaumaan liittyvä teoria tulee esille myöhemmin tässä kappaleessa.

3.1.3 Suunnittelu ja kehittely

Teorian läpikäynnin jälkeen suunnitellaan tutkimuksessa eteneminen. Mitä tarvitaan alkuasetelmaa varten, entä kokeen aikana ja tulosten tallentamista varten. Miten saavutetaan tutkimusongelman ratkaisu.

Tämän työn tavoitteena on algoritmi, joten sen hahmotteleminen aloitetaan kirjoittamalla kaikki algoritmin vaiheet pseudokielellä ja muilla havainnollistavilla keinoilla. Pyritään saamaan kokoon kaikki tarvittavat osat, ja tarkistetaan, onko kaikki tarvittavat tiedot saatu alkuasetelmaa varten. Katsotaan, mikä on algoritmin etenemisjärjestys ja sen merkitys. Tulosten talteenottaminen, ja määritellään mitä niistä halutaan katsoa ja vertailla. Kappaleessa 4 on tarkemmin pseudokielisen ohjelman sisältö, ja kappaleessa 5 on koeasetelma.

3.1.4 Havainnollistaminen

Tutkimuksen artefaktina toimii algoritmi tai ohjelma, jolla saavutetaan ongelmaan ratkaisu. Havainnollistamiseen kuuluu ohjelman toiminnan ja implementoinnin esittelyn lisäksi perustelu, miksi valittua ohjelmaa käytetään ja sen toiminnan esittely. Esittelyiden tulee olla tarpeeksi selkeitä, jotta kokeen toistettavuus on mahdollista jälkeenpäin itse ja muiden toteuttamana.

Algoritmin sisällön lisäksi toiminnan tulos tulee saada tarkasteltavaan muotoon, esimerkiksi kuvana, käyränä tai numeroina. Näkymättömät osiot selvitetään muilla keinoin, ja tarpeeton poistetaan. Tuloksia vertailemalla teoriaan ja tavoitteisiin päätellään datan todenmukaisuus.

3.1.5 Arviointi

Arviointivaiheessa tarkastellaan työn tehokkuutta ja tavoitteisiin pääsyä. Jos tehokkuudessa tai ratkaisussa todetaan parantamisen varaa, palataan suunnittelu- ja kehittelyvaiheeseen ja toistetaan osiot arviointivaiheeseen saakka.

Erityisenä arvioinnin kohteena tässä työssä on todennäköisyys T:n vaikutus jakaumaan. Ohjelman suoriutumista ja tuloksia arvioidaan varmasti jo ennen tähän vaiheeseen pääsyä, ja tarvittavia muokkauksia tehdään havaintojen mukaan. Loppuvaiheilla kuitenkin arvioidaan kokonaisuutta.

3.1.6 Viestintä

Teoria, tulokset, päätelmät ja johtopäätökset esitellään tässä diplomityössä. Jokainen suunnittelutieteellisen tutkimuksen vaihe esitellään järjestyksessä. Taulukossa 2 on vielä tiivistettynä tämän työn suunnittelutieteellisen tutkimuksen vaiheet Venablen ja muiden (2017) osioiden mukaiseen muotoon.

Taulukko 2. Suunnittelutieteellisen tutkimuksen vaiheet tässä työssä.

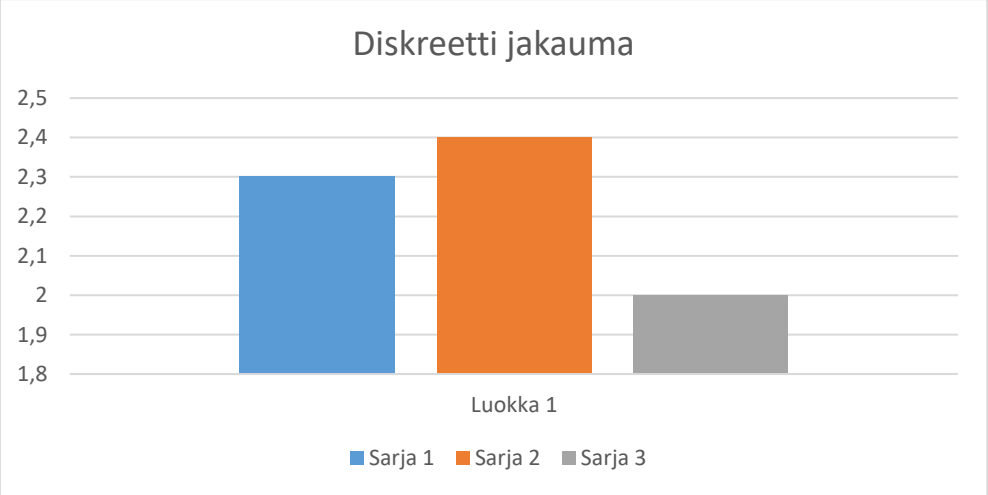
Tutkimusvaihe	Keskeiset asiat tiivistettynä
1. Ongelman määrittely ja tavoite	**Hyperelliptinen satunnaisjoukko jakaumalla. Halutaan tarkastella ja havainnollistaa tehokkaalla menetelmällä.**
2. Ratkaisun tavoitteet	**Tarkoituksenmukainen algoritmin koostaminen ja sopiva ohjelma algoritmin toteuttamiseksi sekä tulosten tallentamiseksi.**
3. Suunnittelu ja kehittely	**Alku- ja koeasetelman määrittely.**
4. Havainnollistaminen	**Ohjelman käyttö, suoritetaan ohjelma ja tallennetaan saavutetut tulokset.**
5. Arviointi	**Tarkastellaan tuloksia ja analysoidaan, tarvittaessa muokkauksia ja iterointia.**
6. Viestintä	**Kaikki koostetaan tähän diplomityöhön.**

3.2 Satunnaisjakaumat

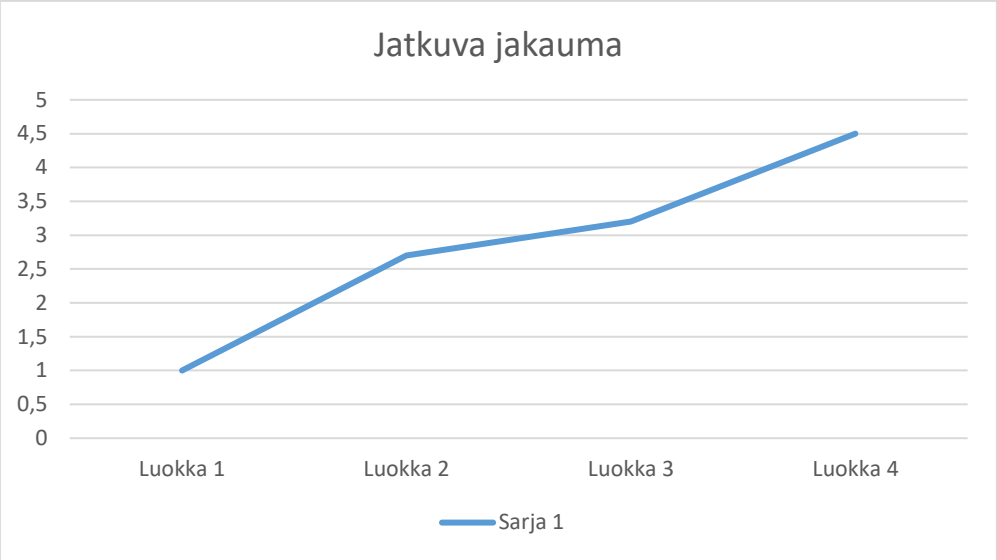
Satunnaisjakauma on todennäköisyysjakauma, jonka muuttujat ovat satunnaismuuttujia. Todennäköisyys kuvastaa mahdollisuutta, jolla piste sijaitsee tarkasteltavalla alueella. Todennäköisyysteorian avulla pystytään arvioimaan tai ennustamaan epävarman tapahtuman mahdollisuus tai määrä (Rajan, 2012).

Todennäköisyysjakaumat jaotellaan satunnaismuuttujien ominaisuuksien mukaan kahteen pääryhmään, jatkuvuuden ja jakaumalle asettumisen mukaan jatkuviin ja diskreetteihin jakaumiin. Lisäksi Hsu (1997, s. 39) määrittelee teoksessaan jakaviin ominaisuuksiin satunnaismuuttujien äärellisyyden jakaumalla ja esittelee jatkuvalle ja diskreetille jakaumalle sopivat funktiot. Toisaalta samaan dataan voidaan käyttää molempien jakaumien tekniikoita löytämään optimaalisin tulos, kuten Wallhoffin ja muiden (2001) tutkimuksessa.

Kuvassa 3 on esimerkki, miltä diskreetti jakauma voi näyttää. Piste kuuluu tai ei kuulu joukkoon, ja joukon määrä on laskettavissa (Shynk, 2012, s. 51).



Kuva 3. Esimerkki diskreetistä jakaumasta.



Kuva 4. Esimerkki jatkuvasta jakaumasta.

Kuvassa 4 on esimerkki siitä, miltä jatkuva jakauma voi näyttää. Pisteiden avulla voi hahmotella käyrän ja arvioida ilmoittamattomia arvoja pisteiden välillä. Jakauma voi olla myös määritelty jatkuvaksi vain tietyllä alueella. Shynk (2012, s. 104) mainitsee, että jakauma voi olla sekoitus diskreettiä ja jatkuvaa osiota. Datalla voi olla numeroitavia ja ei-numeroitavia ominaisuuksia.

3.2.1 Satunnaisjakauma neuroverkoissa

Neuroverkkojen mallinnuksessa on otettu mallia aivojen neuronien toiminnasta. Neuroverkkoja hyödynnetään tekniikassa ja tieteessä laajalti erilaisissa malleissa ja oppimisalgoritmeissa. Satunnaisjakaumaan perustuvia neuroverkon malleja käytetään luokittelussa, tunnistamisessa, kuvankäsittelyssä, viestinnässä ja paljon muussa. (Timotheou, 2009, s. 1)

Satunnaisjakaumat ovat tärkeä osa neuroverkkoja, sillä niitä käytetään neuroverkkojen kouluttamisessa ja optimoinnissa. Neuroverkkojen koulutuksessa satunnaisjakaumia käytetään määrittelemään satunnaiset aloitusparametrit tai alustamaan neuroverkkojen painot. Tähän voidaan käyttää esimerkiksi normaalijakaumaa tai tasajakaumaa, erikseen

tai yhdistelmänä, riippuen tavoitteesta ja matemaattisista odotusarvoista (Hu ja muut, 2022).

Normaalijakaumalla voidaan alustaa neuroverkkojen painot satunnaisina, mutta samalla tasapainottaa niiden alkuarvoja. Tätä voidaan hyödyntää koulutusprosessissa ja optimaalisimman ratkaisun löytämisessä.

Tasajakaumaa käytetään, kun neuroverkkomallin alkuarvoissa tarvitaan laajempi vaihtelu, ja alkuarvoja ei tarvitse tasapainottaa. Tasajakauma voi olla hyödyllinen erityisesti sellaisissa tilanteissa, joissa halutaan varmistaa, että painot ovat alussa jakautuneet laajalle alueelle, mikä voi auttaa verkkoa oppimaan monimutkaisempia kuvioita ja ominaisuuksia.

Satunnaisjakaumat tarjoavat tehokkaan tavan aloittaa neuroverkkojen koulutusprosessi, joka optimoi verkkojen suorituskyvyn ja koulutuksen tehokkuuden. Nämä lähestymistavat auttavat varmistamaan, että verkot voivat oppia ja sopeutua monimutkaisiin tehtäviin ja datamalleihin parhaalla mahdollisella tavalla.

3.2.2 Geometrinen todennäköisyys

Geometrinen todennäköisyys on todennäköisyysteoria, jossa tarkastellaan satunnaisilmiöitä jatkuvassa tilassa, kuten tasossa tai avaruudessa. Perinteisesti se liittyy ongelmiin, joissa todennäköisyys määritetään pinta-alojen, tilavuuksien tai muiden geometristen mittasuhteiden perusteella. Tällaisia ongelmia esiintyy esimerkiksi satunnaisten pisteiden jakautumisessa tietyllä alueella tai etäisyyksien jakauman analysoinnissa.

Hyperelliptisen satunnaisjakauman algoritmin kehittämisessä geometrinen todennäköisyys voidaan käyttää työkaluna analysoitaessa pisteiden ja jakaumien ominaisuuksia monimutkaisessa geometrisessa rakenteessa. Tällä voidaan tutkia, miten satunnaisesti valitut pisteet sijoittuvat hyperelliptiselle käyrälle tai sen osajoukkoihin, ja miten niiden

todennäköisyysjakaumat käyttäytyvät jatkuvassa tilassa. Tällainen analyysi on erityisen hyödyllistä, kun pyritään kehittämään tehokkaita algoritmeja, jotka säilyttävät tietyt matemaattiset ominaisuudet satunnaisprosessien yhteydessä.

Lin ja muiden (2022, s. 228) mukaan geometristä todennäköisyyttä voidaan käyttää sekä yksinkertaisiin, että monimutkaisiin malleihin, kuten teoreettisiin malleihin, simulointeihin, suunnitteluun tai ohjelmistoissa. Li ja muut täsmentävät geometrisillä todennäköisyyksillä ja sen datalla saavutettavan geometrisiä ratkaisumalleja todennäköisyysongelmiin, ja niillä lisättävän ymmärrystä satunnaisuuteen, suhteelliseen tasaisuuteen ja simulointiin.

3.2.3 Todennäköisyyspinta

Todennäköisyyspinta on matemaattinen käsite, joka kuvaa muuttujien satunnaisjakauman tiheysfunktion muotoa jatkuvassa tilassa. Se kuvaa todennäköisyyden jakaantumista eri ulottuvuuksissa, jolloin korkeus tietyssä pisteessä kertoo kyseisen pisteen suhteellisen esiintymistodennäköisyyden. (Hauser ja muut, 2007) Todennäköisyyspintoja hyödynnetään erityisesti tilanteissa, joissa halutaan visualisoida ja analysoida satunnaisjakauman rakenteellisia ominaisuuksia, kuten symmetriaa, huippuja tai monimodaalisuutta. Yksinkertaistettuna, kappaleen kehällä sijaitseva piste on kappaleen pinnalla.

Hyperelliptisen satunnaisjakauman algoritmin kehittämisessä todennäköisyyspinnan teoriaa yhdistämällä ellipsin ja ellipsoidin ominaisuuksiin mahdollistetaan tutkiminen monimutkaisessa matemaattisessa ympäristössä. Koska hyperelliptiset käyrät ovat itsessään ei-triviaaleja geometrisia rakenteita, niiden päälle rakentuva satunnaisjakauma voi muodostaa todennäköisyyspintoja, joilla on monimutkainen topologia. Näiden pintojen analyysi voi auttaa ymmärtämään, miten satunnaisuus jakautuu hyperelliptisen käyrän yli, ja optimoimaan algoritmia siten, että se tuottaa haluttuja jakaumaominaisuuksia eri sovelluksissa ja malleissa.

3.3 Alkuasetelman suunnittelu

Algoritmin kehittämisessä on taustalla tavoite monikäyttöisestä ohjelmasta, jota voidaan muokata omien tarpeiden mukaan. Ratkaisun tavoitteena on tarkoituksenmukaisen algoritmin koostaminen ja sopiva ohjelma algoritmin toteuttamiseksi sekä tulosten tallentamiseksi. Lähtökohtana ja testauksessa käytetään määrättyjä alkuarvoja, joita voidaan myöhemmin muokata. Ohjelmassa voidaan myös vaihtaa järjestystä, poistaa tai lisätä osioita.

Kokonaisuudessaan alkuasetelman tavoitteena on luoda perusta hyperelliptisen satunnaisjakauman algoritmin kehittämiselle, määritellä keskeiset seikat sekä antaa yleiskuva käytettävistä menetelmistä, parametreistä ja niiden rajoitteista.

Tavoitteena on ensin algoritmin toteuttaminen elliptiselle hahmotelmalle, ja sen jälkeen ellipsoidille. Ellipsoidi voidaan muodostaa ellipsin pyörähdyskappaleelle, joten siihen ei ole montaa vaihetta lisättäväksi.

3.3.1 Alkuarvot ja -tiedot

Alussa luodaan sadan pisteen joukko, populaatio. Satunnaisesti luodut pisteet asettuvat arvojen [-100, 100] välille. Satunnaismuuttujien joukosta valitaan satunnaisesti kolme pistettä A, B ja C. Alussa määritellään myös parametrit todennäköisyys T ja iteraatiokerrat n, mutta niitä voidaan vapaasti vaihdella testausvaiheessa.

T kuvaa todennäköisyyttä, jolla korvattava piste on määritetyn muodon sisäpuolella. T voi saada arvoja välillä [0, 1]. Jos todennäköisyys on pienempi kuin 0,5, korvatut pisteet luodaan todennäköisemmin muodon ulkopuolelle, eli pistejoukko hajaantuu. Jos todennäköisyys on suurempi kuin 0,5, korvatut pisteet kerääntyvät todennäköisemmin muodon sisäpuolella, eli pistejoukko suppenee. Jos todennäköisyys on 0,5, pistejoukon

hajaantuminen pysyy suurin piirtein ennallaan. Ääriarvoissa 0 ja 1 vaikutus on suurimmillaan.

Iteraatiokerrat n ovat kerrat, joilla piste korvataan ja muoto määritellään uudelleen. Alussa kerraksi voidaan määrittää 1, ja katsoa, kuinka yksi iteraatiokerta vaikuttaa. Kertoja voidaan kasvattaa ensin yksittäin, sitten kymmeniin, satoihin ja edelleen tuhansiin. Kertoja kasvatetaan sen mukaan, kuinka huomattavaa muutosta havaitaan tavoitteeseen nähden. Vähitellen kasvattamalla iteraatiokertoja huomataan muutosten nopeus ja tarve lisäiteraatioille, sekä hallitaan järjestelmän tekniset vaatimukset. Kasvattamalla kertoja liikaa kerralla saatetaan tuhlata resursseja, eikä tuloksia pystytä ottamaan talteen tarkasti ja tehokkaasti.

3.3.2 Satunnaisotanta

Satunnaismuuttujien joukko kuvastaa kaikkia mahdollisuuksia, millä data voi valikoitua ja esiintyä. Couso ja muut (2014 s. 1) korostavatkin, että satunnaisenmuuttujien datan määrittelyn tulee olla täsmällinen lopputuloksen onnistumisen kannalta. Raja-alueet ja erikoistapaukset tulee ottaa huomioon ja tarvittaessa ehtoja täydennetään.

Satunnaisotanta on tilastollinen menetelmä, jossa poimitaan satunnaisesti yksilöitä perusjoukosta niin, että otos edustaa mahdollisimman hyvin koko populaatiota. Satunnaisotannan perusidea on vähentää valintaharhaa ja mahdollistaa tilastollisten päätelmien tekeminen otoksen perusteella. Menetelmiä on useita, kuten yksinkertainen satunnaisotanta, jossa jokaisella perusjoukon alkiolla on yhtä suuri todennäköisyys tulla valituksi, ja systemaattinen otanta, jossa havainnot valitaan tietyin välein populaation joukosta. (Nummenmaa, 2023, s. 26–27)

Tämän työn algoritmissa käytetään satunnaisotantaa, kun populaatiosta valitaan satunnaista pistettä, ja näistä määritellään tarkasteltava muoto. Tämä tehdään uudelleen ja

uudelleen iteraatiokertojen määrän verran. Satunnaisotannassa uusi valinta ei ole riippuvainen edellisestä valinnasta.

Satunnaisotannalla otetuista pisteistä yksi piste korvataan satunnaisella uudella luodulla pisteellä, eli yhtä satunnaisotettua pistettä ei palauteta. Pistettä luodessa käytetään sijainnille todennäköisyyttä T luomaan piste, ja suoritetaan uusi satunnaisotanta.

Geometrinen todennäköisyys liittyy todennäköisyyksien tarkasteluun jatkuvassa tilassa, jossa tapahtumien esiintymistodennäköisyys riippuu pinta-aloista, tilavuuksista tai muista geometrisista mitoituksista. Hyperelliptisten jakaumien tapauksessa tämä tarkoittaa satunnaispisteiden sijoittumista hyperelliptiselle käyrälle ja niiden esiintymistiheyden analysointia. Todennäköisyyspinta puolestaan visualisoi, miten todennäköisyys jakautuu käyrän eri alueille ja millaisia huippuja ja symmetrioita se muodostaa. Satunnaisotannalla otetuista pisteistä riippuu, millainen ja minne ellipsi tai ellipsoidi muodostetaan.

Algoritmien suunnittelussa on keskeistä ottaa huomioon laskennallinen tehokkuus, tarkkuus ja satunnaistamisen laatu. Tässä työssä halutaan tasaisesti jakautunut joukko satunnaismuuttujia, joten satunnaisjakaumana käytetään tasajakaumaa. Tasajakaumaa hyödynnetään alustuksessa, satunnaisesti valittujen pisteiden valinnassa sekä uusien muuttujien luomisessa. Todennäköisyyttä T käytetään uuden muuttujan todennäköisen sijainnin luomisessa, joten painotetuille arvoille ei ole tarvetta.

3.3.3 Algoritmiin liittyvät valinnat

Algoritmiin liittyen on tehty joitain valintoja, joita voidaan vaihtaa myöhemmässä vaiheessa algoritmin tehostamiseksi tai korjaamiseksi. Esimerkiksi ohjelmistoksi on valittu MATLAB, mutta se ei ole välttämätön. Valinta on perustunut aiempaan käyttökokemukseen ja ohjelmiston ennalta arvioituihin käyttömahdollisuuksiin datan käsittelyssä, havainnollistamisessa ja tulosten tallentamisessa. Toinen valinta olisi voitu tehdä

perustuen maksuttomaan vaihtoehtoon. Valittu ohjelmisto voidaan vaihtaa toiseen algoritmin kehittämisen aikana, jos ilmenee tehokkaampi tai käytännöllisempi vaihtoehto.

Myöskään algoritmin muodostaminen ellipsille ei ole välttämätön vaihe. Sen voi jättää pois ja kehittää algoritmin suoraan pyörähdysellipsoidille. Varsinkin jatkotutkimuksissa, jos tarvetta on käsitellä vain pyörähdysellipsoidin tai moniulotteisemman kappaleen osalta algoritmia. Algoritmin kehittämisessä on haluttu pitää lähtökohtana yksinkertaista mallia, jonka sujuessa on laajennettu mallia ja muuta toteutusta. Kun on löydetty joku mahdollinen ratkaisu, voidaan lähteä etsimään optimaalista ratkaisua (Gammel ja muut 2018, s. 968).

Seuraavassa pääluvussa käydään läpi algoritmi vaihe vaiheelta pseudokielellä. Samalla varmistetaan, että alkuasetelmassa on huomioitu kaikki seikat, ja tarvittaessa täydennetään. Alkuasetelman tietoihin lisätään vakiot ja muut arvot, joilla testataan algoritmin toimivuutta.

4 Algoritmin kuvaus pseudokielellä

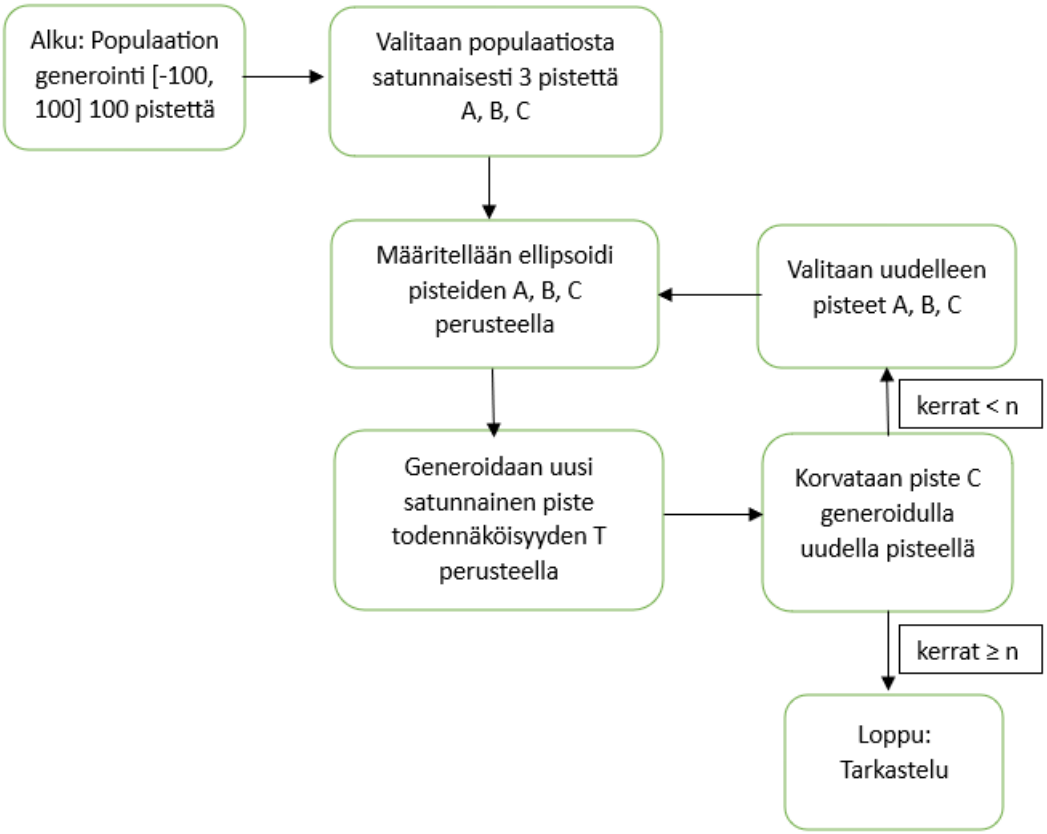
Algoritmin kuvaus ja läpikäynti tehdään ensin ellipsille, sitten pyörähdysellipsoidille. Valitaan kolme pistettä, joista muodostetaan ellipsi tai ellipsoidi. Muodoille on omat piirtofunktiot, muuten menetelmä on molemmille kappaleille pääpiirteittäin sama. Ellipsin muodostaminen on hieman yksinkertaisempi prosessi. Muodostusta voidaan hyödyntää jatkamalla asetelmaa pyörähdysellipsoidille, sillä ellipsoidi on pyörähdyskappale ellipsistä.

4.1 Algoritmi

Ellipsin ja ellipsoidin määrittelyyn ja piirtämiseen tarvitaan kolme pistettä. Näitä pisteitä voidaan kutsua pisteiksi A, B ja C. Pisteet valitaan pistejoukosta satunnaisesti. Pisteet A ja B ovat muodon polttopisteet, ja C on piste kehältä.

Kuvassa 5 on kuvattuna kaaviossa algoritmin kulku kokonaisvaltaisesti vaihe vaiheelta ellipsoidille. Tulevaisuudessa on mahdollisuus muokata algoritmi sopimaan yhä moniulotteisemmille pyörähdyshyperellipsoideille. Yhä moniulotteisemmille ellipsoideille tarvitaan vain kolme pistettä, kun malli on pyörähdyskappale.

Silmukan toistokertojen määrä riippuu siitä, kuinka nopeasti joukon suppeneminen, hajaantuminen tai paikoillaan olo voidaan havaita. Toistokertojen määrä asetetaan alussa, ja voidaan muokata testauksen aikana.



Kuva 5. Algoritmin kulku ellipsoidille.

Ellipsoidin kaavioon kuuluvat seuraavat vaiheet:

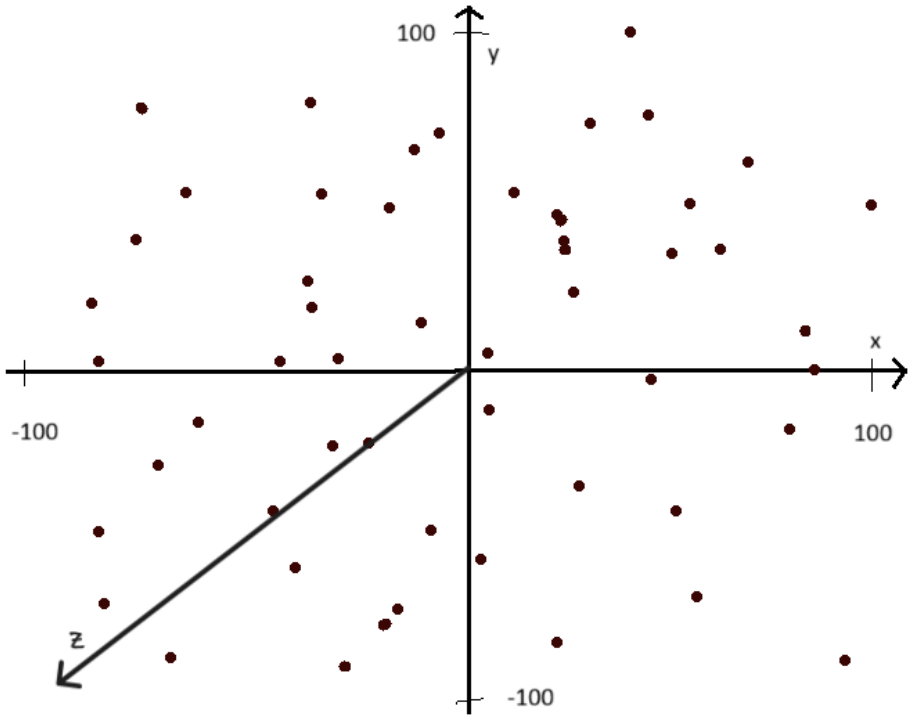
1. Populaation generointi [-100, 100]
2. Valitaan satunnaisesti 3 pistettä A, B, C
3. Määritellään ellipsoidi
4. Generoidaan uusi satunnainen piste
5. Korvataan piste C
6. Toistetaan n kertaa, valitaan uudelleen pisteet A_n, B_n, C_n
7. Kun toistettu n kertaa, tulosten tarkastelu

Näitä kaavion vaiheita toistetaan järjestyksessä, lukuun ottamatta alun populaation generointia. Vaiheet ovat saman molemmille ellipsille ja pyörähdysellipsoidille. Algoritmin vaiheet on koottu teorian pohjalta ja suunniteltu etenemisen kannalta oleellisessa

järjestyksessä. Seuraavaksi jokainen vaihe tarkennetaan ja määritellään erikseen, ja huomioidaan mahdolliset erot ellipsin ja pyörähdysellipsoidin välillä. Kuvissa on algoritmin kaavion vaiheiden mukaisesti piirrokset, joissa vaiheet näkyvät selkeämmin. Kuvat ovat suuntaa antavia, eivät täsmällisiä koon eikä määrän suhteen.

4.2 Populaation generointi, alku

Populaatio on tutkittava pistejoukko. Algoritmin aluksi luodaan sata satunnaismuuttujan pistettä, jotka voivat saada XYZ-koordinaatistossa mitä tahansa arvoja -100 ja 100 välillä. Pisteiden arvojen luomiseen käytetään satunnaislukugeneraattoria ja tasajakaumaa. Tämä alustus tehdään vain kerran, muut lukujen muokkaamiset tehdään yksittäin muussa osassa algoritmia. Näin muiden tutkimusten osalta riittää alkuarvojen muokkaaminen tässä osiossa, jos vaihteluväliä tai määrää halutaan vaihtaa.

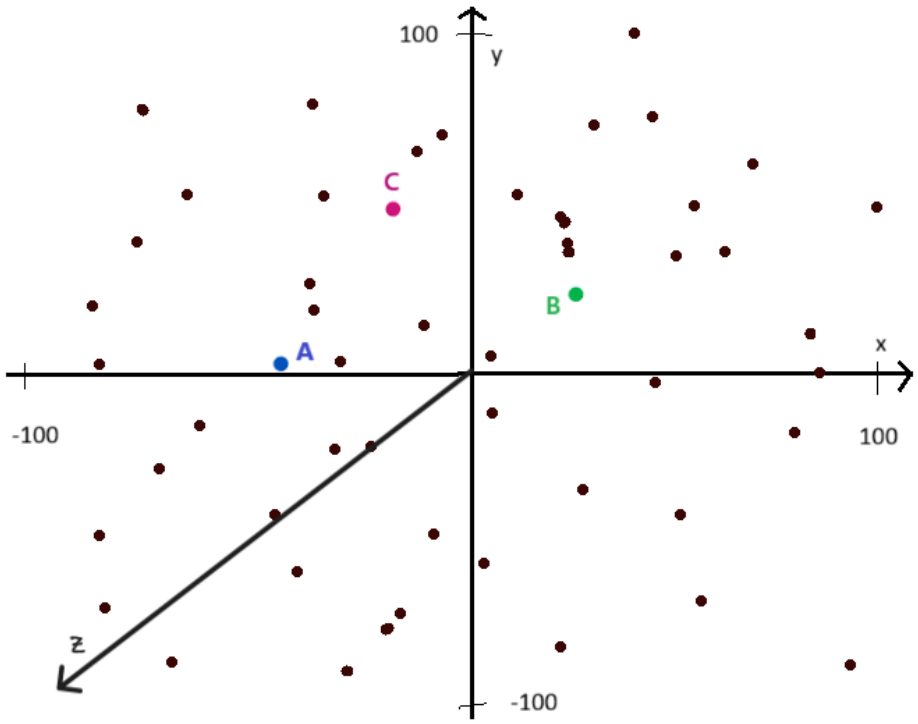


Kuva 6. Populaatio XYZ-koordinaatistossa.

Kuvassa 6 on alkutilanne, kun populaatio on generoitu ja pisteet lisätty XYZ-koordinaatistoon ellipsoidille. Ellipsin populaatio generoidaan XY-koordinaatistoon.

4.3 Valitaan satunnaisesti pisteet

Ellipsille tai pyörähdysellipsoidille valitaan kolme pistettä A, B ja C. Näitä parametreja käytetään uudelleen valituille pisteille, jolloin edelliset pisteet kirjoitetaan yli uudelleenvalituilla. Ensimmäisenä satunnaisesti valittu on A, toisena B ja kolmantena C.



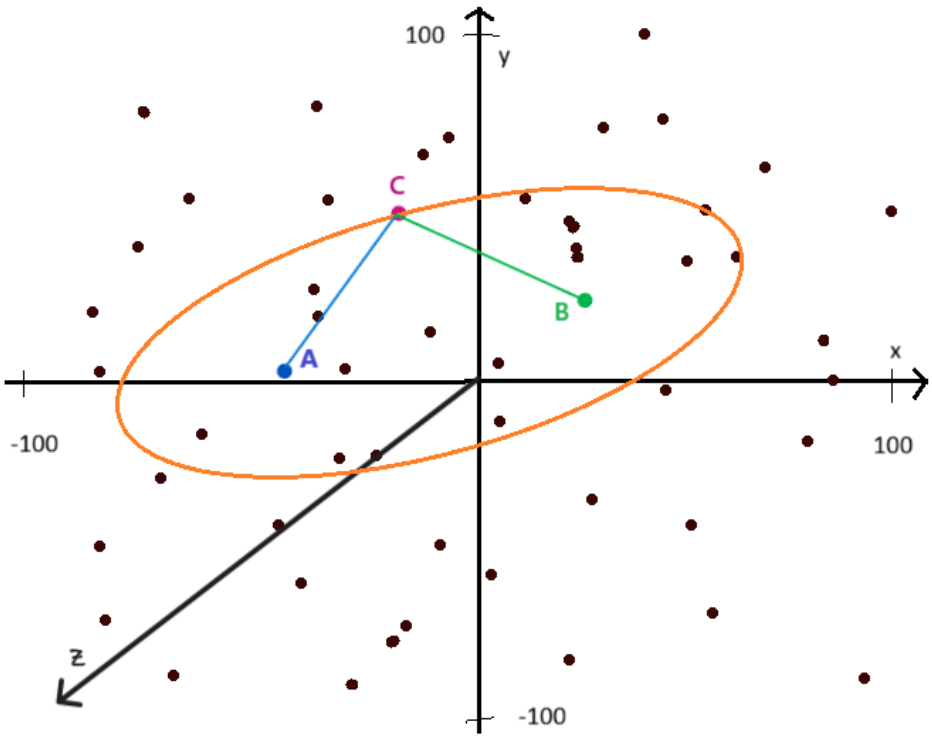
Kuva 7. A, B ja C pisteet valittu.

Kuvassa 7 on XYZ-koordinaatisto ja valitut pisteet. Valittuja pisteitä käytetään ellipsin tai pyörähdysellipsoidin määrittelyyn. Piste A on ensimmäinen polttopiste, piste B on toinen polttopiste, ja C on kehäpiste.

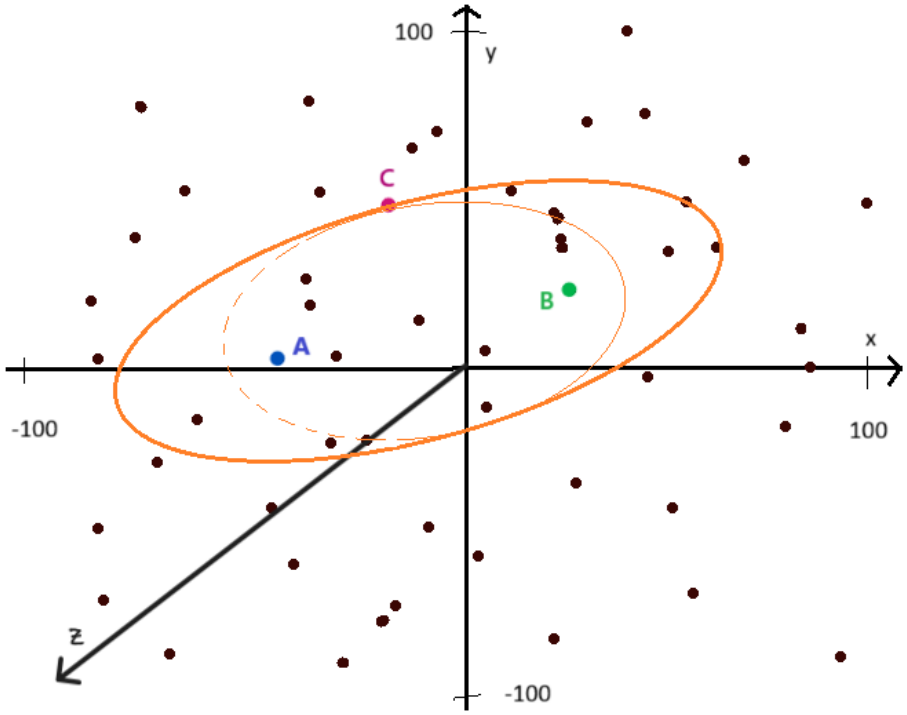
4.4 Määritellään ellipsi tai pyörähdysellipsoidi

Pisteiden A, B ja C perusteella määritellään pyörähdysellipsoidi, ja piirretään se. Pisteitä A ja B käytetään määrittelemään muodon polttopisteet, ja piste C on kehältä.

Määrittelyssä hyödynnetään ellipsin ja ellipsoidin ominaisuuksia, pyörähdyskappaleen ominaisuuksia, euklidista etäisyyttä, polttopisteitä sekä puolipituusakseleita. Näitä yhdistelemällä saadaan muodostettua koordinaatistoon haluttu muoto. Satunnaisesti valittujen pisteiden sijaintien perusteella saadaan arvot puolipituusakseleille ja polttopisteille, ja näillä piirrettyä kappaleen kehä.



Kuva 8. Muodon piirtäminen pisteiden perusteella.



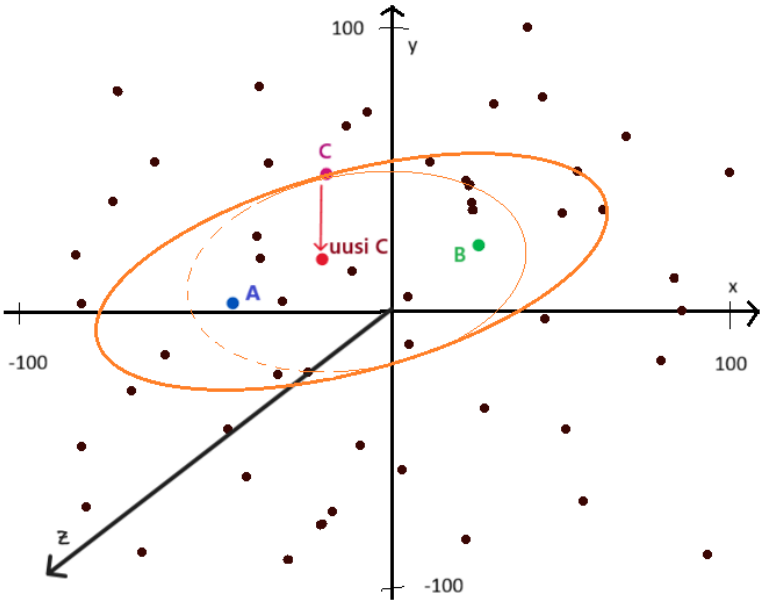
Kuva 9. Pyörähdysellipsoidin piirtäminen pisteiden A, B ja C perusteella.

Kuvissa 8 ja 9 on mallin piirtäminen. Kuvaan 8 on lisätty apuviivat euklidisesta etäisyydestä määritelmien mukaisesti. Kuvassa 8 on hahmoteltu ellipsi, ja kuvaan 9 jatkettu ellipsistä pyörähdysellipsoidi.

4.5 Korvataan piste

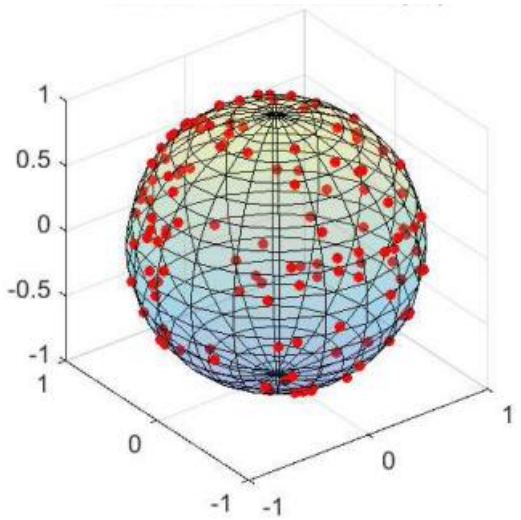
Viimeisimpänä valittu piste korvataan uudella pisteellä, ja siihen käytetään kehäpistettä C. Vanha piste poistuu, eli populaatiopisteiden kokonaismäärä pysyy vakiona. Uusi piste muodostetaan satunnaisesti, mutta sen sijaintiin vaikuttaa todennäköisyys T.

Todennäköisyys T määritetään alussa, ja sen suuruus on 0 ja 1 välillä. Todennäköisyyden suuruus kuvaa, kuinka todennäköisesti uusi piste tulee sijaitsemaan ellipsin tai pyörähdysellipsoidin sisäpuolella. T arvolla 0 uusi piste tulee todennäköisemmin sijaitsemaan pyörähdysellipsoidin ulkopuolella, ja T arvolla 1 pyörähdysellipsoidin sisällä.



Kuva 10. Pisteen C korvaaminen uudella pisteellä.

Kuvassa 10 on pisteen korvaaminen uudella pisteellä. Uusi piste saadaan niin, että generoidaan tasaisella jakaumalla satunnaispisteitä n-dimensioisen yksikköhyperpallon pinnalle, tästä kuva 11. Ellipsille tämä vaihe tehdään yksikköympyrälle kaksiulotteisuuden takia. Pisteistä seulotaan halutut pisteet.

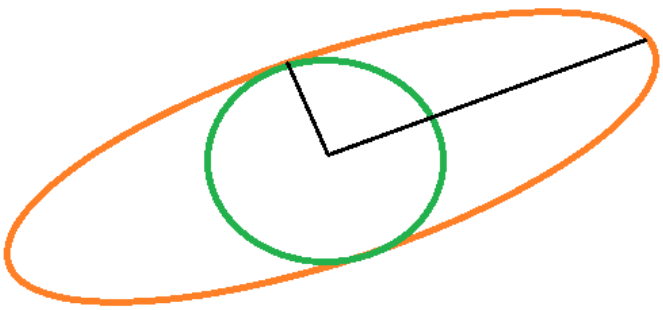


Kuva 11. Yksikköpallo ja pisteet sen pinnalla. (Li ja muut, 2019.)

Yksikköhyperpallo skaalataan ja siirretään koordinaatistoon pisteiden A, B ja C perusteella muodostetun muodon mukaisesti. Näin nämä kaksi muotoa ovat koordinaatistossa kohdikkain, ja muotojen kesken voidaan suorittaa vertailuja ja valita uusi piste. Uutta satunnaispistettä luodessa muodostetaan tavoiteltu satunnaisvektori, jolloin koordinaatiston kiertoa ei tarvita.

Todennäköisyyden T arvo määrittää todennäköisyyden, jolla uusi piste on pyörähdysellipsoidin sisällä. T:n ollessa lähellä yhtä, valittu uusi piste on todennäköisemmin pyörähdysellipsoidin sisällä, ja lähellä nollaa uusi piste on todennäköisemmin pyörähdysellipsoidin ulkopuolella. Ohjelmassa arvotaan tasajakaumalta satunnainen luku ja verrataan sitä arvoon T. Jos arvo on suurempi kuin T, valitaan piste pyörähdysellipsoidin sisältä, ja jos pienempi, valitaan pyörähdysellipsoidin ulkopuolelta. Satunnaisluvun ollessa tasan 0,5, arvonta suoritetaan uudelleen.

Ellipsin kehänsuuntaisen jakauman korjaamiseksi tasaista jakaumaa vastaavaksi lisätään vielä yksi vaihe algoritmiin. Alkuperäinen jakauma yksikköympyrällä on tasainen, sillä esimerkiksi yhtä kulma-astetta vastaavat ympyrän kaaret ovat kaikki yhtä pitkiä. Sen sijaan ellipsillä säde vaihtelee suunnan mukaan, minkä seurauksena lyhyen puoliakselin suuntaan muodostuu tiheämmin satunnaispisteitä kuin pitkän puoliakselin suuntaan. Tämä ilmiö korostuu erityisesti tapauksessa, jossa ellipsi tai pyörähdysellipsoidi on voimakkaasti litistynyt. Kuvassa 12 on esimerkki litteän ellipsin ja yksikköympyrän puoliakselien erosta.



Kuva 12. Litteä ellipsi ja yksikköympyrä.

Tämän epätasaisuuden korjaamiseksi käytetään hylkäämismenetelmää. Jokainen satunnaisesti muodostettu piste hyväksytään tai hylätään erikseen määritellyn hylkäämistodennäköisyyden perusteella. Jos piste hylätään, palataan algoritmissa takaisin ja muodostetaan uusi piste.

$$P(hylkäys) = \frac{|R_{ellipsoidi} - O_{ellipsoidi}|}{a} \quad (14)$$

Kaavassa 14 P(hylkäys) lasketaan $R_{ellipsoidi}$, $O_{ellipsoidi}$ ja a perusteella, eli hylkäämistodennäköisyys perustuu pisteen etäisyyteen pyörähdysellipsoidin kehältä, pyörähdysellipsoidin keskipisteeseen ja pyörähdysellipsoidin suureen puolipituusakseliin. P(hylkäys) arvoa verrataan vielä satunnaislukuun tasajakaumalta, ja jos P(hylkäys) on suurempi kuin kyseinen satunnaisluku, päätetään tarkastella seuraavaa pistevaihtoehtoa.

Kappaleessa 5.2.3 on pisteen C korvaamisesta tarkentavaa tietoa ohjelman kannalta ja siihen tarvittavia funktioita. Kappaleessa 5.2.4 selvennetään hylkäämismenetelmää ohjelmassa.

4.6 Toistorakenne, lopetusehto

Kun piste on korvattu, algoritmissa mennään toistorakenteen alkuun. Populaation generointi tehdään vain kerran, eli sitä ei tarvitse toistaa, käytetään samaa populaatiota algoritmin suorituksen loppuun.

Toistorakenne alkaa pisteiden valinnasta satunnaisesti. Pisteet A, B ja C valitaan joukosta uudelleen. Pisteistä käytetään samoja nimityksiä, joten ne kirjoitetaan yli ohjelmassa uusilla arvoilla.

Seuraavaksi jälleen määritellään ellipsi tai pyörähdysellipsoidi pisteiden perusteella. Jälkimmäisin piste korvataan uudella. Toistetaan haluttu määrä. Kun toistokerrat täyttyvät, ohjelma lopetetaan ja voidaan tarkastella tuloksia. Ohjelma tulostaa visualisoinnin kuvaajasta tietyn välein, sekä jokaisen iteraation arvot populaation keskimääräisestä etäisyydestä muodon keskipisteestä ja varianssin.

Toistokertojen määrä asetetaan alussa, ja alkuun se on 1. Toistokertoja saa kasvatettua kymmeniin, satoihin ja tuhansiin, kun Matlab-ohjelmaa suorittaessa on todettu ohjelman toimivuus ja ajankesto. Alussa on määritelty myös todennäköisyys T, joka vaikuttaa toistokertoihin, ja sitä voidaan vaihdella testauksen aikana. Tuloksia ei oteta talteen jokaisesta iteraatiokerrasta toistokertojen ollessa satoja tai tuhansia, vaan sopivin välein.

Jo testauksessa arvioidaan silmämääräisesti tulosten toteutumista, ja tarvittaessa ohjelmaan tehdään korjauksia. Testaukset päätteeksi lopulliset tulokset otetaan ylös ja niihin tehdään jälkeenpäin vertailuja ja arviointeja. Molemmille ellipsin ja pyörähdysellipsoidin algoritmille tehdään omat arviot. Tuloksissa halutaan katsoa kokonaisuutta ja joukon suppenemista tai hajaantumista. Myös ohjelmiston suoriutumista arvioidaan.

5 Koeasetelma

Ohjelmistona käytetään MATLAB-työpöytäohjelmistoa (myöhemmin Matlab), koetta aloittaessa uusinta julkistettua versiota R2023b. Matlabista on tullut työn aloituksen jälkeen uudempi versio R2024a. Ohjelmisto suoritetaan HP:n valmistamalla tietokoneella, jossa on Windowsin versio Windows 11 Home, 64-bittinen käyttöjärjestelmä ja 8,00 Gt sisäinen muisti. Tietokone on tavanomainen kotitietokone.

Jos suorituksessa kohdataan ongelmia käytettävässä ohjelmistossa, voidaan harkita toiseen vastaavaan ohjelmistoon vaihtamista. Tätä voi myös harkita myöhemmin koetta toistaessa tulevaisuudessa. Myös Matlabin uudistetut versiot voivat tarjota ratkaisuja ilmenneihin ongelmiin.

5.1 Matlab esittely

Tyypillisesti Matlabia käytetään matemaattiseen suunnitteluun ja ohjelmointiin. Matlabia käytetään matemaattisten mallien luomiseen, simuloimiseen ja analysointiin eri tieteenaloilla. Ohjelmistoa hyödynnetään esimerkiksi signaalianalyysissä, kuvankäsittelyssä, ohjausjärjestelmissä sekä tilastollisessa datassa. Sen avulla voidaan toteuttaa monimutkaisia laskutoimituksia ja visualisointeja. Matlab tarjoaa myös laajan valikoiman lisätyökaluja ja kirjastoja, jotka mahdollistavat ratkaisujen löytämisen yksilöityihin ongelmiin ja ratkaisujen tulosten esittämisen.

Matlab käyttää ohjelmointikielenä omaa syntaksiaan, tosin muiden ohjelmointikielillä toteutettujen tiedostojen tuominen ohjelmointiympäristöön on mahdollista. Ohjelma voidaan suorittaa komentona komentoriville tai .m-tiedostona, eli scriptinä. Scriptit mahdollistavat ohjelmien muokkaamisen ja suorittamisen ohjelmistossa, niiden mutkattoman tallentamisen tai jakamisen itselle ja muille.

Tässä työssä käytetään Matlabin tämänhetkistä uusinta julkistettua versiota, R2023b. Kuitenkin, Matlab-versioita päivitetään suunnilleen puolivuosittain yleiseen käyttöön, ja vielä useammin beta-versioita, joten uusin versio tulee tarkistaa Matlabin verkkosivuilta. Osa versioista on tarkoitettu yksityiseen käyttöön ja osa julkiseen.

5.2 Valmistelu

Ellipsille ja pyörähdysellipsoidille tehdään omat ohjelmatiedostot. Ensin tehdään ellipsin ohjelma toimivaksi kokonaisuudeksi, ja sen pohjalta laajennetaan pyörähdysellipsoidin ohjelma. Ohjelmaan sisällytetään visualisointiominaisuuksia tulosten tallentamiseksi.

Algoritmi noudattelee pseudokielisen algoritmin vaiheita Matlab-ohjelmassa. Matlab-ohjelmat sisältävät kommentit jokaisesta vaiheesta, mutta yleisiä huomiota ja valmiita funktioita on hyvä selventää erikseen. Niistä lisätietoa seuraavaksi.

5.2.1 Parametrit käyttäjältä

Käyttäjältä tarvittavat alkuparametrit ovat todennäköisyys T ja toistokerrat n iteraatiolle. Alussa todennäköisyys on vaikkapa 1,00. Tämä tarkoittaa, että iteraatiokertoja lisätessä, pistejoukon odotetaan suppenevan. Toistokerta n on ensin 1. Kun ohjelma on toimiva, sitä kasvatetaan ensin 10, ja sitten satoihin ja siitä eteenpäin. Toistokertoja kasvatetaan, kunnes pistejoukossa havaitaan odotettuja tuloksia. Näitä alkuparametreja voidaan testatessa muutella, ja tarkastella niiden muutosten vaikutusta pistejoukkoon.

Vaihdeltavia parametreja ovat myös populaation parametrit, esimerkiksi pisteiden määrä ja niiden saamat arvot. Tässä algoritmissa pisteiden määräksi on asetettu sata pistettä, ja ne voivat saada alussa arvoja [-100, 100] välillä. Nämä pysyvät samoina kokeen ajan muutoin, mutta pisteiden saamat arvot voivat vaihdella ohjelman edetessä. Parametrit ovat muokattavissa yksilöllisiin tarpeisiin.

5.2.2 Funktiot

Matlabin valmiit funktiot rand() ja randi() ovat satunnaislukugeneraattoreita, jotka tuottavat lukuja annetulta väliltä. Tuotettu satunnaisluku on tasajakaumalta. Rand() palauttaa desimaaliluvun väliltä 0 ja 1, ja randi() palauttaa kokonaisluvun määritetyltä väliltä. Näitä funkioita käytetään ohjelmassa esimerkiksi populaation satunnaispisteiden luomisessa, valitsemisessa ja luomisessa. Rand() ja randi() satunnaislukugeneraattorit ovat näennäissatunnaislukugeneraattoreita, joten ne eivät oikeasti tuota satunnaisia lukuja, mutta ovat näennäisesti riittävän satunnaisia ja läpäisevät satunnaisuuden vaatimukset ja testit (MathWorks, n.d.).

Matlab käyttää satunnaislukugeneroinnissa oletuksena järjestelmän kellonaikaa, mikä tuottaa eri tulokset jokaisella suorituksella. Kuitenkin, jos halutaan määritellä mahdollisimman satunnaiset ja ennakoimattomat satunnaisluvut, satunnaislukugeneraattorin hyödyntämä siemenluku voidaan alustaa satunnaiseksi komennolla rng('shuffle'). Funktio varmistaa satunnaisuuden hyödyntämällä nykyistä aikaa.

Norm() on Matlabin valmis funktio vektorin euklidisen etäisyyden eli pituuden palauttamiseksi. Tässä työssä norm()-funktiota käytetään pisteiden etäisyyden laskemiseen ellipsin tai pyörähdysellipsoidin origosta, eli pituusakselien suuruuden laskemiseen.

DrawEllipse() ja drawEllipsoid() ovat itsetehtyjä funktioita ellipsin ja pyörähdysellipsoidin piirtämiseksi, ja näitä voi muokata ohjelman sisällä. DrawEllipse() piirtää kaksiulotteisen ellipsin määriteltyjen pääakselien ja keskipisteen perusteella. DrawEllipsoid() puolestaan käyttää yksikköpallon skaalattua muunnosta luodakseen kolmeulotteisen pyörähdysellipsoidipinnan annetuille akselipituuksille ja suuntauksille. Näitä funktioita käytetään algoritmien toiminnan havainnollistamiseen ja tulosten analysointiin graafisesti.

5.2.3 Pisteen generoiminen

Ellipsi- ja ellipsoidi-algoritmeissa uusi piste generoidaan satunnaisesti määritellystä jakaumasta ja tarkistetaan sen kelpoisuus suhteessa muodostettuun ellipsiin tai pyörähdysellipsoidiin. Tasajakaumasta valitaan satunnainen luku 0 ja 1 välillä, ja verrataan lukuun T. Jos T ja satunnainen luku ovat molemmat tasan 0,5, valitaan uusi satunnaisluku. Jos satunnaisluku on pienempi kuin T, muodostetaan piste ellipsin tai pyörähdysellipsoidin sisäpuolelle. Jos suurempi kuin T, piste tulee muodon ulkopuolelle.

Jos piste luodaan muodon sisäpuolelle, niin ensin luodaan satunnaisia pisteitä yksikköympyrän tai yksikköpallon muodostamiseksi. Ellipsi on kaksiulotteinen kappale, niin sille muodostetaan yksikköympyrä ja kolmeulotteiselle pyörähdysellipsoidille yksikköpallo. Yksikköympyrä tai yksikköpallo ja sen pisteet siirretään ellipsin tai pyörähdysellipsoidin kohdalle ja skaalataan sopimaan verrattavaan muotoon. Vain muodostetulla alueella sijaitsevat pisteet kelpuutetaan. Sitten valitaan yksikköympyrän tai yksikköpallon pisteistä satunnainen piste käyttämällä Matlabin randi()-funktiota. Varmistetaan, että piste täyttää matemaattiset ehdot pysyäkseen halutussa joukossa. Jos piste hylätään, prosessi toistetaan ennalta määritellyllä maksimimäärällä yrityksiä.

Jos piste luodaan muodon ulkopuolelle, luodaan satunnainen piste [100, 100] alueelle. Pistettä ei kuitenkaan valita, jos se on ellipsin tai pyörähdysellipsoidin sisällä tai kehällä, vaan pisteen luominen toistetaan, kunnes sopiva piste löytyy. Jos 20 yrityskerran jälkeen sopivaa pistettä ei ole saatu muodostettua, määrittelyaluetta kasvatetaan 1,2-kertaiseksi.

Lopulta hyväksytty piste otetaan käyttöön ja korvaa aiemmin valitun pisteen iteratiivisessa prosessissa. Generoitu ja korvaava piste on C, muodon kehällä sijaitseva piste. Piste korvataan, ja seuraavat pisteet valitaan uudelleen uutta muotoa varten.

5.2.4 Hylkäämismenetelmä

Algoritmiin on sisällytetty menetelmä hylkäämään osan muodostetun alueen pisteistä. Tämä vaihe on korjaamaan jakauma ellipsin tai pyörähdysellipsoidin kehänsuunnassa lähemmäs tasaista jakaumaa. Yksikköympyrän tai yksikköpallon jakauma on tasainen, kun taas ellipsin tai pyörähdysellipsoidin lyhyimmällä puolipituusakselille luodaan suhteessa useammin satunnaispisteitä. Ilmiötä esiintyy erityisesti litteillä muodoilla.

Hylkäysmenetelmän laskukaava perustuu hylkäämisen todennäköisyyteen ja pituusakselien suhteeseen. Menetelmä sisällytetään ohjelman osaan, jossa uusi piste generoidaan ellipsin tai pyörähdysellipsoidin sisälle. Jos tarkasteltava piste hylätään, valitaan uusi satunnainen piste tarkasteltavaksi muodon sisältä tai pinnalta. Jos hylkäyksiä tulee 20 peräkkäin, viimeisin tarkasteltava piste valitaan joka tapauksessa, jotta ohjelma ei ajaudu loputtomaan silmukkaan. Lopuksi tuloksissa tarkastellaan vielä, tapahtuuko paljon maksimimäärän hylkäyksiä.

5.3 Ellipsi-ohjelma

Ellipsin ohjelma tallennetaan ellipsi.m-tiedostoon. Ohjelman kommenteissa on osioita välitestailua varten, mutta ne otetaan lopullisesta ohjelmasta pois. Ne eivät siis ole oleellisia algoritmin toiminnan kannalta. Ohjelmassa on siirretty kommentteihin ja jätetty lopullisesta versiosta pois esimerkiksi testauksen osa, jossa varmistetaan pisteen generoinnin oikea sijainti.

Algoritmissa 1 on algoritmin Matlab-ohjelma ellipsille, ja se löytyy kokonaisuudessaan liitteestä 1. Ohjelma sisältää myös tulosten graafisen tulkinnan, ja tulosten lukujen tulostamisen.

Algoritmia on testailtu eri arvoilla T ja n, ja testauksessa on tarkasteltu kuvaajassa uuden pisteen C toiminta. Ohjelma on todettu näiden pohjalta toimivaksi niin, että voidaan edetä ohjelman muokkaukseen ellipsoidille.

5.4 Ellipsoidi-ohjelma

Pyörähdysellipsoidin ohjelma tallennetaan ellipsoidi.m-tiedostoon. Ohjelman kommenteissa on osioita välitestailua varten, mutta ne otetaan lopullisesta ohjelmasta pois. Ellipsoidin ohjelma on muokattu ellipsi-ohjelman pohjalta, ja siihen on lisätty kolmannen ulottuvuuden tuomat ominaisuudet. Tämä on pyritty huomioimaan kokonaisuudessaan pyörähdysellipsoidin algoritmissa. Ellipsoidi-ohjelmalle on tehty samat välitestaukset kuin ellipsi-ohjelmalle, jotta mahdolliset virheet huomataan ja korjataan aikaisessa vaiheessa.

Algoritmissa 2 on algoritmin Matlab-ohjelma pyörähdysellipsoidille, ja se löytyy kokonaisuudessaan liitteestä 2. Ohjelma sisältää myös tulosten graafisen tulkinnan, ja tulosten lukujen tulostamisen. Tuloksiin talletetaan osa kuvaajista ja results-vektorin tulokset. Results-vektoriin on tallennettu iteraatiokerta, populaatiopisteiden keskietäisyys origosta sekä varianssiarvo.

6 Tulokset

Tuloksista odotetaan, että tarpeeksi monella iteraatiokerralla populaatio alkaa hajaantua, supistua tai pysyy samana, ja tähän vaikuttaa todennäköisyys T:n arvo. Raja-arvoja ovat 0, 0,5 ja 1. T:n arvolla T<0,5 populaatio hajaantuu, T>0,5 populaatio supistuu ja T=0,5 populaatio pysyy suurin piirtein samana. Tätä voidaan tarkastella silmämääräisesti kuvaajista sekä laskemalla iteraatiokertojen jälkeen populaatiopisteiden keskimääräinen etäisyys origosta ja variaatio. Olennaisinta on kokonaisuuden muutos.

Tulokset halutaan molemmista ellipsin ja pyörähdysellipsoidin algoritmista. Ohjelmalla voidaan tuottaa kuvaajat määritellyn välein, ja tulostaa arvoja. Populaatiopisteiden keskimääräinen etäisyys populaation origosta kertoo, millä etäisyydellä uudet pisteet alkavat sijaita populaation origosta nähden. Koska luku on keskiarvo, arvo voi aluksi vaihdella molempiin suuntiin, mutta lopullinen kokonaisuuden keskiarvo merkitsee. Alun lukuja tulee verrata viimeisiin iteraatiokertoihin, ja kuvaaja näistä luvuista on havainnollistava. Varianssi puolestaan kertoo, kuinka hajaantuneita populaation pisteet ovat keskipisteeseen nähden. Korkea varianssi tarkoittaa suurta hajaantumista, pieni varianssi että pisteet ovat tiheämmin keskipisteen ympärillä.

Tuloksissa halutaan myös tarkastella algoritmin käytännöllisyyttä, ja mitata eri iteraatiokertojen kestoa. Ohjelman suoritusten kestoja mitataan eri T arvoilla, koska sillä on eniten merkitystä ohjelman suorituksen kestoon. Mittaukset tehdään ellipseille ja pyörähdysellipsoideille.

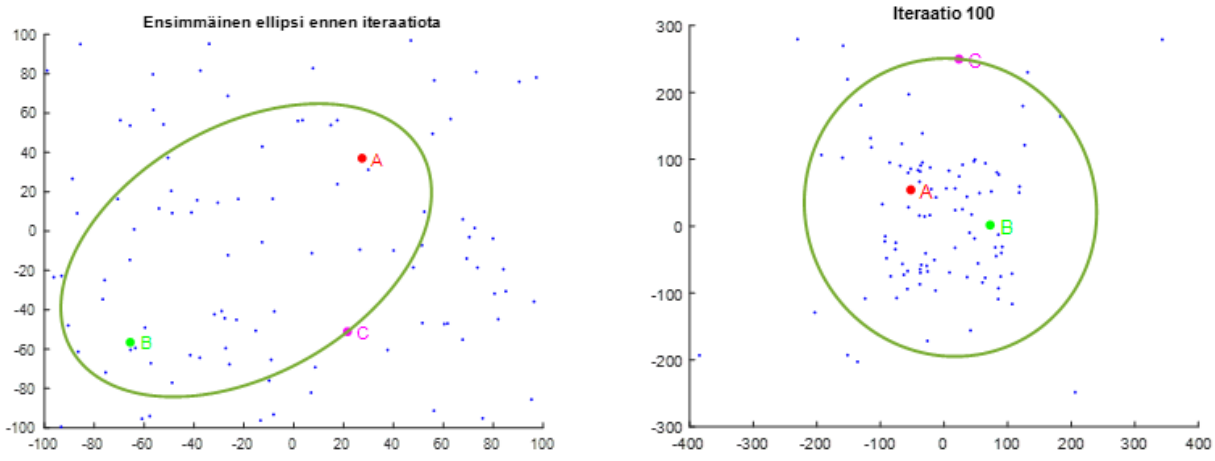
Ellipsi- tai ellipsoidi-ohjelma suoritetaan ja tulokset kirjataan ylös. Vaihdeltava parametri on T, ja iteraatiokerroiksi n määritellään 300. Tarvittaessa n kasvatetaan, mutta tuloksia saattaa näkyä jo varhaisemmassa vaiheessa. Jokaisesta iteraatiosta on tulostettu arvo, joten ei ole merkitystä, jos tuloksia saadaan jo pienemmällä määrällä iteraatiokertoja. 300 iteraatiokertaa ei ole havaittu testauksen aikana vievän liikaa aikaa ollakseen epäkäytännöllinen, mutta näyttävän jo alustavasti tuloksia.

Ilmiöiden tehostamiseksi ohjelmat suoritetaan uudelleen pienemmällä populaatiolla ja kasvattamalla iteraatiokertoja. Samalla tarkkaillaan algoritmin antamien tulosten ja populaation stabiiliutta.

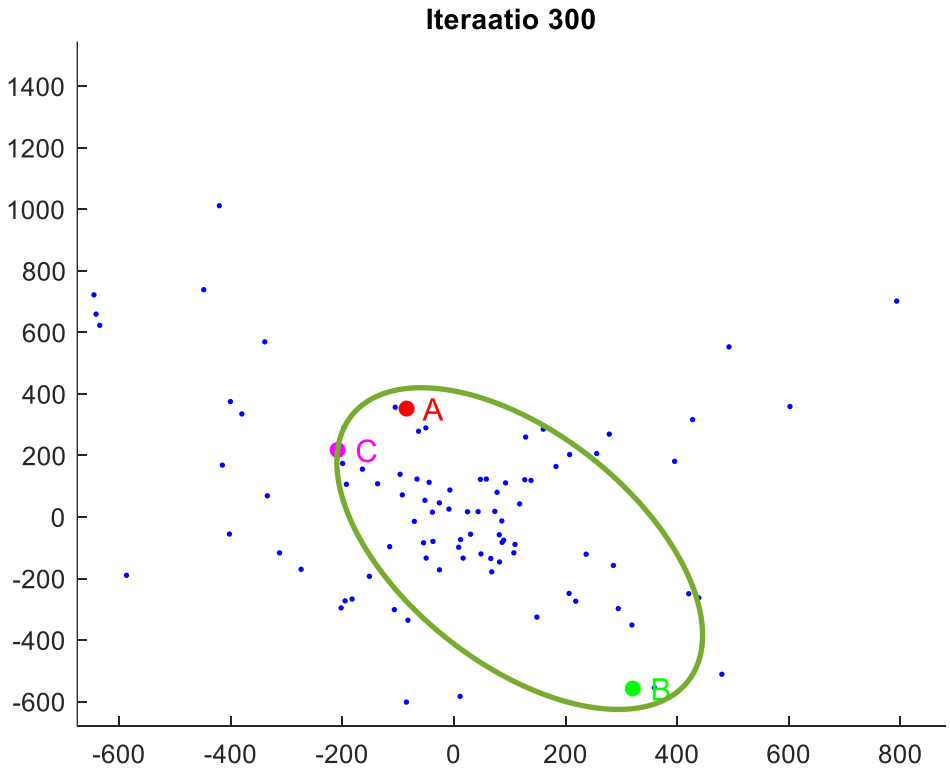
6.1 Ellipsi, T=0,00

Ellipsi-ohjelma suoritetaan, määritetään todennäköisyydeksi T=0,00. Kuvaajista poimitaan kolme kuvaa, ja tarkastellaan joukkoa silmämääräisesti. Hypoteesi T:n arvoon perustuen on, että joukko hajaantuu. Lopuksi vahvistetaan asia tulostettujen arvojen ja niiden muutosten perusteella.

Kuvassa 13 on ellipsi-ohjelman alkuvaiheita, ja kuvassa 14 iteraatio 300. Kuvassa 14 näkyy paremmin, kuinka alkuperäinen pistejoukko on alkanut hajaantumaan määritellyn alueen reunoille, ja pisteitä lähtee keskiöstä. Tämä on hypoteesin mukaisesti. Populaation hajaantumisen myötä koordinaatiston asteikko muuttuu.



Kuva 13. Ellipsi, T=0,00, alkuiteraatiot.



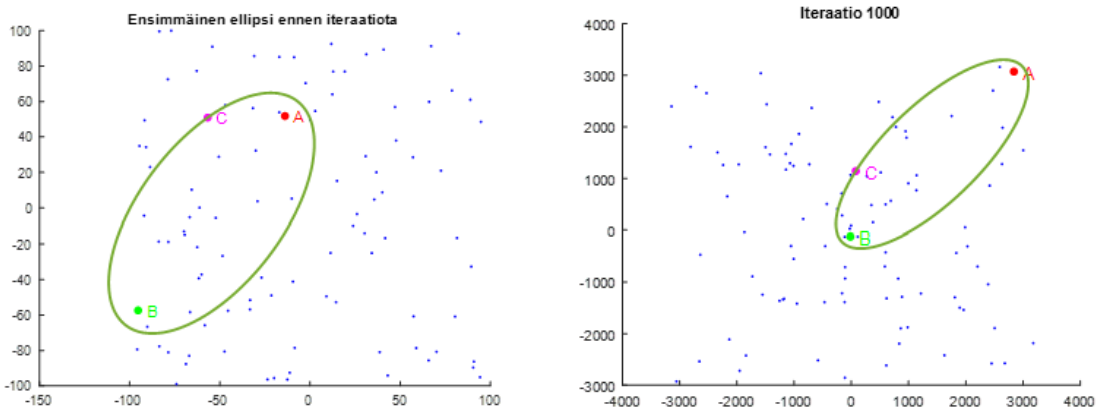
Kuva 14. Ellipsi, T=0,00, iteraatio 300.

Taulukossa 3 on ellipsi-ohjelman pistejoukolle keskimääräinen etäisyys populaation origosta ja pistejoukon varianssin arvoja joka 50. iteraatio. Yksittäisten arvojen sijaan muutoksissa tulee tarkastella kokonaisuutta ja alku- ja lopputilannetta. Keskietäisyyden suureneminen alkutilanteesta tarkoittaa, että keskimäärin pisteet hajaantuvat populaation origosta. Varianssin tasainen kasvu samoin.

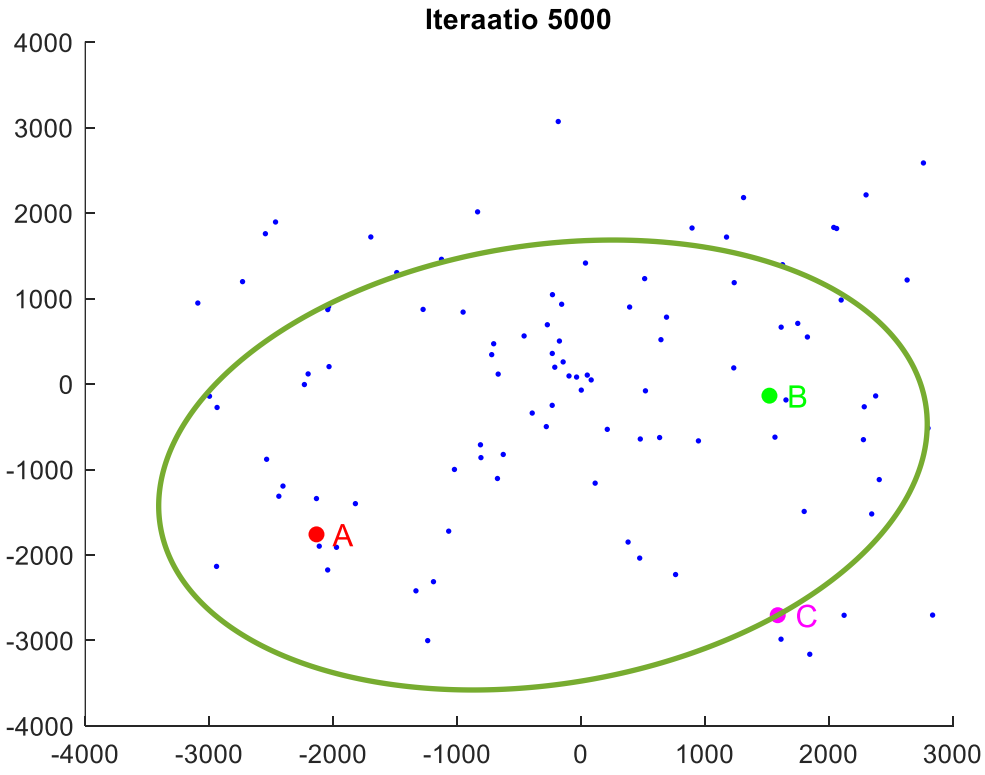
Taulukko 3. Ellipsi, T=0,00, keskietäisyys ja varianssi.

Iteraatio	Keski-etäisyys	Varianssi 1.0e+04
0	71	6,29
50	134	13,52
100	126	21,95
150	190	41,12
200	256	79,41
250	538	173,75
300	441	380,61

Kun iteraatiokertojen määrää lisätään, voidaan tarkastella populaation hajaantumiskäyttäytymistä. Samalla voidaan tutkia populaation stabiiliutta sekä hajaantumisen nopeutta suhteessa odotettuihin tuloksiin.



Kuva 15. Ellipsi, T=0,00, populaation kehittyminen.



Kuva 16. Ellipsi, T=0,00, populaation kehittyminen, iteraatio 5000.

Kuvassa 15 ja 16 populaation hajautuminen. Populaatiopisteet kerääntyvät maltillisesti määrittelyalueen reunoille ja nurkkiin, ja keskiosa harvenee. Iteraatiokertojen nostaminen 5000 saakka ei enää merkittävästi korosta hajaantumisilmiötä.

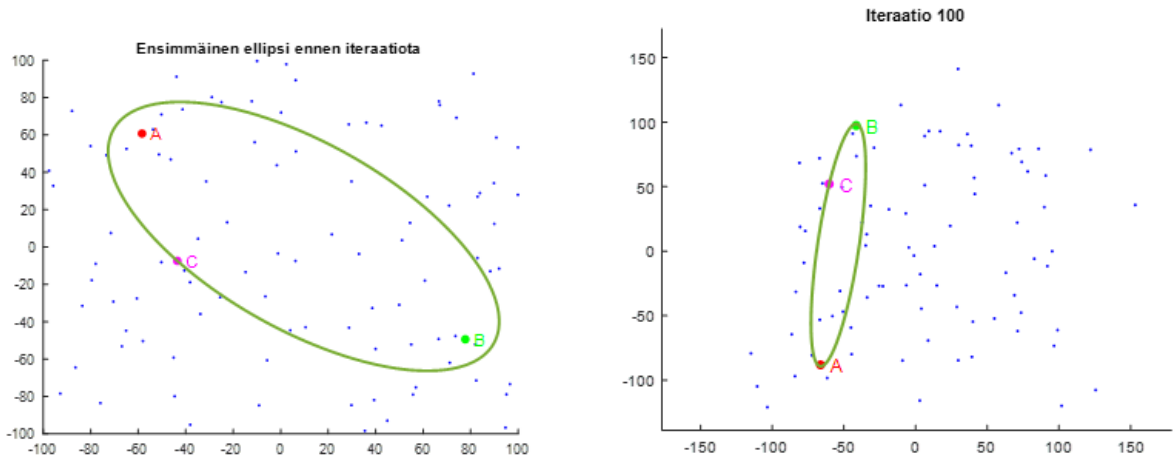
Hajaantumista ja pisteiden kertymistä reunoille voitaisiin vielä korostaa erityisillä ehdoilla, kuten painotetulla satunnaisuudella ja pisteiden generointi suhteessa ellipsiin. Tässä tutkimuksessa siihen ei ole tarvetta, sillä tarkoitus on vain ymmärtää ja havainnollistaa hajaantumista.

6.2 Ellipsi, T=0,50

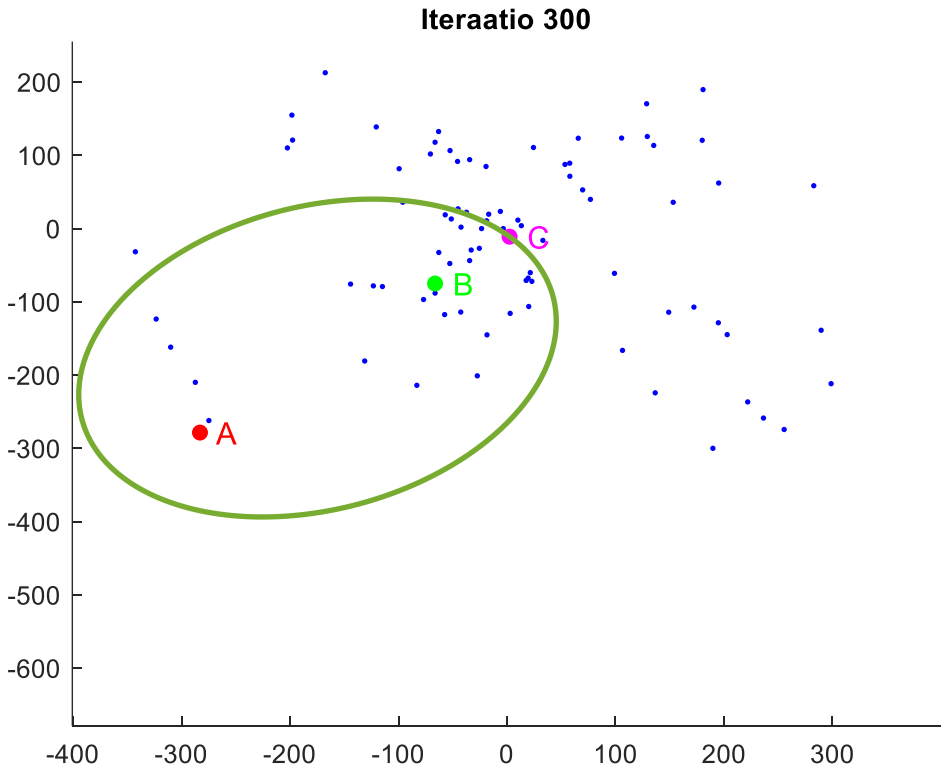
Ellipsi-ohjelma suoritetaan, määritetään todennäköisyydeksi T=0,50. Kuvaajista poimitaan kolme kuvaa, ja tarkastellaan joukkoa silmämääräisesti. Hypoteesi T:n arvoon

perustuen on, että joukko pysyy suurin piirtein samalla alueella. Lopuksi vahvistetaan asia tulostettujen arvojen ja niiden muutosten perusteella.

Kuvassa 17 on ellipsi-ohjelman alkuvaiheita, ja kuvassa 18 iteraatio 300. Kuvissa ei ole huomattavaa muutosta pistejoukossa, millekään alueelle ei kasaannu pisteitä ja missään ei ole erityisen harvaa aluetta, kun verrataan alkuperäiseen pistejoukkoon. Tämä on hypoteesin mukaisesti. Yksittäisiä pisteitä erkaantuu pääpopulaatiosta, mutta kokonaisuus pysyy tasaisena.



Kuva 17. Ellipsi, T=0,50, alkuiteraatiot.



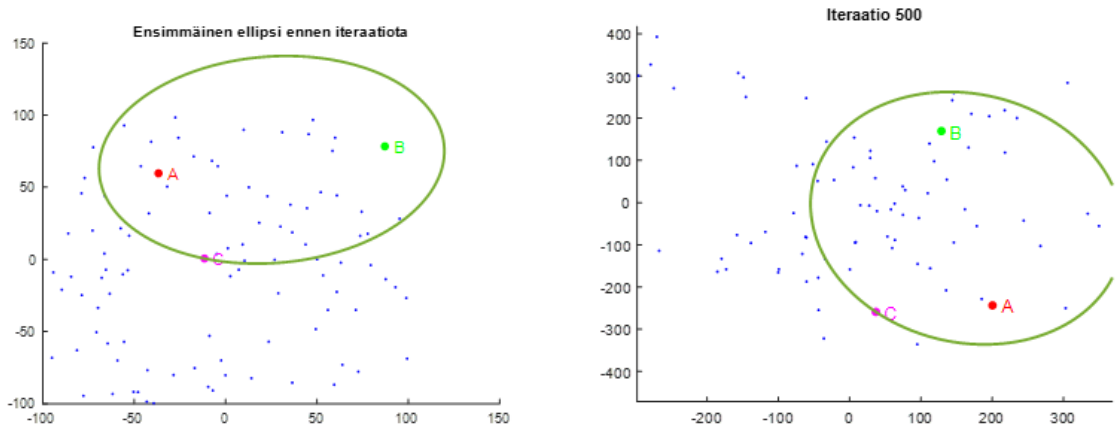
Kuva 18. Ellipsi, T=0,50, iteraatio 300.

Taulukossa 4 on ellipsi-ohjelman pistejoukolle keskimääräinen etäisyys populaation origosta ja pistejoukon varianssin arvoja joka 50. iteraatio. Yksittäisten arvojen sijaan muutoksissa tulee tarkastella kokonaisuutta ja alku- ja lopputilannetta. Keskietäisyyden ja varianssin pieni kasvu selittyy yksittäisten pisteiden satunnaisuudella ja sillä, että ellipsin kehä yltää hieman populaatiojoukon yli. Iteraatiokertoja lisätessä ilmiö tasaantuu.

Taulukko 4. Ellipsi, T=0,50, keskietäisyys ja varianssi.

Iteraatio	Keski-etäisyys	Varianssi 1.0e+04
0	**78**	**6,76**
50	**119**	**13,10**
100	**121**	**18,79**
150	**184**	**26,18**
200	**189**	**33,17**
250	**195**	**44,48**
300	**315**	**63,65**

Populaation käyttäytymisen tehostamiseksi, iteraatiokerroiksi asetetaan 500. Populaation alkutilannetta verrataan lopputilanteeseen, ja tuloksista tarkastellaan populaation tilan kehittymistä.



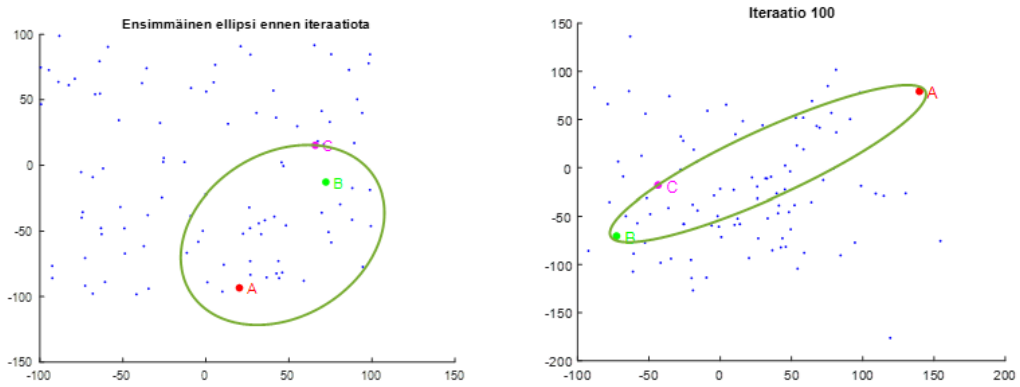
Kuva 19. Ellipsi, T=0,50, populaation kehittyminen.

Kuvassa 19 ei ole huomattavaa populaatiopisteiden kerääntymistä eikä voimakasta hajaantumista. Iteraatiokertojen nostamisella ei ole kokonaisuuden kannalta merkitystä. Tämä on odotettujen tulosten mukaisesti.

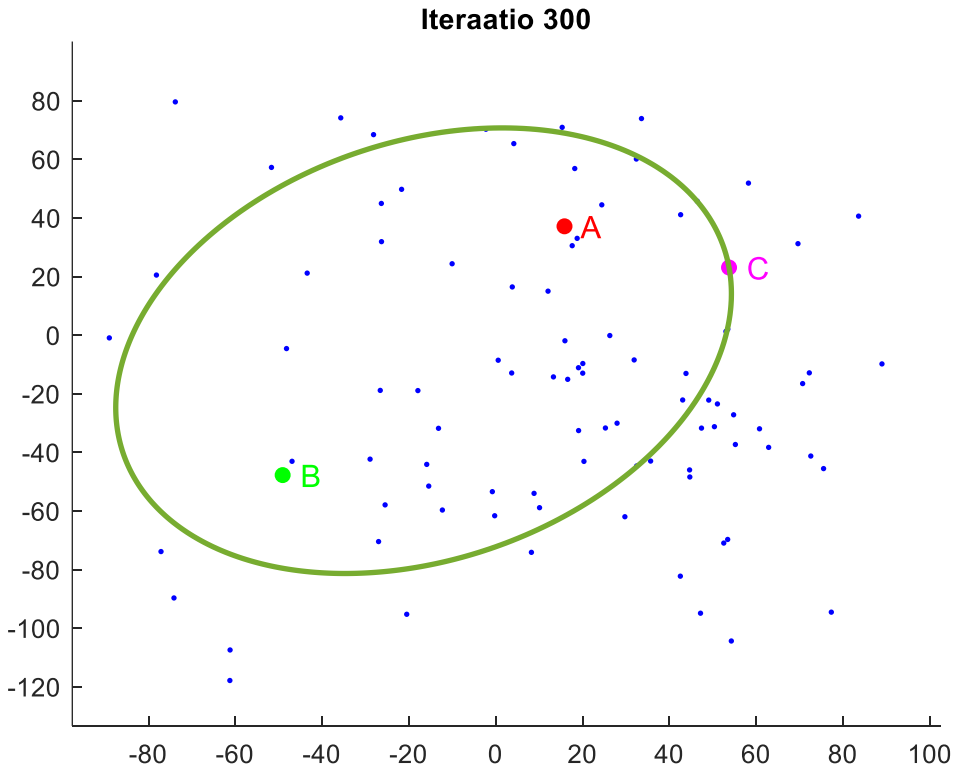
6.3 Ellipsi, T=1,00

Ellipsi-ohjelma suoritetaan, määritetään todennäköisyydeksi T=1,00. Kuvaajista poimitaan kolme kuvaa, ja tarkastellaan joukkoa silmämääräisesti. Hypoteesi T:n arvoon perustuen on, että joukko suppenee. Lopuksi vahvistetaan asia tulostettujen arvojen ja niiden muutosten perusteella.

Kuvassa 20 on ellipsi-ohjelman alkuvaiheita, ja kuvassa 18 iteraatio 300. Kuvassa 21 näkyy paremmin, kuinka alkuperäinen pistejoukko on alkanut suppenemaan määritellyn alueen keskelle, ja pisteitä lähtee määritellyn alueen reunoilta. Tämä on hypoteesin mukaisesti.



Kuva 20. Ellipsi, T=1,00, alkuiteraatiot.



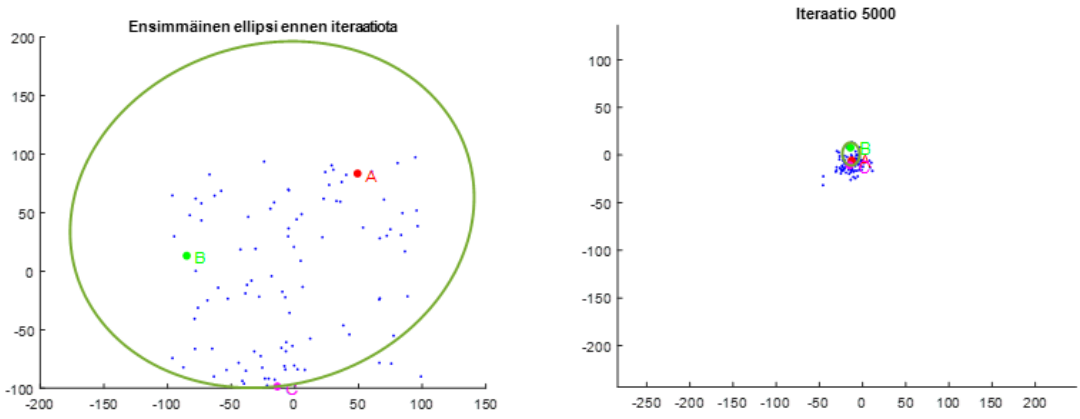
Kuva 21. Ellipsi, T=1,00, iteraatio 300.

Taulukossa 5 on ellipsi-ohjelman pistejoukolle keskimääräinen etäisyys populaation origosta ja pistejoukon varianssin arvoja joka 50. iteraatio. Yksittäisten arvojen sijaan muutoksissa tulee tarkastella kokonaisuutta ja alku- ja lopputilannetta. Keskietäisyyden pieneneminen alkutilanteesta tarkoittaa, että keskimäärin pisteet supistuvat origoon nähden. Varianssin tasainen pieneneminen samoin.

Taulukko 5. Ellipsi, T=1,00, keskietäisyys ja varianssi.

Iteraatio	Keski-etäisyys	Varianssi 1.0e+04
0	**91**	**7,21**
50	**97**	**7,64**
100	**79**	**6,83**
150	**90**	**6,20**
200	**78**	**5,82**
250	**61**	**4,84**
300	**70**	**5,06**

Populaation käyttäytymisen tehostamiseksi, iteraatiokerroiksi asetetaan 5000. Populaation alkutilannetta verrataan lopputilanteeseen, ja tuloksista tarkastellaan populaation tilan kehittymistä.



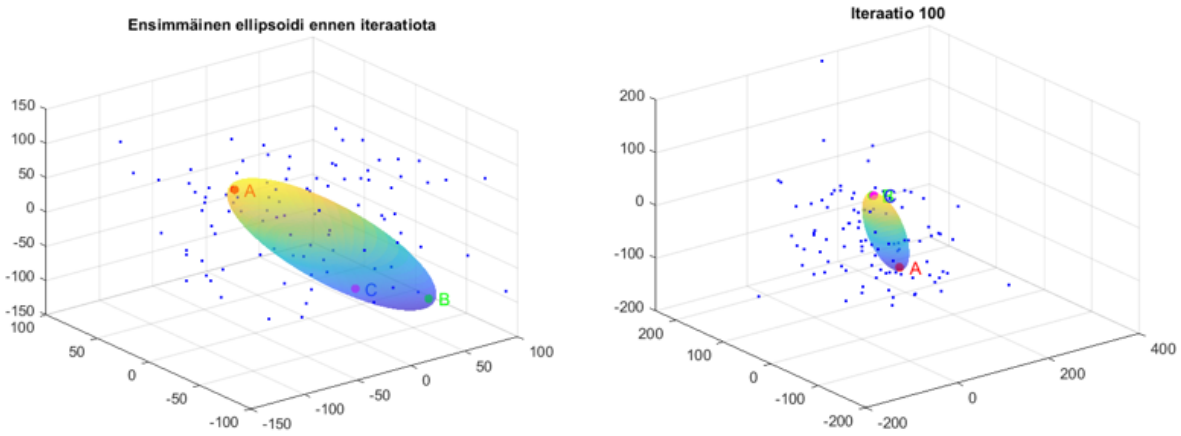
Kuva 22. Ellipsi, T=1,00, populaation kehittyminen.

Kuvassa 22 on populaation kehittyminen alkuasetelmasta lähes pistemäiseksi muodoksi. Pistejoukon kehittyessä se jatkaa supistumista yhä pienemmälle alueelle, ja alkuperäisestä näkökulmasta yksittäisiä populaation pisteitä ei pysty erottamaan toisistaan. Alkuperäisen määrittelyalueen reunat ja nurkat tyhjenevät pisteistä.

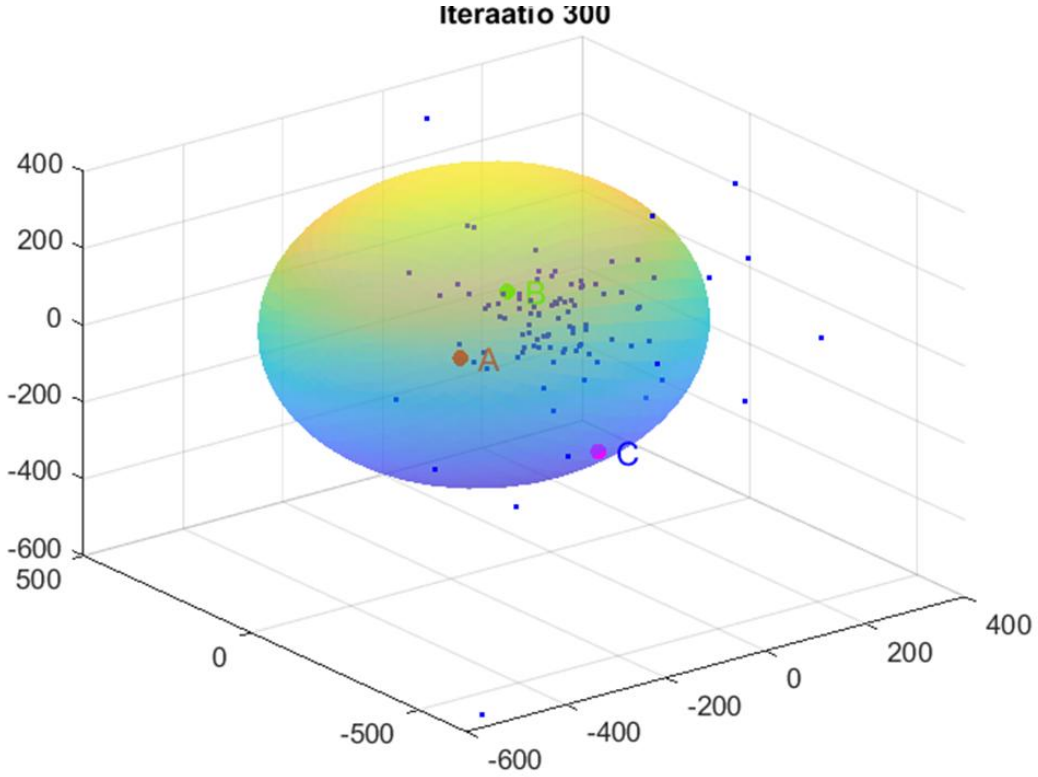
6.4 Ellipsoidi, T=0,00

Ellipsoidi-ohjelma suoritetaan, määritetään todennäköisyydeksi T=0,00. Kuvaajista poimitaan kolme kuvaa, ja tarkastellaan joukkoa silmämääräisesti. Hypoteesi T:n arvoon perustuen on, että joukko hajaantuu. Lopuksi vahvistetaan asia tulostettujen arvojen ja niiden muutosten perusteella.

Kuvassa 23 on ellipsoidi-ohjelman alkuvaiheita, ja kuvassa 24 iteraatio 300. Kuvassa 20 näkynee paremmin, kuinka alkuperäinen pistejoukko on alkanut hajaantumaan määritellyn alueen reunoille ja nurkkiin, ja keskiön pisteet harvenevat. Tämä on hypoteesin mukaisesti. Ilmiötä voi olla hankala havaita 3D-mallin 2D-kuvasta.



Kuva 23. Ellipsoidi, T=0,00, alkuiteraatiot.



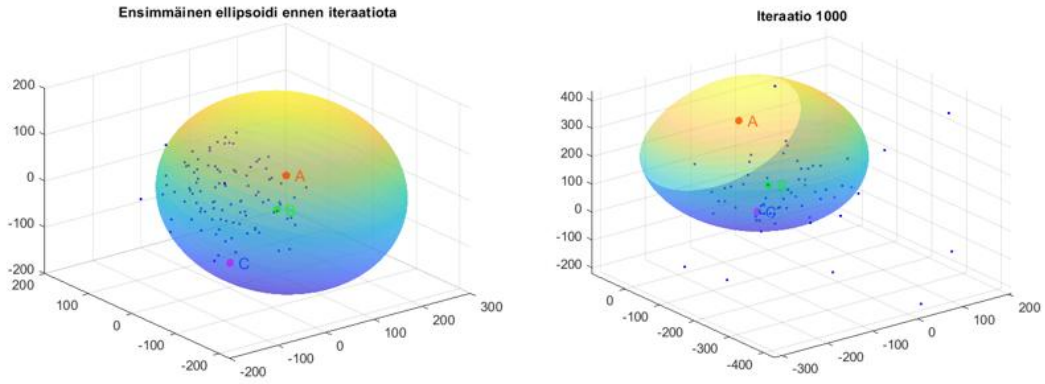
Kuva 24. Ellipsoidi, T=0,00, iteraatio 300.

Taulukossa 6 on ellipsoidi-ohjelman pistejoukolle keskimääräinen etäisyys populaation origosta ja pistejoukon varianssin arvoja joka 50. iteraatio. Yksittäisten arvojen sijaan muutoksissa tulee tarkastella kokonaisuutta ja alku- ja lopputilannetta. Keskietäisyyden suureneminen alkutilanteesta tarkoittaa, että keskimäärin pisteet hajaantuvat origosta. Varianssin tasainen kasvu myös.

Taulukko 6. Ellipsoidi, T=0,00, keskietäisyys ja varianssi.

Iteraatio	Keski-etäisyys	Varianssi 1.0e+04
0	**105**	**0,98**
50	**156**	**1,36**
100	**122**	**1,73**
150	**138**	**2,14**
200	**162**	**3,06**
250	**189**	**4,00**
300	**223**	**5,43**

Populaation käyttäytymisen tehostamiseksi, populaation kooksi asetetaan 20 ja iteraatiokerroiksi 1000. Populaation alkutilannetta verrataan lopputilanteeseen, ja tuloksista tarkastellaan populaation tilan kehittymistä.



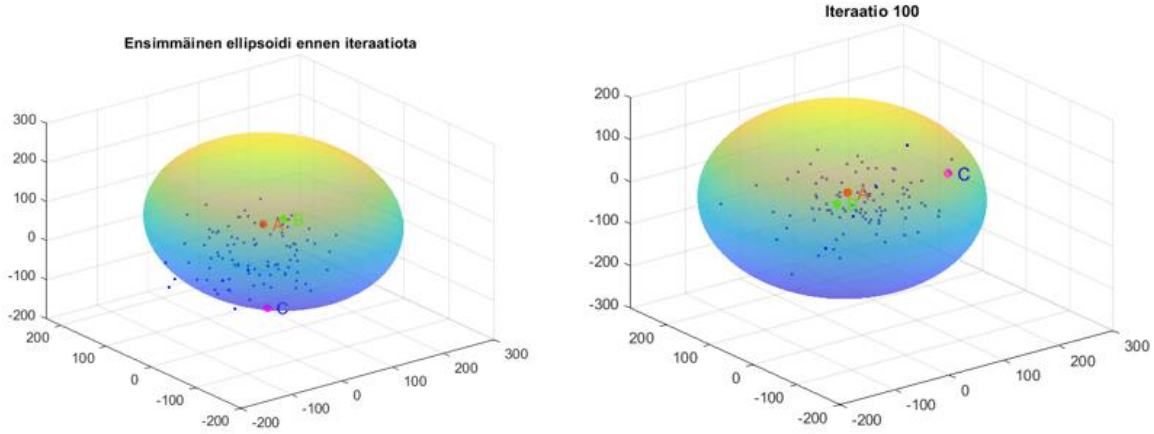
Kuva 25. Ellipsoidi, T=0,00, populaation kehittyminen.

Kuvassa 25 on populaation hajautuminen. Kuten ellipsillä, populaatiopisteet kerääntyvät maltillisesti määrittelyalueen reunoille ja nurkkiin, ja keskiosa harvenee.

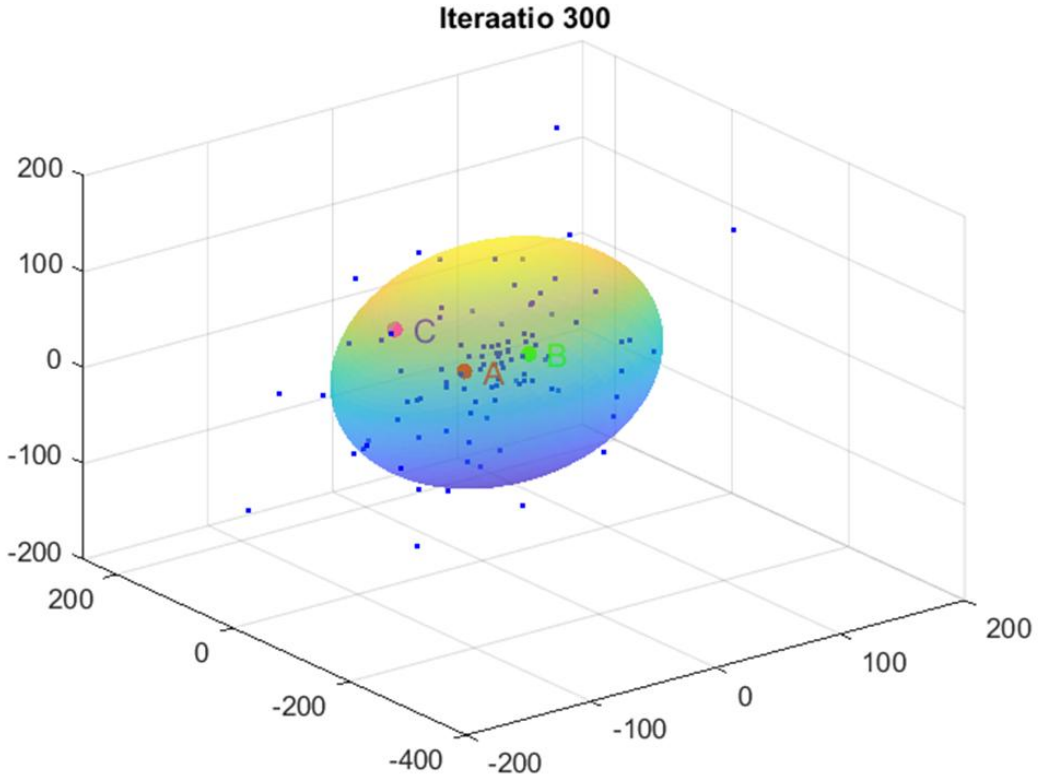
6.5 Ellipsoidi, T=0,50

Ellipsoidi-ohjelma suoritetaan, määritetään todennäköisyydeksi T=0,50. Kuvaajista poimitaan kolme kuvaa, ja tarkastellaan joukkoa silmämääräisesti. Hypoteesi T:n arvoon perustuen on, että joukko suurin piirtein samalla alueella. Lopuksi vahvistetaan asia tulostettujen arvojen ja niiden muutosten perusteella.

Kuvassa 26 on ellipsoidi-ohjelma alkuvaiheita, ja kuvassa 27 iteraatio 300. Kuvissa ei ole huomattavaa muutosta pistejoukossa, millekään alueelle ei kasaannu pisteitä ja missään ei ole erityisen harvaa aluetta, kun verrataan alkuperäiseen pistejoukkoon. Tämä on hypoteesin mukaisesti.



Kuva 26. Ellipsoidi, T=0,50, alkuiteraatiot.



Kuva 27. Ellipsoidi, T=0,50, iteraatio 300.

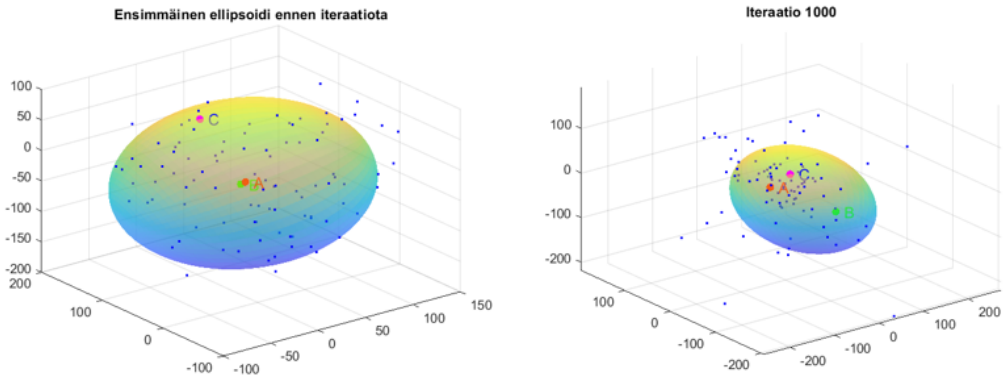
Taulukossa 7 on ellipsoidi-ohjelman pistejoukolle keskimääräinen etäisyys populaation origosta ja pistejoukon varianssin arvoja joka 50. iteraatio. Yksittäisten arvojen sijaan muutoksissa tulee tarkastella kokonaisuutta ja alku- ja lopputilannetta. Keskietäisyyden

pysyminen suunnilleen samana alkutilanteesta tarkoittaa, että keskimäärin pisteet esiintyvät origosta katsottuna samalla etäisyydellä. Varianssin tasaisena pysyminen samoin. Yksittäiset arvot saattavat vaihdella satunnaisuuden vuoksi, mutta tasaantuvat iteraatiokertojen kasvaessa.

Taulukko 7. Ellipsoidi, T=0,50, keskietäisyys ja varianssi.

Iteraatio	Keski-etäisyys	Varianssi 1.0e+04
0	**122**	**0,96**
50	**101**	**1,00**
100	**118**	**0,90**
150	**104**	**0,93**
200	**126**	**1,05**
250	**92**	**1,01**
300	**94**	**1,21**

Populaation käyttäytymisen tehostamiseksi, iteraatiokerroiksi asetetaan 1000. Populaation alkutilannetta verrataan lopputilanteeseen, ja tuloksista tarkastellaan populaation tilan kehittymistä.



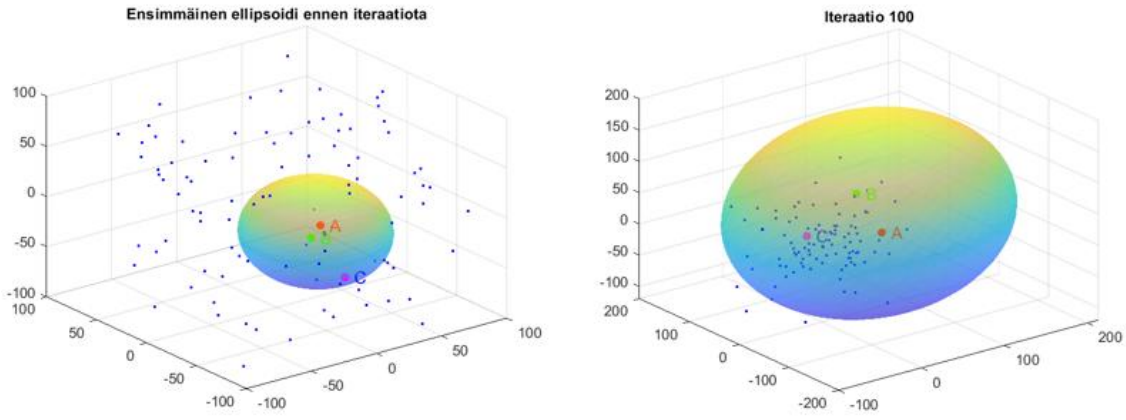
Kuva 28. Ellipsoidi, T=0,50, populaation kehittyminen.

Kuvassa 28 ei ole huomattavaa populaatiopisteiden kerääntymistä eikä hajaantumista, kuten ellipsillä. Iteraatiokertojen nostamisella ei ole kokonaisuuden kannalta merkitystä. Tämä on odotettujen tulosten mukaisesti.

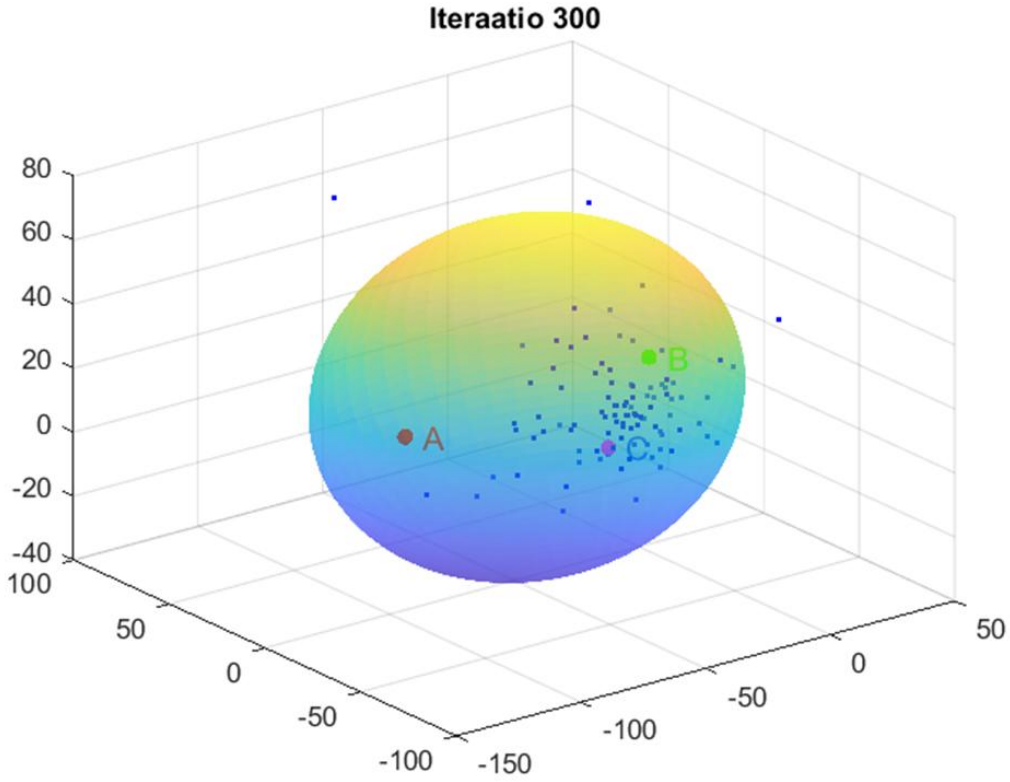
6.6 Ellipsoidi, T=1,00

Ellipsoidi-ohjelma suoritetaan, määritetään todennäköisyydeksi T=1,00. Kuvaajista poimitaan kolme kuvaa, ja tarkastellaan joukkoa silmämääräisesti. Hypoteesi T:n arvoon perustuen on, että joukko suppenee. Lopuksi vahvistetaan asia tulostettujen arvojen ja niiden muutosten perusteella.

Kuvassa 29 on ellipsoidi-ohjelman alkuvaiheita, ja kuvassa 30 iteraatio 300. Kuvassa 24 näkyy paremmin, kuinka alkuperäinen pistejoukko on alkanut suppenemaan määritellyn alueen keskelle, ja pisteitä lähtee määritellyn alueen reunoilta ja kulmista. Tämä on hypoteesin mukaisesti.



Kuva 29. Ellipsoidi, T=1,00, alkuiteraatiot.



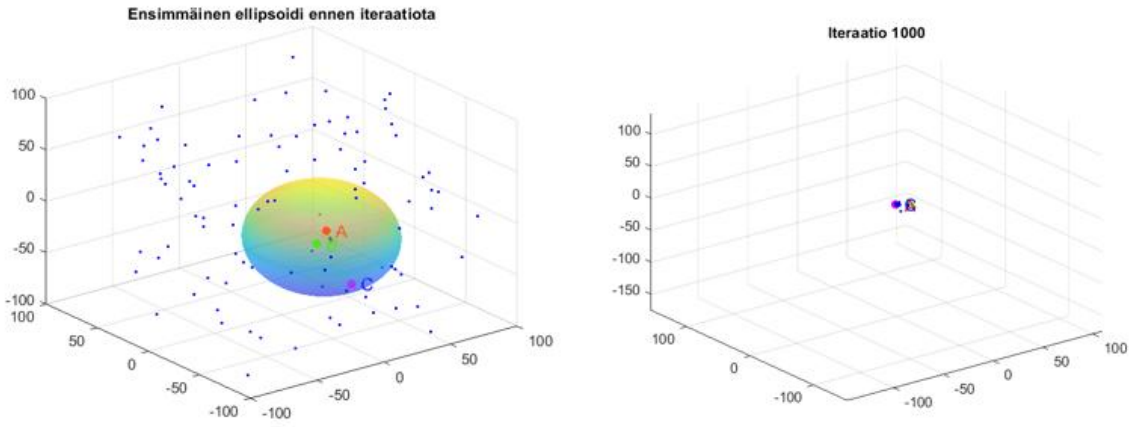
Kuva 30. Ellipsoidi, T=1,00, iteraatio 300.

Taulukossa 8 on ellipsoidi-ohjelman pistejoukolle keskimääräinen etäisyys populaation origosta ja pistejoukon varianssin arvoja joka 50. iteraatio. Yksittäisten arvojen sijaan muutoksissa tulee tarkastella kokonaisuutta ja alku- ja lopputilannetta. Keskietäisyyden pieneneminen alkutilanteesta tarkoittaa, että keskimäärin pisteet supistuvat origoon nähden. Varianssin tasainen pieneneminen samoin.

Taulukko 8. Ellipsoidi, T=1,00, keskietäisyys ja varianssi.

Iteraatio	Keski-etäisyys	Varianssi 1.0e+04
0	**105**	**1,00**
50	**122**	**0,69**
100	**88**	**0,51**
150	**96**	**0,34**
200	**57**	**0,22**
250	**37**	**0,17**
300	**58**	**0,10**

Populaation käyttäytymisen tehostamiseksi, iteraatiokerroiksi asetetaan 1000. Populaation alkutilannetta verrataan lopputilanteeseen, ja tuloksista tarkastellaan populaation tilan kehittymistä.



Kuva 31. Ellipsoidi, T=1,00, populaation kehittyminen.

Kuvassa 31 on populaation kehittyminen alkuasetelmasta lähes pistemäiseksi muodoksi. Pistejoukon kehittyessä se jatkaa supistumista yhä pienemmälle alueelle, ja alkuperäisestä näkökulmasta yksittäisiä populaation pisteitä ei pysty erottamaan toisistaan. Määrittelyalueen reunat ja nurkat tyhjenevät pisteistä.

6.7 Ajanmittaus, hylkäysmenetelmän tarpeellisuus

Ohjelmaan on sisälletty ajanotto millisekunteina ohjelman suoritukselle. Taulukoissa 9 ja 10 on ajanotto ellipsi- ja ellipsoidi-ohjelmille. T:n arvo pysyy samana 0,5, ja kuvaajia piirretään joka 50. iteraatio huolimatta kokonaisiteraatiomäärästä. Iteraatiomäärä aloitetaan yhdestä ja kasvatetaan merkittyyn määrään, ja jokaisesta suoritteesta merkitään tuloksiin aika.

Taulukko 9. Ellipsi-ohjelman ajanmittaus.

Iteraatiot, ellipsi-ohjelma	Aika [ms]
1	176.91
2	187.25
3	211.16
10	333.51
100	1703.72
300	4700.24
500	7139.02
1000	14690.13

Taulukko 10. Ellipsoidi-ohjelman ajanmittaus.

Iteraatiot, ellipsi-ohjelma	Aika [ms]
1	201.94
2	216.65
3	240.50
10	334.95
100	1683.98
300	4201.11
500	7182.29
1000	14589.92

Yksittäisten iteraatioiden vaihtelevuus voi selittyä ohjelmiston evästeillä. Ohjelmisto voi tallentaa ohjelmasta tietoa optimoidakseen seuraavan suorituksen. Keskenään ellipsi- ja ellipsoidi-ohjelmaa vertailemalla nähdään, että ellipsoidi-ohjelma on hieman hitaampi, ja se johtunee kolmannen ulottuvuuden lisäämästä kuormituksesta ohjelman suoritukseen.

Kuitenkin jopa 1000 iteraatiolla ohjelman ajankesto on maltillinen molemmille ohjelmille, noin 14–15 sekuntia. Tutkitulla pistejoukolla tutkittava tieto saavutetaan pienemmällä iteraatiomäärällä. Jos on tarve tutkia erilaista pistejoukkoa tai ohjelmasuorituksia tullaan tekemään huomattavan monta kertaa, voi olla hyvä tarkastella ohjelman sisältöä uudelleen tai perehtyä toiseen ohjelmistoon. Tässä tutkimuksessa ohjelman suoritusten ajankesto on suhteellisen pieni.

Taulukko 11. Hylkäysmenetelmän käyttö.

	Tapaukset	**Osuus (tapaukset/kaikki)**
Ellipsi, T=0,0	0	0
Ellipsi, T=0,5	82	0,27
Ellipsi, T=1,0	153	0,51
Ellipsoidi, T=0,0	0	0
Ellipsoidi, T=0,5	78	0,26
Ellipsoidi, T=1,0	129	0,43

Taulukossa 11 on iteraatiokohtaiset tapaukset hylkäämismenetelmän käytöstä ohjelman suorituksen aikana. Yhden iteraatiotapauksen aikana on käytetty yhden tai useamman kerran menetelmää. Jokaisessa ohjelmansuorituksessa on ollut 300 iteraatiokertaa, ja molempia ellipsi- ja ellipsoidi-ohjelmaa on suoritettu T:n arvoilla 0, 0,5 ja 1.

T:n ollessa 0, ohjelma ei käytä kertaakaan hylkäysmenetelmää. Tämä on johdonmukainen tulos, sillä hylkäysmenetelmää käytetään vain, kun piste lisätään ellipsin tai pyörähdysellipsoidin sisälle. Eniten käyttökertoja hylkäämismenetelmälle on T:n ollessa 1, ja ellipsi-ohjelma käyttää hylkäämismenetelmää enemmän kuin ellipsoidi-ohjelma. Hylkäämismenetelmä on valmisteltu erityisen litteitä ellipsejä ja pyörähdysellipsoideja varten, joten ellipsi saattaa olla alttiimpi muodostumaan litteäksi. T:n arvolla 0,5, tulokset ovat T:n muiden arvojen välissä.

Hylkäysmenetelmää käytetään mahdollisesti jopa puolella iteraatiokerroista. Osuus kaikista tapauksista on sen verran korkea, että hylkäysmenetelmä on hyvä sisällyttää koodiin tasaamaan jakauman oikeanlaiseksi. Poikkeus on, jos ennalta tietää käyttävänsä vain arvoa T=0.

7 Johtopäätökset

Algoritmia voidaan käyttää tutkittaessa hyperelliptistä satunnaisjakaumaa, ja Matlab sopii tulosten tuottamiseen käytännössä. Algoritmissa pystytään säätämään todennäköisyysparametria T, generoimaan pistejoukko, mallintamaan kuvaajaan satunnaisjakauma ja piirtämään ellipsit ja pyörähdysellipsoidit. Iteraatiokertoja pystytään kasvattamaan. Kuvaajat ja muut tulokset pystytään tallentamaan. Ohjelmiston vaatima aika ohjelmien suoritukseen on suhteellisen pieni.

Tulokset täsmäävät niille asetettuja odotusarvoja. Ennakolta asetetut oletusarvot olivat, että joukko suppenee todennäköisyyden T ollessa suurempi kuin 0,5, ja laajenee T ollessa pienempi kuin 0,5. T ollessa tasan 0,5 joukko ei suppene eikä laajene, vaan pysyy suunnilleen tasaisena. Koetta toistettaessa todetaan pisteistä joukon suppeneminen tai laajeneminen. Tulokset saadaan molemmille ellipsille ja pyörähdysellipsoidille, ja näistä saadaan kuvaajien lisäksi tulokset pistejoukon keskimääräiselle etäisyydelle origosta ja pistejoukon varianssille.

Tulevaisuudessa algoritmia voidaan laajentaa käsittelemään yhä moniulotteisempia pyörähdyshyperellipsoideja, ja pyörähdyskappaleen määritelmän vuoksi moniulotteisillekin pyörähdyshyperellipsoideille riittää kolme pistettä pyörähdyskappaleen muodostamiseen. Näille n-ulotteisille pyörähdyskappaleille määritellään kolme pistettä, kaksipolttopistettä ja yksi kehäpiste, ja niistä lasketaan puolipituusakselit. Tätä tietoa ja pyörähdyshyperellipsoidin ominaisuuksia hyödyntämällä noudatellaan algoritmin kaavaa. Samoin kaikki sovelletut prosessit ovat suoraan yleistettävissä tai skaalattavissa korkeampiulotteisiin tapauksiin. Kovin monimutkaisten mallien käsittelyssä on mahdollisuus joutua käyttämään toista ohjelmistoa, mutta algoritmin kaava pysyy samana.

Kehitetty algoritmi on yleisluontoinen, ja sitä on mahdollista hyödyntää erilaisiin käyttötarkoituksiin. Esimerkkinä neuroverkot, sen käsitteen alla on monia hyperellipsejä ja populaatioita yhdistäviä käyttökohteita. Algoritmi on muokattavissa yksilöllisiin tarpeisiin yksinkertaisimmillaan parametrejä vaihtelemalla. Säteittäissuunnassa käytetty

satunnaisjakauma on mahdollista vaihtaa tarvittaessa mihin tahansa muuhun satunnaisjakaumaan, esimerkiksi normaalijakaumaan. Ohjelmasta voidaan myös muokata, poistaa tai lisätä osioita.

Ohjelmaan tuleva, pienehkö, kehityskohde voisi olla käyttäjäystävällisyyden lisääminen. Tällä hetkellä todennäköisyyden T:n ja iteraatiokertojen n määrä vaihdetaan ohjelman sisällä. Ohjelman voisi muokata kysymään käyttäjältä parametrit ohjelmaa suorittaessa, ja käyttäjä syöttäisi luvut. Tämä olisi hyvä lisä myös piirrettyjen kuvaajien määrän luvulle. Käyttäjäystävällisyyden lisääminen olisi optimaalisin siinä vaiheessa, kun käyttökohteeseen otetaan ohjelma työkaluksi. Silloin voidaan ottaa huomioon myös alkupopulaation parametrien vaihdot ja muut muokkaukset.

Olisi myös mielenkiintoista verrata, päästäänkö samaan johtopäätökseen eri tutkimusmenetelmällä. Tässä työssä käytettiin suunnittelutieteellistä tutkimusmenetelmää tutkimusongelman ratkaisun löytämiseksi, mutta toisia tutkimusmenetelmiä, kuten laadullista tai määrällistä, voisi käyttää arvioimaan algoritmin luotettavuutta tai tarkkuutta. Niitä voisi myös hyödyntää jatkojalostamaan tuloksia eri käyttötarkoituksia varten, tai yhdistää näitä menetelmiä toisessa tutkimuksessa.

Lähteet

Abbadia, J. (2023, 24 heinäkuuta). Mitä eroa on: Kvalitatiivinen vs. kvantitatiivinen tutkimus? *Mind The Graph*. Noudettu 19.3.2024 osoitteesta: https://mindthegraph.com/blog/fi/kvalitatiivinen-vs-kvantitatiivinen-tutkimus/

Alestalo, P. (2019). *MS-C1540 Euklidiset avaruudet* [Kurssimateriaali. Aalto-yliopisto]. Noudettu 30.3.2024 osoitteesta: https://mycourses.aalto.fi/pluginfile.php/881366/mod_resource/content/3/eahandout.pdf

Bork, D. & Smajevic, M. (2021, 10-15 lokakuuta). From Conceptual Models to Knowledge Graphs: A Generic Model Transformation Platform. *IEEE. 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C),* s. 610–614. https://doi.org/10.1109/MODELS-C53483.2021.00093

Boyer, C. (1994). *Tieteiden kuningatar – Matematiikan historia, osa I.* Art House. ISBN: 951-884-150-0

Brocke, J. vom, Hevner A., Maedche A. (2020) Introduction to Design Science Research. In: vom Brocke J., Hevner A., Maedche A. (eds) *Design Science Research. Cases,* s. 1–13. Cham. https://doi.org/10.1007/978-3-030-46781-4_1

Couso, J. I., Dubois, D., & Sánchez, L. (2014). *Random Sets and Random Fuzzy Sets as Ill-Perceived Random Variables*. Springer International Publishing. ISBN: 978-3-319-08611-8

Gammell, J. D., Barfoot T. D. & Srinivasa, S. S. (2018, 22 kesäkuuta). Informed Sampling for Asymptotically Optimal Path Planning. *IEEE Transactions on Robotics*, vol. 34 (no. 4), s. 966–984. https://doi.org/10.1109/TRO.2018.2830331

Gogolla, M., Clarisó, R., Selic, B. & Cabot, J. (2021, 10-15 lokakuuta). Towards Facilitating the Exploration of Informal Concepts in Formal Modeling Tools. *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C),* s. 610–614. https://doi.org/10.1109/MODELS-C53483.2021.00044

Han, M., Kan, J., Wang, Y. (2019, 4 marraskuuta). Ellipsoid Fitting Using Variable Sample Consensus and Two-Ellipsoid-Bounding-Counting for Locating Lingwu Long

Jujubes in a Natural Environment. *IEEE Access, vol. 7, s. 164374–164385.* https://doi.org/10.1109/ACCESS.2019.2951473

Hauser, D., Caudal, G., Guimbard, S., & Mouche, A. (2007, 23–28 heinäkuuta). Probability density function of ocean surface slopes from radar observations. *2007 IEEE International Geoscience and Remote Sensing Symposium, Barcelona, Spain, 2007.* s. 1346–1349. https://doi.org/10.1109/IGARSS.2007.4423055.

Hoffman, K. (2013). *Analysis in Euclidean Space.* Courier Corporation. ISBN: 0-486-45804-0

Hsu, H. (1997). Schaum's Outline of Theory and Problems of Probability, Random Variables, and Random Processes. The McGraw-Hill Companies, Inc. ISBN: 0-07-030644-3

Hu, X., Wen, S. & Lam, H.K. (2022, heinäkuu). Dynamic random distribution learning rate for neural networks training. *Applied Soft Computing. Volume 124, July 2022.* https://doi.org/10.1016/j.asoc.2022.109058

Huang, Y.-H. (2013, marraskuu). A note on hyper ellipse method for classifying biological and medical data. *Computers in Biology and Medicine. Volume 43, Issue 11, November 2013, Pages 1978-1986.* https://doi.org/10.1016/j.compbiomed.2013.08.007

Kesäniemi, M. & Virtanen, K. (2017, 25 tammikuuta). Direct Least Square Fitting of Hyperellipsoids. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 40 (no. 1), s. 63–76. https://doi.org/10.1109/TPAMI.2017.2658574

Kivelä, S. (n.d.) *Ellipsi.* [Oppimateriaali. Aalto-yliopisto]. Noudettu 19.10.2024 osoitteesta: https://matta.hut.fi/matta2/isom/html/2astkayr2.html

Krishnan, V., Kumar, A. & Srikanthan S. (2011, 8–10 huhtikuuta). Accelarating the Euclidean distance matrix computation using GPUs. *2011 3rd International Conference on Electronics Computer Technology*, s. 422–426. https://doi.org/10.1109/ICECTECH.2011.5941934

Laksito, A. D., Prabowo, D., Sakti, S. M. & Sari, B. W. (2022. 13–14 joulukuuta). Music Recommendation System Using Content-based Filtering Method with Euclidean Distance Algorithm. *2022 6th International Conference on Information*

Technology, Information Systems and Electrical Engineering (ICITISEE), s. 385–390, https://doi.org/10.1109/ICITISEE57756.2022.10057753

Li, S., Wan, X. & Xue, X. (2022, 16–18 joulukuuta). Stochastic Simulation and Analysis of Geometric Model of Probability. *2022 Euro-Asia Conference on Frontiers of Computer Science and Information Technology (FCSIT), Beijing, China, 2022,* s. 228–231. https://doi.org/10.1109/FCSIT57414.2022.00054.

Li, W., Mclernon, D., Lei, J., Ghogho, M., Zaidi, S. A. R. & Hui, H. (2019, 3 toukokuuta). Cryptographic Primitives and Design Frameworks of Physical Layer Encryption for Wireless Communications. *IEEE Access, vol. 7, s. 63660–63673, 2019.* https://doi.org/10.1109/ACCESS.2019.2914720.

Lu, B., Charlton, M., Harris, P. (2012). Geographically Weighted Regression using a non-euclidean distance metric with simulation data. *2012 First International Conference on Agro- Geoinformatics (Agro-Geoinformatics), Shanghai, China, 2012,* s. 1–4. https://doi.org/10.1109/Agro-Geoinformatics.2012.6311652

MathWorks. (n.d.). *rand*. MathWorks. Noudettu 9.10.2024 osoitteesta: https://se.mathworks.com/help/matlab/ref/rand.html

McMeekin, D. A. & Lin, T. G., Palmer, R., Xia, J. C. (2014, 19–21 elokuuta). Automatic generation of station catchment areas: A comparison of Euclidean distance transform algorithm and location-allocation methods. *2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, s. 963–967. https://doi.org/10.1109/FSKD.2014.6980969

OpenAI. (2024). *ChatGPT* (9.10.2024) [ellipsin ja ellipsoidin ohjelmien luominen ja muokkaaminen tekoälyä hyödyntäen]. Noudettu 9.10.2024 osoitteesta: https://chat.openai.com/chat

Park, Y. S. & Lee, Y. S. (2011). Diagnostic cluster analysis of mathematics skills. *IERI Monograph Series: Issues and Methodologies in Large-Scale Assessments. 4.* s. 75–108. Noudettu 6.4.2024 osoitteesta: https://www.researchgate.net/publication/263889770_Diagnostic_cluster_analysis_of_mathematics_skills

Peda.net. (n.d.). *Pythagoraan lause.* Peda.net. Noudettu 15.4. osoitteesta: https://peda.net/kotka/perusopetus/kotkansaaren-koulu/kt/oppiaineet/matematiikka/jrl/8-luokka/geometria/pythagoraan-lause

Peffers, K., Tuunanen, T., Rothenberger, M., Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems. Winter 2007–8, Vol. 24.* s. 45–77. https://doi.org/10.2753/MIS0742-1222240302

Pote, S., Lande, B. K., Mammen, P. M. (2016). Elliptic Curve arithmetic over extension field to intensify security and privacy. *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India, 2016,* s. 1890–1893. https://doi.org/10.1109/WiSPNET.2016.7566470

Raisamo, R. (2006). *Tutkimuskurssi.* [julkaisu, Tampereen yliopisto]. Noudettu 19.3.2024 osoitteesta: https://www.researchgate.net/publication/233810408_Tutkimuskurssi?enrichId=rgreq-d51203a68695362236410ec94dbaf558-XXX&enrichSource=Y292ZXJQYWdlOzIzMzgxMDQwODtBUzoxMDMxMDI2NzgwNDQ2ODVAMTQwMTU5MzAwMzIwMQ%3D%3D&el=1_x_2&_esc=publicationCoverPdf

Rajan, D. (2012) *Mathematical Foundations for Signal Processing, Communications, and Networking.* Boca Raton. CRC Press. 1st Edition. https://doi.org/10.1201/9781351105668

Shynk, J. J. (2012). *Probability, Random Variables, and Random Processes*. Wiley-Interscience. ISBN: 9780470242094

Smith, K. (2013). *Precalculus: A Functional Approach to Graphing and Problem Solving.* Jones & Bartlett Learning. ISBN: 978-1-4496-4916-6

Starodubov, K. V., Shelpuck, M. I., Pushkin, P. Y., Hussein Najm Abd Ali, A. A., Artemova, S, V., Mikhaylova, V. S. (2023) Improving the Efficiency of Data Analysis in Information Systems through the Use of Elliptic Curves. *2023 5th International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA), Lipetsk, Russian Federation, 2023,* s. 802–805. https://doi.org/10.1109/SUMMA60232.2023.10349544

Tatum, J. (n.d.) The Ellipse. Noudettu 19.10.2024 osoitteesta: https://phys.libretexts.org/Bookshelves/Astronomy__Cosmology/Celestial_Mechanics_(Tatum)/02%3A_Conic_Sections/2.02%3A_The_Ellipse

Timotheou, S. (2009, 30 huhtikuuta). The Random Neural Network: A Survey. *The Computer Journal. Oxford University Press. The Author 2009.* https://doi.org/10.1093/comjnl/bxp032

Vasile-George, M. (2014). Hyper-ellipsoid Clustering of Time Series. A Case Study for Daily Stock Returns. *Procedia Economics and Finance,* vol. 15 (no. 2014), s. 777–783. https://doi.org/10.1016/S2212-5671(14)00497-3

Venable, J. R. Pries-Heje, J., Baskerville, R. L. (2017). Choosing a Design Science Research Methodology. *Australasian Conference on Information Systems. 2017, Hobart, Australia. ACIS 2017 Proceedings. 112.* AISeL: https://aisel.aisnet.org/acis2017/112

Voronova, A.V. & Tsareva, P.E. (2022) The Problem of Analyzing Data Models in Non-Euclidean Spaces in Artificial Intelligence Systems. *2022 Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), Saint Petersburg, Russian Federation*, 2022, s. 493–496. https://doi.org/10.1109/ElConRus54750.2022.9755848

Wallhoff, F., Eickeler, S., & Rigoll, G. (2001). A comparison of discrete and continuous output modeling techniques for a pseudo-2D hidden Markov model face recognition system. *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205), Thessaloniki, Greece, 2001, s. 685–688 vol.2,* https://doi.org/10.1109/ICIP.2001.958586

Wood Mt. Design, LLC. (n.d.) *Hyper Ellipse.* Noudettu 19.10.2024 osoitteesta: https://www.woodmtdesign.com/design-musings/hyper-ellipse/

Liitteet

Liite 1. Algoritmi 1.

```
tic; % Käynnistetään ajastin

% Määritellään todennäköisyys, että piste on ellipsin sisällä
(0-1)
T = 0.50;

population_size = 100;
population = -100 + (200).*rand(population_size, 2);

% Valitaan alkuperäiset pisteet
A = population(randi(population_size), :);
B = population(randi(population_size), :);
C = population(randi(population_size), :);
center = real((A + B) / 2);  % Keskipiste
d1 = norm(C - A);  % Polttopiste1
d2 = norm(C - B);  % Polttopiste2
circle = norm(A - center);  % Etäisyys kehälle

% Piirretään ensimmäinen ellipsi
figure;
scatter(population(:,1), population(:,2), 'b.');
hold on;
scatter(A(1), A(2), 'ro', 'filled');
scatter(B(1), B(2), 'go', 'filled');
scatter(C(1), C(2), 'mo', 'filled');
text(A(1), A(2), '  A', 'Color', 'r', 'FontSize', 12);
text(B(1), B(2), '  B', 'Color', 'g', 'FontSize', 12);
text(C(1), C(2), '  C', 'Color', 'm', 'FontSize', 12);
drawEllipse(center, A, B, C);
title('Ensimmäinen ellipsi ennen iteraatiota');

% Neliöjuurifunktio, imaginääriluvun välttäminen
safe_sqrt = @(x) sqrt(max(x, 0));

% Keskimääräinen etäisyys ja varianssi
avg_distance = mean(sqrt((population(:,1) - center(1)).^2 +
(population(:,2) - center(2)).^2));
total_variance = var(population(:,1)) + var(population(:,2));
rejection_rounds = 0;

n = 300; % Iteraatiot

%Tulostaulukko
results = zeros(n, 3);
results(1, :) = [0, avg_distance, total_variance];

for iter = 1:n
```

```
    rng('shuffle');

    % Satunnaiset pisteet yksikköympyrästä
    x = -1 + 2*rand(10000,1);
    y = -1 + 2*rand(10000,1);
    distances = sqrt(x.^2 + y.^2);
    x = x(distances <= 1);
    y = y(distances <= 1);

    % Skaalataan ja siirretään generoidut pisteet
    ellipse_x_scale = (d1 + d2)/2;
    ellipse_y_scale  =  safe_sqrt(ellipse_x_scale^2  -  cir-
cle^2);
    x_ellipse = center(1) + x * ellipse_x_scale;
    y_ellipse = center(2) + y * ellipse_y_scale;

    % Tarkistetaan, mitkä pisteet ovat ellipsin sisällä
    distances_from_center  =  ((x_ellipse  -  center(1)).^2  /
ellipse_x_scale^2) + ...
                             ((y_ellipse  -  center(2)).^2  /
ellipse_y_scale^2);
    x_inside = x_ellipse(distances_from_center <= 1);
    y_inside = y_ellipse(distances_from_center <= 1);

    satun = rand;

    % Valitaan uusi piste C todennäköisyydellä T
    if satun < T && ~isempty(x_inside)
        random_index = randi(length(x_inside));
        x_new = x_inside(random_index);
        y_new = y_inside(random_index);
        distance_from_center = ((x_new - center(1))^2 / el-
lipse_x_scale^2) + ...
                               ((y_new - center(2))^2 / el-
lipse_y_scale^2);

        if distance_from_center <= 1
            max_rejections = 20;  % Enimmäismäärä hylkäyksiä
ennen hyväksyntää
            rejection_count = 0;

            while true
                R_ellipsi  =  sqrt((x_new  -  center(1))^2  +
(y_new - center(2))^2);
                O_ellipsi = ellipse_x_scale;
                P_reject  =  R_ellipsi  /  O_ellipsi;   %  Hyl-
käystodennäköisyys
                r = rand;

                % Jos hylkäysehto ei täyty tai ollaan hylätty
liian monta kertaa, hyväksytään piste
```

```
                if P_reject <= r || rejection_count >=
max_rejections
                    C_new = [x_new, y_new];
                    break;
                end

                random_index = randi(length(x_inside));
                x_new = x_inside(random_index);
                y_new = y_inside(random_index);
                distance_from_center = ((x_new - cen-
ter(1))^2 / ellipse_x_scale^2) + ...
                                ((y_new - center(2))^2 / el-
lipse_y_scale^2);

                rejection_count = rejection_count + 1;

                if rejection_count >= max_rejections
                    disp('Maksimihylkäys täynnä.')
                end
            end
                if rejection_count > 0
                rejection_rounds = rejection_rounds+1;
                end
                rejection_count = 0;
        else
            random_index = randi(length(x));
            C_new = [x(random_index) * ellipse_x_scale,
y(random_index) * ellipse_y_scale];
        end

    % Jos tasan 0.5, arvotaan uudelleen
    elseif satun == 0.5 && T == 0.5
        satun = rand;

    % Piste ellipsin ulkopuolelle
    elseif satun > T
        max_attempts = 20;
        attempt = 1;
        range = 100;  % Alkuperäinen alueen puolikas
        found = false;

        while ~found && attempt <= max_attempts
            x_rand = -range + 2 * range * rand;
            y_rand = -range + 2 * range * rand;
            random_point = [x_rand, y_rand];

            major_axis = ellipse_x_scale;
            minor_axis = safe_sqrt(ellipse_x_scale^2 - cir-
cle^2);

            ellipse_equation = (random_point(1)^2 / ma-
jor_axis^2) + (random_point(2)^2 / minor_axis^2);
```

```
            if ellipse_equation > 1
                found = true;
                C_new = random_point;
            else
                attempt = attempt + 1;
                range = range * 1.2;  % Laajennetaan aluetta
asteittain
            end
        end

        % Jos ei löytynyt kelvollista pistettä, hyväksytään
viimeinen yritys
        if ~found
            C_new = random_point;
        end

    else
        random_index = randi(length(x));
        C_new = [x(random_index) * ellipse_x_scale, y(ran-
dom_index) * ellipse_y_scale];
    end

    found = false;
    % Päivitetään C uuteen arvoon
    population(find(all(population == C, 2), 1), :) =
real(C_new);
    C = real(C_new);

    % Valitaan uusi A, B ja C populaatiosta ja päivitetään
    A = population(randi(population_size), :);
    B = population(randi(population_size), :);
    C = population(randi(population_size), :);
    d1 = norm(C - A);
    d2 = norm(C - B);
    center = real((A + B) / 2);
    circle = norm(A - center);

    avg_distance = mean(sqrt((population(:,1) - cen-
ter(1)).^2 + (population(:,2) - center(2)).^2));
    total_variance = var(population(:,1)) + var(populat-
ion(:,2));
    results(iter+1, :) = [iter, avg_distance, total_vari-
ance];

    % Kuvien piirtäminen määrätyin välein
    if mod(iter, 100) == 0
        figure;
        scatter(population(:,1), population(:,2), 'b.');
        hold on;
        scatter(A(1), A(2), 'ro', 'filled');
        scatter(B(1), B(2), 'go', 'filled');
        scatter(C(1), C(2), 'mo', 'filled');
```

```
        text(A(1), A(2), '   A', 'Color', 'r', 'FontSize',
12);
        text(B(1), B(2), '   B', 'Color', 'g', 'FontSize',
12);
        text(C(1), C(2), '   C', 'Color', 'm', 'FontSize',
12);
        drawEllipse(center, A, B, C);
        title(['Iteraatio ', num2str(iter)]);
    end
end

time = toc;
% Tulostetaan taulukko lopuksi
disp('Iteraatio | Keski-etäisyys | Varianssi');
disp(results);
display(rejection_rounds)
fprintf('Ellipsi-laskenta kesti %.2f ms\n', time * 1000);

% Funktio ellipsin piirtämiseen
function drawEllipse(center, A, B, C)
    t = linspace(0, 2*pi, 1000);
    AB = B - A;
    d1 = norm(C - A);
    d2 = norm(C - B);
    circle = norm(A - center);

    semi_major_axis = (d1 + d2)/2;
    semi_minor_axis = sqrt(max(semi_major_axis^2 - circle^2,
0));

    angle = atan2(AB(2), AB(1));

    x = semi_major_axis * cos(t);
    y = semi_minor_axis * sin(t);

    R = [cos(angle), -sin(angle); sin(angle), cos(angle)];
    xy = R * [x; y];

    plot(center(1) + xy(1,:), center(2) + xy(2,:), 'Lin-
eWidth', 2);
    hold on;
end
```

Algoritmi 1. Ellipsin Matlab-ohjelma. Algoritmi on luotu tekoälyä hyödyntäen yhdistämällä useaa kehotetta (OpenAI, 2024).

Liite 2. Algoritmi 2.

```
tic; % Käynnistetään ajastin

% Määritellään todennäköisyys, että piste on ellipsoidin si-
sällä (0-1)
T = 0.50;

% Määritellään populaation koko ja generoidaan populaatiopis-
teet
population_size = 100;
population = -100 + (200).*rand(population_size, 3);

% Alustetaan kolme pistettä A, B, C
A = population(randi(population_size), :);
B = population(randi(population_size), :);
C = population(randi(population_size), :);
center = real((A + B) / 2);  % Keskipiste
d1 = norm(C - A);  % Polttopiste1
d2 = norm(C - B);  % Polttopiste2
circle = norm(A - center);  % Etäisyys kehälle

% Piirretään ensimmäinen ellipsoidi
figure;
scatter3(population(:,1), population(:,2), population(:,3),
'b.');
hold on;
scatter3(A(1), A(2), A(3), 'ro', 'filled');
scatter3(B(1), B(2), B(3), 'go', 'filled');
scatter3(C(1), C(2), C(3), 'mo', 'filled');
text(A(1), A(2), A(3),'  A', 'Color', 'r', 'FontSize', 12);
text(B(1), B(2), B(3), '  B', 'Color', 'g', 'FontSize', 12);
text(C(1), C(2),  C(3), '  C', 'Color', 'b', 'FontSize', 12);
drawEllipsoid(center, A, B, C);
title(['Ensimmäinen ellipsoidi ennen iteraatiota']);

% Neliöjuurifunktio, imaginääriluvun välttäminen
safe_sqrt = @(x) sqrt(max(x, 0));

avg_distance = mean(sqrt(sum((population - center).^2, 2)));
total_variance = sum(var(population));
rejection_rounds = 0;

% Iteraatiot
n = 300;
results = zeros(n, 3);
results(1, :) = [0, avg_distance, total_variance];

for iter = 1:n
    rng('shuffle');

    % Generoidaan satunnaisia pisteitä yksikköpallosta
```

```
    x = -1 + 2*rand(10000,1);
    y = -1 + 2*rand(10000,1);
    z = -1 + 2*rand(10000,1);
    distances = sqrt(x.^2 + y.^2 + z.^2);
    x = x(distances <= 1);
    y = y(distances <= 1);
    z = z(distances <= 1);

    % Ellipsoidin akselit
    ellipsoid_x_scale = (d1 + d2)/2;
    ellipsoid_y_scale = safe_sqrt(ellipsoid_x_scale^2 - cir-
cle^2);
    ellipsoid_z_scale = safe_sqrt(ellipsoid_x_scale^2 - cir-
cle^2);

    % Siirretään ja skaalataan yksikköpallo ellipsoidiin
    x_ellipsoid = center(1) + x * ellipsoid_x_scale;
    y_ellipsoid = center(2) + y * ellipsoid_y_scale;
    z_ellipsoid = center(3) + z * ellipsoid_z_scale;

    % Tarkistetaan, mitkä pisteet ovat ellipsoidin sisällä
    distances_from_center = ((x_ellipsoid - center(1)).^2 /
ellipsoid_x_scale^2) + ...
                            ((y_ellipsoid - center(2)).^2 /
ellipsoid_y_scale^2) + ...
                            ((z_ellipsoid - center(3)).^2 /
ellipsoid_z_scale^2);
    x_inside = x_ellipsoid(distances_from_center <= 1);
    y_inside = y_ellipsoid(distances_from_center <= 1);
    z_inside = z_ellipsoid(distances_from_center <= 1);
    satun = rand;

    % Valitaan uusi piste C ellipsoidin sisälle
    if satun < T && ~isempty(x_inside)

        random_index = randi(length(x_inside));
        x_new = x_inside(random_index);
        y_new = y_inside(random_index);
        z_new = z_inside(random_index);
        distance_from_center = ((x_new - center(1)).^2 / el-
lipsoid_x_scale^2)  +  ((y_new  -  center(2)).^2  /  ellip-
soid_y_scale^2) + ...
                               ((z_new - center(3)).^2 / el-
lipsoid_z_scale^2);

        if distance_from_center <= 1
        % Määritellään ellipsoidille pääakselit
        aa = max([ellipsoid_x_scale, ellipsoid_y_scale, el-
lipsoid_z_scale]); % Suurin pääakseli
        cc = min([ellipsoid_x_scale, ellipsoid_y_scale, el-
lipsoid_z_scale]); % Pienin pääakseli
        max_rejections = 20;
        rejection_count = 0;
```

```
        accepted = false;

        while ~accepted && rejection_count < max_rejections

            % Generoidaan satunnaispiste yksikköpallolta
            C_new = [0.5*x_inside(random_index), 0.5*y_in-
side(random_index), 0.5*z_inside(random_index)];

            % Lasketaan etäisyys origosta (O_ellipsoidi)
            O_ellipsoidi = norm(C_new - center);

            % Lasketaan teoreettinen etäisyys ellipsoidilla
tähän suuntaan (R_ellipsoidi)
            v = (C_new - center) / O_ellipsoidi; % Yksikkö-
suuntavektori
            R_ellipsoidi = (aa * cc * ellipsoid_z_scale) /
sqrt((cc * ellipsoid_z_scale * v(1))^2 + (aa * ellip-
soid_z_scale * v(2))^2 + (aa * cc * v(3))^2);

            % Lasketaan hylkäystodennäköisyys
            P_reject = abs(R_ellipsoidi - O_ellipsoidi) / aa;

            % Hylätään todennäköisyydellä P_reject
            if rand > P_reject
                accepted = true;
            else
                random_index = randi(length(x_inside));

                x_new = x_inside(random_index);
                y_new = y_inside(random_index);
                z_new = z_inside(random_index);
                distance_from_center = ((x_new - cen-
ter(1))^2 / ellipsoid_x_scale^2) + ((y_new - center(2))^2 /
ellipsoid_y_scale^2) + ...
                                       ((z_new - cen-
ter(3))^2 / ellipsoid_z_scale^2);
                aa = max([ellipsoid_x_scale, ellip-
soid_y_scale, ellipsoid_z_scale]);
                cc = min([ellipsoid_x_scale, ellip-
soid_y_scale, ellipsoid_z_scale]);

                rejection_count = rejection_count + 1;

                if rejection_count >= max_rejections
                    disp('Maksimihylkäys täynnä.')
                end
            end
        end

        if rejection_count > 0
        rejection_rounds = rejection_rounds+1;
        end
        rejection_count = 0;
```

```
        % Jos maksimi hylkäysmäärä saavutettiin, kelpuute-
taan viimeinen piste
        if ~accepted
            %C_new = [x_inside(random_index), y_inside(ran-
dom_index), z_inside(random_index)];
            accepted = false;
        end
        else
            random_index = randi(length(x));
            C_new = [x(random_index) * ellipsoid_x_scale,
y(random_index) * ellipsoid_y_scale, z(random_index) * el-
lipsoid_z_scale];
        end

    %Jos tasan 0.5, arvotaan uudelleen
    elseif satun == 0.5 && T == 0.5
            satun = rand;

    %Piste ellipsoidin ulkopuolelle
    elseif satun > T
        % Lasketaan ellipsoidin akselien pituudet
        majora = (d1 + d2)/2;
        minorb = safe_sqrt(majora^2 - circle^2);
        minorc = safe_sqrt(majora^2 - circle^2);

        max_attempts = 20;
        attempt = 1;
        range = 100;  % Alkualue
        outside = false;

        while ~outside && attempt <= max_attempts
            % Generoidaan satunnainen piste laajenevalta
alueelta
            random_point = -range + (2 * range) * rand(1, 3);

            % Tarkistetaan, onko piste ellipsoidin ulkopuo-
lella
            ellipsoid_value = ((random_point(1) - cen-
ter(1))^2 / majora^2) + ...
                              ((random_point(2) - cen-
ter(2))^2 / minorb^2) + ...
                              ((random_point(3) - cen-
ter(3))^2 / minorc^2);

            if ellipsoid_value > 1
                outside = true;
                C_new = random_point;
            else
                attempt = attempt + 1;
                range = range * 1.2;  % Kasvatetaan aluetta
            end
        end
```

```
        % Jos ei löytynyt ulkopuolista pistettä, hyväksytään
viimeinen
        if ~outside
            C_new = random_point;
        end
    else
        random_index = randi(length(x));
        C_new = [x(random_index) * ellipsoid_x_scale, y(ran-
dom_index)  *  ellipsoid_y_scale,  z(random_index)  *  ellip-
soid_z_scale];
    end

    % Päivitetään C uuteen arvoon
    population(find(all(population == C, 2), 1), :) = C_new;
    C = C_new;

    % Päivitetään A, B, C ja keskipiste
    A = population(randi(population_size), :);
    B = population(randi(population_size), :);
    C = population(randi(population_size), :);
    d1 = norm(C - A);
    d2 = norm(C - B);
    center = real((A + B) / 2);
    circle = norm(A - center);

    % Keskimääräinen etäisyys ja varianssi
    avg_distance = mean(sqrt(sum((population - center).^2,
2)));
    total_variance = sum(var(population));
    results(iter+1, :) = [iter, avg_distance, total_vari-
ance];

    % Joka 10. iteraatiossa piirretään kuva
    if mod(iter, 100) == 0
        figure;
        scatter3(population(:,1),  population(:,2),  popula-
tion(:,3), 'b.');
        hold on;
        scatter3(A(1), A(2), A(3), 'ro', 'filled');
        scatter3(B(1), B(2), B(3), 'go', 'filled');
        scatter3(C(1), C(2), C(3), 'mo', 'filled');
        text(A(1), A(2), A(3),'  A', 'Color', 'r', 'FontSize',
12);
        text(B(1), B(2), B(3), '  B', 'Color', 'g', 'Font-
Size', 12);
        text(C(1), C(2),  C(3), '  C', 'Color', 'b', 'Font-
Size', 12);
        drawEllipsoid(center, A, B, C);
        title(['Iteraatio ', num2str(iter)]);
    end
end
```

```
time = toc; % Tallennetaan aika
disp('Iteraatio | Keski-etäisyys | Varianssi');
disp(results);
display(rejection_rounds)

fprintf('Ellipsoidi-laskenta kesti %.2f ms\n', toc * 1000);

% Funktio ellipsoidin piirtämiseen
function drawEllipsoid(center, A, B, C)
    [x, y, z] = sphere(50);  % Generoidaan yksikköpallo
    AB = B - A;
    AC = C - A;
    d1 = norm(C - A);
    d2 = norm(C - B);
    circle = norm(A - center);

    % Lasketaan ellipsoidin pääakselien pituudet
    semi_major_axis = (d1 + d2)/2;
    semi_minor_axis = sqrt(max(semi_major_axis^2 - circle^2,
0));

    % Käännös matriisit AB:n ja AC:n mukaan
    [U, S, ~] = svd([AB.' AC.']);

    % Skaalataan ja pyöritetään ellipsoidiksi
    ellipsoid_points = U * diag([semi_major_axis, semi_mi-
nor_axis, semi_minor_axis]) * [x(:)'; y(:)'; z(:)'];

    % Siirretään ellipsoidi keskelle
    ellipsoid_points = center.' + ellipsoid_points;

    % Piirretään ellipsoidi
    X = reshape(ellipsoid_points(1,:), size(x));
    Y = reshape(ellipsoid_points(2,:), size(y));
    Z = reshape(ellipsoid_points(3,:), size(z));

    surf(X, Y, Z, 'FaceAlpha', 0.5, 'EdgeColor', 'none'); %
Piirretään ellipsoidi läpinäkyvänä
    hold on;
end
```

Algoritmi 2. Ellipsoidin Matlab-ohjelma. Algoritmi on luotu tekoälyä hyödyntäen yhdistämällä useaa kehotetta (OpenAI, 2024).