Kandidaatintutkielma
Fysikaalisten tieteiden kandidaattiohjelma
Fysikaalisten tieteiden kandiohjelma/Opintosuunta

Kvantti-Monte Carlo: Numeerinen ratkaisu atomien perustilan energialle ja aaltofunktiolle

Naiyu Deng

3.7.2023

Ohjaaja(t): dosentti/Ilja Makkonen

Tarkastaja(t): dosentti/Ilja Makkonen

Helsingin Yliopisto
Matemaattis-luonnontieteellinen tiedekunta

PL 64 (Gustaf Hällströmin katu 2a)
00014 Helsingin yliopisto

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Matemaattis-luonnontieteellinen tiedekunta	Fysikaalisten tieteiden kandidaattiohjelma Fysikaalisten tieteiden kandiohjelma/Opintosuunta

Tekijä — Författare — Author

Naiyu Deng

Työn nimi — Arbetets titel — Title

Kvantti-Monte Carlo: Numeerinen ratkaisu atomien perustilan energialle ja aaltofunktiolle

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Kandidaatintutkielma	3.7.2023	30

Tiivistelmä — Referat — Abstract

Lääketieteessä, materiaalifysiikassa ja ilmatieteessä mallinnetaan atomeja, molekyyliyhdisteitä ja kidehiloja. Kvanttifysiikan mukaan näiden systeemien mallintamiseksi tarvitaan tietoa systeemien energiatiloista ja aaltofunktiosta, joiden ratkaisemiseen vaaditaan usein numeerisia menetelmiä. Tässä tutkielmassa käytetään variaatio-Monte Carlo -menetelmää, jonka pohjalta kehitetään yleinen helposti ymmärrettävä algoritmi, jolla ratkaistaan atomisysteemien perustilan energiat ja aaltofunktio, ja joka toteutetaan Fortran-ohjelmointikielellä ja Mathematica-ohjelmalla. Lopussa testataan ohjelman toimivuutta heliumatomilla.

Algoritmin kehittämisessä huomioitiin, että algoritmin olisi oltava helposti ymmärrettävä ja vaatisi lukijalta vain vähän taustatietoa. Algoritmissa otettiin mallia olemassa olevasta kirjallisuudesta, mutta samalla vältettiin teoreettisesti haastavia vaiheita. Aaltofunktion ja perustilan energian saamiseksi tarvittiin tietää, mikä on yleinen yriteaaltofunktio ja Hamiltonin operaattori kaikille mahdollisille atomeille. Päädyttiin Slater-Jastrow-yriteaaltofunktioon ja Born-Oppenheimer-approksimaatioon perustuvaan Hamiltonin operaattorin approksimaatioon. Yriteaaltofunktio ja siten Hamiltonin operaattori ovat parametrisoituja, joita rajoittavat kvanttivariaatioperiaate. Kvanttivariaatioperiaate muuttaa ongelman optimointiongelmaksi, ja tehtäväksi tulee sen aaltofunktion parametriyhdistelmän löytämiseksi, jolla saadaan pienin energian arvio. Optimointiongelman ratkaisemiseksi käytettiin gradient descent -menetelmää.

Energian laskemisessa analyyttinen integrointi ei usein onnistu ja siksi käytettiin Monte Carlo –integrointia, jonka avulla muutettiin energian lauseke paikallisten energioiden keskiarvoksi. Tämä pätee vain kuin elektronikonfiguraatioille on käytetty painotettua otantaa, jolloin elektronikonfiguraatiot jakautuvat aaltofunktion neliön mukaisesti. Painotetussa otannassa käytettiin Metropolis-Hastings algoritmia, joka mahdollistaa konfiguraatioiden otannan ilman, että tiedetään yriteaaltofunktion normalisointivakio.

Algoritmien pohjalta tehtiin koodit Mathematica-ohjelmalla, jolla saadaan mikä tahansa atomin yriteaaltofunktio ja paikallinen energia. Edellä saadut lausekkeet käännetään Python-ohjelmalla kelvollisiksi Fortran-ohjelmaan. Fortran-ohjelmointikielellä suoritettiin parametrien optimointi ja energian lasku. Lopussa testattiin ohjelmaa heliumatomilla.

Tuloksena saadaan heliumatomille -2.898 ± 0.004(Hartree-yksikköjä), joka on lähellä kirjallisuusarvoa $-2.90338583 \pm 0.00000013$. Täten ohjelma ja algoritmi toimivat. Algoritmia olisi jatkossa mahdollista kehittää muun muassa muuttamalla algortimi numeerisemmaksi, mikä nopeuttaisi ohjelman ajoa.

Avainsanat — Nyckelord — Keywords

Mathematica, Fortran, variaatio-Monte Carlo, Monte Carlo -integrointi, perustilan energia, aaltofunktio

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Sisällys

1. Johdanto

Kvanttimekaniikan ensimmäisen oletuksen mukaan systeemin aaltofunktio sisältää kaiken informaaation systeemistä. Systeemin aaltofunktio on usein mahdotonta ratkaista analyyttisesti, ja siksi turvaudumme tässä tutkimuksessa numeeriseen menetelmään. Systeemin perustilan energia ja aaltofunktio usein ratkaistaan kvantti-Monte Carlo -menetelmillä (QMC, quantum Monte Carlo), kuten **variaatio-Monte Carlo -menetelmällä (VMC, variational Monte Carlo)** tai diffuusio-Monte Carlo -menetelmällä (diffusion Monte Carlo). Tässä tutkimuksessa käytämme variaatio-Monte Carlo -menetelmää. Tavoitteenamme on löytää yleinen algoritmi, jolla voimme ratkaista atomien perustilan energia ja aaltofunktio, ja toteuttaa sen jollakin mahdollisimman laskennallisesti tehokkaalla ohjelmointikielellä. Lopussa testataan ohjelmaa heliumatomilla.

1.1 Motivaatio

Perustilan energian ja aaltofunktion tunteminen on olennaista systeemin mallintamisessa ja siten sen ymmärtämisessä. Systeemi voi olla yksikertaisimmillaan atomi tai monimutkaisimmillaan molekyyli, molekyyliyhdisteet ja makroskooppiset kidehilat. Näiden jälkimmäisten systeemien mallintaminen on tärkeä kvanttikemiassa, jossa mallinnetaan ihmisyhteiskunnalle merkittäviä systeemejä. Esimerkiksi ilmastonmuutoksen mallintamisessa mallinnetaan ilmakehän molekyylejä, lääketieteessä etsitään mahdollisia lääkemolekyylejä tietokoneilla ja materiaalifysiikassa tutkitaan materiaalien rakennehiukkasia ja etsitään eksoottisia materiaaleja, jotka voivat olla hyödyllisiä tulevissa energiantuotannossa. Kaikki edellä mainitut tarvitsevat tiedon systeemien perustilan energiasta ja niiden aaltofunktiosta.

Tutkimuksessa käytetään VMC, joka hyödyntää kvanttimekaniikan variaatioperiaatetta, josta puhutaan myöhemmin lisää. On olemassa myös muita variaatioperiaatteeseen perustuvia menetelmiä, joilla voidaan ratkaista systeemin aaltofunktio ja perustilan energia. Esimerkiksi Hartree-Fock menetelmä, density matrix renormalization group -menetelmä ja Ritz-menetelmä ovat tällaisia menetelmiä. Edellä mainitut ovat kuitenkin teoreettisesti hankalia ja vaatii tutkijoilta syvää perehtymistä kvantti-

mekaniikkaan ja menetelmien algoritmiin. Siksi tässä tutkielmassa kehitämme algoritmin, jonka ymmärtämiseeen tarvitaan tutkijoilta vain suhteellisen vähän taustatietoa. Tämä tutkielma kuuluu perustutkimukseen, jonka sovellusmahdollisuudet ovat suuria, kuten edellä mainittujen mahdollisten lääkeaineiden etsiminen.

Uutta tutkimuksessa on muun muassa VMC-menetelmään perustuvan algoritmin implementointi käyttämällä Fortran-ohjelmointikielen ja Mathematica-ohjelman yhdistelmää, ja sekä Python-ohjelmointikieltä edellisten ohjelmien yhdistämisessä. Käytämme algoritmissa systeemin paikallisen energian (local energy) kokonaan sievennettyä funktionaalista muotoa *, joka on tunnetusti laskennallisesti epätehokasta, koska joudumme käsittelemään pitkiä lausekkeita Fortran-ohjelmassamme. Tämä onkin yksi algoritmin puutteista. Kuten aiemmin mainittiin, algoritmimme on vastapainona helpommin ymmärtää.

*eli mahdolliset derivaatat paikallisessa energiassa on laskettu auki Mathematica-ohjelmalla.

2. Teoria

2.1 Taustatiedot

2.1.1 Kvanttimekaaniset operaattorit ja Hamiltonin operaattori

Kvanttimekaniikassa mitattavat suureet vastaavat hermiittisiä operaattoreita. Tässä tutkimuksessa käytetään vain yhtä operaattoria, Hamiltonin operaattoria. Se vastaa systeemin kokonaisenergiaa [†]. Hamiltonin operaattorin ominaisarvot kertovat meille, mitä mahdollisia energia-arvoja systeemi voi saada. Koska nyt halutaan ratkaista perustilan energiaa, joka on pienin mahdollisin energia, tarvitaan vain Hamiltonin operaattorin odotusarvon pienintä energiaa. Tarvitsemme siis Hamiltonin operaattorin odotusarvo, joka on

$$E \equiv \langle H \rangle = \langle \Psi | H | \Psi \rangle, \tag{2.1}$$

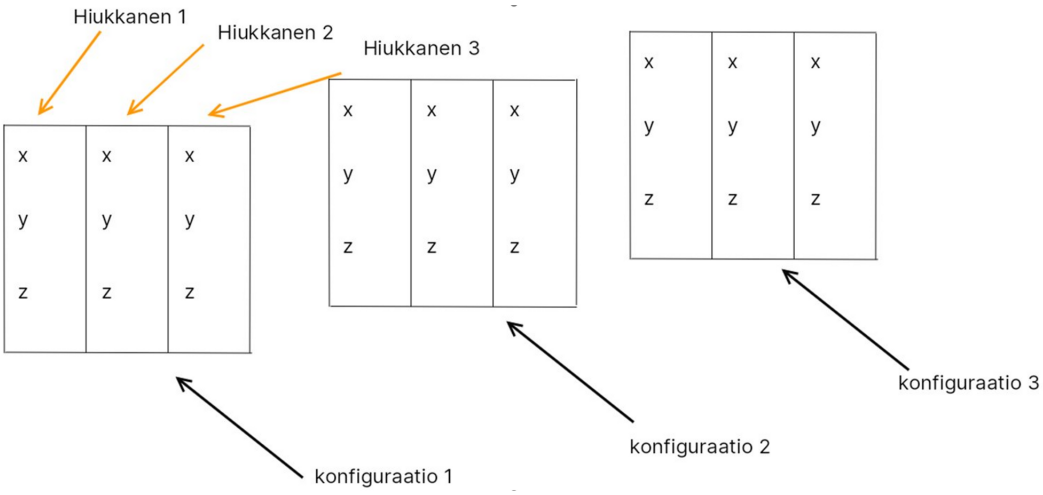
jossa E on energia, $|\Psi\rangle$ on systeemin aaltofunktio ja H systeemin Hamiltonin operaattori.

2.1.2 Termistö: Elektronikonfiguraatiot, konfiguraatioavaruus

Myöhemmin esittämässämme algoritmissamme tuotamme joukon satunnaisia paikkakoordinaatteja kullekin elektronille, ja siksi esitämme tässä nyt, miten se ilmentyy ohjelmassamme.

Ottakaamme esimerkiksi litiumatomi, jolla on kolme elektronia. Kunkin elektronin paikkaa merkitään kolmella koordinaatilla (x, y, z). Kutsun kaikkien elektronien paikkoja yhdellä hetkellä *elektronikonfiguraatioksi* (ks. kuva 2.1), joka on siis 2D-taulu tai matriisi, jossa kukin sarake edustaa yhden hiukkasen paikkakoordinaatit. Hiukkasten paikkoja seuraavalla hetkellä edustaa konfiguraatio 2, ja seuraavalla hetkellä konfiguraatio 3 jne. Koko tensori, eli kaikki matriisit, kutsun *elektronikonfiguraatioavaruudeksi.*

[†] kineettinen energia ja potentiaalienergia



Kuva 2.1: Litiumsysteemin elektronikonfiguraatioavaruus.

2.2 Variaatioperiaate ja variaatiomenetelmä

Variaatio-Monte Carlo -menetelmä, johon meidän lopullinen algoritmimme perustuu, käyttää kvanttimekaniikan variaatioperiaatetta. Variaatioperiaate on yleisesti periaate, joka muuttaa jonkin fysikaalisen ongelman optimointiongelmaksi.* Se ilmenee yleisesti epäyhtälönä.

Kvanttimekaaninen variaatioperiaate on seuraavanlainen: [1]

$$\langle H \rangle \equiv E = \frac{\int \Psi^*(\mathbf{R}) H \Psi(\mathbf{R}) d\mathbf{R}}{\int \Psi^*(\mathbf{R}) \Psi(\mathbf{R}) d\mathbf{R}} = \frac{\int \Psi^2 E_L(\mathbf{R}) d\mathbf{R}}{\int \Psi^2(\mathbf{R}) d\mathbf{R}} \geq E_0, \tag{2.2}$$

jossa E on laskettu arvio perustilan energialle, E_0 on todellisen perustilan energian arvo, Ψ on aaltofunktio, H on Hamiltonin operaattori , $E_L \equiv \frac{H\Psi}{\Psi}$ on paikallinen energia (local energy) ja $\mathbf{R} = \{\mathbf{r}_1, \mathbf{r}_2 ... \mathbf{r}_n\}$ on elektronikonfiguraatio, jossa esim. $\mathbf{r}_1 = (x_1, y_1, z_1)$ on hiukkasen 1 paikkakoordinaatit.

Epäyhtälö kertoo meille, että kun laskemme systeemin energian arviota (eli Hamiltonin operaattorin odotusarvoa), mitä pienempi on saadun energian arvio, sitä lähempi arvio on systeemin todellista perustilan energian arvoa.

Tässä Hamiltonin operaattorin muoto on ennalta määrätty †, mutta aaltofunktio Ψ ei ole. Me siis parametrisoimme aaltofunktion ja etsimme parametrit, jotka minimoivat energian. Kun energian arvo ei pienene enää, olemme saavuttaneet oletetusti energianminimin.

*Eli etsitään parametrit parametrisoidulle funktiolle, jotka minimoivat laskettavan funktion arvon.
†esitetään seuraavassa luvussa

2.3 Hamiltonin operaattori

Hamiltonin operaattorina käytetään Born-Oppenheimerin approksimaatiosta saatua muotoa: [2]

$$H = T + V = -\frac{\hbar^2}{2m}\sum_i \nabla^2_{\mathbf{r}_i} + \frac{1}{4\pi\epsilon_0}\left(\frac{1}{2}\sum_i\sum_{j\neq i}\frac{e^2}{|\mathbf{r}_i - \mathbf{r}_j|} - \sum_i \frac{Ze^2}{r_i}\right), \tag{2.3}$$

jossa Z on atomin järjestysluku (protonien lukumäärä), e on alkeisvaraus, m on hiukkasen eli elektronin massa, $\nabla^2_{\mathbf{r}_i}$ on Laplace-operaattori muuttujan $\mathbf{r}_i$ suhteen ja $\mathbf{r}_i$ on hiukkasen i paikkakoordinaatit.

Hartree-yksiköissä (ks. 3.1) Hamiltonin operaattori saa yksinkertaisemman muodon:

$$H = -\frac{1}{2}\sum_i \nabla^2_{\mathbf{r}_i} + \frac{1}{2}\sum_i\sum_{j\neq i}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} - \sum_i \frac{Z}{r_i}. \tag{2.4}$$

Tässä approksimaatiossa oletetaan seuraavia: [3]

1. Atomin ydin pysyy paikallaan elektroneiden nähden.
2. Systeemiin ei kohdistu ulkoista voimaa ja ulkoista vääntömomenttia.

1. oletus perustuu siihen, että atomin ydin on yleensä paljon raskaampi kuin atomin elektronit. * Koska tässä tutkimuksessa haluamme tarkastella vain atomeja, yksinkertaisuuden vuoksi oletamme muut epäolennaiset voimat mitättömiksi, minkä seurauksena saamme 2. oletus.

2.4 Monte Carlo -integraali

Miten yhtälössä (2.2) esiintyviä integraaleja lasketaan? Näemme yhtälöstä (2.3), että Hamiltonin operaattori muuttuu nopeasti hyvin monimutkaiseksi, kun elektroneiden määrä lisääntyy systeemissä. Näin myös epäyhtälössä (2.2) olevien integraalien laskeminen analyyttisesti muuttuvat todella vaikeiksi. Meidän on siis käytettävä numeerista integrointia, ja tutkimuksessamme käytämme Monte Carlo -integrointia (MC-integrointi). †

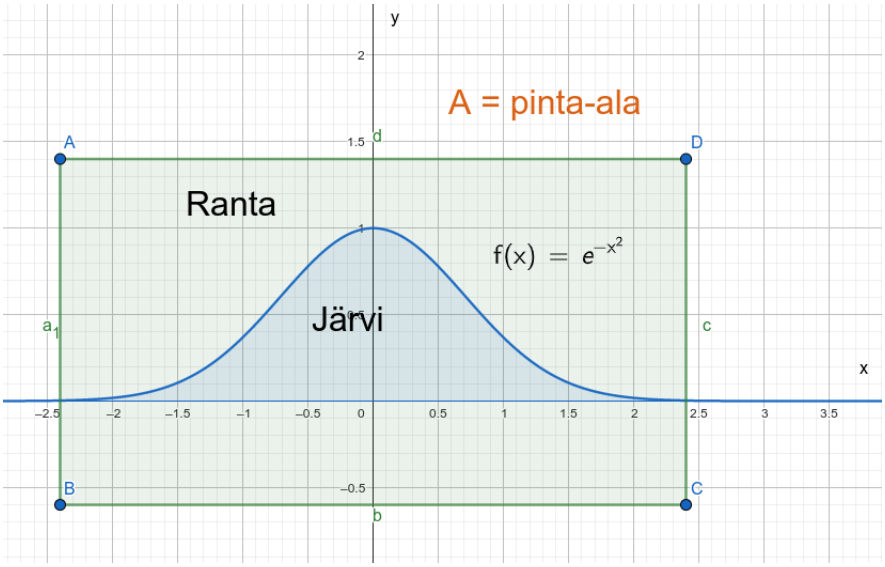
2.4.1 Monte Carlo -integraali: intuitio

MC-integraali tarkoittaa yleisesti joukko stokastisia integraaliapproksimaatioita, jotka saadaan hyödyntämällä suuri määrä satunnaislukuja. Käytännössä monet MC-integraalit käyttävät jonkinlaista keskiarvoistamista.

*Protoni on noin 2000 kertaa raskaampi kuin elektroni.

†MC-integroinnin perusteet on selitetty hyvin kirjassa [1].

Kuvitellaan meillä on järvi (kuva 2.2), joka on sininen alue kuvassa, ja jota rajoittaa funktio $f(x) = e^{-x^2}$ ja x-akseli. Vihreä alue on ranta. Tiedämme rannan ja järven yhteisen pinta-alan olevan A. Kysytään, mikä on järven pinta-ala. [1]



Kuva 2.2: Järvi ja ranta

Järven pinta-ala on helppo laskea analyyttisesti:

$$F = \int_a^b f(x)dx, \tag{2.5}$$

jossa a ja b on järven alku- ja päätepiste.

Kuvitellaan, että olemme alueen A (rannan ja järven) ulkopuolella ja meillä on käytössä rajattomasti kiviä. Voimme arvioida järven pinta-alaa heittämällä kiviä satunnaisesti, mutta tasaisesti jokaiseen suuntaan alueeseen A . Lasketaan, kuinka monesta kivestä syntyy vesiroisketta n_s (kivet, jotka putoavat järveen) ja kuinka monesta ei (kivet, jotka putoavat rantaan). * Käytettyjen kivien yhteismäärä merkitään n. Kun olemme saaneet heitettyä suuren määrän kiviä, saadaan järven pinta-alan arvioksi:

$$F = A\frac{n_s}{n}. \tag{2.6}$$

Näemme, että MC-integraali ei siis ole mitä muuta kuin eräänlainen keskiarvoistamisapproksimaatio ja muut MC-integraalit toimivat samanlaisella tavalla.

*Vaihtoehtoisesti voit päätellä, onko kivi pudonnut rantaan vai veteen kuuntelemalla siitä syntyvää ääntä.

2.4.2 Painotettu otanta ja Metropolis-Hastings algoritmi

Painotettu otanta: perusteet

Näimme edellisessä osiossa, että jotta saamme tarkan integraaliapproksimaation, meidän on käytettävä suuri määrä satunnaislukuja (suuri määrä kiviä), jos ne ovat tasaisesti jakautuneesta tiheysjakaumasta. Tässä osiossa tutustumme tekniikkaan, joka voisi vähentää käytettyjen satunnaislukujen määrää, mutta silti saada suunnilleen yhtä tarkat approksimaatiot.

Halutaan laskea seuraava integraali numeerisesti:

$$F = \int_a^b f(x)dx. \tag{2.7}$$

Sample mean -menetelmässä (yksi MC-integrointimentelmistä) integraalista tulisi tällaiseksi: [1]

$$F = \int_a^b f(x)dx = (b-a)\langle f \rangle \approx (b-a)\frac{1}{n}\sum_{i=1}^{n} f(x_i), \tag{2.8}$$

jossa $\langle f \rangle$ on funktion keskiarvo välillä $[a, b]$, x_i ovat koordinaatin lukuarvot tasaisesta jakaumasta väliltä $[a, b]$ ja n on satunnaislukujen määrä.

Jos $f(x) \geq 0$, kaava siis laskee suorakulmion pinta-alan, jossa pituus on $\langle f \rangle$ (funktion arvo keskimäärin lukuvälillä $[a, b]$) ja leveys on $(b - a)$.

Edellä mainittu tapa arvioida integraali on helppo ymmärtää käsitteellisesti, mutta ei ole laskennallisesti tehokasta. Meidän on käytettävä suuri määrä mittauspisteitä x_i, kun lukuväli $[a, b]$ on suuri, jotta saadaan hyvä arvio integraalille (2.7).

Monte Carlo -integraalin virhearviona voimme käyttää keskiarvon keskivirhettä: [1]

$$\sigma_m = \frac{\sigma}{\sqrt{n}}, \tag{2.9}$$

jossa σ on Monte Carlo -integraaliestimaatin keskihajonta ja n on käytettyjen mittauspisteiden määrä. Monte Carlo -integraalin varianssi lasketaan seuraavasti:

$$\sigma^2 = \langle f^2 \rangle - \langle f \rangle^2, \tag{2.10}$$

jossa keskiarvot lasketaan seuraavasti:

$$\langle f^2 \rangle = \frac{1}{n}\sum_{i=1}^{n}(f(x_i))^2, \tag{2.11}$$

$$\langle f \rangle = \frac{1}{n} f(x_i). \tag{2.12}$$

Monte Carlo -integraaliapproksimaation virhe on siis riippuvainen integraaliapproksimaation varianssista ja keskihajonnasta kaavan (2.9) mukaisesti. Voimme siis

vähentää virhettä kahdella tavalla: joko pienentämällä integraaliapproksimaation keskihajontaa tai käyttämällä enemmän mittauspisteitä. Painotettu otanta on tekniikka, jolla voidaan vähentää Monte Carlo -integraalin virhettä pienentämällä laskettavan integraaliapproksimaation varianssia. Tämä on hyödyllistä, koska meidän ei tarvitse lisätä mittauspisteiden määrää saavutaksemme tarkempia tuloksia. Mittauspisteiden lisääminen olisi lisännyt laskennan aikaa. [1]

Painotetussa otannassa integraalin (2.7) approksimaatio on: [1]

$$F = \int_a^b \frac{f(x)}{p(x)} p(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} \frac{f(x_i)}{p(x_i)}, \tag{2.13}$$

jossa $p(x)$ on todennäköisyysjakauma, joka toteuttaa todennäköisyysjakauman ehdot:

$$p(x) \geq 0, \tag{2.14}$$

$$\int_a^b p(x)dx = 1. \tag{2.15}$$

Nyt x_i otetaan todennäköisyysjakaumasta $p(x)$. Tarkoituksena on valita sellainen $p(x)$, joka minimoi termin $\frac{f(x)}{p(x)}$ varianssia. Yleensä $p(x)$ valitaan siten, että se muistuttaa $f(x)$.

Perustilan energian arvio

Yleistetään edellä saatu tulos meidän tapaukseemme. Eli meidän perustilan energianarviomme approksimaatio on:

$$E = \frac{\int \Psi^*(\mathbf{R}) H \Psi(\mathbf{R}) d\mathbf{R}}{\int \Psi^*(\mathbf{R})\Psi(\mathbf{R})d\mathbf{R}} = \frac{\int \Psi^2(\mathbf{R}) \frac{H\Psi(\mathbf{R})}{\Psi} d\mathbf{R}}{\int \Psi^*(\mathbf{R})\Psi(\mathbf{R})d\mathbf{R}} \approx \frac{1}{N}\sum_{i}^{N} E_L(\mathbf{R}_i), \tag{2.16}$$

jossa E on laskettu arvio perustilan energialle, Ψ^2 on aaltofunktion neliö ja myös todennäköisyystiheys hiukkasille, H on Hamiltonin operaattori, $E_L \equiv \frac{H\Psi}{\Psi}$ on paikallinen energia, $\mathbf{R} = \{\mathbf{r}_1, \mathbf{r}_2 ... \mathbf{r}_n\}$ on elektronikonfiguraatio muuttujana, jossa on kaikkien hiukkasten paikkakoordinaatit muuttujina, ja $\mathbf{R}_i$ on elektronikonfiguraatio, jossa on kaikkien hiukkasten paikkakoordinaatit lukuarvoina.

Nyt haluamme ottaa mittauspisteitä $\mathbf{R}_i$ jakaumasta Ψ^2. Yleisesti aaltofunktiossa Ψ on normalisointivakio, joka vaikeuttaa otannan tekemistä, koska emme yleensä tunne sitä. Me esitämme Metropolis-Hastings -algoritmin, jossa tiheysjakauman vakiolla ei ole vaikutusta otannan tekemiseen ja helpottaa siten otannan tekemistä. [1]

Metropolis-Hastings algoritmi

Olettakaamme, että haluamme tehdä siirtoa vain yhdelle hiukkaselle. Jotta siirto olisi Metropolis-Hastings algoritmin mukainen, hiukkasen paikkakoordinaateille suoritetaan seuraavat vaiheet: [1]

1. Valitse yritepaikkakoordinaatit hiukkaselle $\mathbf{r}_{trial} = \mathbf{r}_i + \boldsymbol{\delta}_i$, jossa $\boldsymbol{\delta}_i$ on 3-komponenttinen vektori, jonka jokainen komponentti on tasaisesta jakaumasta saatu satunnaisluku väliltä $[-\delta, \delta]$. Yleensä $\delta \sim 0.5$.

2. Laske $w = p(\mathbf{r}_{trial})/p(\mathbf{r}_i)$, jossa $p \equiv \Psi^2$ on tiheysjakauma ja aaltofunktion neliö.

3. (a) Jos $w \geq 1$, hyväksy yritepaikka: $\mathbf{r}_{i+1} = \mathbf{r}_{trial}$.

   (b) Jos $w < 1$, luo satunnaisluku r tasaisesta jakaumasta väliltä $[0, 1]$.

      i. Jos $r \leq w$, hyväksy yritepaikka: $\mathbf{r}_{i+1} = \mathbf{r}_{trial}$.

4. Jos yritepaikka $\mathbf{r}_{trial}$ ei ole hyväksytty, niin aseta $\mathbf{r}_{i+1} = \mathbf{r}_i$.

   Tämän algoritmin avulla, kun käydään koko elektronikonfiguraatioavaruuden läpi tarpeeksi monta kertaa, saadaan elektronien paikkakoordinaatit jakautumaan Ψ^2 mukaisesti, mikä mahdollistaa kaavan (2.16) käyttöä perustilan energian arvioimiseksi. Tätä prosessia kutsutaan *termalisaatioksi*.

Huomaa, että Metropolis-ehdotusalgoritmia tehdään kukin hiukkaselle erikseen, eli esim. heliumatomille $\mathbf{r} = (x_1, y_1, z_1) \neq \mathbf{R} = (x_1, y_1, z_1, x_2, y_2, z_2)$.

2.5 Yriteaaltofunktio

2.5.1 Edellytykset sopivalle yriteaaltofunktiolle

Tarkastellaan heliumatomia esimerkkinä. Yksinkertaisimmassa tapauksessa heliumatomin aaltofunktio voidaan approksimoida vedyn perustilan aaltofunktion ψ_{100} avulla näin:

$$\Psi = \psi_{100}(\mathbf{r}_1)\psi_{100}(\mathbf{r}_2), \tag{2.17}$$

jossa $\psi_{100} \equiv (\frac{Z^3}{\pi a_0^3})^{1/2} e^{-Zr/a_0}$, ja jossa Z on optimoitava parametri. Tämä approksimaatio ei kuitenkaan päde enää, kun approksimoidaan atomeja, joissa on enemmän elektroneita.

Tarkastellaan vähimmäisedellytykset yriteaaltofunktiolle, joka pätee kaikille atomille. Vähimmäisedellytykset ovat seuraavat:

1. Antisymmetrisyys: $\Psi(\mathbf{r}_1, \mathbf{r}_2) = -\Psi(\mathbf{r}_2, \mathbf{r}_1)$.

2. Tarpeeksi optimoitavia parametrejä.

3. Ottaa huomioon tärkeät vuorovaikutukset.

4. Ottaa huomioon käännepiste-ehdot (CUSP conditions).

1. edellytys on peräisin Paulin kieltosäännöstä fermioneille (tässä tapauksessa elektroneille), jonka mukaan fermionit eivät voi sijaita samassa paikassa, ja kun systeemin fermionit vaihtavat paikkojaan, miinus-merkki tulee aaltofunktion eteen. 2. edellytys ei anna meille mitään rajoitteita parametrien määrälle, mutta vaatii, että yriteaaltofunktiossa on oltava vähintäänkin yksi parametri. Erilaiset mallit käyttävätkin eri määrä parametreja. 3. ja 4. kohta liittyvät toisiinsa ja siitä puhutaan myöhemmin, kun ollaan esitetty yriteaaltofunktion muoto.

2.5.2 Yriteaaltofunktio: Slater-Jastrow muoto

Yriteaaltofunktio, joka täyttää edellä mainitut ehdot, on niin sanottu Slater-Jastrow-yriteaaltofunktio: [1]

$$\Psi(\mathbf{X}) = e^{J(\mathbf{X})} D(\mathbf{X}), \tag{2.18}$$

jossa $\mathbf{X} \equiv (\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_N)$ on lista kunkin hiukkasen (elektronin) spin-paikkakoordinaatit. Esim. jos hiukkasen 1 spin on $\uparrow$, sen spin-paikkakoordinaatti on $\mathbf{x}_1 = \{\mathbf{r}_1, \sigma_1\} = \{(x_1, y_1, z_1), \uparrow\}$. $D(\mathbf{X})$ on Slater-determinantti, joka ottaa huomioon aaltofunktion antisymmetrisyyden, ja jossa on atomin kunkin orbitaalin aaltofunktion approksimaatiot. Jos atomilla on N määrä elektronia se olisi

$$D(\mathbf{X}) = \begin{vmatrix} \psi_1(\mathbf{x}_1) & \psi_1(\mathbf{x}_2) & \dots & \psi_1(\mathbf{x}_N) \\ \psi_2(\mathbf{x}_1) & \psi_2(\mathbf{x}_2) & \dots & \psi_2(\mathbf{x}_N) \\ \vdots & \vdots & \vdots & \vdots \\ \psi_N(\mathbf{x}_1) & \psi_N(\mathbf{x}_2) & \dots & \psi_N(\mathbf{x}_N) \end{vmatrix}, \tag{2.19}$$

jossa ψ_i on nyt parametrisoitu aaltofunktion approksimaatio orbitaalille i ja $J(\mathbf{X})$ on Jastrow-tekijä, joka ottaa huomioon elektroni-elektroni ja elektroni-ydin vuorovaikutukset, ja joka on seuraavanlainen:

$$J(\mathbf{X}) = \sum_{i=1}^{N} \chi(\mathbf{x}_i) - \frac{1}{2} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} u(\mathbf{x}_i, \mathbf{x}_j) \tag{2.20}$$

Me emme käytä Slater-Jastrow-aaltofunktiota sellaisenaan (2.18), vaan jätämme spin-riippuvuudet huomioimatta. Kirjallisuuden mukaan spin-riippumattomien operaattoreiden odotusarvon laskemisessa – kuten Hamiltonin operaattori – spin-koordinaatti ei vaikuta tulokseen [1]. Eli spin-paikkakoordinaatista tulee $\mathbf{x}_i = \mathbf{r}_i$, jossa $\mathbf{r}_i = (x_i, y_i, z_i)$ on vain i:s elektronin paikkakoordinaatit. Slater-Jastrow-yriteaaltofunktiosta tulee seuraavanlainen: [1]

$$\Psi(\mathbf{X}) = e^{J(\mathbf{X})} D^{\uparrow}(\mathbf{r}_1, \mathbf{r}_2 ..., \mathbf{r}_{N_\uparrow}) D^{\downarrow}(\mathbf{r}_{N_{\uparrow+1}}, ..., \mathbf{r}_N), \tag{2.21}$$

jossa $D^{\uparrow}$ on Slater-determinantti, joka sisältää kaikki spin-ylös elektronit. $D^{\downarrow}$ on Slater-determinantti, joka sisältää kaikki spin-alas elektronit.

Nyt tarkennamme Jastrow-tekijän muoto. Ensinnäkin χ, joka ottaa huomioon elektroni-ydin korrelaatiot, jätetään pois Jastrow-tekijästä (2.20), sillä me oletamme tasaista ja avaruudellisesti homogeenistä elektronikaasupilveä joka ajanhetkellä (eli elektronit eivät ole diskreettejä yksittäisiä pisteitä missäkään hetkessä). [1]. Termin χ ottaminen huomioon lisäisi aaltofunktion variaatiovapautta, ja siten antaisi meille pienempiä energian arvioita, mutta se lisäisi kuitenkin joitakin ratkaistavia ehtoja ja Jastrow-tekijän lausekkeen pituutta. Siksi yksinkertaisuuden vuoksi jätämme χ huomiotta edellisen argumentin nojalla. * Ja nyt funktio u, joka ottaa huomioon elektroni-elektroni korrelaatiot, on:

$$u_{\sigma_i,\sigma_j}(\mathbf{r}_i, \mathbf{r}_j) \equiv \frac{A_{\sigma_i,\sigma_j}}{r_{ij}}(1 - e^{-r_{ij}/F_{\sigma_i,\sigma_j}}), \tag{2.22}$$

jossa σ_i on hiukkasen i spin-arvo (joko ylös ($\uparrow$) tai alas $\downarrow$), $A_{\sigma_i,\sigma_j}, F_{\sigma_i,\sigma_j}$ ovat käännepiste-parametrit, ϕ_i on aaltofunktio-approksimaatio i:s orbitaalille ja $r_{ij} \equiv |\mathbf{r}_i - \mathbf{r}_j|$ on hiukkasten i ja j etäisyys toisistaan.

Käyttämämme Jastrow-tekijä saa siis seuraavan muodon:

$$J(\mathbf{X}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N} u(\mathbf{x}_i, \mathbf{x}_j) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N} \frac{A_{\sigma_i,\sigma_j}}{r_{ij}}(1 - e^{-r_{ij}/F_{\sigma_i,\sigma_j}}). \tag{2.23}$$

2.5.3 Jastrow-tekijä ja käännepiste-ehdot: johdanto

Matemaattisesti käännepisteet ovat funktion kohtia, joissa funktio kääntyy jyrkästi. Esimerkkinä olisi $f(x) = x^{2/3}$, jolla on yksi käännepiste pisteessä $x = 0$ (ks. kuva 2.3). Tässä käännepisteessä ensimmäinen derivaatta ei ole määritetty:

$$\frac{df(x)}{dx} = \frac{2}{3}\frac{1}{x^{1/3}}, \tag{2.24}$$

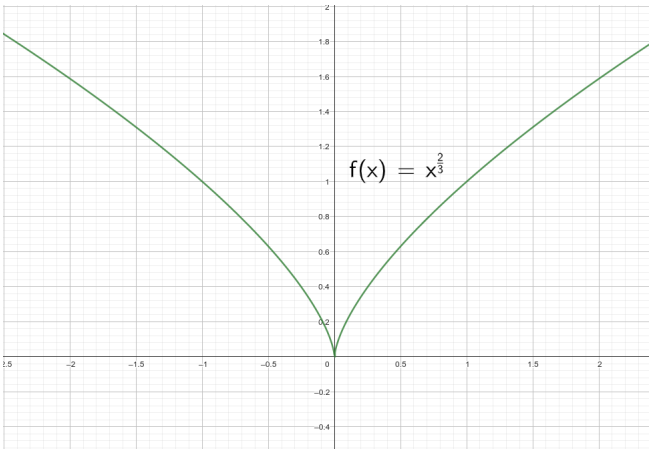
jossa derivaatta lähestyy äärettömyyteen, kun $x \to 0$. Käännepisteet ovat siis eräänlaisia matemaattisia singulariteetteja. Käännepisteet esiintyvät myös fysikaalisten systeemien aaltofunktiossa. Seuraavaksi havainnollistamme yksinkertaisimmalla atomilla eli vedyllä, miten käännepisteet esiintyvät atomisysteemissä.

Käytetään Hartree-yksikköjä (3.1). Vedyn normalisoitu perustilan aaltofunktio ja Hamiltonin operaattori ovat seuraavanlaisia:

$$\psi = \frac{e^{-r}}{\sqrt{\pi}}, \tag{2.25}$$

$$H = -\frac{1}{2}\nabla^2 - \frac{1}{r}, \tag{2.26}$$

*Halutessa lukija voi helposti lisätä χ-termi muokkaamalla Mathematica-koodin hieman.



Kuva 2.3: Funktio, jolla on yksi käännepiste kohdassa $x = 0$

jossa Hamiltonin operaattorin 1. termi on kineettinen energia ja 2. termi potentiaalienergia.

Nyt laskun mukaan Shrödingerin yhtälöksi saadaan:

$$H\psi = \frac{1}{2}\left(\frac{2e^{-r}}{\sqrt{\pi}r} - \frac{e^{-r}}{\sqrt{\pi}}\right) - \frac{e^{-r}}{\sqrt{\pi}r} = E\psi, \tag{2.27}$$

jossa E on energian ominaisarvo.

Tarkastellaan molempia termejä erikseen, kun $r \to 0$:

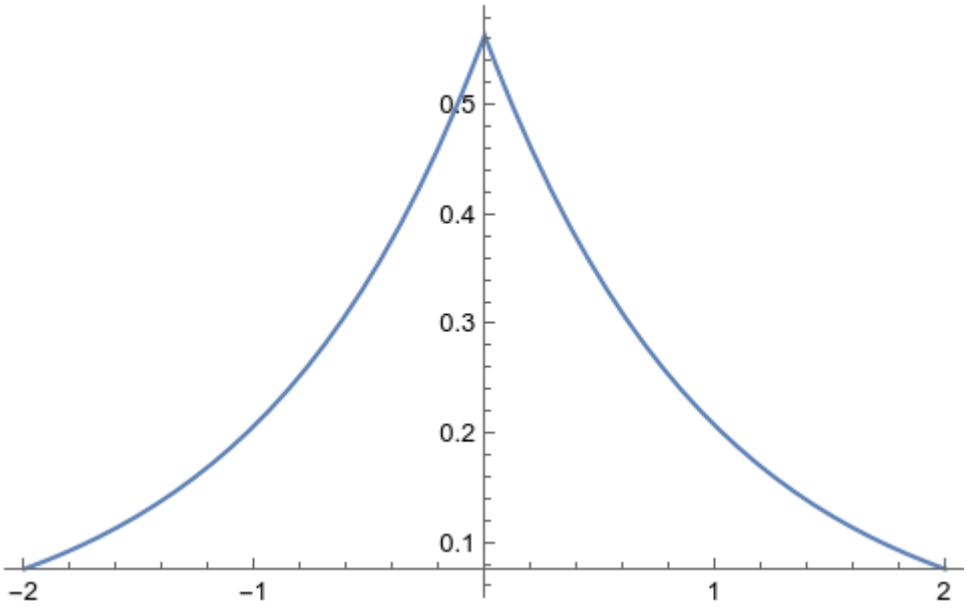
$$\lim_{r\to 0} \frac{1}{2}\left(\frac{2e^{-r}}{\sqrt{\pi}r} - \frac{e^{-r}}{\sqrt{\pi}}\right) = \infty, \tag{2.28}$$

$$\lim_{r\to 0} -\frac{e^{-r}}{\sqrt{\pi}r} = -\infty, \tag{2.29}$$

ja siten lausekkeesta (2.27) tulee ei-määrätty. Siis mitä näytimme, on se, että kun lähestytään koordinaatin origoa (jossa atomiydin sijaitsee), potentiaalienergia divergoi negatiiviseen äärettömyyteen. Mutta kineettisen energian divergenssi positiiviseen äärettömyyteen tasapanottaa potentiaalienergian divergenssiä. $H\psi$ on siis nyt äärellinen. Käännepiste nähdään myös vedyn perustilan aaltofunktiossa pisteeessä $r = 0$ (kuva 2.4).

Edellä tarkasteltiin vetyatomia, mutta tapaus voidaan yleistää muihin atomeihin, joilla on monia elektroneita. [1] Nämä atomien käännepisteet lisäävät yriteaaltofunktion energian arvion suuruutta ja sen varianssia [1], mitä ei haluta. * Siksi yriteaaltofunktiossa meidän on huomioitava käännepisteet keksimällä siihen käännepiste-ehdot, jotka tunnetusti pienentävät energian arvoa. [1]

*Meidän algoritmi, joka perustuu kvanttivariaatioperiaatteeseen, käyttää minimienergiaperiaatetta. Eli mitä pienempi energia, sitä parempi ja lähempi todellista perustilan energiaa. Samalla energian varianssi lähestyy nollaa, kun energian arvo pienenee.



Kuva 2.4: Vedyn aaltofunktiolla on käännepiste kohdassa $r = 0$

2.5.4 Jastrow-tekijä ja käännepiste-ehdot: käännepiste-ehdot ja parametrit

Oletetaan atomi ei ole spin-polarisoitunut. Meillä on neljä parametriä jokaiselle tarkastavalle elektroniparille: [1]

$$\begin{aligned} A_{\uparrow\uparrow} &= A_{\downarrow\downarrow}, \\ A_{\uparrow\downarrow} &= A_{\uparrow\downarrow}, \\ F_{\uparrow\downarrow} &= F_{\uparrow\downarrow}, \\ F_{\uparrow\uparrow} &= F_{\downarrow\downarrow}, \end{aligned}$$

joista kaksi parametria voidaan ratkaista käännepiste-ehtojen avulla:

$$\frac{du_{\sigma_i,\sigma_j}(r_{ij})}{dr_{ij}}\Big|_{r_{ij}=0} = \begin{cases} -1/4,\ \sigma_i = \sigma_j \\ -1/2,\ \sigma_i \neq \sigma_j \end{cases} . \tag{2.30}$$

Laskelmieni mukaan, saadaan:

$$F = \sqrt{2A},\ \sigma_i = \sigma_j, \tag{2.31}$$

$$F = \sqrt{A},\ \sigma_i \neq \sigma_j. \tag{2.32}$$

Johdettu tulos kertoo meille, että systeemin aaltofunktion Jastrow-tekijässä on enimmillään kaksi parametria. *

*Atomin aaltofunktiossa on kaksi parametria, jos sillä on enemmän kuin kolme elektronia.

2.6 Gradient descent: parametrien optimointi

2.6.1 Perustiedot algoritmista

Miten energian minimi löydetään? Muistakaamme, että olemme parametrisoineet aaltofunktiomme (ja siten myös paikallisen energian). Yritämme etsiä sellaisen parametriyhdistelmän, joka antaa energianminimin atomisysteemille. Ongelma on siis funktion optimointi-/minimointiongelma ja tutkielmassa käytämme yksi yksinkertaisimpia funktion minimointimenetelmiä, gradient descent -menetelmää, optimaalisin parametriyhdistelmän löytämisessä. Gradient descent -kaava on seuraavanlainen:

$$\boldsymbol{\alpha}_{i+1} = \boldsymbol{\alpha}_i - \eta \nabla_{\boldsymbol{\alpha}} E|_{\boldsymbol{\alpha}=\boldsymbol{\alpha_i}}, \tag{2.33}$$

jossa $\boldsymbol{\alpha}_i = \{\alpha_i, \beta_i, \gamma_i ...\}$ on parametrivektori, joka koostuu aaltofunktiossa esiintyvistä parametreistä, $\boldsymbol{\alpha}_{i+1}$ on seuraava iteraatio parametrivektorista, η on (eng. learning rate) parametri, joka säätää askeleen suuruus ja E on energia funktio.

Gradient descent toimii siis näin:

1. Alkuarvaus parametreille. Voimme olettaa, että kaikki parametrit ovat positiivisia. *

2. Laske seuraava askel kaavan mukaisesti.

3. Toista edellistä vaihetta, kunnes

$$\nabla_{\boldsymbol{\alpha}} E|_{\boldsymbol{\alpha}=\boldsymbol{\alpha_i}} = 0.$$

Menetelmän yksinkertaisuudesta johtuen se sisältää muutamia puutteita, joita ovat mm. seuraavat :

1. Saattaa konvergoida lokaaliin minimiin, eikä haluamaamme globaaliin minimiin.

   - Voimme olettaa, että yksinkertaisimmissa systeemeissä – kuten atomit ovat verrattuna molekyyleihin tai ioniyhdisteisiin – on vain yksi minimi.

2. Learning rate -parametrin η on säädettävä.

   - Kun ollaan kaukana minimistä, η-parametrin on hyvä olla suuri, jotta konvergoidaan nopeasti minimipisteeseen.
   - Kun lähestytään minimiä, η-parametri on säädettävä pienemmäksi.

*Negatiiviset tai nolla saattavat tuottaa epäfysikaalisia tuloksia. Katso esimerkiksi, mitä tapahtuisi, jos heliumin yksinkertaisin yritefunktion $e^{-\alpha r_1} e^{-\alpha r_2}$ parametri olisi $\alpha \leq 0$.

3. On käytettävä turhan paljon askeleita.

    - Atomien, joilla on enemmän elektroneja, aaltofunktio ja paikallinen energia ovat pitkiä, minkä takia niiden arvioiminen on laskennallisesti kallista. Tämän takia myös energian gradientin laskeminen ja vain yhden askeleen eteneminen vie paljon aikaa.

2.6.2 Gradientin laskeminen numeerisesti

Gradient descent -menetelmässä haluamme laskea gradientin $\nabla_{\boldsymbol{\alpha}} E$. Gradientti on:

$$\nabla_{\boldsymbol{\alpha}} E = \frac{\partial E}{\partial \alpha}\hat{\alpha} + \frac{\partial E}{\partial \beta}\hat{\beta} + \frac{\partial E}{\partial \gamma}\hat{\gamma}... = (\frac{\partial E}{\partial \alpha}, \frac{\partial E}{\partial \beta}, \frac{\partial E}{\partial \gamma}),$$

jossa $\boldsymbol{\alpha} = (\alpha, \beta, \gamma...)$ on parametrivektori ja esimerkiksi $\hat{\alpha}$ on parametrin α yksikkövektori (muut parametrit samalla tavalla).

Osittaisderivaattaa ei pystytä laskemaan algebrallisesti Fortran-ohjelmointikielellä ja käytämme differenssimenetelmää derivaattojen approksimoimisessa. Erityisesti käytämme kahden pisteen tai viiden pisteen differenssimenetelmää. Esimerkki osittaisderivaatasta ensimmäisen parametrin α suhteen: [4]

- kahden pisteen differenssimenetelmä:

$$\frac{\partial E}{\partial \alpha} \approx \frac{E(\alpha + h, \beta, \gamma...) - E(\alpha - h, \beta, \gamma...)}{2h},$$

- viiden pisteen differenssimenetelmä:

$$\frac{\partial E}{\partial \alpha} \approx \frac{E(\alpha - 2h, \beta, \gamma...) - 8E(\alpha - h, \beta, \gamma...) + 8E(\alpha + h, \beta, \gamma...) - E(\alpha + 2h, \beta, \gamma...)}{12h},$$

jossa h on askelen pituus derivaattoja laskiessa. Yleensä $h \sim 0.0001$, mutta säädämme tarvittaessa ohjelman ajaessamme, jos ei saavuta minimiä tällä arvolla.

3. Aineisto ja menetelmät

3.1 Fysikaaliset yksiköt

Tässä tutkielmassa käytämme Hartree-atomiyksiköitä, missä alkeisvaraus, redusoitu Planckin vakio, elektronin massa ja Coulombin vakio asetetaan ykköseksi:

$$e \equiv 1, \tag{3.1}$$

$$\hbar \equiv 1, \tag{3.2}$$

$$m_e \equiv 1, \tag{3.3}$$

$$k_e \equiv 1. \tag{3.4}$$

3.2 Algoritmi

Seuraavaksi esitetään algoritmi, joka on kehitetty ohjaajani [†] kanssa.

Algoritmi on seuraavanlainen:

1. Paikallinen energia ja aaltofunktio [‡] saadaan Mathematica-koodilla.

   - Aaltofunktio sisältää parametreja, joita myöhemmin optimoidaan löytääksemme energian minimi (ks. 2.2 miksi energian minimi). Paikallinen energia on johdettu aaltofunktiosta, ja siksi se sisältää myös samat parametrit.
   - Esimerkiksi heliumin yksinkertaisin approksimatiivinen aaltofunktio, jossa käytetään vedyn perustilan elektronin aaltofunktiot yksittäisten heliumin elektronien aaltofunktiona, on:

     $$\Psi = \psi_{100}(\mathbf{r}_1)\psi_{100}(\mathbf{r}_2) = e^{-\alpha r_1}e^{-\alpha r_2},$$

     jossa r_1, r_2 ovat elektronin 1 ja 2 paikkakoordinaatit [§], α on optimoitava parametri ja ψ_{100} on vedyn elektronin perustilan aaltofunktio.

[†] Dosentti Ilja Makkonen

[‡] tarkalleen tarvitsemme aaltofunktion neliön

[§] tarkalleen paikkakoordinaattien absoluuttinen arvo origonkeskisessä koordinaatissa.

2. Python-ohjelmointikielellä tehdyllä ohjelmalla käännetään edellä saadut lausekkeet siten, että lausekkeet voidaan arvioida Fortran-ohjelmassa.

3. Parametrien optimointi: etsitään parametriyhdistelmä, joka minimoi energian arvion.

   (a) Alkuarvaus parametreille: $\boldsymbol{\alpha}_i = \{\alpha_i, \beta_i, \gamma_i ...\}$.

   (b) Termalisaatio: Saadaan hiukkasten konfiguraatiot asettumaan aaltofunktion neliön mukaiseksi (ks. 2.4.2).

   (c) Gradient descent -menetelmällä etsitään minimipisteet: (ks. 2.6)

$$\boldsymbol{\alpha}_{i+1} = \boldsymbol{\alpha}_i - \eta \nabla_{\boldsymbol{\alpha}} E|_{\boldsymbol{\alpha}=\boldsymbol{\alpha_i}}. \tag{3.5}$$

   - Katkaistaan optimointi, kun $\nabla_{\boldsymbol{\alpha}} E|_{\boldsymbol{\alpha}=\boldsymbol{\alpha_i}} \sim 0$.
   - Tallennetaan optimaaliset parametrit: $\boldsymbol{\alpha}_{optimal}$.

4. Energian lasku optimoiduilla parametreillä:

   (a) Kohdassa 3 saatiin optimoidut parametrit $\boldsymbol{\alpha}_{optimal}$.

   (b) Suoritamme rinnakkaislaskentaa: * Luodaan $M \sim 10$ toisistaan riippumatonta elektronikonfiguraatioavaruutta (ks. 2.1.2). Elektronikonfiguraation elementit ovat satunnaislukuja tasaisesta jakaumasta jostakin lukuvälistä † .Jokaiselle konfiguraatioavaruudelle suoritamme seuraavat vaiheet:

      i. Termalisaatio:
         - Tehdään $\sim 10^3$ kertaa Metropolis-ehdotusta koko elektronikonfiguraatioavaruuden läpi.

      ii. Paikallisen energian lasku jokaiselle elektronikonfiguraatiolle R_i.

         A. Otetaan esimerkiksi heliumatomi, jonka elektronikonfiguraatio yhdellä hetkellä on $R = (x_1, y_1, z_1, x_2, y_2, z_2)$, jossa alaindeksit viittaa elektroniin 1 ja 2.

         B. Tee Metropolis-ehdotusta yhdelle hiukkaselle. Voimme tehdä Metropolis-ehdotusta vuorotellen hiukkaselle:
            - Tehdään Metropolis-ehdotusta 1. elektronille tällä elektronikonfiguraatiolla R_i, seuraavalla vuorolla elektronikonfiguraatiolla R_{i+1} tehdään 2. elektronille, sitä seuraavalla vuorolla R_{i+2} taas 1. elektronille jne.

*Dosentti Ilja Makkosen ehdottama ratkaisu autokorrelaation ongelmalle. Autokorrelaatioon tutustaan myöhemmin.

†Lukuväli voi olla mikä vain, koska termalisaation avulla elektronikonfiguraatiot asettuvat kuitenkin aaltofunktion neliön mukaiseksi ja todellinen lukuväli määräytyy lopulta tämän pohjalta.

C. Laske paikallinen energia tälle konfiguraatiolle: $E_L(R_i)$.

D. tallenna $E_L(R_i)$ listaan

iii. Kohdan 4(b)ii jälkeen saadaan joukko paikallisia energioita, jotka tallennetaan:

$$\{E_L(R_1), E_L(R_2), E_L(R_3), ...\}.$$

(c) Kohdan 4b jälkeen saadaan seuraavanlainen matriisi, jossa jokainen sarake edustaa kustakin riippumattomasta konfiguraatioavaruudesta saadut lokaalienergiat:

$$E = \begin{pmatrix} E_{L1}(R_1) & E_{L2}(R_1) & \cdots & E_{LM}(R_1) \\ E_{L1}(R_2) & E_{L2}(R_2) & \cdots & E_{LM}(R_2) \\ \vdots & \vdots & \cdots & \vdots \\ E_{L1}(R_n) & E_{L2}(R_n) & \cdots & E_{LM}(R_n) \end{pmatrix}, \tag{3.6}$$

jossa esim. E_{L1} on paikallinen energia ensimmäisestä konfiguraatioavaruudesta ja R_i on konfiguraatio i vastaavassa konfiguraatioavaruudessa.

i. Lasketaan matriisin kaikista paikallisesta energiasta (3.6) keskiarvo, varianssi ja keskiarvon keskivirhe.

- Varianssi:

$$varianssi = \langle E^2 \rangle - \langle E \rangle^2.$$

- Keskiarvon keskivirhe:

$$\sqrt{varianssi/Size(E)},$$

jossa $Size(E)$ on matriisin (3.6) alkioiden määrä.

3.3 Lisäselitykset algoritmista

3.3.1 Autokorrelaatio ja sen ratkaisu

Tämä alaluku liittyy algoritmin kohtaan 4b, jossa loimme noin kymmentä konfiguraatioavaruutta ja suoritimme rinnakkaislaskentaa. Aloitamme yleisesti, mikä on autokorrelaatio.

Autokorrelaatio viittaa korrelaatioon saman muuttujan arvoissa eri havainnoissa. Se voi esiintyä aikasarja- tai poikkileikkausaineistoissa. Tyypillinen esimerkki aikasarja-aineistosta on lämpötila-muuttuja, joka on yleensä autokorreloitunut. Kahden peräkkäisen päivän lämpötilan odotetaan todennäköisemmin olevan lähellä toisiaan ja korreloituneet toistensa kanssa kuin kahden kaukana toisistaan olevien päivien lämpötila.

Algoritmin kohdassa 4b käytimme rinnakkaislaskentaa autokorrelaation ongelman ratkaisuksi. Mutta missä voi syntyä autokorrelaatiota, jos emme käyttäisi rinnakkaistamista? Käytännössä autokorrelaatioita voi tapahtua silloin, kun käytämme eri elektronikonfiguraatiota jonkin funktion lukuarvon saamisessa. Funktion lukuarvot peräkkäisten elektronikonfiguraation välillä ei pitäisi olla mitään yhteyttä, mutta jos niissä on jotain korrelaatioita, se on ongelmallista, koska se voi vaikuttaa lopulliseen energianarvioon. Tällaista tilannetta voi esiintyä algoritmissa:

1. vaiheessa 4(b)i Metropolis-otannassa laskiessa $\psi^2(\mathbf{r}_{trial})/\psi^2(\mathbf{r}_i)$,
2. ja vaiheessa 4(b)ii paikallisen energian arvioimisessa.

Rinnakkaislaskennassa saadaan joukko energian arvioita eri riippumattomasta konfiguraatioavaruudesta, joista lasketaan keskiarvo, ja näin vähennetään autokorrelaation vaikutusta lopullisen energian arvoon.

Autokorrelaatiota voi tietysti myös esiintyä, jos satunnaisluvun generaattorit eivät tuota riippumattomia satunnaislukuja algoritmissamme eri vaiheissa. Satunnaisluvun generaattorin valinnasta puhutaan seuraavaksi.

3.3.2 Satunnaisluvun generoiminen

Tyypillinen satunnaislukugeneraattorin valinta Monte Carlo -simulaatioissa on The Mersenne Twister (lyh. MT)[5]. Erillisen satunnaislukugeneraattorin on tarpeen, sillä yleensä ohjelmointikielen sisäiset satunnaislukugeneraattorit eivät generoi tarpeeksi satunnaisia lukuja, eli ne tuottavat korreloivia satunnaislukuja. MT on hyvä, sillä se on laajasti käytetty eri MC-simulaatioissa ja on myös läpäissyt useat satunnaislukutestit. MT:stä on kirjoitettu ilmaisia ja lisenssi-vapaita koodeja eri ohjelmointikielellä. Meidän koodissamme käytämme Hiroshi Takanon kirjoittamaa Fortran-koodia [6].

3.4 Algoritmin toimivuuden testaaminen

Testaamme algoritmimme toimivuutta heliumatomilla. Seuraavat ovat käyttämämme yriteaaltofunktiot heliumatomille:

1. Dosentti Ilja Makkosen käyttämä yritefunktio heliumatomille:

$$e^{-\mathrm{A1}\sqrt{\mathrm{x1}^2+\mathrm{y1}^2+\mathrm{z1}^2}-\mathrm{A1}\sqrt{\mathrm{x2}^2+\mathrm{y2}^2+\mathrm{z2}^2}}\left(\mathrm{A2}\sqrt{|\mathrm{x1}-\mathrm{x2}|^2+|\mathrm{y1}-\mathrm{y2}|^2+|\mathrm{z1}-\mathrm{z2}|^2}+1\right), \tag{3.7}$$

   jossa $A1$ ja $A2$ ovat optimoitavat parametrit ja esim. $x1$ on hiukkasen 1 x-paikkakoordinaatti ja muut koordinaatit toimivat samalla periaatteella.

2. Slater-Jastrow yritefunktio heliumatomille:

$$\exp\left(-\frac{\mathrm{A2}\left(1-e^{-\frac{\sqrt{|\mathrm{x1}-\mathrm{x2}|^2+|\mathrm{y1}-\mathrm{y2}|^2+|\mathrm{z1}-\mathrm{z2}|^2}}{\sqrt{\mathrm{A2}}}}\right)}{\sqrt{|\mathrm{x1}-\mathrm{x2}|^2+|\mathrm{y1}-\mathrm{y2}|^2+|\mathrm{z1}-\mathrm{z2}|^2}}-\mathrm{A3}\sqrt{\mathrm{x1}^2+\mathrm{y1}^2+\mathrm{z1}^2}-\mathrm{A3}\sqrt{\mathrm{x2}^2+\mathrm{y2}^2+\mathrm{z2}^2}\right), \tag{3.8}$$

jossa $A2$ ja $A3$ ovat optimoitavat parametrit ja esim. $x1$ on hiukkasen 1 x-paikkakoordinaatti.

4. Tulokset

4.1 Ohjelman tulosteet

Näemme taulukosta 4.1 ohjelmamme tulokset.

Atomi	Helium, Makkonen (3.7)	Helium, Slater-Jastrow (3.8)
Energian arvio keskivirheineen	-2.893 ± 0.003	-2.898 ± 0.004
Varianssi	0.09	0.09
Kirjallisuusarvo [7]	$-2.90338583 \pm 0.00000013$	$-2.90338583 \pm 0.00000013$
Ero kirjallisuusarvoon	0.011	0.005
Optimoidut parametrit	2.0, 0.5	$3.599, 1.965$

Taulukko 4.1: Ohjelman tulosteet eri atomeille. Energian arviot ovat Hartree-yksiköillä. Ensimmäisen rivin viittaukset viittavat yritefunktioihin, joita käytimme.

.

Näemme, että Slater-Jastrow-yriteaaltofunktiolla saimme pienemmän energian arvion ja lähemmäs kirjallisuusarvoa verrattuna Makkosen käyttämän yriteaaltofunktiolla saatuun arvioon. Slater-Jastrow-yritteen tuloksen mukaan löydämme 95% todennäköisyydellä todellisen perustilan energian arvion välillä:

$$[\bar{x} - 1.96\frac{s}{\sqrt{n}}, \bar{x} + 1.96\frac{s}{\sqrt{n}}] = [-2.904, -2.891].$$

Makkosen yritteen tuloksen mukaan luottamusväli on $[-2.898, -2.888]$.

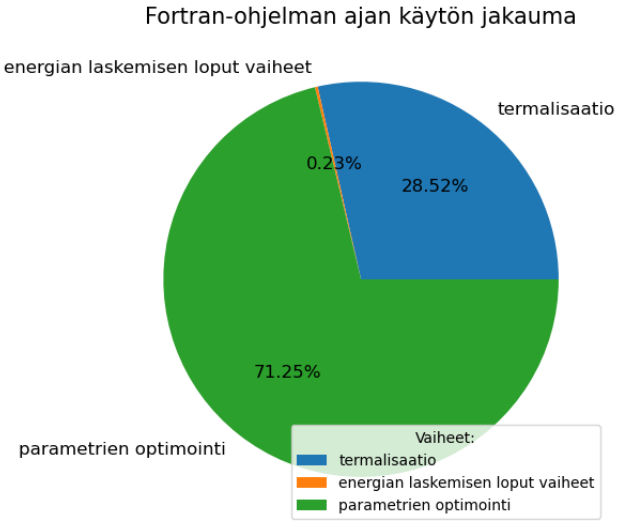
Tulokset ovat järkeviä, sillä Markkosen yrite on yksinkertaistettu versio Slater-Jastrow yritteestä ja siksi ei ole yhtä lähellä kirjallisuusarvoa kuin täydellisemmästä Slater-Jastrow yritteestä saatua energian arviota.

4.2 Ohjelman ajankäytön analyysi

Hyödynnämme mahdollisimman paljon Fortranin rinnakkaislaskennan ominaisuutta, ja kääntäessämme Fortran-ohjelma konekieleksi käytämme optimointiin liittyvää argumenttia `-O3`, mikä vähentää ohjelman varsinaisen ajon aikaa. Havainnollistamiseksi

analysoimme ajan käyttöä, kun käytämme Makkosen käyttämän yritefunktion ohjelmamme syötteenä.

Makkosen käyttämällä yritefunktiolla energian laskemiseen kului yhteensä noin 2088.17 minuuttia. Noin 99% energian laskemisen ajasta (eli termalisaatio ja energian laksemisen loput vaiheet) kului termalisaatioon. Parametrien optimointiin aikaa kului noin 5177 minuuttia, eli noin 2.5 kertaa enemmän aikaa kuin energian laskemiseen käytetty aika. Tähän aikaan sisältyy myös konfiguraatioavaruuden termalisaatio. Yhteensä koko Fortran-ohjelman ajo kului siis noin 120 tuntia. Mathematica-koodin ajamiseen kului aikaa enintään noin minuutista muutamaan minuuttiin.



Kuva 4.1: Fortran-ohjelman ajan käytön jakauma. ”Energian laskemisen loput vaiheet” ja ”termalisaatio” on osa energian laskemista.

5. Yhteenveto

Tässä tutkielmassa saimme toimivan ja helposti ymmärrettävän algoritmin, jonka pohjalta teimme ohjelman Fortranin ja Mathematica-ohjelman avulla. Ohjelmalla saimme heliumatomin perustilan energian arviot, jotka ovat lähellä kirjallisuusarvoa.

Ajankäytön analyysissa huomasimme, että ohjelman ajamiseen kului paljon aikaa jo heliumatomin tapauksessa. Voimme odottaa, että hankalampien atomien tapauksessa, ohjelman ajamiseen on käytettävä vähintään muutamia päiviä, mikä on yksi ohjelman suurimmista puutteista: joudumme arvioimaan liian pitkiä lausekkeita, erityisesti paikallisen energian arvioiminen vie paljon aikaa. Esimerkiksi Makkosen aaltofunktion neliön lausekkeessa on 121 merkkiä ja paikallisessa energiassa on 6524 merkkiä. Lausekkeiden arvioimiseksi käytimme Fortran-ohjelmassamme Jacopo Chevallardin FortranParser-ohjelmapakettia [8], jolla on rajoite, joka rajoittaa, kuinka pitkä arvioitava lauseke voi olla. Pisin mahdollinen lauseke on 2147483647-merkkinen [†]. Tämä rajoitus helposti ylittyy monimutkaisemmissa atomisysteemeissä paikallisen energian tai jopa yriteaaltofunktion lausekkeessa. Ensimmäisessä tapauksessa voimme aina pilkkoa paikallisen energian lauseketta osiin ja laskea lopulta osat yhteen, vaikka tämä ratkaisu on hieman aikaa vievää.

Parannettavaa ohjelmassa on, että emme käyttäisikään FortranParser-ohjelmapakettia lausekkeiden arvioimiseen, vaan käytämme perinteisempiä numeerisempia lausekkeiden arvioimismenetelmiä, joita mainitaan tässä [9]. Käytännössä tarvitsemme FortranParser-pakettia vain Metropolis-otannassa, jossa lasketaan aaltofunktioiden neliöiden $w = \Psi^2(\mathbf{r}_{trial})/\Psi^2(\mathbf{r}_i)$ suhdetta, ja paikallisen energian laskemisessa. Suhteen w laskemisessa voimme laskea pelkästään molempien Slater-determinanttien suhdetta, sillä aaltofunktioiden muut termit supistavat toistensa pois. Determinanttien suhteen laskemisessa on olemassa algoritmi, joka nopeuttaisi suhteen laskemista, ja joka on selitetty tässä [10]. Myös paikallisen energian laskemisessa on olemassa numeerinen menetelmä, joka nopeuttaa sen laskemista. [9] Edellä mainittujen parannuksien jälkeen, rajoitus lausekkeen pituudelle poistuu ja ohjelma on nopeampi.

Ohjelman testaamisesta ja luotettavuusarvion parantamista varten olisimme voi-

[†]joka on peräisin Fortran-ohjelmointikielen ominaisuudesta: kuinka suuri luku KIND(4) kokonaislukumuuttuja voi tallentaa

neet testata enemmänkin atomisysteemejä, mutta tutkielman aikarajoituksen takia, näin emme voineet tehdä.

Liite A. Tietokoneohjelma ja käytetyt ohjelmistot ja ohjelmapaketit

Linkki tutkielman tietokoneohjelmaan (Github)

Seuraavassa luettelossa luetellaan käytetyt ohjelmapaketit ja ohjelmointikielet:

Ohjelma/paketti	Lisenssi
Python-ohjelmointikieli [11]	PSF Licencse Agreement
Fortran-ohjelmointikieli [12]	Public domain
Mathematica-ohjelmisto [13]	Wolfram Mathematica® License Agreement
re-moduuli [14]	PSF Licencse Agreement
Jupyter lab -ohjelmisto [15]	BSD-3-Clause license
OpenMP-moduuli [16]	GNU General Public License
GNU Fortran compiler [17]	GNU General Public License
FortranParser-moduuli [16]	BSD 3-Clause ”New”
Mersenne Twister -moduuli [6]	GNU General Public License

Kirjallisuutta

[1] Harvey Gould, Jan Tobochnik, and Wolfgang Christian. An introduction to computer simulation methods applications to physical system, 2016.

[2] Leon van Dommelen. *Quantum Mechanics for Engineers.* Version 5.63 alpha edition, 2018.

[3] Adrian E. Feiguin. *Phys 5870: Modern Computational Methods in Solids.* 2009. URL: https://web.northeastern.edu/afeiguin/phys5870/phys5870/node9.html.

[4] Cameron R Taylor. Finite difference coefficients calculator, 2016. URL: https://web.media.mit.edu/~crtaylor/calculator.html.

[5] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, 1998.

[6] Hiroshi Takano. mt1997.f, 1 1999. URL: http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/VERSIONS/FORTRAN/fortran.html.

[7] Alexander Kramida, Yuri Ralchenko, Joseph Reader, and et al. Atomic spectra database, 2022. URL: http://physics.nist.gov/PhysRefData/ASD/ionEnergy.html.

[8] Jacopo Chevallard. Fortranparser, 2019. URL: https://github.com/jacopo-chevallard/FortranParser.

[9] W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal. Quantum monte carlo simulations of solids. *Reviews of Modern Physics*, 73, 1 2001.

[10] Anderson James B. *Quantum Monte Carlo : Origins, Development, Applications.* Oxford University Press, 2007. URL: https://search-ebscohost-com.libproxy.helsinki.fi/login.aspx?direct=true&db=e000xww&AN=209622&site=ehost-live&scope=site.

[11] Python Software Foundation. Python ohjelmointikieli, 2022. URL: http://www.python.org.

[12] Fortran lang Community. Fortran 95 ohjelmointikieli, 1997. URL: https://fortran-lang.org/.

[13] Wolfram Research. Wolfram mathematica, 2023. URL: https://www.wolfram.com/mathematica/.

[14] Python Software Foundation. re module, 2019.

[15] Project Jupyter. Jupyter lab, 2014. URL: https://jupyter.org/.

[16] OpenMP Architecture Review Board. Openmp, 2021. URL: openmp.org.

[17] Inc. Free Software Foundation. gfortran, 2021. URL: https://gcc.gnu.org/.