Kvanttivariaatiomenetelmä: numeerinen ratkaisu atomien perusenergioille Monte Carlolla

Naiyu Deng, 17.4.2023

Kvanttimekaniikan periaatteen mukaan aaltofunktio sisältää kaikki informaatiot systeemistä (esim. yksi hiukkanen tai hiukkasryhmä). Aaltofunktion analyyttinen muoto on usein mahdotonta selvittää monimutkaisille systeemeille, siksi käytetämme numeerista menetelmää selvitäksemme aaltofunktio ja tässä tutkimuksessa käytämme **kvanttivariaatiomenetelmää**. Tavoitteena on saada selville yksittäisten atomien perusenergia (ground state energy) ja sen kautta atomien aaltofunktiot.

Kvanttivariaatiomenetelmä perustuu **kvanttivariaatioperiaatteeseen** ja se toimii seuraavasti: Tehdään arvaus atomin aaltofunktiolle (kutsutaan **yriteaaltofunktioksi**), jonka pohjalta lasketaan arvio atomin perusenergialle kvanttimekaanisella kaavalla. Mistä tiedetään, että tämä arvaus aaltofunktiolle on oikein ja tämä laskettu perusenergian arvio on juuri atomin oikea perusenergia? Kvanttivariaatiomenetelmä kertoo meille, että mikä tahansa satunnainen aaltofunktio tuottaa aina suurempi perusenergian arvio kuin atomin todellisen aaltofunktion tuottama energia. Kvanttivariaatioperiaatteeseen perustuen saadaan seuraavanlainen yksinkertaistettu algorimi:

1. Tehdään alkuarvaus atomin aaltofunktiolle. Yriteaaltofunktio on parametrisoitu mahdollisimman joustavasti, jotta voidaan myöhemmin vaihtamalla parametrien arvoja saada energiaa, toivottavasti, yhä pienemmäksi.
2. Lasketaan perusenergian arvio yriteaaltofunktiolla.
3. Optimoidaan parametrit.
4. Suorita 2 – 3, kunnes energia ei pienene enää.
5. verrataan lopussa kirjallisuusarvoon.

Yriteaaltofunktion on täytettävä tiettyjä ehtoja, joiden kautta päädytään ns. **Slater-Jastrow yriteaaltofunktioon**.

Hamiltonin operaattorina käytetään **Born-Oppenheimer-approksimaatiota**, joka olettaa atomin ytimen olevan äärettömän raskas elektroneiden näkökulmasta ja siksi ytimen liikettä voidaan jättää huomioimatta.

Perusenergian arvioiminen on usein analyyttisesti mahdotonta, joten käytämme numeerista integraalimenetelmää: **Monte Carlo -integraalia**. Se perustuu ideaan, että generoidaan suuri määrä satunnaista datapistettä (elektroneiden paikkakoordinaatteja), joista lasketaan lokaalienergiaa kukin datapisteessä. Tämän jälkeen keskiarvoistetaan lokaalienergiaa. Tässä käytetään lisäksi **Metropolis-Hastings samplays -menetelmää**, jonka avulla voidaan valita sopivat satunnaispisteet, jotka mukautuvat aaltofunktion mukaisesti. Ideana on vähentää tarvittavien datapisteiden määrän, mutta silti saada riittävän tarkat integraaliapproksimaatiot.

Yriteaaltofunktiossa esiintyy parametrejä, joita meidän pitää optimoida. Tämä tarkoittaa sitä, että etsitään sellainen parametriarvojen yhdistelmä, joka minimoi perusenergiaa (, jota saadaan edellä mainitulla Monte Carlo -integraalilla). Tässä algoritmissa käytämme **gradient descent -menetelmää**, joka on yleisin optimointimenetelmä.

Meillä on kuitenkin seuraavat ongelmat: lokaalienergian laskemisen vaikeudet, satunnaisluvun generoiminen Metropolis-Hastings samplayksessä, autokorrelaatio ja toteutus Fortranilla vai Pythonilla.

Lokaalienergiaa lasketaan tässä suoraan Mathematicalla, vaikka se tuottaakin todella pitkän lausekkeen, joka hidastaa ohjelman ajoa ja vaikeuttaa algoritmin implementointia eri ohjelmointikielillä. Tietokoneen tai ohjelmointikielen sisältämät satunnaislukugeneraattorit eivät tuota todellisia satunnaislukuja, eli peräkkäiset satunnaisluvut riippuvat toisistaan jollain tavoin. Ratkaisuna käytämme The Mersenne Twister, klassinen valinta Monte Carlo -simulaatioihin. Autokorrelaatio on korrelaatio kahden peräkkäisen mittauspistejoukkojen korrelaatiota kahden peräkkäisen ajanjaksolla. Tätä korrelaatiota ei haluta ja siksi lopullisessa algoritmissa toteutetaan vielä algoritmin 2. vaiheen rinnakkaistaminen: kopioidaan 10–20 riippumatonta konfiguraatioavaruutta, jota kutsutaan yksittäisesti konfiguraatio-osa-avaruudeksi, jossa on siis hiukkasten satunnaiset paikat. Suoritetaan riippumattomasti Monte Carlo – Metropolis-Hastings integraalia kullekin konfiguraatio-osa-avaruudelle ja lopussa yhdistetään ja keskiarvoistetaan tulokset.

Alustavasti, saimme Pythonin toteutuksella Heliumille melko hyvän tuloksen -2.77 (verrattuna kirjallisuuteen -2.90). Parannettavaa olisi mm. käyttää Fortrania, sillä se toimii nopeammin ja tukee hyvin rinnakkaislaskentaa, jota tarvitaan meidän autokorrelaation ratkaisussamme.