Osmo Parviainen

Aikariippuvan Diracin yhtälön numeerisesta ratkaisemisesta

Tietotekniikan kandidaatintutkielma

31. tammikuuta 2017

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Osmo Parviainen

Yhteystiedot: oseeparv@student.jyu.fi

Työn nimi: Aikariippuvan Diracin yhtälön numeerisesta ratkaisemisesta

Title in English: On the numerical solution of the time-dependent Dirac equation

Työ: Kandidaatintutkielma

Sivumäärä: 25+0

Tiivistelmä: Tässä opinnäytetyössä tarkastellaan Diracin yhtälön numeerista ratkaisemista sekä tähän liittyviä haasteita. Diracin yhtälö on keskeinen kvanttifysiikan ilmiöitä kuvaava yhtälö, jonka ratkaisemiseen on esitetty kirjallisuudessa useita erilaisia menetelmiä. Tässä opinnäytetyössä keskitytään tarkastelemaan menetelmää, jossa yhtälön ratkaisemiseen sovelletaan operaattorinjakoa ja karakteristista menetelmää. Yhtälön ratkaiseminen vaatii paljon aikaa ja laskentaresursseja, joten laskennan tehostaminen rinnakkaislaskennan keinoin on välttämätöntä.

Avainsanat: Diracin yhtälö, aikariippuva, rinnakkaislaskenta

Abstract: In this paper the numerical solution of the Dirac equation is studied along with its challenges. The Dirac equation is a central equation describing the phenomenons of the quantum physics and there are plenty of different methods suggested in the literature on solving the equation. This paper concentrates on solving the Dirac equation with a combination of operator splitting and the method of characteristics. Solving the equation requires a lot of time and computational resources and hence it is necessary to enhance the computation by means of parallel computing.

Keywords: Dirac equation, time-dependent, parallel computing

Sisältö

1 Johdanto

Tavanomaisesti tietokoneissa on ollut vain yksi prosessori, jolla laskutoimituksia on laskettu peräkkäin. Tietokoneen nopeuden kasvattaminen toteutettiin kasvattamalla tietokoneen kellotaajuutta, minkä jatkuva transistoreiden lukumäärän kasvu on mahdollistanut. Mooren laki on ennustanut, että noin kerran vuodessa tietokoneeseen kustannustehokkaasti toteutettavien transistoreiden lukumäärä kasvaa aina kaksinkertaiseksi. Kuitenkin nykyään kellotaajuuden kasvattamisella saavutettava nopeuden kasvu on saavuttanut rajansa, joten transistoreiden lukumäärää on alettu kohdistamaan prosessorien lukumäärän kasvattamiseen. (Bauke ja Keitel 2011.) Usean prosessorin hyödyntäminen paremman laskutehon saavuttamiseksi edellyttää rinnakkaislaskennan toteuttamista.

Rinnakkaislaskennassa useampi prosessori laskee laskutoimituksia samanaikaisesti. Tämä voidaan toteuttaa keskusprosessoreilla (engl. *Central Processing Unit, CPU*) tai grafiikkaprosessorilla (engl. *Graphics Processing Unit, GPU*). Ongelmia, joissa rinnakkaislaskennan hyödyntäminen on käytännössä välttämätöntä, tulee vastaan useilla eri tieteenaloilla, kuten fysiikassa, taloustieteessä ja kemiassa. Rinnakkaislaskentasovelluksia on kehitetty raskaan laskennallisen työn helpottamiseksi esimerkiksi graafien värjäämiseen (Lu ym. 2016), geenien (Le ym. 2016) ja atomien mallintamiseen ja simulointiin (Nakano ym. 2001) sekä kolmiulotteisten kudos- ja eliömallien tarkasteluun (Mishra ym. 2006). Lisäksi rinnakkaislaskenta on erittäin hyödyllistä esimerkiksi käänteisongelmien (Akcelik, Biros ja Ghattas 2002) sekä optimointisovellusten (Migdalas, Toraldo ja Kumar 2003) tapauksessa, joissa samantyyppisiä laskutoimituksia joudutaan toistamaan useita kertoja peräkkäin.

Tässä opinnäytetyössä tarkastellaan Diracin yhtälön numeerista ratkaisemista, sekä tähän liittyviä ongelmia. Diracin yhtälö on keskeinen kvanttifysiikan ilmiöitä kuvaava aikariippuva osittaisdifferentiaaliyhtälö, jonka ratkaiseminen vaatii paljon laskentaresursseja. Yhtälön tehokas käsittely edellyttää rinnakkaislaskennan hyödyntämistä numeerisen ratkaisumenetelmän tukena. Tässä opinnäytetyössä keskitytään läheisemmin yksittäiseen Diracin yhtälön numeerisen laskennan menetel-

mään, joka on esitetty artikkelissa (Fillion-Gourdeau, Lorin ja Bandrauk 2012), sillä kirjallisuudessa on esitetty hyvin paljon erilaisia ratkaisumenetelmiä Diracin yhtälölle.

Tässä opinnäytetyössä käsitellään aluksi luvussa 2 rinnakkaislaskentaa yleisellä tasolla. Luvussa 3 Diracin yhtälö esitellään ensin lyhyesti, jonka jälkeen luvussa 3.1 tarkastellaan Diracin yhtälön numeerista ratkaisemista sekä diskretointia ajan ja paikan suhteen, sekä joitakin yhtälön numeeriseen tarkasteluun liittyviä haasteita, kuten fermionien kahdentumista. Lopuksi luvussa 3.2 käsitellään rinnakkaistamisen toteuttamista Diracin yhtälön numeerisen ratkaisemisen menetelmälle.

2 Rinnakkaislaskenta

Rinnakkaislaskenta on tietokonelaskennan menetelmä, jossa suurempi perusongelma jaetaan pienemmiksi osaongelmiksi. Nämä osaongelmat jaetaan sitten usealle eri prosessorille, jotka kykenevät suorittamaan ne samanaikaisesti. Tehtävien samanaikainen suorittaminen eri prosessoreilla vähentää ongelman ratkaisemiseen kuluvaa kokonaisaikaa suhteessa rinnakkaistamisessa käytettävien prosessorien lukumäärään. Tehtävien jaottelu eri prosessoreille voidaan toteuttaa esimerkiksi jakamalla ohjelman samankaltaisia laskutoimituksia eri prosessoreille, tai diskretoimalla aineistoa, jolle laskutoimituksia ollaan suorittamassa, tai suorittamalla eri ohjelmia eri prosessoreilla. Moniydinprosessorijärjestelmä kykenee kasvattamaan tietokoneen suoritustehoa entisestään pitäen tietokoneen energiankulutuksen, koon sekä painon merkittävästi pienempänä verrattuna kellotaajuuden kasvattamisella saavutettavaan suoritustehoon (Bhat, Babu ja Anala 2015). Käytännössä rinnakkaistaminen toteutetaan yleensä useiden keskusprosessorien välillä, mutta myös grafiikan käsittelyyn optimoitua grafiikkaprosessoria on alettu hyödyntämään rinnakkaislaskennassa tietynlaisissa tilanteissa.

2.1 Rinnakkaislaskennan yleisiä piirteitä

Vaikka rinnakkaislaskennalla on mahdollista säästää huomattavasti ongelman ratkaisemiseen kuluvaa kokonaisaikaa, siihen liittyy peräkkäiseen laskentaan verrattuna erityispiirteitä, jotka asettavat rajoitteita ohjelman rinnakkaistamisella saavutettavalle ajalliselle hyödylle. Keskeisimpiä tällaisia huomioitavia asioita ovat informaation synkronisointi prosessorien välillä, prosessorien muistinkäyttö sekä erilaiset virhetilanteet (Bhat, Babu ja Anala 2015). Informaation synkronisointiin prosessorien välillä kuluu aikaa datan siirrosta sekä muistipaikkoihin kirjoittamisesta johtuen, ja aikaa kuluu myös kun prosessorit joutuvat odottamaan, että muut prosessorit saavat suoritettua niiden käsittelemät laskutoimitukset. Tästä syystä jos informaation synkronisointia on paljon, rinnakkaislaskenta-algoritmin tehokkuus heikentyy merkittävästi. Lisäksi prosessoreilla on tehokkaasti käytettävissään vain

rajallinen määrä muistia, joten käsiteltävien muistialueiden tulee olla sopivan kokoisia yksittäisten prosessorien käytössä olevien muistialueiden koon mukaisesti, jotta algoritmi olisi mahdollisimman tehokas. Rinnakkaislaskenta-algoritmin tulee myös ottaa huomioon erilaiset virhetilanteet, kuten kilpailutilanteet. Kilpailutilanteissa useampi prosessori yrittää kirjoittaa samaan muistipaikkaan, jolloin prosessorien tuottamaa informaatiota kadotetaan.

Pääsääntöisesti ongelman ratkaisemiseen kuluvaa aikaa saadaan vähennettyä lisäämällä ongelmaa käsittelevien rinnakkaisten prosessorien lukumäärää. Kuitenkin prosessorien lukumäärän kasvattamisessa tulee yleensä raja vastaan, jolloin niitä lisäämällä ei enää saavuteta ajallista hyötyä. Erityisesti prosessorien väliseen informaation synkronisointiin kuluu paljon aikaa, mikä on ongelmana erityisesti silloin, kun prosessorien lukumäärä on suuri. Tästä johtuen osaongelmat tulisi jaotella siten, että ne olisivat mahdollisimman pieniä, ja samalla vaatisivat mahdollisimman vähän kommunikointia toistensa kanssa (Maday 2008). Myös rinnakkaisten säikeiden luominen sekä ylläpitäminen edellä kuvattujen erityispiirteiden mukaisesti lisää hieman vaadittujen laskutoimitusten lukumäärää käytettäessä rinnakkaislaskentaa (Bhat, Babu ja Anala 2015). Näin ollen, jos prosessorien lisäämisellä ei juurikaan saada pienennettyä osaongelmien kokoa, ei prosessorien lisäämisestä ole hyötyä. Lisäksi tietynlaisia riippuvuuksia sisältävät ongelmat eivät sovellu rinnakkaisella laskennalla toteutettavaksi.

Käytännön tasolla erilaiset rinnakkaislaskennan ohjelmakirjastot huomioivat edellä kuvatut rinnakkaislaskentaan liittyvät erityispiirteet, ja käyttäjän toteuttavaksi jää varmistaminen, että kirjoitettu ohjelmakoodi soveltuu tehokkaasti rinnakkaislaskentaa varten (Diaz, Muñoz-Caro ja Niño 2012). Ohjelman rinnakkaistamisen prosessin automaattiseen toteutukseen on olemassa joitakin ohjelmointirajapintoja, mutta rinnakkaistaminen voidaan myös tarvittaessa toteuttaa manuaalisesti ohjelmoijan toimesta tilanteesta riippuen (Bhat, Babu ja Anala 2015). Automaattinen rinnakkaistaminen vähentää vaatimusta käyttäjän osaamiselle rinnakkaislaskennan suhteen, sekä säästää sen toteuttamisen vaivalta. Automaattista rinnakkaistamista tukevia alustoja ovat esimerkiksi OpenMP, CUDA ja MPI, joista CPU-rinnakkaistami-

sen lisäksi CUDA tarjoaa myös automaattista GPU-rinnakkaistamista (Bhat, Babu ja Anala 2015).

Kuitenkaan automaattisella rinnakkaistamisella ei kyetä takaamaan optimaalista laskutehoa, vaan mahdollisimman hyvän rinnakkaislaskenta-algoritmin saamiseksi tulee rinnakkaistaminen toteuttaa manuaalisesti. Tämä on järkevää esimerkiksi silloin, kun ohjelman yksittäinen osa on laskennallisesti erittäin vaativa ja täten hidas muuhun ohjelmaan nähden. Sovellusten manuaalinen rinnakkaistaminen voi olla järkevää myös silloin, kun käsitellään sellaisia ongelmia, joiden ratkaisemiseen kuluu hyvin paljon aikaa. Esimerkiksi joissakin tieteellisissä sovelluksissa ongelman ratkaisemiseen voi kulua useita päiviä tai jopa viikkoja jokaisella tarkasteltavalla aineistolla.

Rinnakkaislaskennan kapasiteettia on myös mahdollista kasvattaa muodostamalla tietokoneista klustereita (Ma, Li ja Samatova 2007; Sun, Lin ja Wang 2016). Klusteri koostuu keskenään kommunikoivista solmuista useiden tietokoneiden muodostamassa verkossa. Ne ovat kustannustehokas ja joustava tapa muodostaa rinnakkaislaskentaympäristö, kun käsitellään laskennallisesti haastavia laskutoimituksia (Sun, Lin ja Wang 2016), sillä klusterin voi koota esimerkiksi työpisteistä (Ma, Li ja Samatova 2007).

2.2 Rinnakkaislaskenta grafiikkaprosessorilla

Rinnakkaislaskenta toteutetaan yleensä usean keskusprosessorin (*CPU*) välillä. Kuitenkin viimeisen vuosikymmenen aikana myös grafiikkaprosessorien (*GPU*) käyttö rinnakkaislaskennassa on alkanut yleistyä. Grafiikkaprosessori on alunperin kehitetty ja optimoitu raskaiden grafiikanmallinnukseen liittyvien laskutoimitusten laskentaan, mutta sen on havaittu myös sopivan muidenkin vastaavantyyppisten ongelmien tehokkaaseen rinnakkaiseen käsittelyyn (Tan ym. 2015; L'Ecuyer 2015). Joissakin tapauksissa rinnakkaislaskenta on huomattavasti nopeampaa grafiikkaprosessorilla kuin keskusprosessoreilla, ja voi vähentää vaadittua laskenta-aikaa jopa tunneista minuutteihin (Chen 2015).

Grafiikkaprosessorin rakenne asettaa kuitenkin rajoitteita sille, millaisissa tilanteissa sen käytölle rinnakkaislaskennassa on hyötyä. Grafiikkaprosessorilla voidaan nimittäin suorittaa yksittäinen käsky suurelle määrälle säikeitä samanaikaisesti, mutta tämän käskyn tulee olla sama kaikille säikeille jokaisella aika-askeleella (L'Ecuyer 2015). Tämän rajoitteen etuna on vähäisempi tarve tehtävien aikataulutukselle grafiikkaprosessoria käytettäessä (L'Ecuyer 2015). Lisäksi grafiikkaprosessorin rinnakkaisilla säikeillä on pääsy vain erittäin rajoitettuun muistitilaan, mikä tulee huomioida sovellettaessa grafiikkaprosessoria rinnakkaislaskentaan. Grafiikkaprosessorin käyttämistä rinnakkaislaskentaan muussakin tarkoituksessa kuin grafiikan mallinnuksessa käytetään nimitystä yleiskäyttöinen grafiikkaprosessori (engl. *General Purpose Graphics Prosessing Unit*, *GPGPU*).

Grafiikkaprosessoreilla tehokkaasti käsiteltäviä ongelmatyyppejä ovat erityisesti matriisioperaatiot, kuten matriisien kertolasku sekä matriisin kääntäminen, kun matriisien sarakkeiden sekä rivien lukumäärä on useiden tuhansien luokkaa (Liu, Liang ja Ansari 2016; Chen 2015). Esimerkiksi suuria tietoaineistoja (engl. *big data*) käsiteltäessä joudutaan tekemään matriisioperaatioita erittäin suurille matriiseille. Suuria tietoaineistoja ovat esimerkiksi internetin sosiaalisia verkostoja käsittelevät sovellukset sekä internetissä toimivat suosituspalvelut. (Liu, Liang ja Ansari 2016.) GPU-laskenta soveltuu hyvin myös suurten lineaaristen yhtälöryhmien ratkaisemiseen, kuten myös tieteellisen laskennan sekä simuloinnin tehostamiseen.

3 Diracin yhtälö

Diracin yhtälö kehitettiin vuonna 1928 mallintamaan kvanttifysiikan ilmiöitä suhteellisuusteorian mukaisella tavalla

$$i\hbar\frac{\partial\psi(t,\mathbf{x})}{\partial t} = \hat{H}\psi(t,\mathbf{x}), \tag{3.1}$$

missä $\hbar$ on redusoitu Planckin vakio, $\hat{H}$ on Hamiltonin operaattori ja $\psi(t,\mathbf{x})$ on ratkaistava aaltofunktio, joka kuvaa elektronin lepomassaa. Diracin yhtälössä (3.1) esiintyvä Hamiltonin operaattori $\hat{H}$ voidaan esittää muodossa

$$\hat{H} = \alpha\cdot[cp - eA(t,\mathbf{x})] + \beta mc^2 + e\mathbb{I}_4 V(t,\mathbf{x}), \tag{3.2}$$

missä c on valon nopeus, p on momenttioperaattori, e on alkeisvaraus, m on hiukkasen massa, $A(t,\mathbf{x})$ kuvaa sähkömagneettisen vektoripotentiaalin paikka-avaruuden komponentteja, $V(t,\mathbf{x})$ on skalaaripotentiaali, ja α ja β ovat Diracin matriiseja, joissa spin-operaattoreina on käytetty Paulin matriiseja. Diracin yhtälössä esiintyville spin-operaattoreille on esitetty kirjallisuudessa useita eri operaattoreita (Bauke ym. 2014). Kirjallisuudessa eniten käytetty spin-operaattori on Paulin matriisit, mutta Bauke ym. (2014) esittävät, että Foldy-Wouthuysenin sekä Prycen spin-operaattorit voisivat olla paras valinta relativististen ilmiöiden kuvaamiseen. Tässä opinnäytetyössä tarkasteltavassa Diracin yhtälön numeerisessa ratkaisumenetelmässä spin-operaattoreina on käytetty Paulin matriiseja.

Diracin yhtälö (3.1) on aikariippuva aaltoyhtälö, joka mallintaa spin-$\frac{1}{2}$ hiukkasten, kuten elektronien ja kvarkkien käyttäytymistä, ja sillä on ollut keskeinen asema hiukkasfysiikan sekä molekyylikemian ilmiöiden mallintamisessa. Käytännön tasolla yhtälön kuvaamia ilmiöitä tutkitaan törmäyttämällä hiukkasia toisiinsa sekä käyttämällä voimakkaita laserpulsseja, jotka ovat kehittyneet merkittävästi viimeisten vuosikymmenten aikana. Nykyään laserin avulla voidaan kiihdyttää hiukkasia jopa valon nopeutta lähestyviin nopeuksiin, mikä mahdollistaa relativististen

eli suhteellisuusteorian mukaisten ilmiöiden tarkastelua (Salamin ym. 2006; Braun, Su ja Grobe 1999), ja elektronien kiihdyttämiseen voidaan hyödyntää laserin lisäksi myös magneettikenttiä (Krekora ym. 2001). Diracin yhtälö mallintaa täsmällisesti yhden elektronin järjestelmiä, mutta useamman elektronin järjestelmille ei ole olemassa tarkkaa relativistista mallia. Raskaampien atomien ja molekyylien tapauksessa relativististen vaikutusten merkitys kasvaa suureksi. (Nakatsuji ja Nakashima 2005.)

Diracin yhtälö yhdistää kvanttimekaniikan sekä erityisen suhteellisuusteorian (Lorin ja Bandrauk 2011; Succi 1996), ja sitä voidaan pitää Schrödingerin yhtälön relativistisena vastikkeena. Relativistisen ja ei-relativistisen mallin suurin ero on sidosenergia, jolla alkaa olla merkitystä mallin kannalta, kun hiukkasten varaukset ovat suuria (Milosevic, Krainov ja Brabec 2002). Massattoman hiukkasen $m = 0$ tapauksessa mallissa (3.2) saadaan Diracin yhtälön erikoistapauksena Weylin yhtälö.

Diracin yhtälöllä on useita keskeisiä sovelluskohteita. Yhtälön avulla on muun muassa ennustettu antimaterian olemassaolo (Hammer, Pötz ja Arnold 2014), jonka lisäksi Diracin yhtälö selittää tehokkaasti taustalla olevaa fysiikkaa esimerkiksi kaksi- ja kolmiulotteisille eristeille sekä optisille hiloille, kuten myös grafeiinin elektroneille kokeellisen tarkastelun yhteydessä (Hammer, Pötz ja Arnold 2014). Diracin yhtälön avulla voidaan myös tutkia tunneli-ionisaatiota (Milosevic, Krainov ja Brabec 2002), ja lisäksi yhtälö ennustaa joitakin epäklassisia ilmiöitä, kuten Kleinin paradoksia ja oskillointi-ilmiötä (saks. *Zitterbewegung*) (Lorin ja Bandrauk 2011).

3.1 Diracin yhtälön ratkaiseminen numeerisesti

Diracin yhtälö (3.1) on aikariippuva aaltoyhtälö, jossa ratkaistavana on osittaisdifferentiaaliyhtälö. Yhtälön analyyttinen tarkastelu toimii vain hyvin rajoitetuissa tilanteissa, ja usein analyyttisten menetelmien kehittäminen on haastavaa (Fillion-Gourdeau, Lorin ja Bandrauk 2016). Tästä johtuen Diracin yhtälöä käsitellään pääosin numeerisesti.

Diracin yhtälön ratkaiseminen mahdollisimman tarkasti on tärkeää tulosten hyö-

dyntämiseksi teorian sekä käytännön havaintojen tukena (Lorin ja Bandrauk 2011; Salomonson ja Öster 1989). Tarkoissa laskutoimituksissa elektronien välinen korrelaatio, eli parinmuodostuminen, tulee laskea relativistisesti (Salomonson ja Öster 1989), ja Diracin yhtälön kannalta on myös tärkeää huomioida miten käsitellään sen negatiivisten energioiden ratkaisut (Selstø, Lindroth ja Bengtsson 2009). Yhtälön numeerisessa ratkaisemisessa joudutaan käyttämään hyvin tiheitä laskentahiloja lyhyillä aika-askeleilla (Bauke ym. 2011), ja tästä johtuen sen ratkaiseminen vaatii paljon laskentaresursseja, mikä asettaa rajoitteita yhtälön tarkastelulle. Usein Diracin yhtälöä tarkastellaan yksinkertaistetussa muodossa esimerkiksi rajoituttumalla yhteen tai kahteen paikka-avaruuden dimensioon (Selstø, Lindroth ja Bengtsson 2009).

Diracin yhtälön ratkaisemiseen liittyy useita ongelmakohtia, joihin saatetaan törmätä, jos sitä yritetään ratkaista liian suoraviivaisesti. Eräs keskeinen ongelma on fermionien kahdentuminen (engl. *Fermion doubling*). Fermioneja eli ainehiukkasia ovat hiukkaset, joiden spin on joko $\frac{1}{2}$ tai $\frac{3}{2}$, eli leptonit, kvarkit sekä baryonit. Fermionien kahdentumisessa jokaista alkuperäistä fermionia kohden saadaan 2^D kappaletta fermioneja, missä D on diskretoitavien ulottuvuuksien lukumäärä. Tämä johtuu tiettyjen numeeristen ratkaisumenetelmien yhteydessä esiintyvästä numeerisesta virheestä ja on ristiriidassa teorian kanssa. Esimerkiksi on osoitettu, että fermionien kahdentumisen sallittaessa malli mahdollistaisi aaltojen etenemisen valon nopeutta suuremmalla nopeudella (Müller, Grün ja Scheid 1998).

Fermionien kahdentumisena havaittavan virheen korjaaminen on tärkeää, jotta tulokset olisivat todenmukaisia ja vertailukelpoisia mittaustulosten kanssa (Bottcher ja Strayer 1985). Virhettä on yritetty pienentää useilla eri tavoilla. Fillion-Gourdeau, Lorin ja Bandrauk (2012) mukaan Wilsonin fermioneihin ja lomitettuihin fermioneihin (engl. *staggered fermions*) perustuviin menetelmiin liittyy lieveilmiönä symmetrian rikkoutuminen. Viime vuosina hyviksi havaittuja menetelmiä ovat olleet operaattorinjako (engl. *operator split*) (Hammer ja Pötz 2014) ja lomitetun hilan menetelmä (engl. *staggered grid*) (Hammer, Pötz ja Arnold 2014).

Differentiaaliyhtälöiden, kuten Diracin yhtälön, numeerinen laskenta vaatii myös tietynlaisia reunaehtoja (Hammer, Pötz ja Arnold 2014). Vapausasteiden lukumää-

rän tulee olla äärellinen, ja aika- ja paikkamuuttujien diskretointi reaaliavaruudessa tulee rajoittaa äärelliseen alueeseen sekä ajallisesti, että paikallisesti. Lisäksi tulee asettaa sopivia reunaehtoja siten, että saatava ratkaisu rajoitetussa alueessa on riittävän tarkka approksimaatio rajoittamattomassa alueessa esitetylle ongelmalle. (Hammer, Pötz ja Arnold 2014.)

Diracin yhtälön numeeriseen ratkaisemiseen on esitetty kirjallisuudessa useita erilaisia menetelmiä. Diracin yhtälön (3.1) diskretointi ajan suhteen voidaan toteuttaa esimerkiksi operaattorinjakomenetelmällä (Mocken ja Keitel 2008) tai differenssimenetelmällä (engl. *finite difference method*) (Hammer, Pötz ja Arnold 2014). Paikan suhteen diskretointi puolestaan voidaan toteuttaa esimerkiksi tilavuusmenetelmällä (engl. *finite volume method*) (Momberger, Belkacem ja Sørensen 1996), elementtimenetelmällä (engl. *finite element method*) (Lorin ja Bandrauk 2011) tai differenssimenetelmällä (Braun, Su ja Grobe 1999), mutta kirjallisuudessa on esitetty myös muitakin menetelmiä. Tässä opinnäytetyössä valitaan tarkasteltavaksi artikkelissa (Fillion-Gourdeau, Lorin ja Bandrauk 2012) esitetty Diracin yhtälön numeerisen laskennan menetelmä, jossa aikadiskretointi on toteutettu operaattorinjakomenetelmällä ja paikkadiskretointi tilavuusmenetelmällä, sillä operaattorinjakoon perustuvat menetelmät vaikuttavat olevan yleisiä kirjallisuudessa, esimerkiksi (Mocken ja Keitel 2008) ja (Braun, Su ja Grobe 1999). Kyseinen menetelmä on myös toteutettu siten, että se tukee rinnakkaislaskentaa, eikä siinä esiinny fermionien kahdentumista (Fillion-Gourdeau, Lorin ja Bandrauk 2012).

3.1.1 Aikadiskretointi

Tarkastellaan seuraavaksi Diracin yhtälön diskretointia ensin ajan suhteen. Fillion-Gourdeau, Lorin ja Bandrauk (2012) toteuttavat ajan diskretoinnin operaattorinjakomenetelmällä, jolla saadaan diskretoitua aaltofunktion koordinaattien x, y ja z suuntaiset komponentit erilleen toisistaan, jolloin Diracin yhtälö (3.1) voidaan esittää muodossa

$$i\partial_t \psi(t)^{(1)} = -ic\alpha_x \partial_x \psi(t)^{(1)}, \quad \psi(t_n)^{(1)} = \psi^n, \qquad t \in [t_n, t_{n+1}) \tag{3.3}$$

$$i\partial_t \psi(t)^{(2)} = -ic\alpha_y \partial_y \psi(t)^{(2)}, \quad \psi(t_n)^{(2)} = \psi^{(1)}, \quad t \in [t_n, t_{n+1}) \tag{3.4}$$

$$i\partial_t \psi(t)^{(3)} = -ic\alpha_z \partial_z \psi(t)^{(3)}, \quad \psi(t_n)^{(3)} = \psi^{(2)}, \quad t \in [t_n, t_{n+1}) \tag{3.5}$$

$$i\partial_t \psi(t)^{(4)} = \left[\beta mc^2 + e\mathbb{I}_4 V(t,\mathbf{x}) - e\boldsymbol{\alpha}\cdot A(t,\mathbf{x})\right] \psi(t)^{(4)},\ \psi(t_n)^{(4)} = \psi^{(3)},\ t \in [t_n, t_{n+1}) \tag{3.6}$$

$$\psi^{n+1} = \psi^{(4)}(t_{n+1}). \tag{3.7}$$

Menetelmän suurin numeerinen virhe tulee käytetystä operaattorien jaosta (3.3) – (3.7). Tässä esitetty ensimmäisen kertaluokan operaattorinjakomenetelmä johtaa kertaluokan $O(\delta t^2)$ virhetermiin. Toisen kertaluokan operaattorinjaolla virheluokaksi saadaan $O(\delta t^3)$, ja korkeamman kertaluokan operaattorinjaoilla voitaisiin saavuttaa vieläkin parempi tarkkuus. Kuitenkin korkeamman kertaluokan operaattorinjakomenetelmien käyttäminen kasvattaa laskenta-aikaa huomattavasti, joten artikkelissa (Fillion-Gourdeau, Lorin ja Bandrauk 2012) on pitäydytty ensimmäisen ja toisen kertaluokan operaattorinjakomenetelmissä.

Yhtälöihin (3.3), (3.4) ja (3.5) sovelletaan karakteristista menetelmää (engl. *method of characteristics*), joka on osittaisdifferentiaaliyhtälöiden ratkaisutekniikka. Diracin yhtälön tapauksessa tämä toteutetaan siten, että ensin Diracin matriisi α muotoillaan uudelleen, jolloin se saadaan diagonalisoitua spinor-komponenttien erottamiseksi toisistaan. Tämän jälkeen karakteristisella menetelmällä etsitään yhtälöille täsmällinen analyyttinen ratkaisu, ja lopuksi Diracin matriisi muunnetaan takaisin alkuperäiseen esitysmuotoonsa. Näin operaattorinjaolla saadut yhtälöt (3.3), (3.4) ja (3.5) voidaan esittää seuraavassa muodossa (Fillion-Gourdeau, Lorin ja Bandrauk 2012):

$$\psi^{(1)}(t_{n+1},\mathbf{x}) = \frac{1}{2}\{[\mathbb{I}_4 + \alpha_x]\psi^n(x - c\delta t, y, z) + [\mathbb{I}_4 - \alpha_x]\psi^n(x + c\delta t, y, z)\} \tag{3.8}$$

$$\psi^{(2)}(t_{n+1},\mathbf{x}) = \frac{1}{2}\{[\mathbb{I}_4 + \alpha_y]\psi^{(1)}(t_{n+1}x, y - c\delta t, z) + [\mathbb{I}_4 - \alpha_y]\psi^{(1)}(t_{n+1}x, y + c\delta t, z)\} \tag{3.9}$$

$$\psi^{(3)}(t_{n+1},\mathbf{x}) = \frac{1}{2}\{[\mathbb{I}_4 + \alpha_z]\psi^{(2)}(t_{n+1}x, y, z - c\delta t) + [\mathbb{I}_4 - \alpha_z]\psi^{(2)}(t_{n+1}x, y, z + c\delta t)\}, \tag{3.10}$$

ja yhtälö (3.6) voidaan kirjoittaa muodossa

$$\psi^{(4)}(t_{n+1},\mathbf{x}) = exp\left[-i\int_{t_n}^{t_{n+1}} d\tau[\beta mc^2 - e\alpha\cdot A(\tau,\mathbf{x})\right] \times exp\left[-ie\int_{t_n}^{t_{n+1}} d\tau V(\tau,\mathbf{x})\right] \psi^{(3)}(t_{n+1},\mathbf{x}). \tag{3.11}$$

Yhtälön (3.11) oikean puolen edessä käytetään usein aikajärjestelyoperaattoria T, mutta se korvataan tarkasteltavassa menetelmässä kertoimella $T = 1$ yksinkertaisuuden vuoksi sekä laskennallisen työn vähentämiseksi, sillä tämä ei kasvata virhettä merkittävästi. Aikajärjestelyoperaattorin T poisjättämisestä aiheutuva virhetermi on kertaluokkaa $O(\delta t^3)$, joka on parempi kuin ensimmäisen kertaluokan operaattorinjaosta aiheutuva virhetermi $O(\delta t^2)$, ja samaa kertaluokkaa kuin toisen kertaluokan operaattorinjaon virhetermi $O(\delta t^3)$. Korkeamman kertaluokan operaattorinjakomenetelmiä käytettäessä aikajärjestelyoperaattori voi olla hyvä ottaa tarkasteluun mukaan.

3.1.2 Paikkadiskretointi

Aikadiskretoinnin lisäksi tehtävä tulee diskretoida myös paikan suhteen. Tässä tutkielmassa keskitytään tapaukseen, jossa käytetään äärellisten tilavuuksien menetelmää (Fillion-Gourdeau, Lorin ja Bandrauk 2012; Divitiis, Petronzio ja Tantalo 2004; Momberger, Belkacem ja Sørensen 1996). Äärellisten tilavuuksien menetelmässä paikka-avaruus jaetaan kuutioelementteihin, joiden sivun pituus on $a = \delta x = \delta y = \delta z$. Aaltofunktion arvoa yksittäisen tällaisen kuutioelementin sisällä kuvaa vakio.

Menetelmällä saadaan seuraavanlainen diskretointi aaltofunktiolle $\psi(t,\mathbf{x})$ ja sähkömagneettiselle kentälle $A(t,\mathbf{x})$:

$$\psi_h(t,\mathbf{i}) = \sum_{m=1}^{N} \mathbf{1}_m(\mathbf{i})\psi(t,\bar{\mathbf{x}}_m) \tag{3.12}$$

$$A_h(t,\mathbf{i}) = \sum_{m=1}^{N} \mathbf{1}_m(\mathbf{i})A(t,\bar{\mathbf{x}}_m), \tag{3.13}$$

missä N on kuutioelementtien kokonaislukumäärä, $\mathbf{i} = (i,j,k)$ ovat kuutioelementtien indeksit ja $\bar{\mathbf{x}}_m$ kuvaa jatkuvan funktion arvoa yksittäisen kuutioelementin kes-

kipisteessä.

Nyt varsinainen numeerinen menetelmä saadaan yhdistämällä paikan ja ajan diskretointi toisiinsa. Valitsemalla $c\delta t = a$, missä c on valon nopeus, saadaan diskreetti malli

$$\psi_h^{n_1}(\mathbf{i}) = \frac{1}{2}\{[\mathbb{I}_4 + \alpha_x]\psi_h^n(i-1,j,k) + [\mathbb{I}_4 - \alpha_x]\psi_h^n(i+1,j,k)\} \tag{3.14}$$

$$\psi_h^{n_2}(\mathbf{i}) = \frac{1}{2}\{[\mathbb{I}_4 + \alpha_y]\psi_h^{n_1}(i,j-1,k) + [\mathbb{I}_4 - \alpha_y]\psi_h^{n_1}(i,j+1,k)\} \tag{3.15}$$

$$\psi_h^{n_3}(\mathbf{i}) = \frac{1}{2}\{[\mathbb{I}_4 + \alpha_z]\psi_h^{n_2}(i,j,k-1) + [\mathbb{I}_4 - \alpha_z]\psi_h^{n_2}(i,j,k+1)\} \tag{3.16}$$

$$\psi_h^{n+1}(\mathbf{i}) = exp\left[-i\beta mc^2\delta t - i\tilde{V}_h^n(\mathbf{i}) + i\alpha\cdot\tilde{A}_h(\mathbf{i})\right]\psi_h^{n_3}(\mathbf{i}), \tag{3.17}$$

missä $\tilde{V}$ ja $\tilde{A}$ ovat sähkömagneettisia kenttiä kuvaavia integraalilausekkeita, joiden numeeriset arvot lasketaan summina. Aika-askel δt tulisi valita siten, että $\delta t \ll \frac{1}{mc^2}$, sillä tarkkoja tuloksia saavutetaan valitsemalla $\delta t < \frac{1}{5}\cdot\frac{1}{mc^2}$. (Fillion-Gourdeau, Lorin ja Bandrauk 2012.)

3.2 Diracin yhtälön rinnakkaistaminen

Jotta Diracin yhtälö saataisiin ratkaistua nopeammin, sille on alettu kehittämään rinnakkaislaskentamenetelmiä. Yleensä rinnakkaistaminen eri prosessorien kesken toteutetaan jakamalla laskettava tehtävä joko paikka-avaruuden, ajan, tai näiden molempien suhteen. Ajan suhteen rinnakkaistaminen on kuitenkin usein huomattavasti haastavampaa kuin paikan suhteen rinnakkaistaminen. Kun paikan suhteen rinnakkaistaminen saavuttaa rajansa, on mahdollista diskretoida aineistoa myös ajan suhteen, mikä mahdollistaa laskentanopeuden kasvattamista edelleen (Gander 2015).

Ajan suhteen rinnakkaisen laskennan (engl. *time parallel*) toteuttaminen on ollut mahdollista vasta viimeisen vuosikymmenen aikana, kun massiivisesti rinnakkaiset tietokoneet ovat yleistyneet (Gander 2015). Aikariippuville ongelmille kyetään

nykyään saavuttamaan tarkkoja ratkaisuja hyvällä konvergenssinopeudella, ja riittävän suurella määrällä prosessoreita aikarinnakkaiset algoritmit ovat nopeampia kuin klassiset algoritmit (Gander 2015).

Tässä opinnäytetyössä tarkasteltavassa artikkelissa (Fillion-Gourdeau, Lorin ja Bandrauk 2012) Diracin yhtälön numeerinen ratkaisumenetelmä kehitettiin siten, että se soveltuu hyvin rinnakkaislaskentaan. Fillion-Gourdeau, Lorin ja Bandrauk (2012) toteuttavat menetelmässään Diracin yhtälön rinnakkaistamisen hyödyntämällä aluejakomenetelmää (engl. *domain decomposition*), joka on yleinen tapa toteuttaa rinnakkaistamista. Aluejaossa laskentahilan alueet jaetaan pienemmiksi alialueiksi (engl. *subdomains*), jotka suoritetaan itsenäisesti eri prosessoreilla. Ainoastaan alialueiden reunoja tarvitsee päivittää vierekkäisiä alueita käsittelevien prosessorien kesken, joten prosessorien välisen kommunikoinnin tarve on pieni. Prosessorien välinen kommunikointi puolestaan on toteutettu MPI:llä (*Message Passing Interface*) (Fillion-Gourdeau, Lorin ja Bandrauk 2012). MPI on yleisesti käytetty prosessienvälisen viestinvälityksen ohjelmakirjasto, joka on erityisesti suunniteltu hajautetun muistin mallia varten. Hajautetun muistin arkkitehtuureja ovat esimerkiksi klusterit ja grafiikkaprosessorit.

Opinnäytetyössä tarkasteltavan menetelmän etuna on sen yksinkertaisuus sekä skaalautuvuus erityyppisiin ongelmatapauksiin. Fillion-Gourdeau, Lorin ja Bandrauk (2012) esittävät, että menetelmä on helppo toteuttaa ohjelmointikielillä 1–3 paikkaulotteisissa tapauksissa, ja että paikkaulottuvuuksien lukumäärän muutos vaatii vain pieniä muutoksia koodiin, kuten matriisien koon muuttamista ja operaattoreiden lisäämistä tai poistamista. Kirjallisuudessa esiintyvät menetelmät keskittyvät usein tietynsuuruisen paikkaulottuvuuden tapauksen tarkasteluun yksittäisessä tilanteessa, joten tämän menetelmän skaalautuvuus erityyppisiin tilanteisiin on hyvä, vaikka tapauskohtaisella menetelmällä pystytään todennäköisesti optimaalisempaan suoritustehoon. Ohjelmakoodi toteutettiin kyseisessä artikkelissa C++ kielellä.

Tarkasteltavassa artikkelissa toteutettiin ensimmäisen ja toisen kertaluvun operaattorinjakomenetelmät, ja menetelmän tarkkuutta voitaisiin kasvattaa helposti toteut-

tamalla korkeamman kertaluvun operaattorinjako. Lisäksi rinnakkaistamisen toteuttaminen menetelmälle voidaan tehdä yksinkertaisesti aluejaolla vähäisellä tarpeella prosessorienväliselle kommunikoinnille. (Fillion-Gourdeau, Lorin ja Bandrauk 2012.) Numeerisen menetelmän tarkkuus ja tehokkuus osoitettiin käyttämällä sitä analyyttisesti tunnettujen tapausten ratkaisemiseen, Gaussin aaltopaketteihin ja Kleinin paradoksiin, ja vertaamalla saatuja tuloksia teoreettisiin arvoihin (Fillion-Gourdeau, Lorin ja Bandrauk 2012). Tarkasteltavassa artikkelissa myös osoitetaan, ettei siinä esiinny fermionien kahdentumista, ja esitetään, että menetelmä säilyttää useimmat Diracin yhtälön symmetriat.

Menetelmän haittapuolena on kuitenkin, että Diracin yhtälö on raskasta laskea kyseisellä menetelmällä, kun paikkaulottuvuuksia on kolme. Tällöin on mielekästä tarkastella vain ajallisesti lyhytkestoisia ilmiöitä, kuten raskaiden ionien törmäyksiä. (Fillion-Gourdeau, Lorin ja Bandrauk 2014.) Pidempiaikaisten ilmiöiden tapauksessa, kuten laserin ja aineen vuorovaikutuksessa, tarvittaisiin tehokkaampi menetelmä kolmen paikkaulottuvuuden tapauksen tarkastelemiseen. Vaihtoehtoisesti Diracin yhtälö voitaisiin myös palauttaa kahden paikkaulottuvuuden tapaukseksi käyttäen symmetriaehtoja. (Fillion-Gourdeau, Lorin ja Bandrauk 2014.) Lisäksi kyseisellä menetelmällä vaadittu laskenta-aika kasvaa nopeasti käytettäessä kolmannen tai korkeamman kertaluvun operaattorinjakomenetelmiä. Artikkelissa ei myöskään käsitelty kuinka voimakkaasti monimutkaisempien sähkömagneettisten kenttien tarkastelu vaikuttaa menetelmän laskenta-aikaan.

Myös grafiikkaprosessoreilla on toteutettu useita toimivia sovelluksia osittaisdifferentiaaliyhtälöiden ratkaisemiseen liittyen (Chen ja Shen 2013), ja grafiikkaprosessoria on ehdotettu myös Diracin yhtälön tarkasteluun (Bauke ja Keitel 2011). Eräässä tarkastelussa sovellettiin grafiikkaprosessoreita aikariippuvan Diracin yhtälön ratkaisemiseen käyttäen Fourierin operaattorinjakomenetelmää (engl. *Fourier split operator*), ja kyseisellä menetelmällä saavutettiin grafiikkaprosessoreilla parempi laskuteho tarkasteltavassa tilanteessa kuin keskusprosessoreilla. Fourierin operaattorinjako voidaan toteuttaa tehokkaasti grafiikkaprosessoreilla, sillä grafiikkaprosessoreiden muistikaista on huomattavasti suurempi kuin keskusprosessoreilla, mikä ra-

joittaa menetelmän tehokasta hyödyntämistä keskusprosessoreilla. (Bauke ja Keitel 2011.)

4 Yhteenveto

Diracin yhtälö on keskeinen kvanttifysiikan ilmiöitä kuvaava yhtälö. Diracin yhtälön ratkaisemiseen on esitetty kirjallisuudessa useita erilaisia menetelmiä, joista huomattava osa perustui operaattorinjakoon. Perinteistä elementti- tai differenssimenetelmää käytettäessä voi esiintyä epäfysikaalisia, fermionien kahdentumista kuvaavia, numeerisia ratkaisuja. Niiden välttämiseen on käytetty tyypillisesti operaattorinjakomenetelmää. Viime vuosina on todettu, että lomitetun hilan menetelmiä käytettäessä vältytään fermionien kahdentumiselta.

Rinnakkaislaskenta on nykyään keskeisin menetelmä laskentanopeuden kasvattamiseksi peräkkäisen laskennan nopeuden saavutettua rajansa. Diracin yhtälön numeerinen ratkaiseminen vaatii paljon laskentaresursseja, joten soveltamalla rinnakkaislaskentaa yhtälön numeerisen laskennan menetelmän tukena voidaan saavuttaa merkittävä ajallinen hyöty. Yhtälön rinnakkaistamiseen on tyypillisesti käytetty aluejakoa, ja prosessorien välinen kommunikointi on toteutettu MPI:llä. Opinnäytetyössä tarkasteltavan menetelmän etuna on sen toteutuksen yksinkertaisuus, skaalautuvuus erityyppisiin tapauksiin, sekä menetelmän soveltuvuus rinnakkaislaskentaan. Menetelmän heikkouksia ovat laskenta-ajan nopea kasvu tarkasteltaessa Diracin yhtälöä kolmessa paikkaulottuvuudessa tai käytettäessä kolmannen tai korkeamman kertaluokan operaattorinjakomenetelmiä. Diracin yhtälön rinnakkaislaskentaan on myös ehdotettu grafiikkaprosessorien käyttämistä, joilla mahdollisesti voitaisiin säästää laskenta-aikaa. Lisäksi kehittämällä laskentamenetelmä tapauskohtaisesti voidaan saavuttaa laskentaresurssien kannalta tehokkaampi menetelmä, mutta tässä opinnäytetyössä käsitelty menetelmä soveltuu Diracin yhtälön tarkastelemiseen useissa erityyppisissä ongelmatapauksissa.

Lähteet

Akcelik, Biros ja Ghattas. 2002. "Parallel Multiscale Gauss-Newton-Krylov Methods for Inverse Wave Propagation". Teoksessa *Supercomputing, ACM/IEEE 2002 Conference*, 41–41. doi:10.1109/SC.2002.10002.

Bauke, Ahrens, Keitel ja Grobe. 2014. "Relativistic spin operators in various electromagnetic environments". *Phys. Rev. A* 89 (5): 052101. doi:10.1103/PhysRevA.89.052101.

Bauke, Hetzheim, Mocken, Ruf ja Keitel. 2011. "Relativistic ionization characteristics of laser-driven hydrogenlike ions". *Phys. Rev. A* 83 (6): 063414. doi:10.1103/PhysRevA.83.063414.

Bauke ja Keitel. 2011. "Accelerating the Fourier split operator method via graphics processing units". *Computer Physics Communications* 182 (12): 2454–2463. ISSN: 0010-4655. doi:http://dx.doi.org/10.1016/j.cpc.2011.07.003.

Bhat, Babu ja Anala. 2015. "Towards automatic parallelization of 'for' loops". Teoksessa *Advance Computing Conference (IACC), 2015 IEEE International*, 136–142. doi:10.1109/IADCC.2015.7154686.

Bottcher ja Strayer. 1985. "Numerical Solution of the Time-Dependent Dirac Equation with Application to Positron Production in Heavy-Ion Collisions". *Phys. Rev. Lett.* 54 (7): 669–672. doi:10.1103/PhysRevLett.54.669.

Braun, Su ja Grobe. 1999. "Numerical approach to solve the time-dependent Dirac equation". *Phys. Rev. A* 59 (1): 604–612. doi:10.1103/PhysRevA.59.604.

Chen. 2015. "A New Framework of GPU-Accelerated Spectral Solvers: Collocation and Glerkin Methods for Systems of Coupled Elliptic Equations". *Journal of Scientific Computing* 62 (2): 575–600. ISSN: 1573-7691. doi:10.1007/s10915-014-9868-3.

Chen ja Shen. 2013. "A GPU parallelized spectral method for elliptic equations in rectangular domains". *Journal of Computational Physics* 250:555–564. ISSN: 0021-9991. doi:http://dx.doi.org/10.1016/j.jcp.2013.05.031.

Diaz, Muñoz-Caro ja Niño. 2012. "A Survey of Parallel Programming Models and Tools in the Multi and Many-Core Era". *IEEE Transactions on Parallel and Distributed Systems* 23, numero 8 (): 1369–1386. ISSN: 1045-9219. doi:10.1109/TPDS.2011.308.

Divitiis, Petronzio ja Tantalo. 2004. "On the discretization of physical momenta in lattice QCD". *Physics Letters B* 595 (1–4): 408–413. ISSN: 0370-2693. doi:http://dx.doi.org/10.1016/j.physletb.2004.06.035.

Fillion-Gourdeau, Lorin ja Bandrauk. 2012. "Numerical solution of the time-dependent Dirac equation in coordinate space without fermion-doubling". *Computer Physics Communications* 183 (7): 1403–1415. ISSN: 0010-4655. doi:http://dx.doi.org/10.1016/j.cpc.2012.02.012.

———. 2014. "A split-step numerical method for the time-dependent Dirac equation in 3-D axisymmetric geometry". *Journal of Computational Physics* 272:559–587. ISSN: 0021-9991. doi:http://dx.doi.org/10.1016/j.jcp.2014.03.068.

———. 2016. "Galerkin method for unsplit 3-D Dirac equation using atomically/kinetically balanced B-spline basis". *Journal of Computational Physics* 307:122–145. ISSN: 0021-9991. doi:http://dx.doi.org/10.1016/j.jcp.2015.11.024.

Gander. 2015. "50 Years of Time Parallel Time Integration". Teoksessa *Multiple Shooting and Time Domain Decomposition Methods: MuS-TDD, Heidelberg, May 6-8, 2013*, toimittanut Thomas Carraro, Michael Geiger, Stefan Körkel ja Rolf Rannacher, 69–113. Cham: Springer International Publishing. ISBN: 978-3-319-23321-5. doi:10.1007/978-3-319-23321-5_3.

Hammer ja Pötz. 2014. "Staggered grid leap-frog scheme for the Dirac equation". *Computer Physics Communications* 185 (1): 40–52. ISSN: 0010-4655. doi:http://dx.doi.org/10.1016/j.cpc.2013.08.013.

Hammer, Pötz ja Arnold. 2014. "A dispersion and norm preserving finite difference scheme with transparent boundary conditions for the Dirac equation in (1+1)D". *Journal of Computational Physics* 256:728–747. ISSN: 0021-9991. doi:http://dx.doi.org/10.1016/j.jcp.2013.09.022.

Krekora, Wagner, Su ja Grobe. 2001. “Dirac theory of ring-shaped electron distributions in atoms”. *Phys. Rev. A* 63 (2): 025404. doi:10.1103/PhysRevA.63.025404.

Le, Hoang, Li, L. Liu, H. Liu ja Hu. 2016. “A fast PC algorithm for high dimensional causal discovery with multi-core PCs”. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* PP (99): 1–1. ISSN: 1545-5963. doi:10.1109/TCBB.2016.2591526.

L’Ecuyer. 2015. “Random number generation with multiple streams for sequential and parallel computing”. Teoksessa *2015 Winter Simulation Conference (WSC)*, 31–44. doi:10.1109/WSC.2015.7408151.

Liu, Liang ja Ansari. 2016. “Spark-Based Large-Scale Matrix Inversion for Big Data Processing”. *IEEE Access* 4:2166–2176. ISSN: 2169-3536. doi:10.1109/ACCESS.2016.2546544.

Lorin ja Bandrauk. 2011. “A simple and accurate mixed solver for the Maxwell–Dirac equations”. *Nonlinear Analysis: Real World Applications* 12 (1): 190–202. ISSN: 1468-1218. doi:http://dx.doi.org/10.1016/j.nonrwa.2010.06.007.

Lu, Halappanavar, Chavarria-Miranda, Gebremedhin, Panyala ja Kalyanaraman. 2016. “Algorithms for Balanced Graph Colorings with Applications in Parallel Computing”. *IEEE Transactions on Parallel and Distributed Systems* PP (99): 1–1. ISSN: 1045-9219. doi:10.1109/TPDS.2016.2620142.

Ma, Li ja Samatova. 2007. “Automatic Parallelization of Scripting Languages: Toward Transparent Desktop Parallel Computing”. Teoksessa *2007 IEEE International Parallel and Distributed Processing Symposium*, 1–6. doi:10.1109/IPDPS.2007.370488.

Maday. 2008. *The parareal in time algorithm.* Tekninen raportti. https://www.ljll.math.upmc.fr/publications/2008/R08030.html.

Migdalas, Toraldo ja Kumar. 2003. “Nonlinear optimization and parallel computing”. Parallel computing in numerical optimization, *Parallel Computing* 29 (4): 375–391. ISSN: 0167-8191. doi:http://dx.doi.org/10.1016/S0167-8191(03)00013-9.

Milosevic, Krainov ja Brabec. 2002. “Semiclassical Dirac Theory of Tunnel Ionization”. *Phys. Rev. Lett.* 89 (19): 193001. doi:10.1103/PhysRevLett.89.193001.

Mishra, Joshi, Schoenbach ja Clark. 2006. “A Fast Parallelized Computational Approach Based on Sparse LU Factorization for Predictions of Spatial and Time-Dependent Currents and Voltages in Full-Body Biomodels”. *IEEE Transactions on Plasma Science* 34, numero 4 (): 1431–1440. ISSN: 0093-3813. doi:10.1109/TPS.2006.876485.

Mocken ja Keitel. 2008. “FFT-split-operator code for solving the Dirac equation in 2+1 dimensions”. *Computer Physics Communications* 178 (11): 868–882. ISSN: 0010-4655. doi:http://dx.doi.org/10.1016/j.cpc.2008.01.042.

Momberger, Belkacem ja Sørensen. 1996. “Numerical treatment of the time-dependent Dirac equation in momentum space for atomic processes in relativistic heavy-ion collisions”. *Phys. Rev. A* 53 (3): 1605–1622. doi:10.1103/PhysRevA.53.1605.

Müller, Grün ja Scheid. 1998. “Finite element formulation of the Dirac equation and the problem of fermion doubling”. *Physics Letters A* 242 (4): 245–250. ISSN: 0375-9601. doi:http://dx.doi.org/10.1016/S0375-9601(98)00218-7.

Nakano, Kalia, Vashishta, Campbell, Ogata, Shimojo ja Saini. 2001. “Scalable Atomistic Simulation Algorithms for Materials Research”. Teoksessa *Supercomputing, ACM/IEEE 2001 Conference*, 10–10. doi:10.1145/582034.582035.

Nakatsuji ja Nakashima. 2005. “Analytically Solving the Relativistic Dirac-Coulomb Equation for Atoms and Molecules”. *Phys. Rev. Lett.* 95 (5): 050407. doi:10.1103/PhysRevLett.95.050407.

Salamin, Hu, Hatsagortsyan ja Keitel. 2006. “Relativistic high-power laser–matter interactions”. *Physics Reports* 427 (2–3): 41–155. ISSN: 0370-1573. doi:http://dx.doi.org/10.1016/j.physrep.2006.01.002.

Salomonson ja Öster. 1989. “Relativistic all-order pair functions from a discretized single-particle Dirac Hamiltonian”. *Phys. Rev. A* 40 (10): 5548–5558. doi:10.1103/PhysRevA.40.5548.

Selstø, Lindroth ja Bengtsson. 2009. “Solution of the Dirac equation for hydrogenlike systems exposed to intense electromagnetic pulses”. *Phys. Rev. A* 79 (4): 043418. doi:10.1103/PhysRevA.79.043418.

Succi. 1996. “Numerical solution of the Schrödinger equation using discrete kinetic theory”. *Phys. Rev. E* 53 (2): 1969–1975. doi:10.1103/PhysRevE.53.1969.

Sun, Lin ja Wang. 2016. “Implementing dynamical pattern recognition algorithm on computer cluster”. Teoksessa *2016 35th Chinese Control Conference (CCC)*, 5249–5254. doi:10.1109/ChiCC.2016.7554172.

Tan, Zhang, Du ja Wang. 2015. “GPU Parallel Implementation of Support Vector Machines for Hyperspectral Image Classification”. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 8, numero 10 (): 4647–4656. ISSN: 1939-1404. doi:10.1109/JSTARS.2015.2453411.