TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Hanna-Britt Reinpõld 213124IAIB

ESIMESE LAHUSTI KIHI LOOMINE SOLVENDIEFEKTI MODELLEERIMISEKS ARVUTUSKEEMIAS

Bakalaureusetöö

Juhendaja: Toomas Tamm
Ph.D

Kaasjuhendaja: Irina Osadchuk
Ph.D

Tallinn 2025

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Hanna-Britt Reinpõld

15.01.2025

Annotatsioon

Käesoleva lõputöö peamine eesmärk oli realiseerida algoritm, mis looks lahustunud molekuli süsteemi ning jätaks alles, vaid esimese lahusti kihi.

Selleks analüüsiti seitset olemasolevat programmi, mis sisaldavad lahustamise funktsionaalsust. Tulemuste põhjal disainiti uus algoritm, mis põhineb Packmoli rakendusel ning lisab sellele funktsionaalsuse, mis võimaldab luua esimese lahusti kihi. Parima lahenduse saavutamiseks kasutati Packmoli pakkimisalgoritmi, kuna selle väljundfaili töötlemine on lihtne ja arvutuslikult tõhus.

Tulemuseks oli käsurea rakendusel põhinev algoritm kuue parameetriga, mis kasutab Python 3.8 või uuemat versiooni, Packmoli rakendust ja Fortrani kompilaatorit. Algoritmi valideeriti kuue kriteeriumi põhjal. Kuna automaatset valideerimismeetodit ei eksisteeri, kasutati Avogadro tarkvara visualiseerimiseks ning tulemuste manuaalseks ülevaatamiseks. Algoritm töötas XYZ-failidega, jättes alles ainult esimese kihi, ning arvutas õigesti lõppaatomite arvu, säilitades molekulide terviklikkuse. Parima tulemuse saavutamiseks on võimalik parameetreid varieerida, arvestades modelleeritava molekuli pikima telje pikkust.

Töö oli interdistsiplinaarne, sidudes arvutiteadust, programmeerimist ja keemia valdkonda. Loodud programm täitis püstitatud eesmärgid, võimaldades luua lahusti esimese kihi.

Lõputöö on kirjutatud eesti keeles, sisaldades 38 lehekülge, 6 peatükki ja 6 joonist.

Abstract

Building the First Solvation Shell for Modeling of Solvent Effects in Computational Chemistry

The main goal of the thesis was to develop an algorithm for creating a solvated system containing the first solvation shell.

Seven programs that included the solvation functionality were analyzed. From the gathered data, a new algorithm was created that uses the Packmol program to create an initial system and remove excess molecules, leaving behind the first solvation shell.

The outcome was a command line interface application using Python 3.8 or newer version, Packmol application, and FORTRAN compiler. Algorithm was validates using six criteria, since there is no automatic tool to validate the results, the Avogadro application was used to visualize the outcome and manually check its correctness. The algorithm uses XYZ file format, creates the first solvation shell, and generates the output file correctly, calculating the number of molecules and leaving them complete. To achieve the best outcome, the user should tune the algorithm parameters according to the type of molecules being used.

The thesis was interdisciplinary combining computer science, programming, and chemistry. The created algorithm satisfied the main criteria, being able to create the first solvation shell.

The thesis is written in Estonian and is 36 pages long, including 6 chapters, and 6 figures.

Lühendite ja mõistete sõnastik

BSD	Berkeley tarkvara levitamine (*Berkeley Software Distribution*)
CRD	CHARMM koordinaadi fail (*CHARMM coordinate file*)
GUI	Graafiline kasutajaliides (*Graphical User Interface*)
MIT	Massachusettsi Tehnoloogiainstituut (*Massachusetts Institute of Technology*)
PDB	Valkude andmebaas (*Protein Data Bank*)
PSF	Valgu struktuuri fail (*Protein Structure File*)
SASA	Solvendile ligipääsetav pind (*Solvent-accessible surface area*)
VMD	Visuaalne molekulaardünaamika (*Visual Molecular Dynamics*)

Sisukord

1 Sissejuhatus **9**
1.1 Ülesandepüstitus 9
1.2 Arvutuskeemia 10
1.2.1 Lahus 10
1.2.2 Solvendiefekt 10
1.2.3 Lahusti kiht 10
1.3 Kasutatavad faili tüübid 10

2 Algoritmi kirjeldus **12**
2.1 Probleemi ülevaade 12
2.2 Vajaduste analüüs 12
2.3 Lahustamise programmid 12
2.3.1 Autosolvate 13
2.3.2 CHARMM-GUI 14
2.3.3 VMD 14
2.3.4 Octave algoritm 14
2.3.5 Amber Tools 15
2.3.6 GROMACS 15
2.4 Packmol 16

3 Algoritmi disain **17**
3.1 Tarkvararaamistikud ja teegid 18
3.2 Struktuur 19
3.2.1 Solvation_shell 19
3.2.2 Align_molecule 22
3.3 Algoritmi kasutamine 23

4 Algoritmi valideerimine ja tulemused **26**
4.1 Valideerimis meetodid 26
4.2 Tulemused 26

5 Kommentaarid ja arutelu **28**

6 Kokkuvõte **30**

Kasutatud kirjandus **31**

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks **34**

Lisa 2 – Molekulid **35**

Joonistе loetelu

1 Tsink porfüriin 36
2 Kukkurbiturill 36
3 Vesi 37
4 Kloroform 37
5 Atsetonitriil 38
6 Diklorometaan 38

Tabelite loetelu

1. Sissejuhatus

Keemilised reaktsioonid toimuvad alati kindlas keskkonnas, mitte vaakumis. Üks levinumaid keskkondi on lahus, mis võib märkimisväärselt mõjutada uuritavat süsteemi. Kasutatava lahusti valik määrab sageli reaktsiooni produktide tüübi ja koguse. Õige lahusti valik võib suurendada produkti saagist või parandada reaktsiooni selektiivsust [1]. Lahusti mõju ulatub ka spektroskoopiasse, kus lahusti võib muuta aine spektrit, põhjustades valediagnoose uuritavate ühendite tuvastamisel [2].

Ajalooliselt on arvutuskeemias uuritavate süsteemide suurus olnud piiratud. Täpsemate meetodite kasutamisel suureneb süsteemi keerukus eksponentsiaalselt ning suur mäluvajadus sunnib rakendama lihtsustusi, et arvutused oleksid teostatavad. Selle tõttu on sageli kasutatud kontinuummudeleid, mis lihtsustavad lahustatava molekuli ümbruse omadusi, käsitledes seda sujuvalt muutuvate omadustega keskkonnana. Kuigi kontinuummudelid suudavad hästi kirjeldada näiteks reaktsioonimehhanisme, ei ole need sobilikud olukordades, kus lahusti ja lahustatava molekuli vahel tekivad spetsiifilised sidemed.

Biokeemias on lahustite mõju sageli modelleeritud eksplitsiitsete lahustimudelitega, kus üksikud lahustimolekulid paigutatakse simulatsioonikasti. Alternatiivina kasutatakse ka kaudseid mudeleid, mis määravad lahusti dielektrilisi omadusi, või nende kahe meetodi hübriide, et ühendada täpsus ja arvutuslik efektiivsus. Kaasaegse arvutivõimsuse kasv viimase viie aasta jooksul võimaldab nüüd ka arvutuskeemias kasutada eksplitsiitseid lahustimudeleid, et paremini kirjeldada solvendiefekte.

Esimese lahusti kihi modelleerimine on oluline komponent lahusesüsteemide uurimisel, pakkudes põhjalikku ülevaadet lahustunud aine ja lahusti vaheliste interaktsioonide molekulaarsetest mehhanismidest. Need mudelid aitavad ületada lõhet teoreetilise keemia ja praktiliste rakenduste vahel materjaliteaduses [3], farmakoloogias [4] ja muudes valdkondades.

1.1 Ülesandepüstitus

Käesoleva töö eesmärgiks oli luua põhjalik ülevaade olemasolevatest lahusti paigutamise algoritmidest ja nende funktsionaalsustest. Saadud teadmiste põhjal täiendada olemasolevat algoritmi või arendada täiesti uus lahendus, mis suudab paigutada lahusti molekule lahustatava aine ümber, säilitades molekulide terviklikkuse ja vältides keemiliselt ebakorrektset paigutust, nagu molekulide kattumine või tühjad alad. Lisaks on oluline võimaldada

ainult esimese lahusti kihi loomist, mis aitab solvendiefekte täpsemalt modelleerida.

1.2 Arvutuskeemia

Arvutuskeemia ühendab keemia ja arvutiteaduse, pakkudes võimalust lahendada keerukaid matemaatilisi võrrandeid arvutisimulatsioonide abil. See on tänapäeval oluline tööriist keemiliste probleemide uurimiseks ja nähtuste selgitamiseks. Lisaks aitab arvutuskeemia säästa aega, vaeva ja kulusid, võimaldades keemilisi reaktsioone simuleerida enne nende läbiviimist laboris[5].

1.2.1 Lahus

Lahus on homogeenne süsteem, mis koosneb enamasti vedelas faasis olevatest ainetest. Üks nendest on lahusti ehk solvent, mis võib koosneda mitmest komponendist, ning ülejäänud on lahustunud ained. Kui lahusti ja lahustatava aine vahel tekivad sidemed, ei saa lahustit käsitleda makroskoopilise kontinuumina, vaid diskreetsete molekulide kogumina, mis omavahel vastastikku mõjuvad. Lahusti tüüp mõjutab oluliselt lahuse omadusi, nagu reaktsioonimehhanism ja spekter [6].

1.2.2 Solvendiefekt

Solvendiefekt kirjeldab, kuidas lahusti molekulid mõjutavad lahuse keemilist reaktiivsust, geomeetriat, laengut ja muid omadusi. Õige lahusti valik võimaldab keemilisi reaktsioone täpsemalt juhtida, näiteks suurendades produkti kogust või parandades lahustuvust [7].

1.2.3 Lahusti kiht

Lahusti kiht kujutab endast lahusti molekulide paigutumist lahustatava molekuli ümber. Lahustumise käigus klasterduvad solvendi molekulid lahustatava aine ümber, pakkudes kaitset teiste molekulide ja välismõjude eest. Esimene lahusti kiht asub lahustatavale ainele kõige lähemal, omades suurimat mõju selle omadustele. Kihi struktuur ja kompositsioon sõltuvad lahusti ja lahustatava aine omadustest, sealhulgas suurusest, kujust, polaarsusest ning lahuse temperatuurist ja rõhust [8] [9].

1.3 Kasutatavad faili tüübid

Faili tüüpe mida kasutatakse loodusteadustes seal hulgas keemias on mitmeid. Antud töö kontekstis on kõige relevantsemad XYZ ja PDB tüüpi failid.

XYZ fail

XYZ tüüpi faile kasutatakse molekulaargeomeetriate kirjeldamiseks. Sellel puudub kindel standard, kuid tavaliselt antakse ette esimesel real aatomite koguarv, teine rida kommentaaride jaoks ja ülejäänud read vastavalt elemendi sümbol, nagu see perioodilisusetabelis esineb, ja selle kolm koordinaati.

```
<number of atoms>
comment line
<element> <X> <Y> <Z>
...
```

XYZ fail ei sisalda sidemete informatsiooni ehk milliste aatomite vahel sidemed on. See arvutatakse põhimõttel, kui distants kahe aatomi vahel on väiksem kui nende kovalentsete raadiuste summa, siis peetakse neid omavahel ühendatud [10].

PDB fail

PDB ehk *Protein Data Bank* formaat on standartne failide jaoks, mis sisaldavad aatomite koordinaate. Seda kasutatakse *Protein Data Bank* arhiivis[1] olevates failides, sealt ka nimetus. Seal talletatakse informatsiooni valkude 3D struktuuri, nukleiinhapete ja muude kompleksete süsteemide kohta. Sellest tulenevalt kasutatakse PDB formaati peamiselt biomolekulide kirjeldamiseks ja sisaldavad palju rohkem informatsiooni molekuli kohta, võrreldes XYZ formaadiga. Seal hulgas jõuvälja tüübi nimi, ahela id, temperatuuri faktor jms [11].

Arvutuskeemia ja antud töö kontekstis, sisaldab PDB liiga palju üleliigset informatsiooni ja kohati ka välju, mida pole võimalik täita, kui tegemist pole biomolekuliga.

[1] https://www.wwpdb.org/

2. Algoritmi kirjeldus

Eesmärk oli luua ülevaade saadaval olevate programmide plussidest ja miinustest ning selle põhjal luua algoritm, mis vastaks kliendi vajadustele kõige paremini.

2.1 Probleemi ülevaade

Kliendi teadaolevad rakendused sisaldasid mitmeid probleeme. Enamik programme olid suunatud molekulaarbioloogidele ja võtsid vastu ainult PDB-failiformaate, samas kui klient töötas XYZ-failidega. Lahustina sai sageli kasutada vaid vett või molekulaarbioloogias levinud aineid, nagu etanool või kloroform. Orgaanilised keemikud vajavad aga palju mitmekesisemat lahustite valikut [12]. Lisaks on molekulaarbioloogias loodavad süsteemid tavaliselt suured, mistõttu mõne molekuli olemasolu või puudumine ei ole kriitiline. Arvutuskeemias on aga iga aatom oluline, ning süsteem peaks sisaldama piisavalt vähe aatomeid, et solvendiefekte modelleerida, samas tagades süsteemi realistlikkuse.

2.2 Vajaduste analüüs

Kliendiga kohtumisel lepiti kokku kriteeriumid, millele rakendus vastama peaks.

1. Väljatulev fail peaks olema XYZ formaadis ning rakendus, mida selle visualiseerimiseks kasutatakse, peab selle kompileerimisega hakkama saama.
2. Süsteemis olevad molekulid ei tohi üksteise sees asuda ehk nende vahel peab piisav distants olema.
3. Lahus peab säilitama kasutaja määratud tihedust ühtlaselt ehk molekulid ei tohi kontsentreeruda ühte kohta, vaid peavad jaotuma ruumis ühtlaselt.
4. Lahusti molekulid ei tohi olla paigutatud liiga korrapäraselt, moodustades kristalli struktuuri.

2.3 Lahustamise programmid

Analüüsi käigus valiti kuus programmi, mis sisaldavad lahustamise funktsionaalsust. Nende põhjal analüüsiti, kuidas iga programm vastab seatud kriteeriumidele ning kas need suudavad lahendada antud näidisprobleemi.

Näidis probleem

Keskmiseks molekuliks, mida lahustatakse valiti tsinkporfüriin, seda mitmel põhjusel. Esiteks kasutab klient seda konkreetset molekuli oma töös. Teiseks sisaldab, see tsinki. Orgaanilises keemias kasutatavad ained sisaldavad suuremat elementide variatiivsust, kus aga molekulaarbioloogias piirdutakse suuremas osas süsiniku, vesiniku, hapniku, lämmastiku, fosfori, väävli ja vähesel määral mõnede teiste elementidega [13]. Kolmandaks on konkreetne aine kahes osas - suurem porfüriini osa ja kõrval olev väike osa. Molekuli struktuuri on näha joonisel 1. Lahustiks võeti vesi, sest tegemist on lihtsa ja väikese molekuliga, mida iga programm peaks suutma lisada. Kui veega probleeme ei esine, proovitakse diklorometaani, atsetonitriili ja benseeni, mille struktuur on juba keerulisem ja aatomite arv molekulis suurem. Lisaks vaadati järgnevaid programmi omadusi:

1. Kas tegemist on vabavaralise programmiga ehk kas lähtekoodi on võimalik näha;
2. Kui tegemist on vabavaralise programmiga, millise litsentsi alla see kuulus;
3. Kas tegemis on tasuta või tasulise programmiga;
4. Millist faili formaati see võtab sisse ja millise väljastab;
5. Kas saab lisada enda valitud lahustit või peab valima etteantud ainete seast;
6. Mitu lahustit on valikus programmis endas
7. Kas saab lisada lahustite segusid ehk lahusti koosneb kahest või rohkemast ainest;
8. Millises programmeerimise keeles see on kirjutatud;
9. Kui kasutaja sõbralik programm on, kas kasutaja peab olema suur ekspert konkreetse programmi kohapealt, et seda kasutada.

2.3.1 Autosolvate

Autosolvate on vabavaraline tasuta tarkvara, mis on loob automaatselt lahustunud süsteemi ning võimaldab molekulaardünaamika ja kvantkeemia modelleerimist algse süsteemi põhjal. Tarkvara on kirjutatud kasutades Python 3.8 ja kuulub *BSD 3-Clause "New" or "Revised"* litsentsi alla[1]. See võtab sisse ja väljastab faile XYZ formaadis, programmi enda poolt on antud viis lahustit ja teoorias on võimalik lisada teisi lahusteid, segusid pole võimalik teha [14].

Algselt tundus Autosolvate kõige lubavam programm, sest see on loodud keemikute jaoks, kuid näidis probleemi lahendamisega tekkis palju probleeme. See polnud nõus vastu võtma molekuli, mis sisaldab "ebatavalist" elementi, mis selles kontekstist oli tsink. Tuli välja, et molekul peab koosnema ainult vesinikust, süsinikust, lämmastikust, hapnikust, väävlist, fosforist või hallogeenidest[2]. Peale tsingi eemaldamist molekulis tekkis järgmine probleem, molekul ei tohi olla kahes osas, vaid peab olema üks tervik. Kuna programm ei olnud

[1] https://choosealicense.com/licenses/bsd-3-clause/
[2] https://www.britannica.com/science/halogen

nõus vastu võtma test molekuli, siis polnud võimalik katsetada, kui hästi selle lahustamis funktsioon toimib.

2.3.2 CHARMM-GUI

CHARMM-GUI on Lehigh ülikooli poolt loodud veebipõhine platvorm, mis võimaldab genereerida komplekssseid süsteeme ja nende sisend faile, mida saab kasutada erinevates molekulaarsete simulatsioonide programmides, nende seas NAMD[3], AMBER[4] ja Tinker[5]. Tegemist pole vabavaralise programmiga, kuid akadeemilises kontekstis saab sellele tasuta ligipääsu. Selle jaoks peab registreerima läbi ülikooli, kus seda teadustööks kasutatakse ja isegi sellisel viisil pole võimalik ligi pääseda lähtekoodile [15].

Funktsionaalsus, mis antud töös lahendatava probleemi kasutamiseks on *Solution Builder*. See võtab sisse PDB formaadis faili ja lahustab molekuli vees. Selleks, et muud lahustit kasutada, peab looma süsteemi kasutades CHARMM-GUI *Multicomponent Assembler'it*. Selle puhul on vaja PSF ja CRD faile molekuli kohta, mis näidis molekulide puhul puudusid. Sellepärast prooviti lahustada ainult vees, kuid tsinkporfüriini PDB failis puudus *chain id* ja sellest tulenevalt polnud võimalik lahustunud süsteemi luua.

2.3.3 VMD

VMD on molekulaarse visualisatsiooni programm, mida kasutatakse suurte biomolekulaarsete süsteemide kuvamiseks, animeerimiseks ja analüüsimiseks. See on vabavaraline, kirjutatud C++ keeles ja läheb MIT litsentsi[6] alla. VMD jaoks on plugin *Solvate*, mis pakub nii graafilist liidest kui ka tekstilisi käske, mis loovad lahustunud süsteemi [16]. Selleks on vaja nii PSF kui PDB faili lahustatava molekuli kohta. Vaikiv väärtusena kasutatakse lahustina vett, kuid teoorias on võimalik kasutada ka teisi lahusteid. Selle jaoks on vaja süsteemi, mis koosneks valitud ainest, kuid kasutatav programm peaks selle ise genereerima, muutes VMD näidis probleemi lahendamiseks mitte kõlbulikuks.

2.3.4 Octave algoritm

Järgnen skript on kirjutatud Dr. Nicolas Neumani poolt, kes erialapoolest on keemik. Algoritm on kirjutatud kasutades GNU Octave keelt, erinevalt eelnevatest programmidest on tegu skriptiga, mille kood tuleb githubist kloonida ja käsitsi jooksutada [17]. Selleks, et

[3]https://www.ks.uiuc.edu/Research/namd/
[4]https://ambermd.org/
[5]https://dasher.wustl.edu/tinker/
[6]https://opensource.org/license/mit

lisada lahustatavat molekuli peab skriptis olevasse faili käsitsi kirjutama molekuli x,y,z koordinaadid ja aatomi numbrid. Failis endas on olemas kolm lahustit: vesi, atsetoon ja atsetonitriil. Kui on soov muud lahustit kasutada tuleb ka nende koordinaadid ja molekuli aatominumbrid ise käsitsi kirjutada uue muutujana.

Tegemist oli esimese programmiga, mis suutis näidis probleemist reaalse väljund faili luua. Selle suurim probleem oli, et väljund failis olevad molekulid ei paigutunud ruumis ühtlaselt ja tekkisid suured augud tühja vaakumi. Lisaks selle tuleb käsitsi arvutada mitu lahusti molekuli tuleb ette antud ruumi paigutada, et lahusti tihedust määrata. Antud programm, ei vastanud kõikidele soovitud parameetritele, kuid see sisaldas kasulikku informatsiooni, kuidas loodav algoritm saaks paremini töötada.

2.3.5 Amber Tools

AmberTools on Amberi[7] osa, mis sisaldab endas molekulaar simulatsioonide pakette. Sellele omakorda on tehtud tööriist *Solvate*, mis lisab lahusti molekule keskse molekuli ümber. Kasutada saab ainult ette antud lahusteid: kloroform, metanool, N-metüütatsetaatamiid ja vesi ehk pole võimalik kasutada ise valitud muid aineid [18].

Tervel Amber programmil on olemas kastusjuhend[8], mis on üle tuhande lehekülje pikk, aga sealt ei tulnud välja head juhendit, kuidas täpsemlat *Solvate* funktsiooni kasutada saab ning ei leitud viisi kuidas panna see lahendama näidis probleemi.

2.3.6 GROMACS

GROMACS on vabavaraline tarkvara, mida kasutatakse molekulaardünaamika simulatsioonide loomiseks. Eelkõige on see mõeldud biomolekulide nagu valkude, rasvade ja nukleiinhapete kasutuseks, kuid teaoorias töötab see ka teiste molekulide tüüpidega. Sisendiks võtab ta palju erinevaid faili tüüpe, kuid XYZ polnud üks neist.

Kahjuks näidis probleemi polnud võimalik lahendada, sest tsinkporfüriini PDB failis puudusid jõuväljad, mida GROMACS ära tunneks niiet polnud võimalik lahustunud süsteemi luua.

[7] https://ambermd.org/index.php
[8] https://ambermd.org/doc12/Amber24.pdf

2.4 Packmol

Analüüsi jooksul leiti programm Packmol[9], mille olemas olu polnud kliendile teada.

Packmol on vabavaraline Fortranis kirjutatud programm, mis loob algse süsteemi, mida saab molekulaardünaamika simulatsioonides kasutada, pakkides molekulid kokku kasutaja määratud ruumi. Kuna programm ise ei määratle ennast kui lahustamise algoritmi, siis tõenäoliselt on see ka põhjus, miks selle leidmine oli raskendatud antud töö raames.

Programmi enda poolt pole pandud piiranguid milline peaks olema lahustatav molekul või lahusti molekul. Sellega saab luua komplekseid süsteeme ja lahusti võib olla ka segu, see võtab vastu PDB, TINKER, XYZ ja MOLDY faili formaate. Selleks, et süsteemil oleks õige tihedus peab arvutama lahusti molekulide arvu ning boksi suuruse, kuhu sisse molekule paigutama hakatakse. Kasutamiseks peab kasutaja andma lahustatava molekuli ja solvendi molekuli faili ning looma Packmol'ile omase .inp faili, mis peab minimaalselt sisaldama distantsi tolerantsi, väljund faili nime, faili tüüpi ning vähemalt ühte struktuuri, mis sisaldab selle faili nime, molekulide kogust ja boksi minimaalseid ja maksimaalseid x,y,z koordinaate.

```
tolerance 2.0
output test.xyz
filetype xyz
structure water.xyz
    number 2000
    inside cube 0. 0. 0. 40.
end structure
```

Kui sellist sisendfaili jooksutada luuakse süsteem, kus 40 kuupongströmi suurusesse kuupi paigutatakse 2000 vee molekuli. Iga molekuli paari vahe on vähemalt 2.0 Å ja need on ruumis suvaliselt paigutatud [19].

Näidis probleemiga sai Packmol väga hästi hakkama, oli suuteline kasutama eri tüüpi ja suuruses molekule ning segusid. Ainuke miinus oli, et sellel puudus funktsionaalsus kuidas luua ainult esimest lahusti kihti.

Kokkuvõtvalt leiti, et igal programmil on omad plussid ja miinused ning suurimaks probleemiks ostus fakt, et need olid suunatud bioloogidele ja antud töös olevat probleemi polnud need suutelised lahendama.

[9] https://m3g.github.io/packmol/index.html

3. Algoritmi disain

Eelneva analüüsi põhjal otsustati uus algoritm luua Packmoli rakenduse baasil, lisades sellele funktsionaalsuse, mis võimaldab luua ainult esimese lahusti kihi.

Algoritmi loomiseks kaaluti viit võimalikku lahendust:

Variant A

Luua Pythoni skript, mis genereerib Packmoli sisendfaili, paigutades lahusti molekulid ellipsoidi ümber, mille sees asub lahustatav molekul.

Variant B

Kasutada Packmoli, et luua esmane süsteem, ning arendada skript, mis eemaldab väljundfailist üleliigsed molekulid, jättes alles ainult esimese kihi.

Variant C

Arvutada lahustatava molekuli pind (SASA) ning paigutada selle pinna ümber lahusti molekulid, kasutades lihtsat algoritmi.

Variant D

Kalkuleerida lahustatava molekuli pind (SASA) ning paigutada selle pinna ümber mingi naiivse meetodiga lahusti molekulid. Saadud süsteem anda Packmolis olevale optimiseerijale ette kui algolek ja vaadata, mis juhtub.

Variant E

Proovida Packmoli sees asuvat cost funktsiooni muuta selliselt, et ta pakiks lahusti molekulid ühe kihina.

Kõikidest eelnevatest variantidest valiti B, sest selle realiseerimine tundus kõige paremini sobivat antud töö probleemi lahendamiseks. Variant A ei sobinud põhjusel, et see pole piisavalt universaalne. Kui keskel olev lahustatav molekul sarnaneb ise ellipsoidi kujuga,

siis sobib see väga hästi, kuid teiste molekuli kujude puhul ellipsoid ei ümbritse seda hästi ning keskele võib jääda palju tühja ruumi ning sellist vaakumit peaks vältima. Varianti C ei valitud põhjusel, et lahustatava molekuli pinna arvutamiseks tuleks luua algoritm, mis pinna arvutaks ja looma viisi, kuidas sellele pinnale lahusti molekule paigutada. Teoorias võiks see hea lahendus olla, kuid antud töö skoopi arvestades läheks see liiga suureks ja keeruliseks ettevõtmiseks. Packmol on universaalne tööriist, mis on väga hästi loodud ja optimiseeritud. Variandid D ja E lähevad süsteemi alustaladega vastuollu ning tegemist poleks lihtsa muudatuse, vaid terve programmi ümberkirjutamisega, mis sarnaselt B variandile muutuks selle töö raames liiga mahukaks projektiks ning sellest saadav kasu poleks garanteeritud.

Variant B kasutab ära Packmoli loodud pakkimis algoritmi, mis on hästi optimiseeritud, see aitab arvutus aja pealt kokkuhoida. Väljundifail mille see loob on väga kindla struktuuriga, mis tähendab, et selle töötlemine on lihtsustatud, sest on teada kus mingi informatsioon failis täpselt asub.

3.1 Tarkvararaamistikud ja teegid

NumPy

NumPy on vabavaraline Pythoni teek, mida kasutatakse teaduslike arvutuste tegemiseks. See toetab suuri mitme mõõtmelisi massiive ja maatrikseid ning tööriistu nendega manipuleerimiseks [20]. Antud töö raames kasutatakse seda muutes XYZ failis oleva molekuli info NumPy massiiviks, mida kasutatakse arvutuste tegemisel.

Pandas

Pandas on populaarne vabavaraline Pythoni teek, mida kasutatakse andmetega manipuleerimiseks ja nende analüüsimiseks. See sisaldab andmestruktuure ja funktsioone, mis lihtsustavad struktureeritud andmetega töötamist [21]. Antud töös kasutatakse pandase *DataFrame* objekti, et molekulide andmeid manipuleerida ja töödelda.

Periodictable

Periodictable on Pythoni teek, mis sisaldab endas infot elementide massi, tiheduse ja röntgen/neutron laialilaotumise kohta [22]. Antud töös kasutatakse seda, et arvutada molekulide molaarmassi.

3.2 Struktuur

Algoritm loodi käsurea rakendusena, mis on võimeline jooksma nii Windows kui ka Unix'i tüüpi operatsioonisüsteemidel. Algoritmil on kuus muutjuat mille väärtust kasutaja saab vastavalt oma vajadustele muuta:

1. '–solute' lahustatava molekuli faili nimi
2. '-solvent' lahusti molekuli faili nimi
3. '–output' väljastatava faili nimi
4. '–shell' esimese lahusti kihi paksus
5. '–density' lahusti tihedus
6. '–tolerance' molekulide minimaalne distants teineteisest
7. '–packmol-executable' packmoli asukoht arvutis

3.2.1 Solvation_shell

Algoritmi tuum ja põhiprotsessid asuvad faili solavtion_shell.py failis. Esmalt loetakse sisse molekulide failid ning joondatakse nad z-telje järgi. Järgnevalt genereeritakse Packmoli sisendfail, selle jaoks on vaja arvutada lahustatava molekuli suurus, seda ümbritseva boksi suurus ja lahusti molekulide kogus vastavalt kasutaja määratud tihedusele.

Boksi suurused leitakse molekuli koordinaatide järgi: minimaalsed ja maksimaalsed x,y,z väärtused, lisaks leitakse lahusti molekuli kõige pikem telg. Kihi paksuseks võetakse tolerants korrutatud kahega pluss lahuti pikim telg. Saadud väärtus lahutatakse minimaalsetest koordinaatidest ja liidetakse maksimaalsetele juurde. Nii leitakse terve lahustunud süsteemi koordinaadid.

```
def calculate_solvent_box(solute, solvent, tolerance):
    ...

    shell_size = 2 * tolrance + solvent_max_lenght
    box_coordinates = [
    x_min - shell_size,
    y_min - shell_size,
    z_min - shell_size,
    x_max + shell_size,
    y_max + shell_size,
    z_max + shell_size
    ]
```

```
    return box_coordinates
```

Selleks, et lahusti molekulide kogust teada on vaja arvutada boksi maht, lahutsi molekuli molaarmass ja lahustatava molekuli molaarmass. Maht leitakse kasutades kuubi ruumala valemit:

```
def calculate_total_volume(box):
    x_min, y_min, z_min, x_max, y_max, z_max = box
    return (x_max - x_min) * (y_max - y_min) * (z_max - z_min)
```

Molaarmass on aine ühe mooli osakeste mass:

$$M = \frac{m}{n},$$

kus

M on molaarmass,

m on aine mass (kilogrammides),

n on aine hulk (moolides).

Selle arvutamiseks võeti iga aatomi mass ja liideti need kõik kokku.

```
import periodictable
    def calculate_molar_mass(molecule):
    molar_mass = 0
    elements = {e.symbol: e for e in periodictable.elements}
    for atom in molecule['atom']:
        molar_mass += elements[atom].mass
    return molar_mass
```

Terve süsteemi molaarmass saadakse liites lahusti ja lahustatava aine molekuli molaarmassid. Kasutades lihtsustust, et iga molekul võtab umbes sama palju ruumi, kui on selle molaarmass, saab välja arvutada lahusti molekulide arv. See saadakse kui korrutada omavahel süsteemi kogu maht lahusti tihedusega, lahutada sellest süsteemi molaarmass ja jagada lahusti molaarmassiga.

```
def calculate_num_solvent_molecules(solute_molecule,
solvent_molecule, density, tolerance):
```

```
    ...

    num_atoms = int((total_volume * density
        - system_molar_mass) / solvent_molar_mass)
return num_atoms
```

Peale kõikide väärtuste välja arvutamist on võimalik luua Packmoli sisend fail ja lasta sellel genereerida esmane pakitud süsteem.

```
packmol_input = f'''
tolerance {tolerance}
filetype xyz
output {packmol_output_filename}

structure {solute_filename}
  number 1
  center
  fixed 0. 0. 0. 0. 0. 0.
end structure

structure {solvent_filename}
  number {num_solvent_molecules}
  inside box {' '.join(format(x, "0.3f") for x in solvent_box)}
end structure
'''
```

Packmolist saadud faili esimesel real on kogu süsteemi aatomite arv, järgmisena tuleb lahustatava aine molekul, mis on järgmised *len(solute_molecule)* rida ja ülejäänud read on lahusti molekulid. Kusjuures iga rea peal on üks aatom enda andmetega.

Esmalt salvestatakse kahte järjendisse lahustatava aine ja lahusti molekulide andmed.

```
solute_molecule = []
    for i in range(num_solute_atoms):
    atom, x, y, z = xyz.readline().split()
    solute_molecule.append((atom, (float(x), float(y), float(z))))
```

Kuna lahusti puhul pole tegu ühe molekuli vaid mitmega, siis *solvent_molecules* sisaldab endast järjendeid, mille sees on üks lahusti molekul.

```
solvent_molecules = []
num_solvent_molecules = int((num_atoms - num_solute_atoms)
                    / num_solvent_atoms)
for j in range(num_solvent_molecules):
    current_molecule = []
    for i in range(num_solvent_atoms):
        atom, x, y, z = xyz.readline().split()
        current_molecule.append((atom,
                                 (float(x),
                                 float(y),
                                 float(z))))
    solvent_molecules.append(current_molecule)
```

Peale seda saab süsteemist eemaldada kõik lahusti molekulid, mis ei kuulu esimese kihi alla. Selle jaoks kujutatakse, et iga lahustatav aine aatomi ümber on sfäär, mille keskpunkt on aatom ja diameeter on kasutaja määratud –*shell* ehk kihi paksus. Kui üks aatom lahustatava aine molekulist jääb sfääri sisse jäetakse terve molekul alles ning see kuulub esimesse kihti. Nii käiakse terve lahustatav molekul läbi.

```
def is_point_inside_sphere(sphere_center, sphere_radius, point):
a, b, c = sphere_center
d = sphere_radius
x, y, z = point
return (x-a)**2 + (y-b)**2 + (z-c)**2 - d**2 <= 0
```

Kui esimene kiht on kätte saadud genereeritakse väljund fail, mis sisaldab uut kogu süsteemi aatomite arvu, lahustatavat molekuli ja lahusti molekule, mis kuuluvad esimesse kihti.

3.2.2 Align_molecule

See konkreetne protsess võtab sisse molekuli ning joondab selle z-telje järgi. See on vajalik selleks, et välja arvutada molekuli miinimum ja maksimum x,y,z koordinaadid ning seda ümbritseva boksi suurus.

Esmalt arvutatakse molekuli massi kese, tehes lihtsustus, et iga aatomi massi väärtus on üks. Selleks luuakse NumPy ühtetest koosnev massiiv, milles on sama palju elemente kui molekulis on aatomeid ja jagatakse läbi aatomite arvuga. Lõpuks transponeeritakse maatriks, mis sisaldab endas molekuli x,y,z koordinaate ja korrutatakse eelnevalt saadud

massiiviga läbi [23].

```
import numpy as np

    def compute_center_of_mass(points, masses=None):
    if masses is None:
        masses = np.ones(points.shape[0])
        masses /= masses.sum()

    return points.T.dot(masses)
```

Järgnevalt arvutatakse inertsimomendi tensor. Kui tegemist pole iga telje suhtes sümmeetrilise objektiga, siis igal teljel on oma inertsimoment. Tensor on viis kuidas seda iseloomustada ühe objektina, kuna hetkel on tegu kolmemõõtmelise molekuliga, siis tensor on 3x3 maatriks [24].

$$\mathbf{tensor} = \begin{bmatrix} I_{11} & I_{12} & I_{13} \\ I_{21} & I_{22} & I_{23} \\ I_{31} & I_{32} & I_{33} \end{bmatrix}.$$

Selleks arvutatakse skalaarne 'A', mis saadakse kui summerida kokku kõik molekulide vastavate koordinaatide ruudud. Järgnevalt leitaks maatriks 'B', mis representeerib duplikeeritud molekuli koordinatide skalaarkorrutist. 'A' korrutatakse läbi 3x3 ühik maatriksiga ja lahutatakse 'B' [23].

$$\mathbf{tensor} = \mathbf{A} * \mathbf{I} - \mathbf{B}$$

Peale tensori leidmist saab arvutada peainertsimomendi, mis on siis kõige pikem telg, mis objekti läbib ehk selleks, et molekul mööda seda telge pöörata on vaja kulutada kõige rohkem energiat [25]. Selleks võetakse tensor ning leitakse selle omaväärtus ja omavektor

Viimaks roteeritakse peainertsmoment z-telje järgi [26].

3.3 Algoritmi kasutamine

Selleks, et programmi kasutada on vaja vähemalt Python 3.8 versiooni, Packmoli rakendust ja Fortrani kompilaatorit, mis seda jooksutaks.

Esmalt peaks kontrollima, kas antud arvutis on Fortrani kompilaator olemas, macOS süsteemil saab seda teha jooksutades terminalis käsku:

```
which gfortran
```

Kui GNU Fortran puudub ja arvutis on olemas Homebrew[1] saab selle alla tõmmata järgneva käsuga:

```
brew install gcc
```

Juhised kuidas teiste operatsioonisüsteemide või rakendustega seda teha saab leiab GNU Fortrani kodulehelt[2].

Packmoli saab alla laadida aadressilt https://github.com/m3g/packmol/releases/latest. Kui packmol-20.13.0.tar.gz on alla laetud tuleb see lahti pakkida ja kompileerida, seda saab teha järgneva käsuga:

```
tar -xvzf packmol-20.13.0.tar.gz
```

See loob kausta nimega packmol, mille sees on lähtekood ja kompileerimiseks jooksuta käsku:

```
cd packmol
make
```

Töös loodud algoritmi kasutamiseks tuleb see kloonida gitlab'i repost või allatõmmata zip fail lähtekoodiga:

```
git clone https://gitlab.cs.taltech.ee/hreinp/thesis.git
```

Järgnevalt liikuda kausta kus asub projekt:

```
cd [project folder]
```

Selleks, et algoritm saaks ligi molekuli failidele, mida kasutada soovitakse peaksid need olema samas kaustas kui projekt või jooksutamise käsul lisada faili täielik asukoht arvutis.

Viimaks saab programmi panna tööle käsuga:

```
python3 solvation_shell.py --packmol-executable
[packmol folder address]/packmol
```

[1] https://brew.sh/
[2] https://fortran-lang.org/learn/os_setup/install_gfortran

Eelnev käsk on miinimum, et programm tööle läheks ning kõik muutujad nagu lahusti molekul, lahustatav molekul, tihedus jms jäävad vaikeväärtustena, mis algoritmi sees on.

4. Algoritmi valideerimine ja tulemused

4.1 Valideerimis meetodid

Algoritmi valideerimiseks pole ühte parimat viisi, vaid analüüsima peab mitut eri-parameetrit. Selle töö kontekstis vaadati algseid kriteeriumeid algoritmile, töö eesmärke ja kliendi poolset tagasisidet.

Töö alguses pandi kuus kriteeriumit, millele algoritm vastama peaks:

1. Töö käib XYZ formaadis ja tulemust on võimalik korrektselt hiljem visulaiseerida.
2. Molekulid ei tohi teineteise sees asuda.
3. Peab säilima kasutaja poolt määratud tihedus.
4. Lahusti molekulid ei tohi liiga korrapäraselt paigutuda.
5. Lahusti võib koosneda ühest või enamast ainest.
6. Võimalus luua ainult lahusti esimene kiht.

Lisaks luua põhjalik kirjandus ülevaade saadaval olevatest programmidest, mis lubavad sama probleemi lahendada.

Selleks, et esimest nelja kriteeriumi kontrollida anti kliendi poolt eri tüüpi lahuseid, mida luua. Kuna puudub otsene programm, mis seda automaatselt kontrolliks, tehti kõik süsteemid ja kasutades Avogadrot[1], et need visualiseerida ning manuaalselt ülevaadata. Valideerimiseks võeti järgnevad lahustatavad molekulid: tsinkporfüriin ja selle erinevad vormid, kukurbituriil, erinevad paindlikud keerulise struktuuriga molekulid. Lahustiks võeti vesi, kloroform, atsetonitriil ja diklorometaan.

Ainuke kriteerium, mida ei jõutud implementeerida oli viies ehk lahusti võib koosneda ühest või enamast ainest.

4.2 Tulemused

Algoritm on selliselt loodud, et see töötab ainult XYZ failidega ja jätab alles esimese kihi.

Katsetati läbi kõik võimalikud kombinatsioonid lahustatava molekuli ja lahustiga. Igat

[1] https://avogadro.cc/

saadud faili sai visualiseerida Avogadroga ilma probleemiteta ehk algoritm arvutas õigesti lõpp aatomite arvu ning molekulid jäid tervikuks. Esimese kihi loomise protsessi jooksul eemaldati molekule korrektselt, võeti tervik molekul ja suvalisi aatomeid ei jäänud maha. Ükski molekul ei asunud teise sees, vaid nende vahel oli piisav vahe. Selle eest, et lahusti molekulid liiga korrapäraselt ei paigutu vastutab Packmol. Algoritm, mille järgi ta neid paigutab sisaldab endas juhuslikkuse elementi, mis tagab molekulide nii öelda suvalise paigutamise.

Selleks, et saaks parima esimese kihi peab algoritmi parameetritega mängima. Vaikeväärtusena on kihi paksus 2.5 Å, kuid see ei pruugi olla piisav iga lahusti molekuli kohta. Algoritmi jooksmisel kuvatakse käsureale ka lahusti molekuli pikima telje pikkus, seda tuleks arvesse võtta ja vastavalt kihi paksust muuta.

5. Kommentaarid ja arutelu

Käesolev töö näitab, kuidas interdistsiplinaarsed lähenemised ja arvutiteaduse rakendused suudavad lahendada keerulisi probleeme keemia valdkonnas. Algoritmi loomisel olid peamisteks väljakutseteks lahustit ja lahustatavat molekuli puudutavad keemilised spetsifikad ning vajadus tagada arvutuslik efektiivsus.

Õppetunnid ja väljakutsed

Projekti jooksul selgus, et olemasolevad lahustamise algoritmid olid peamiselt suunatud molekulaarbioloogidele, mistõttu need ei vastanud orgaanilise keemia vajadustele. Näiteks PDB-formaadi nõue välistas suure hulga lahustatavate molekulide kasutamise. See rõhutas vajadust lahenduse järele, mis suudaks töötada universaalsemate failiformaatidega, nagu XYZ, ja aktsepteerida keerukamaid molekulistruktuure.

Valitud lahenduseks kujunes Packmoli kasutamine, kuna see pakkus paindlikkust ja efektiivsust erinevate süsteemide loomiseks. Kuigi Packmol ise ei toeta esimese lahusti kihi loomist, võimaldas selle väljundfaili struktureeritud vorm algoritmi edasist töötlemist. Packmoli lisamine suurendas algoritmi kasutusvõimalusi ning vähendas arenduse keerukust.

Arenduse tugevused

1. **Universaalsus ja paindlikkus:** Algoritm suudab töödelda erinevat tüüpi lahustatavaid molekule ja lahusteid, pakkudes laia kasutusvõimaluste spektrit.
2. **Täpne modelleerimine:** Algoritm eemaldab süsteemist üleliigsed molekulid, säilitades ainult esimese lahusti kihi, mis võimaldab täpsemat solvendiefekti modelleerimist.
3. **Kasutajakesksus:** Kasutajale pakutakse võimalust kohandada algoritmi parameetreid vastavalt konkreetsetele vajadustele, näiteks määrata lahusti tihedus ja kihi paksus.

Parandamist vajavad aspektid

Projekti käigus ilmnesid mõned piirangud:

1. **Mitme lahusti kasutamine:** Praegune algoritm toetab ainult üht lahustit, mistõttu pole võimalik luua segusid. Lahusti segude modelleerimine oleks aga paljudes rakendustes vajalik ja suurendaks algoritmi paindlikkust.
2. **Automaatne valideerimine:** Hetkel tugineb algoritmi tulemuste kontroll manuaalsele ülevaatusele. Automaatse valideerimistööriista loomine parandaks protsessi efektiivsust ja täpsust.

Võimalused ja tulevikusuunad

Töö edukas lõpuleviimine näitab, et loodud algoritmil on potentsiaali arvutuskeemia valdkonnas oluliseks tööriistaks kujuneda. Edasiste täiustuste hulka kuuluvad järgmised sammud:

1. **Lahusti segude lisamine:** Algoritmi arendamisel lisatakse funktsionaalsus, mis võimaldab kasutada mitut lahustit korraga. See muudab algoritmi veelgi paindlikumaks ja praktilisemaks.
2. **Integreerimine olemasolevate töövoogudega:** Algoritmi kasutajate hulga suurendamiseks võib seda integreerida teiste arvutuskeemia tööriistade ja tarkvaradega.
3. **Automatiseeritud valideerimine:** Spetsiaalse valideerimistööriista arendamine aitaks parandada algoritmi kasutusmugavust ja vähendada tulemuste kontrollimiseks kuluvat aega.

Kokkuvõte

Töö tulemusena loodi toimiv lahendus, mis ühendab arvutiteaduse ja keemia, pakkudes täpse ja tõhusa viisi lahusti esimese kihi modelleerimiseks. Algoritmi edasiarendus koos kliendi tagasisidega võimaldab tulevikus laiendada selle funktsionaalsust ning rakendatavust.

6. Kokkuvõte

Käesolev töö keskendus algoritmi loomisele, mis võimaldab luua lahustatava molekuli ümber ainult esimese lahusti kihi. Algoritmi väljatöötamise aluseks oli vajadus täpsema solvendiefekti modelleerimise järele, vähendades samal ajal süsteemi suurust ja arvutusressursside vajadust.

Algoritm kasutab Packmoli rakendust, mida täiendati spetsiaalse funktsiooniga üleliigsete molekulide eemaldamiseks. Tulemusena loodi süsteem, mis vastab seatud eesmärkidele: see töötab XYZ-failidega, tagab molekulide õige paigutuse ja säilitab kasutaja määratud tiheduse. Töö käigus katsetati algoritmi erinevate lahustatavate molekulide ja lahustitega, kinnitades selle usaldusväärsust.

Lõputöö tulemusena saavutati järgmised peamised eesmärgid:

1. Algoritm on võimeline looma lahusti esimese kihi.
2. Töö pakub uudse lahenduse arvutuskeemia rakendustele.
3. Projekti raames arendatud lahendus loob aluse edasisteks täiustusteks, sealhulgas mitme lahusti segude ja automaatse valideerimise funktsioonide lisamiseks.

Töö tõestas interdistsiplinaarse lähenemise väärtust, ühendades arvutiteaduse ja keemia eesmärgipäraseks ja tõhusaks lahenduseks. Algoritm täidab olulist lünka olemasolevates tööriistades ning selle edasiarendus avab võimalusi veelgi laiemaks rakendatavuseks.

Kasutatud kirjandus

[1] Artur Noole et al. “Enantioselective Henry Reaction Catalyzed by CuII Salt and Bipiperidine”. In: *The Journal of Organic Chemistry* 75.4 (2010). PMID: 20095538, pp. 1313–1316. DOI: 10.1021/jo902664v. eprint: https://doi.org/10.1021/jo902664v. URL: https://doi.org/10.1021/jo902664v.

[2] Benedetta Mennucci et al. “Modeling solvent effects on chiroptical properties”. In: *Chirality* 23.9 (2011), pp. 717–729. DOI: https://doi.org/10.1002/chir.20984. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/chir.20984. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/chir.20984.

[3] E. F. da Silva, H. Svendsen, and K. Merz. “Explicitly representing the solvation shell in continuum solvent calculations.” In: *The journal of physical chemistry. A* 113 22 (2009), pp. 6404–9. DOI: 10.1021/jp809712y.

[4] A. Lomize, I. Pogozheva, and H. Mosberg. “Anisotropic Solvent Model of the Lipid Bilayer. 1. Parameterization of Long-Range Electrostatics and First Solvation Shell Effects”. In: *Journal of chemical information and modeling* 51 4 (2011), pp. 918–29. DOI: 10.1021/ci2000192.

[5] Hassan H Abdallah. “Computation chemistry as a tool for academic and industrial chemists”. In: (2016). DOI: 10.4172/2469-9764.C1.003.

[6] Christian Reichardt. “Solvents and Solvent Effects in Organic Chemistry”. In: WILEY-VCH Verlag GmbH Co. KGaA, Weinheim, 2003. Chap. 2.1, pp. 5–10.

[7] G. Balakrishnan et al. “Understanding solvent effects on structure and reactivity of organic intermediates: a Raman study”. In: *Faraday Discussions* 145 (2010), pp. 443–466. DOI: 10.1039/B908146A.

[8] Christopher J. Cramer and Donald G. Truhlar. “Density functional theory for transition metals and transition metal chemistry”. In: *Phys. Chem. Chem. Phys.* 11 (46 2009), pp. 10757–10816. DOI: 10.1039/B907148B. URL: http://dx.doi.org/10.1039/B907148B.

[9] Ashok K Adya, Oleg N Kalugin, and W Spencer Howells. “Dynamics and structure of nickel chloride–methanol solutions: quasi-elastic neutron scattering and molecular dynamics simulations”. In: *Journal of Physics: Condensed Matter* 19.41 (Sept. 2007), p. 415120. DOI: 10.1088/0953-8984/19/41/415120. URL: https://dx.doi.org/10.1088/0953-8984/19/41/415120.

[10] Jesus M. Castagnetto M. *CCL: SUMMARY: Some Molecular File Format Descriptions*. Accessed: 02-05-2024. URL: http://www.ccl.net/chemistry/resources/messages/1996/10/21.005-dir/index.html.

[11] *PDB File Format*. Accessed: 02-05-2024. URL: https://www.biostat.jhsph.edu/~iruczins/teaching/260.655/links/pdbformat.pdf.

[12] *Expanding the limits of computational chemistry*. Accessed: 01-05-2024. URL: https://gaussian.com/scrf/?tabid=7.

[13] Charles Molnar and Jane Gair. *Concepts of Biology - 1st Canadian Edition*. pp. 59. BCcampus, 2015. ISBN: 978-1-989623-99-2. URL: https://collection.bccampus.ca/textbooks/concepts-of-biology-1st-canadian-edition-bccampus-87/.

[14] Eugen Hruska et al. "AutoSolvate: A toolkit for automating quantum chemistry design and discovery of solvated molecules". In: *The Journal of Chemical Physics* 156.12 (2022). DOI: https://doi.org/10.1063/5.0084833. URL: https://pubs.aip.org/aip/jcp/article/156/12/124801/2841110/AutoSolvate-A-toolkit-for-automating-quantum.

[15] Sunhwan Jo et al. "CHARMM-GUI: A web-based graphical user interface for CHARMM". In: *Journal of Computational Chemistry* 29.11 (2008). DOI: https://doi.org/10.1002/jcc.20945. URL: https://onlinelibrary.wiley.com/doi/10.1002/jcc.20945/abstract;jsessionid=C59A89FADCB9EBCB5B764444E76AF6E6.f03t02.

[16] *Visual Molecular Dynamics*. Accessed: 02-05-2024. URL: https://www.ks.uiuc.edu/Research/vmd/plugins/solvate/.

[17] Dr. Nicolas Neuman. *solvateMolecule*. Accessed: 02-05-2024. URL: https://github.com/niconeuman/octaveScripts/tree/main/solvateMolecule.

[18] UCSF Computer Graphics Laboratory. *Chimera User's Guide*. Accessed: 02-05-2024. URL: https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/midas/solvate.html.

[19] *PACKMOL: Initial configurations for Molecular Dynamics Simulations by packing optimization*. Accessed: 03-05-2024. URL: https://m3g.github.io/packmol/userguide.shtml.

[20] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[21] Wes McKinney et al. “Data structures for statistical computing in python”. In: *Proceedings of the 9th Python in Science Conference*. Vol. 445. Austin, TX. 2010, pp. 51–56.

[22] Paul Kienzele. *Extensible Periodic Table*. Accessed: 04-05-2024. URL: https://periodictable.readthedocs.io/en/latest/#.

[23] Lewis J. Martin. *aligning principal moments of inertia*. Accessed: 04-05-2024. URL: https://github.com/ljmartin/align/blob/main/0.2%20aligning%20principal%20moments%20of%20inertia.ipynb.

[24] M. Beatty. “The Moment of Inertia Tensor”. In: (2006), pp. 355–404. DOI: 10.1007/978-0-387-31255-2_5.

[25] D. Rowe. “The moment of inertia”. In: (2010), pp. 119–129. DOI: 10.1142/9789812790668_0008.

[26] Alan Morningstar. *rotation*. Accessed: 04-05-2024. URL: https://gist.github.com/aormorningstar/3e5dda91f155d7919ef6256cb057ceee?fbclid=IwAR12CR4r0JVafxwnxYg5mDydEdmRSBX4WTdBlYLj9ranp34bM-urMKg6pnc.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Hanna-Britt Reinpõld

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Esimese lahusti kihi loomine solvendiefekti modelleerimiseks arvutuskeemias", mille juhendaja on Toomas Tamm and Irina Osadchuk
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

15.01.2025

[1] Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 - Molekulid

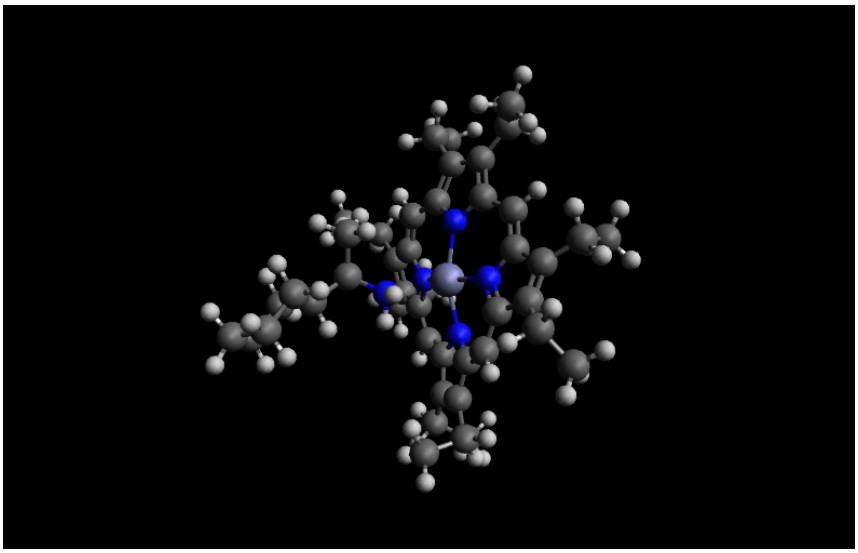

Figure 1. Tsink porfüriin

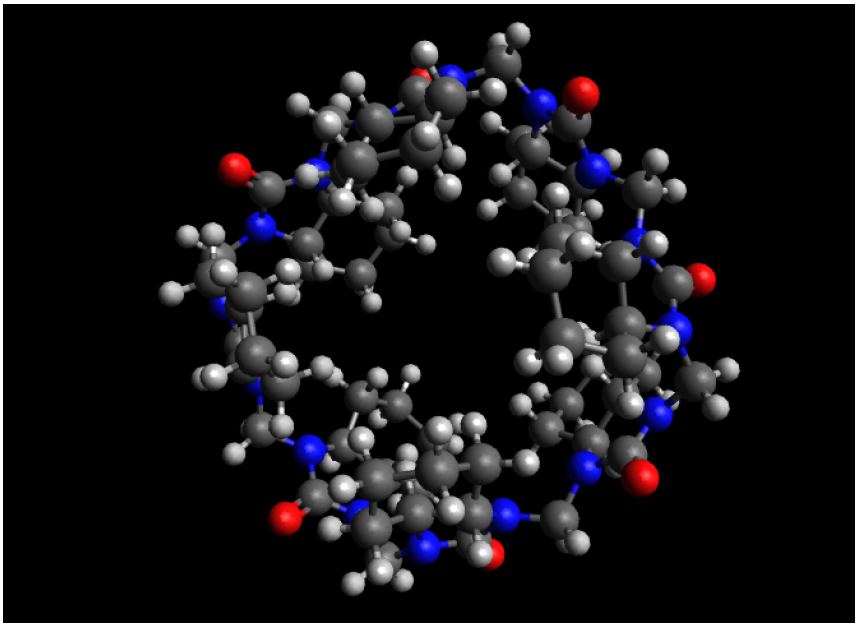

Figure 2. Kukkurbiturill

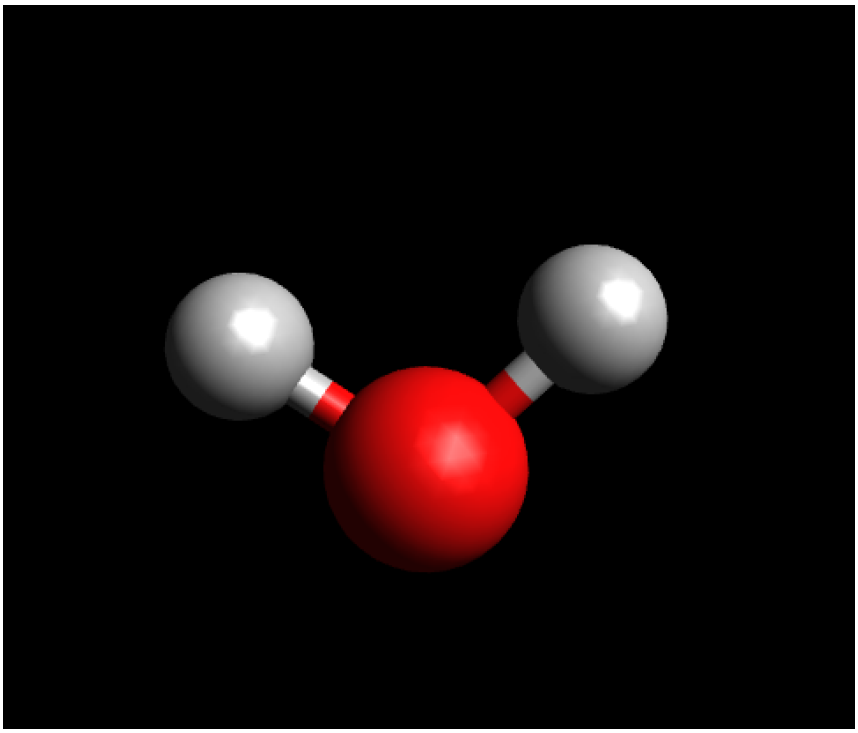

Figure 3. Vesi

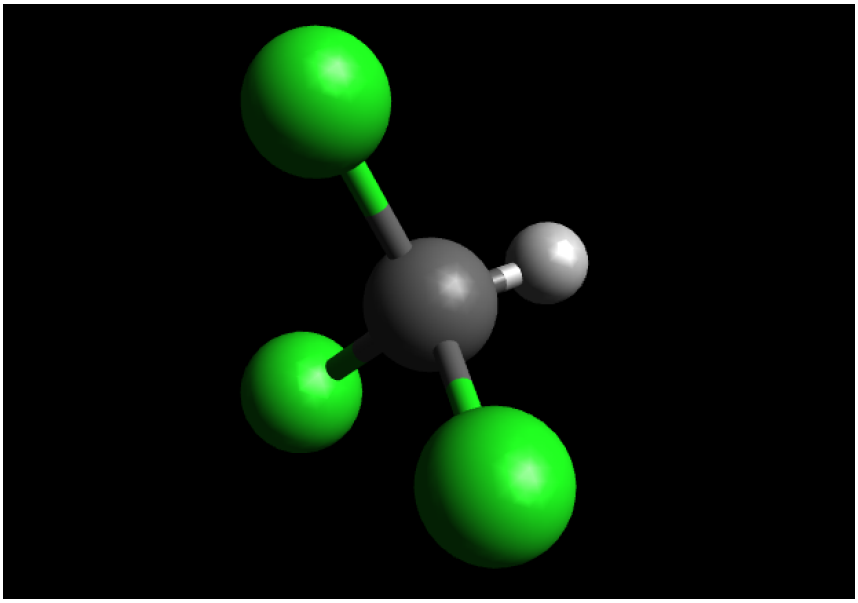

Figure 4. Kloroform

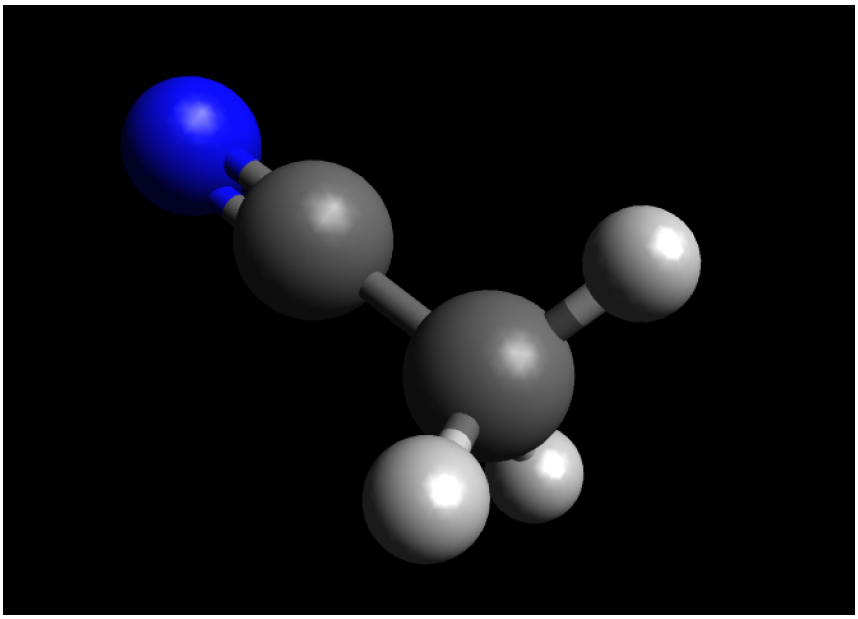

Figure 5. Atsetonitriil

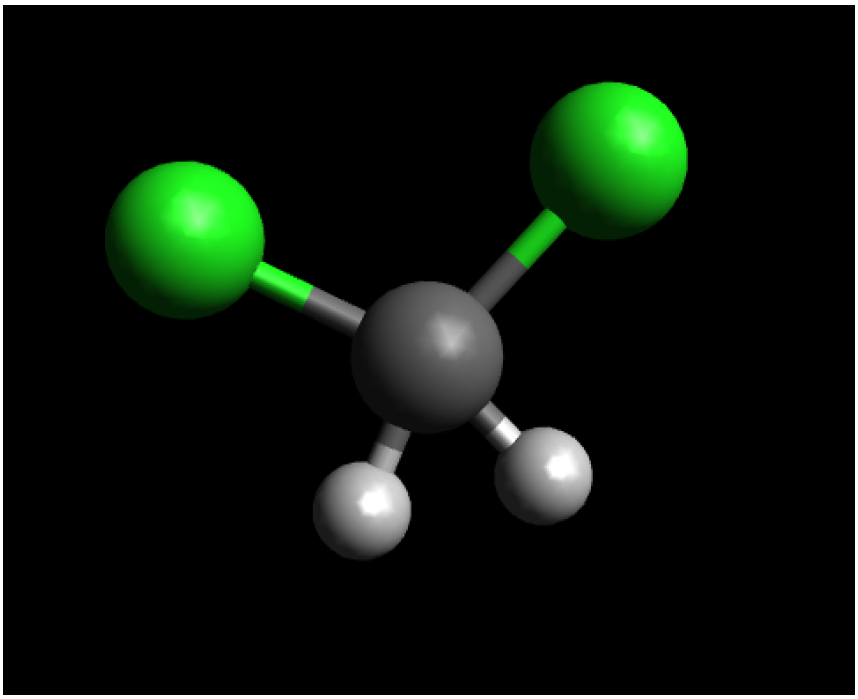

Figure 6. Diklorometaan