Development of molecular dynamics simulation software

Group A

July 26, 2021

Version 0.1

Status

Reviewed		
Approved		

Project Identity

Orderer: Rickard Armiento, Linköping University

E-mail: rickard.armiento@liu.se

Supervisor: Abijith S Parackal

E-mail: abijith.s.parackal@liu.se

Project members

Name	**Responsibility**	**E-mail**
Nicholas Sepp Löfgren (NSL)	Project leader	nicse725@student.liu.se
Emil Jivtegen (EJ)	Documentation	emiji263@student.liu.se
Linda Le (LL)	Scrum master	linle336@student.liu.se
William Stenlund (WS)	Backlogging	wilst106@student.liu.se
Roger Önçü (RÖ)	Nutrition/Group dynamics	rogon956@student.liu.se
Petter Cyrén (PC)	Group dynamics/Nutrition	petcy342@student.liu.se

Abstract

This document concludes the computational physics CDIO project of the course TFYA92 given at Linköping University autumn 2020. A software for molecular dynamics simulations has been developed by a project group of six members, all of whom are students finishing their respective master in physics. With the increase of computational power hypothetical materials can be tested through molecular dynamics simulations. Tests like this are useful in a variety of scientific fields and can help develop future necessities. This report will give an insight in the work of the project group, how it all developed and what results were reached. All produced code is available at the Github repository for the project and results from simulations made by the project group are available at an OPTIMADE database. A molecular dynamics simulation program capable of simulations on orthorhombic, monoatomic and diatomic materials has been developed. It can calculate properties such as energy, lattice constant, self diffusion and more.

Contents

Document history

Version	Date	Changes	Made by	Reviewed
0.1	2021-01-13	Complete project documentation.	EJ	EJ

1. Introduction

To make new and innovative technologies viable, design of materials with desired properties is crucial. It is a typical challenge to to find materials that fulfill requirements on cost, environmental impact, length of life and safety among other properties. A tool used for this challenge is theoretical investigation of hypothetical materials and system configurations by computational materials simulations. An understanding of the physics of materials can be reached through these simulations and continued progression of computational power, storage capability and improved methodology makes it more available. Molecular dynamics is such a simulation that has been utilized where one can extract valuable information about materials behavior to unlock new, desirable materials.

As a part of the master in theoretical physics in the applied physics and electrical engineering program at Linköping University we in this project group has taken a course in CDIO. Throughout this report the project group and molecular dynamics will be addressed as PG and MD, respectively.

All products and documents in this project are free to use according to the MIT open license specified on the projects Github repository[1].

1.1. Problem formulation

The PG shall contribute to the development of materials design via high-throughput materials simulations and materials databases. This is to be achieved by creating software for molecular dynamics simulations, operating the said software to determine materials properties on a large-scale materials database, analyze and visualize results to draw conclusions and make the results available online, see appendix A.

2. Knowledge basis

To understand and be able to complete this project an extended repertoire of knowledge is required. Below is described the project specific knowledge the PG acquired for this project. Added to this is also the aforegoing four years of studying applied physics and electrical engineering with focus on the physics.

1 https://github.com/obsqyr/TFYA92-group-A/blob/master/LICENSE

2.1. LIPs-Scrum

LIPs-Scrum is a kind of Agile project model. It is a mixture of the LIPS model and Scrum, in the way that there are sprints and stand ups from Scrum and decision points from LIPS [1], [2]. All documentation written during the project follows the LIPS model. The stand ups are short meetings several times a week where each group member informs the rest on what he/she has done and will do, if there were any problems and if help is needed. In the sprint planning, the PG schedules the upcoming period of time (sprint) and which tasks are supposed to be implemented. All tasks in a sprint are given an estimated number of hours through a voting in the group. When a sprint is finished a review of it is made: what was good, can something be improved to next sprint and so on. In between these sprints, there are decision point meetings according to LIPS. At these meetings is decided if the PG should continue with the project, if some major changes has to be done and how the delivery of the product will be handled.

2.2. Lectures

In order to get an understanding of the project and its contents, the PG had a number of lectures given by the orderer. These were given in the preparation phase prior to the start of the project. The content of these lectures ranged from programming to physics to security and history. Much of this content was just for a general understanding but the main purpose was to guide the PG in the project.

2.3. Seminars

Four different hands-on exercises were implemented in the beginning of the project in order to give the PG an understanding in Git, programming techniques, MD simulations and supercomputer handling. These were used as an example on what the upcoming project would be all about. Github was used for version control and cloud storage of the software. One hands-on exercise showed examples of how to create a repository and make pull requests, merging branches and other important commands specific for Git. Another hands-on exercise gave the PG an opportunity to handle the supercomputers at NSC, explaining how to add programs to the supercomputer queue and how to enable parallel calculations.

2.4. Theory

MD is a simulation of movements of atoms. The implementation of MD varies but the main idea is to calculate the energy and forces corresponding to the simulated system. This is done by using Newton's equations of motion. The calculations are repeated for each desirable time step and together they show how the atoms interact with each other and what forces are acting on them. A potential is used in order to calculate the forces. The most common potential, which happens to be the one used for this project, is the Lennard-Jones potential. How the movement of atoms is made also depends on the ensemble of the system. Different properties are kept constant in simulations with different ensembles and can therefore yield different results. It is common to approximate a system to be infinitely large in relation to a single atom. This method uses a small unit cell with its boundaries being a copy of it self.

3. Implementation

The implementation of this product was divided in to three phases: preparation, execution and end game. The tasks and workload for each phase differed quite much but each phase is equally important. For more details about the layout of the project, see the technical documentation D. We communicated within the group through Discord and with the examiner via Microsoft Teams. This was necessary because of Linköping University being in distance mode for all of its education due to the Covid-19 pandemic.

3.1. Preparation phase

The preparation phase starts by putting together a project group, followed by appointing responsibilities among the members. All hands-on exercises and lectures were placed in the preparation phase to help us get started. Along with this is also the making of the specification of requirements (see appendix B). Together with the project backlog on Github, this is where the project is planned out and all prerequisites made clear. When the examiner has approved the specification of requirements the project plan (appendix C) is written and the execution phase begins.

3.2. Execution phase

The execution phase contains the bulk work of the project. It is divided into smaller development phases with separating milestones as is custom in LIPS. Each phase coincides with the sprints. Every sprint started with a planning meeting and ended with a review. The pervading theme is software writing, such as functionalities and tests. To keep track of all written code and to make it available to each member, Github was used extensively. Two smaller meetings each week were held to update each member of the current progress as specified by Scrum. Some problems submerged during the execution phase. Via the remote client Thinlinc there could occur an error when trying to run the simulation. Some additional problems and with the time running out the PG renegotiated some requirements with the examiner.

3.3. End game

The end game concludes the project, here the product is being finished and the final presentation is prepared and given. All documents are compiled and handed over to the orderer. A formal delivery will take place according to LIPS and the examiner approves (or fails) the project. Lastly, the PG dissolves and the project is over.

4. Technical description

The MD implementation formed the core of the software. Calculations were made in a separate script with its content being the majority of the functionality of a MD simulation. In a settings file (see section 4.2) parameters can be calibrated to get different and desired results. Potentials and integrators are extracted from online libraries and the most fitting integrator for the simulations ensemble are chosen by the code. For more information, see appendix D.

4.1. Software

It is necessary to have an implementation of time convergence since the MD simulations are constituted by a certain time interval. Functions to check if thermal equilibrium has been reached is implemented in *md.py* along with functionalities and calculations for a single simulation and there are several hyperparameters to consider for this in a settings file. The main program *main.py* handles running MD over a larger set of materials. Different sets of materials to run a simulation on can be specified by a Materials Project json-file in the settings file [3]. The main program will unpack defined materials and put them in a list of atoms objects. The main program then simply run a simulation for each material in the list. This has the ability to be done in parallel to speed up the simulation.

4.1.1. Libraries

Package managers and installers, such as Anaconda, were used to gain access to libraries and dependencies needed for this project [4]. Relevant packages can be found in the technical documentation and users guide (E). Some of the most prominent dependencies are asap and ASE which contain the majority of the simulation software [5], [6]. The results of this project are uploaded to a database by using MongoDB. The uploaded materials has a certain predefined structure for the presentation of properties. The entries specific for this project all start with *_groupa_*. The description of these can be found in appendix D.

4.2. Input and output

In *settings.json* the user can define the desired parameters needed for a simulation. The user can for example choose which ensemble (and consequently which integrator) will be used for the simulations, which materials they wish to simulate, at what temperature the simulations should be done in and more. How to manage this is written in appendix E.

After a simulation the results are presented in a table showcasing interesting properties. Plots showing the correlation between different properties are created and can be found in corresponding folder in the repository. MD simulations are performance demanding and can take a long time to finish. However, the results can be worth while. If an animation is wanted on each system structure it can be done.

4.3. Supercomputer

Throughout this project the PG has had access to the supercomputers at NSC at Linköping University, specifically Sigma. Running MD simulations for many materials requires the power of supercomputers, as the time required to run the simulations would be too large to reasonably run on personal computers. The strength of running simulations on Sigma is the possibility to run several simulations in parallel. The large list of atoms objects is divided and distributed among computing nodes.

5. Result and discussion

A MD program for orthorombic materials has been developed and tested against values for Argon. The tests yielded sufficiently good comparisons to consider the project finished. For more detail on this, see appendix D.

References

[1] Christian Kryssander and Tomas Svensson. *Projektmodellen LIPS*. Studentlitteratur AB, Lund, 1:1 edition, 2011. ISBN 978-91-44-07525-9.

[2] Jeff Sutherland and Ken Schwaber. The 2020 scrum guide. https://www.scrumguides.org/scrum-guide.html, 2020. Visited: 2020-12-07.

[3] Materials project. The Materials Project. [Online]. Tillgänglig: https://next-gen.materialsproject.org/. [Visited: 20201211].

[4] Anaconda Inc. Individual edition. [Online]. Tillgänglig: https://www.anaconda.com/products/individual. [Visited: 20201223].

[5] University Corporation for Atmospheric Research. ASAP tools. [Online]. Tillgänglig: https://asappytools.readthedocs.io/en/latest/. [Visited: 20201211].

[6] ASE-developers. Atomic Simulation Environment. [Online]. Tillgänglig: https://wiki.fysik.dtu.dk/ase/. [Visited: 20201211].

A. Project directive

Project directive

Computational physics project

Version: 1.0
Date: September 1, 2020

Contents

1 Involved parties

- Customer and expert: Rickard Armiento
- Supervisor: Abhijith S Parackal
- The project team

2 Background

The design of new materials with desired properties is a crucial step in making innovative technologies viable. A typical challenge in materials design is to find materials that fulfill specific requirements on efficiency, cost, environmental impact, length of life, safety, and other properties. Theoretical investigation of hypothetical material candidates and system configurations by computational materials simulations have become a standard tool in this field. These simulations help us understand the physics of materials and systems of molecules and can be used to predict many different materials properties. The continued progression of computational power, storage capability, and improved methodology, has led to the application of these methods in high-throughput to produce large, openly available, databases of theoretically predicted materials properties that are a major asset for addressing materials design challenges.

3 Project idea

The project team will participate in the development of materials design via high-throughput materials simulations and materials databases by creating software for molecular dynamics simulations, operate this software to determine materials properties on a large-scale materials database, analyze and visualize the results to uncover any interesting conclusions, make these results available online by participating in a network of materials property databases, and report the outcome to the customer.

4 Purpose

The purpose of the project is to use modern software engineering practices for conceiving, designing, and implementing computational physics software and to operate this software in a study aligned with present trends in database-driven materials design. The project is meant to keep a high scientific and technical standard, providing documentation in the form of relevant reports and a final presentation.

5 Goals

- Design, implement and operate a Molecular Dynamics (MD) program. This will require the team members to get insight into the inner workings of an MD code, how to interpret results obtained with the program, and assert the quality of these results. The work includes the correct selection and writing of code subroutines, debugging, testing, and operation of the program.
- Extract an extensive set of hypothetical materials using one of the large online materials databases available.
- Apply high-throughput methodology to run the MD code on supercomputers to build a database of relevant materials properties for the materials in the extracted dataset.
- Analyze, datamine, and visualize the results, e.g., using property scatter-plots to look at interesting general correlations and outliers.
- Make the results available via the OPTIMADE network of materials property databases.

6 Effect goals

- Advance currently available data for, and understanding of, materials via custom-made materials simulations based on molecular dynamics, making the resulting data available as open data.

7 Description of the project

7.1 Available resources

- Scientific software
  - The idea is to build the MD program using the ASAP and ASE software libraries, which should be helpful for this implementation. It is also recommended to use the OpenKIM support in these libraries for access to interesting interatomic potentials.
  - Python, ASAP and ASE are free software that can be downloaded and installed on any computer. A recommended way to run these is via the Anaconda Python data science platform.
  - For information about the OPTIMADE open API, see https://www.optimade.org/

    Helper software is available at https://github.com/Materials-Consortia/optimade-python-tools
- Hardware

    - The team is expected to do most of the development in the university computer labs or on their own personal computers. However, if there is interest, it may be possible to arrange a few computers in a dedicated lab space.
- Training
    - A number of lectures and hands-on exercises will be provided.
    - The team is expected to seek out relevant literature themselves (guided by the lectures), with help from the customer and any project experts.

7.2 Project parts / main deliverables

- An MD program capable of simulating the materials that will be considered.
- Tests that assure the quality of the MD simulations.
- A large dataset of materials for which to apply the MD code.
- Helper software / scripts for running MD simulations in high-throughput for the large dataset on supercomputers.
- A results dataset created from the high-throughput MD simulations for the large dataset of materials.
- Helper software / scripts for analysis, data-mining, and visualization of the results of the high-throughput MD simulations.
- Software for making the results dataset available via the OPTIMADE open API and as part of the OPTIMADE network of materials property databases.
- Main Reports / Documentation
    - Specification of requirements
    - Project plan
    - User's guide
    - Technical documentation
    - Final report
- Final presentation

The technical documentation should describe the inner workings of all the software components that are delivered. It must also include a review of test simulations performed with the MD software that convincingly show that it works as intended.

The user's guide should describe how the software is used from an end user's perspective.

The final report should include all documentation that has been created as part of the project.

The final presentation should cover the project and its result and give a brief demonstration of the provided software.

7.3 Requirements

- The source code, the user's guide and the technical documentation must specify a license. It would be nice if it is an open source license, e.g., the MIT license.
- The development must be done in such a way that the development history is preserved by source code version management in git, preferably hosted at GitHub.
- Code development done in open source software maintained externally by others (e.g, libraries, toolkits, etc.) should be contributed back to those projects for possible future inclusion.
- Produced code should be covered by unit tests.

- The software should be user-friendly and possible to operate for someone not so familiar with the internals.
- The MD code needs to be configurable using an external configuration file to specify atomic configurations, MD parameters, boundary conditions, etc.
- The user of the MD code needs to be able to specify an interaction potential definition and parameterization.
- The MD code needs to use an efficient integrator for the numerical integration of the equations of motion.
- The MD code needs to use an efficient way to calculate forces.
- The MD code needs to be able to handle periodic boundary conditions.
- The MD code need to have routines to calculate temperature and internal pressure.
- The MD code needs to support constant temperature simulations.
- It must be possible to use the MD code to be able to calculate bulk properties: lattice constant, bulk modulus, cohesive energy.
- The MD code needs to be able to calculate time-averages of the mean square displacement, the self-diffusion coefficient, the Debye temperature, the Lindemann criterion, and the specific heat of the material.
- The MD code may be able to perform 2-dimensional simulations.
- The MD code may have routines for other material properties, e.g., elastic moduli and elastic constants, formation energy, surface energy and surface reconstruction.
- When validating the code, test simulations needs to be carried out for the implemented materials properties. These tests must:
    - have reached equilibrium.
    - have an optimized volume to get minimal (close to zero) internal pressure.
    - reproduce a reasonable lattice constant of the material, as well as other implemented properties that can be compared with reference values.
- The high-throughput calculations need to be carried out interatomic potentials that are somewhat physically relevant for the materials being considered.
- The supercomputer calculations should be conducted in a way where as little as possible supercomputer time is wasted.
- The system for analysis and visualization needs to be able to work with quite large datasets.
- The analysis and visualization should at least investigate possible correlations between pairs of materials properties by placing the considered materials in scatter plots. Other investigations are encouraged.
- The data set generated in high-throughput should be made available via the OPTIMADE API.

8 Final approval

For final approval all deliveries must have been completed and the end product presented with an interactive demonstration.

9 Deliveries

Deliveries of the project components (see Project parts) and the documentation should be done according to instructions and in accordance with specified deadlines. If nothing else is agreed with the customer, all deliveries of the product and documentation should be uploaded to the course Lisam page.

- In the preparation phase a requirement specification and a project plan should be produced and approved by the customer.
- During the execution phase, the customer and supervisor should be invited to the sprint reviews to be able to review what had been achieved and give input on re-prioritization. However, it is not a requirement that they participate on all of them. Brief notes from the sprint review meetings need to be delivered.
- At the end of the project, all documentation should be collected in a final report and a presentation be given of the project and its results along with a product demonstration.

10 Other requirements

- The project should be conducted according to the LIPs-Scrum project model presented at the beginning of the course.
- Reproduction of copyrighted material (e.g., figures or tables) without permission should generally be avoided in the delivered documentation. Nevertheless, some limited such use can be accepted due to the Swedish teacher copyright exemption as long as this is indicated very clearly (''Reproduced from X without permission'') and the documents are not made public. However, no copyrighted material should be included without permission in the delivered source code, the user's guide, and the technical documentation.
- While not part of the formal deliveries, after the project is finished, but before the project group is dissolved, a final project reflection report should be created and handed in.

B. Specification of requirements

Specification of requirements

Group A

December 21, 2020

Version 1.2

Status

Reviewed	2020-09-28	EJ, PC, LL, WS, RÖ, NSL
Approved		

Project Identity

Orderer: Rickard Armiento, Linköping University

E-mail: rickard.armiento@liu.se

Supervisor: Abijith S Parackal

E-mail: abijith.s.parackal@liu.se

Project members

Name	**Responsibility**	**E-mail**
Nicholas Sepp Löfgren (NSL)	Project leader	nicse725@student.liu.se
Emil Jivtegen (EJ)	Documentation	emiji263@student.liu.se
Linda Le (LL)	Scrum master	linle336@student.liu.se
William Stenlund (WS)	Backlogging	wilst106@student.liu.se
Roger Öngü (RÖ)	Nutrition/Group dynamics	rogon956@student.liu.se
Petter Cyrén (PC)	Group dynamics/Nutrition	petcy342@student.liu.se

Contents

Document history

Version	Date	Changes	Made by	Reviewed
0.1	2020-09-22	First draft.	All	EJ
0.2	2020-09-24	Fixed orderers comments.	All	EJ
0.3	2020-09-28	Revised requirements after DP1.	NSL	EJ
1.0	2020-09-28	Approved.		
1.1	2020-12-10	Renegotiated requirements 19, 22 and 23.	EJ, WS	EJ
1.2	2020-12-21	Updated requirements 53 and 70.	EJ	EJ

1 Overview of the system

A Molecular Dynamics (MD) program will be conceived, designed, implemented and operated by the project group (PG). The system will be divided into partial systems: simulation, data mining, visualization and analysis. This system will be able to extract an extensive set of hypothetical materials using data from large online materials databases, which ones will be specified in later documents. High-throughput methodology will be used to run the MD program on supercomputers to build a database of relevant material properties for the materials in the extracted data sets. The PG will also analyze, datamine and visualize the results. What this exactly entails will be specified in future documents. The results will be made available via the OPTIMADE network of materials property databases.

The priority of each requirement is enumerated with 1, 2 or 3. Priority 1 requirements has the highest priority and must be implemented in the project. Priority 2 requirements has the second highest priority and may be implemented, whilst priority 3 requirements has the lowest priority and are considered more of a suggestion for possible upgrades of the software.

Requirement	**Revising**	**Description**	**Priority**
1	Original	The project will be conducted according to the LIPs-Scrum model.	1
2	Original	No copyrighted material will be included without permission in the delivered source code.	1
3	Original	No copyrighted material will be included without permission in the delivered user's guide.	1
4	Original	No copyrighted material will be included without permission in the technical documentation.	1
5	Original	The software will extract an extensive set of hypothetical materials using one of the large online materials databases available.	1
6	Original	A high-throughput methodology will be used to run the MD program on supercomputers to build a database of relevant material properties for the materials in the extracted data sets.	1
7	Original	The user will be able to specify potential functions via a configuration file.	1

2 System parts in separate sections, with requirements.

These requirements describe the different parts of the system.

2.1 Simulation

Simulation is a main part of this project and the requirements concerning simulation are listed below. The requirements describe what properties the final program will have.

Requirement	Revising	Description	Priority
8	Original	The MD program will be written in Python.	1
9	Original	ASAP and ASE software libraries will be used to produce the MD program.	1
10	Original	The provided software will be able to fetch data about atomic positions and potentials from one or more databases that is searchable.	1
11	Original	The MD code will be configurable using an external configuration file to specify atomic configurations, MD parameters and boundary conditions.	1
12	Original	The MD code will have routines to calculate temperature and internal pressures.	1
13	Original	The MD code will support constant temperature simulations.	1
14	Original	The user of the MD code will be able to specify an interaction potential definition and parametrization.	1
15	Original	The MD code will use an efficient way to calculate forces.	1
16	Original	The MD code will be able to handle periodic boundary conditions.	1
17	Original	The MD code will run the simulation in accordance with some ensemble, with constant temperature e.g. NVT.	1
18	Original	The MD code will be able to move the atoms based on the forces and some time step.	1

cont. on next page

Requirement	Revising	Description	Priority
19	Modified	The MD code can stop the simulation if thermal equilibrium is reached[1].	1
20	Original	The MD code will be able to perform 2-dimensional simulations.	2
21	Original	The high-throughput calculations will be carried out such that the interatomic potentials are not completely irrelevant for the physics of the materials being considered.	1

2.2 Analysis and visualization

The following requirements specify what calculations the program will be able to execute based on the output data from the simulation. These requirements define what results our product can give.

Requirement	Revising	Description	Priority
22	Modified	The MD code will be able to calculate the lattice constant for cubic structures.	1
23	Modified	The MD code will be able to calculate the bulk modulus for cubic structures.	1
24	Original	The MD code will be able to calculate the cohesive energy.	1
25	Original	The MD code will be able to calculate time averages of the mean-square displacement.	1
26	Original	The MD code will be able to calculate time averages of the self-diffusion coefficient.	1
27	Original	The MD code will be able to calculate time averages of the Debye temperature.	1

cont. on next page

1 This can be done by checking maximum and minimum values or checking that some parameter is stabilizing.

Requirement	Revising	Description	Priority
28	Original	The MD code will be able to calculate time averages of the Lindemann criterion.	1
29	Original	The MD code will be able to calculate time averages of the specific heat of the material.	1
30	Original	The MD code will able to calculate the elastic moduli.	2
31	Original	The MD code will able to calculate the elastic constants.	2
32	Original	The MD code will able to calculate the formation energy.	2
33	Original	The MD code will able to calculate the surface energy.	2
34	Original	The MD code will able to calculate the surface reconstruction.	2
35	Original	The analysis and visualization will investigate possible correlations between pairs of materials properties by placing the considered materials in scatter plots[2].	1
36	Original	The software can create surface plots of different kinds, temperature versus pressure or composition etc. for visualisation and analysis.	3
37	Original	The software can create phase diagrams for visualisation and analysis.	3
38	Original	The system will visualize the atomic structure, using some helper program that draws the atoms in the supercell.	3
39	Original	The software will be able to visualize/animate atomic motion over the course of the simulation.	3

2 It is enough to do it for diatomic materials.

3 Performance requirements

The performance requirements carried out for this project aims to establish how well the software developed performs under certain conditions given. The implementation should contribute to an enhanced performance of the software.

Requirement	Revising	Description	Priority
40	Original	The supercomputer calculations will be conducted in a way that there will be no unnecessary waiting time for the parallel process of the supercomputer cores.	1
41	Original	The system for analysis and visualization needs to be able to work with data sets of the size the project is intended to handle.	1
42	Original	The MD code will use the best integrator for the numerical integration of the equations of motion, that can reasonably be implemented on the project budget.	1

4 Possibilities to upgrade

The possibilities to upgrade incites further development of the software. These upgrades should be made such that they enhance the utility of the software.

Requirement	Revising	Description	Priority
43	Original	Additional macroscopic properties could be able to be calculated.	3
44	Original	Additional ensembles could be implemented.	3

5 Stability

These requirements describe the stability of the system.

Requirement	Revising	Description	Priority
45	Original	A time limit on calculations will be implemented to stop any diverging process.	1

6 Economy

The main resource used in this project is the working time of the PG and the core hours of the supercomputer. The project covers nine credits, which is equivalent to six working weeks or around 240 hours.

Requirement	Revising	Description	Priority
46	Original	Each member of the PG will work on the project for approximately 240 hours, including 36 hours of training.	1
47	Original	The PG will use allotted amount of core hours of the supercomputers.	1

7 Security requirements

These requirements describe the security of the system.

Requirement	Revising	Description	Priority
48	Original	The system will be secure enough so that an external agent will not be able to cause harm to any systems involved with the MD program, e.g. the supercomputers used.	1

8 Delivery

This section declares what results the orderer wants from the PG and at which times. The deadlines are crucial for the approval of the project.

Deadlines:
week 37: Form groups and go through project directive.
week 40: Finish Specification of requirements and Project plan.
week 50: Finish bulk work of project.
week 52: Present the project.
week 2: All documents and tasks are approved.

Requirement	Revising	Description	Priority
49	Original	Through helper software/scripts the MD program will be run on supercomputers to help create a database of relevant materials properties for a given data set.	1
50	Original	The source will specify the open source MIT license.	1
51	Original	Code development done in open source software maintained externally by others will be contributed back to those projects for possible future inclusion.	1
52	Original	In the preparation phase a requirement specification and a project plan will be produced and approved by the orderer.	1
53	Original	Brief notes from the sprint review meetings will be delivered to the orderer and supervisor.	
54	Original	At the end of the project, all documentation will be collected in a final report and a presentation be given of the project and its results along with a product demonstration.	1
55	Original	After the project is finished, but before the PG is dissolved, a final project reflection report will be created and handed in.	1
56	Original	The results of the project, meaning the data generated by the MD program will be made available via OPTIMADE API.	1

cont. on next page

Requirement	**Revising**	**Description**	**Priority**
57	Original	A high-throughput methodology will be applied to run the MD code.	1

9 Documentation

Below are listed the documents the PG will produce for this project, with a short description of them. Furthermore, requirements on these documents are also listed.

Document	**Language**	**Purpose**	**Target**	**Format**
Specification of requirements	English	Lists the requirements that needs to be fulfilled by the project.	Orderer, PG	pdf
Project plan	English	What has to be done in the project and an overall idea for the design.	PG	pdf
Users guide	English	Describes in detail how the product works and how to handle it.	Public	pdf
Technical documentation	English	Describes how the product is designed and implemented. Detailed documentation. Includes a review of test simulations performed by the MD code.	Orderer	pdf
Final report	English	A collection of all documents in the project.	Orderer	pdf
Project reflection	English	The PG members reflect on their experience regarding working method, collaboration and the usage of the LIPs-Scrum project model.	Orderer	pdf

Requirement	Revising	Description	Priority
58	Original	The user's guide will specify the open source MIT license.	1
59	Original	The technical documentation will specify the open source MIT license.	1
60	Original	All documents will follow the LIPS standard [1].	1
61	Original	Explanatory comments and embedded documentation in the source code will be implemented.	1
62	Original	In relation to the sprint reviews, brief summarizing notes will be sent in by mail to the orderer (Rickard Armiento).	1

10 Training

The following requirements describes the necessary training the group members will go through to be able to carry out the project.

Requirement	Revising	Description	Priority
63	Original	All members of the PG will attend ten lectures in the course.	1
64	Original	All members of the PG will be trained using Git, by doing a hands-on exercise in Git.	1
65	Original	All members of the PG will be trained in different programming language paradigms, by doing a hands-on exercise.	1
66	Original	All members of the PG will be trained in using the libraries ASAP and ASE with the purpose of doing an MD program. This will be established by doing a hands-on exercise concerning the topic.	1
67	Original	All members of the PG will be trained in the usage of NSC supercomputer, by learning to run ASAP-simulations in a hands-on exercise.	1

cont. on next page

Requirement	**Revising**	**Description**	**Priority**
68	Original	All members of the PG will be given 10 lectures related to the content of the project.	1

11 Quality

To make sure our product is up to standard we will develop tests and tweak the program for optimal quality. Below are the requirements for this.

Requirement	Revising	Description	Priority
69	Modified	Relevant produced code will be covered by unit tests.	1
70	Original	Tests must exist to validate that the final state of calculated system, has reached equilibrium.	1
71	Original	Tests must have an optimized volume to get minimal internal pressure (internal pressure close to zero).	1
72	Original	Tests must validate that the MD program produce a reasonable lattice constant of test systems, as well as other implemented properties compared to some reference values.	1

12 Maintainability

Requirement	Revising	Description	Priority
73	Original	The development will be done in such a way that the development history is preserved by source code version management in git, via Github.	1

References

[1] C. K. TOMAS SVENSSON, *Projektmodellen LIPS*, Studentlitteratur AB, Lund, 1:1 ed., 2011. ISBN 978-91-44-07525-9.

C. Project plan

Project plan

Group A

October 2, 2020

Version 1.0

Status

Reviewed		
Approved		

Project Identity

Orderer: Rikard Armiento, Linköping University

E-mail: rickard.armiento@liu.se

Supervisor: Abijith S Parackal

E-mail: abijith.s.parackal@liu.se

Project members

Name	**Responsibility**	**E-mail**
Nicholas Sepp Löfgren (NSL)	Team leader	nicse725@student.liu.se
Emil Jivtegen (EJ)	Document responsible	emiji263@student.liu.se
Linda Le (LL)	Scrum master	linle336@student.liu.se
William Stenlund (WS)	Backlogging	wilst106@student.liu.se
Roger Önçü (RÖ)	Nutrition/Group dynamics	rogon956@student.liu.se
Petter Cyrén (PC)	Group dynamics/Nutrition	petcy342@student.liu.se

Contents

List of Tables

Document history

Version	Date	Changes	Made by	Reviewed
0.1	2020-10-01	First draft.	All	EJ
0.2	2020-10-02	Added risk analysis and information about sprints.	All	EJ
1.0	2020-10-02	Approved.		

1 Description of project

The following section contains information describing the purpose and goal of the project, its deliveries and limitations will also be presented. The project group will be addressed as PG throughout the document.

1.1 Purpose and goal

CDIO is an abbreviation for Conceive, Design, Implement and Operate. These words represent and refer to the engineering process. Conceive is about firstly understanding the problem, and the possible solutions for it. Design is about deciding the practical details of the solution to the problem. Thereafter, the implementation step needs to actualize the decision made for the design chosen. Lastly, operate refers to running and operating the final implemented system. The purpose of the CDIO project is to give the group members an opportunity to develop the skills required to finish a large engineering project. Via this project important skills in group work and cooperation, problem solving and implementation will be acquired. All these skills are vital for an engineer.

1.2 Deliveries

In the following table all deliveries for the project will be presented. Deliveries of the project refer to all products, documents and reports.

Table 1: Table of deliveries.

Delivery	**Type**	**Medium**	**Date**
Specification of requirements	Document	Teams	2020-10-02
Project plan	Document	Teams	2020-10-02
User's guide	Document	Teams	2020-12-18
Technical documentation	Document	Teams	2020-12-18
Notes from sprint review meetings	Notes	E-mail	Continuous frequent delivery
Final report	Document	Lisam	2021-01-15 (w.2)
Final presentation	Oral demonstration	Meeting	2020-12-25 (w.52)

Project reflection	Document	Lisam	
An MD program capable of simulating the materials that will be considered.	Product	Demonstration	2020-12-25 (w.52)
Tests that assure the quality of the MD simulations.	Product	Code delivery	2020-12-25 (w.52)
A large data set of materials for which to apply the MD code.	Product	Demonstration	2020-12-25 (w.52)
Helper software/scripts for analysis, data-mining, and visualization of the results of the high-throughput MD simulations.	Product	Demonstration	2020-12-25 (w.52)
Software for making the results dataset available via the OPTIMADE open API and as part of the OPTIMADE network of materials property databases.	Product	Demonstration	2020-12-25 (w.52)

1.3 Limitations

There will be no in depth analysis or investigation over optimal programming languages, or other tools that can be used for the project other than over what there are already knowledge about within the project group.

Due to the nature of the project, there will be resource limits that will limit the project. This will mainly be the working hours expected of each members, roughly 240 hours, including 36 hours of training. There will also be restrictions placed upon the project, decided by the available contents from the used databases, such as inter atomic potentials from e.g. OpenKIM.

2 Phase plan

There are mainly three phases present in the project. These are "Preparation phase", "Execution phase" and the "End game". There can also be a fourth phase, that takes place before the preparation phase. Hence, not a phase of the project itself, but still an instance for the CDIO course. During this phase the organization, the PG, is created. The organization organizes itself with clear roles and a group contract is signed. Here, the group has their first meeting and the project directive is looked at. The first hands-on exercise will be given.

2.1 Preparation phase

- Some lectures will be held to teach the group skills needed for the project.
- Hands-on exercise 2 and 3 will be given.
- The Requirements Specifications is created and approved by the orderer.
- The project plan is finished and approved by the orderer.
- A product backlog is created, where epics are broken down to tasks and stories.
- The project backlog is finished.
- DP1 and DP2 with accordance to LIPS is done.

2.2 Execution phase

Execution phase is divided into development phases with mile stones separating them.

- Some lectures will be held to teach the group skills needed for the project.
- Hands-on exercise 4 will be given.
- The software will be written.
- Unit tests will be written in conjunction with the code writing.
- The whole group will have 3 meetings per week, known as stand ups, with an intended maximum time limit of 10 minutes. The group will also have at least 4 sprint meetings (meaning print reviews, retrospectives and planning).
- Notes from sprint reviews will be handed in to the orderer via mail.

2.3 End Game

- The orderer will check requirements of the system, and decide to approve the system delivery, DP5.
- The system is formally delivered.
- Final version of documents are delivered.
- A final presentation will be given by the group, which demonstrates results of the project.
- The project group is dissolved.

3 Organization plan

In this section the overall organization and work flow of the project is described. The PG will develop the product according to the orderers demands and wishes. If problems are encountered, the PG has a supervisor that can help.

3.1 Organization plan per phase

Preparation phase: The PG is made and is handed the project. Responsibilities, such as team leader, document responsible etc., are distributed. Specification of requirements and a project plan are written.

Execution phase: The PG works on the project. This is the bilk work and includes implementing code and communicating with stakeholders.

End game: When the product is delivered the PG will write a reflective document, giving their thoughts and experiences of the project. The PG dissolves.

3.2 Terms of co-operation

Th PG is a flat organization where the team leader acts as chairman at meetings and takes care of communication with the orderer. In addition to this, the team leaders tasks are the same as for other members. The PG has jointly developed a group contract that will be followed.

3.3 Definition of responsibilities

Team leader:	Nicholas Sepp Löfgren
E-mail:	nicse725@student.liu.se
Description:	Handles organization and communication.
Document responsible:	Emil Jivtegen
E-mail:	emiji263@student.liu.se
Description:	Makes sure all documents required are written and up to standard.
Scrum master:	Linda Le
E-mail:	linle336@student.liu.se
Description:	Master over meetings.
Backlogging:	William Stenlund
E-mail:	wilst106@student.liu.se
Description:	Keeps track of activities.
Nutrition, Group dynamics:	Petter Cyrén, Roger Önçü
E-mail:	petcy342@student.liu.se, rogon956@student.liu.se
Description:	Brings "fika" to meetings and keeps a pleasant atmosphere.

4 Document plan

Throughout the project several documents will be produced. These documents will be written using Overleaf, an online text editor. Finished documents will be uploaded to the designated Microsoft Teams channel. In addition to these documents, there will also be documentation in Github about specific tasks uploaded to the PG's repository. In contrast to other documents, the technical documentation and users guide will be edited throughout the execution phase.

All documents uploaded to Lisam will be version controlled. Not yet approved documents will have version 0.x, where x, $x \geq 1$, indicates minor changes. Approved documents will have version 1.x, $x \geq 0$. All documents will have a document history which keeps track of the versions. All documents will be written in english.

Table 2: Table of documents.

Document	**Responsible/ Approved by**	**Purpose**	**Distributed to**	**Deadline**
Group contract	LL	Agreement of conflict handling and communication in group.	PG	2020-09-08
Desicion protocol	EJ/ Orderer	Meetings protocol from desicion points.	PG, supervisor, orderer	Running
Meeting protocol	LL	For follow up.	PG	Running
Other internal documents	EJ	Storage of information, factual basis etc.	PG	Running
Specification of requirements	PG/ Orderer	Defines which requirements the project has.	Orderer	2020-10-02
Project plan	PG	Planning and time consumption for each block, who works with what and when.	Orderer	2020-10-02
Time report	PG	Accounting of time.	Orderer, supervisor	Running
Technical documentation	PG	Description of product.	Orderer	2020-12-11
Users guide	PG	Description of how to handle and maintain the product.	Orderer	2020-12-11
Final report	PG	Conclusion of project.	Orderer	2021-12-11
Project reflection	PG	Summary of the PG's experiences.	Orderer	2020-01-15

5 Development methodology

For the project the development methodology will consist of object-oriented programming, version control via Git and GitHub, present data through simulation and visualization. By using the NSC supercomputer, the data generated will be uploaded to OPTIMADE. During the project the software development will be divided in smaller tasks. The tasks will be developed either individually or in pairs.

6 Education plan

The education plan for the project are made up of lectures and hands-on lessons given by the course lecturer. Each lecture covers a different topic critical for the further development of knowledge for the project. The hands-on lessons consists of 4 of such. The purpose of the lessons is to give each project member an idea how to apply the theory given during the lectures and how to use the theory in the project.

1. Hands-on lesson 1: The lesson will provide basic and useful knowledge of version control in Git and Github.
2. Hands-on lesson 2: Advanced programming concepts and visualization.
3. Hands-on lesson 3: Using the ASAP and ASE libraries to construct a MD program.
4. Hands on lesson 4: NSC supercomputer usage by running ASAP-simulations.

The project group will also be given a guest lecture from an experienced software developer to further develop a certain structure within the software development.

7 Reporting plan

The reporting plan describes how and when the project group will report the current status of the project to the orderer and supervisor. The project group will report sprint reviews, which are brief notes taken of sprint review meetings, via e-mail to the orderer. The orderer and supervisor will also be invited to sprint review meetings if they wish to attend. The project group will also report their working time in a document. All the documents that are produced by the project group will be uploaded to a private group in Microsoft Teams.

8 Meeting plan

The PG will primarily be meeting 2-3 times a week for stand-up meetings, as specified by the Scrum model. These meetings will be in the morning and very short, around 10 minutes. Additional meetings may be scheduled if necessary. There will also be four extensive sprint meetings where the project is reviewed and members are briefed of further tasks. These meetings may include the orderer.

9 Resource plan

The resource plan describes what resources the project group has available and how they will be used during the project work.

9.1 People

The project group consists of six members, as specified in the project identity. The project group has a supervisor who assists the group with their expertise. The group is ultimately accountable to the orderer.

9.2 Material

The PG will have access to computer rooms on the university campus and are allotted hours on the supercomputers of NSC.

9.3 Premises

The project group will mainly work in distance mode. Although campus premises will be used when possible or necessary.

9.4 Economy

The primary economic resource the project group has is their working time. Each member of the group should work for around 240 hours, considering the project should cover nine credits.

10 Milestones and decision points

This section describes the milestones and decision points that the project group will reach throughout the project.

10.1 Milestones

Table 3: Table of milestones.

Milestone	Description	Deadline
1	Hand in requirement specification and project plan.	2020-10-02
2	Finish the project backlog and start the execution phase.	2020-10-02
3	The simulation can reach equilibrium.	2020-10-16
4	The software fulfills the requirements placed upon it.	2020-12-11
5	Deliver project, documentation and presentation.	2020-12-25

10.2 Decision points

Table 4: Table of decision points.

Decision Point	Description	Deadline
DP0	Form subgroups, have the first group meeting to define roles, write group contract, and have a look at the project directive.	2020-09-11
DP1	Finish the requirement specification: start the preparation phase.	2020-10-02
DP2	Finish the project plan and the "project backlog": start the execution phase.	2020-10-02
DP5	Approve software against requirement specification, decide to deliver.	2020-12-25
DP6	Deadline for final version of the report. Dissolve project group.	2021-12-16

11 Activities

Below follows a list of activities together with their respective descriptions, time estimates and dependencies. The ”Dependencies” column refers to the activities that must be completed before you can start with the current activity.

Table 5: Table of activities.

No.	Activity	Description	Time (h)	Dependencies
Documentation			**223**	
1	Project directive	The directives given by the costumer.	0	
2	Specification of requirements	Defines the requirements on the project.	42	1
3	Project plan	Describes how the project will be carried out.	42	1, 2
4	Group contract	Describes the agreed upon rules regarding workflow, communication and conflict management.	1	1
5	Technical documentation	Describes how the product works and how it is designed.	54	2, 3
6	User’s guide	Describes how the product is used.	36	5
7	Final report	Concludes the entire project.	36	2, 3, 4, 5, 6
8	Project reflection	Summary of the PG’s experiences.	12	7
Meetings			**48**	
9	Decision point meetings	Meetings between costumer and PG to decide whether or not the project should continue.	12	2, 3
10	Sprint planning	Planning upcoming tasks.	24	3
11	Sprint reviews	Reviewing how the sprints have affected the project.	12	10
Training			**216**	
12	Lectures	Ten lectures in relevant programming skills, software development and physics.	120	1

13	Hands-on exercises	Training in Git, programming language paradigms, programming libraries and usage of supercomputers.	96	1, 12
Software			**694,5**	
14	Calculation code	Write code for calculations that will be used in the simulations.	120	3
15	Forces code	Write code for calculating the forces.	40	14
16	Moving code	Write code that moves the atoms one time step.	40	15
17	Sample code	Write code that samples the system.	40	
18	Simulation code	Write code that calls on the above code/-subroutines in the right order with conditions for stopping the simulation.	20	14-17
19	Diverging code	Write code that stops any diverging processes.	10	14-18
20	Visualization of atom structure	Implement methods to visualize atom structure.	14.5	18
21	Extract materials	Extract structures from materials database.	40	
22	Extract potentials	Extract interatomic potentials from database (OpenKIM).	30	
23	Analysis and visualization code	Write code that will analyse and visualize properties using data.	120	18, 26
24	Scatter plot	Write code to visualize results in scatter plots.	60	18, 26
25	Supercomputer integration code	Write code that will allow our software to interface against the supercomputer.	60	18
26	Database storage code	Write code for saving data into a database.	100	18
Tests			**108**	
27	Unit tests	Write unit tests for the software.	36	
28	Integration tests	Implement integration test for the software.	36	

29	System test	Implement system test for the software.	36	
Other			**60,5**	
30	Additional self studies	Time for members to learn more about the project on their own.	30	1, 3
31	Prepare delivery	Prepare the oral presentation of the project.	30	7
32	Create a NSC account	Create an account to be able to use the NSC supercomputer.	0,5	1, 12
Spare time			**90**	
Total			**1440**	

12 Time plan

A time plan that schedules who and when each activity will be executed together with its estimated time consumption will be modeled. The time plan will be an excel document.

13 Change plan

The project is quite extensive which brings possible risks of complications along the way. It is reasonable that the PG will not be able to carry out the project according to the project plan at all times. Therefore, it is important to have a change plane. In the project, a few meetings with the costumer are scheduled. In these meetings, arised obstacles and possible renegotiation of requirements will be discussed.

14 Quality plan

This project is fairly extensive so the the quality of the product and it's parts must be assured. To achieve this code and documents will be reviewed and tested.

14.1 Reviews

Documents will be reviewed by the group members and the project member responsible for documentation will check with each member before delivery. Any code pushed to the Github master branch will have to be reviewed and approved by at least two group members.

14.2 Test plan

All parts of the MD code and helper programs will be covered by unit tests to assure each part has the correct functionality. Integration tests will be used to verify that parts of the code work together properly, and that they can receive and output data to each other. This will be done at several levels of integration. The complete system will be covered by systems tests to ensure that the system as a whole works as intended and outputs sensible results. For example can a system, where properties like lattice parameter are well known, be simulated and compared to the known properties of the system.

15 Risk analysis

The project is given a budget in form of hours and to minimize the risk of exceeding this limit a risk analysis is a prerequisite. The PG will keep track of spent hours in a time plan. The time plan is a frame work of when each activity should be made and how long it is supposed to take. In accordance to the agile work flow of the LIPs-Scrum model, there is also a modified time plan if changes will be necessary.

Another risk in this project is data storing. Hard drives and computers can break down and to keep the software safe all code will be uploaded to Github. This gives the PG the opportunity to access it from any computer.

16 Priorities

In the event of delays, as a general rule, the simulation will be prioritized over the analysis and visualization. The general functionality of the simulation will be prioritized over its performance.

17 Project finish

After the finished code and all related documentation has been delivered to the orderer an end reflection report looking back at the project will be sent by the project group to the orderer. When the delivery has been approved by the orderer the project group will be dissolved.

D. Technical documentation

Technical documentation

Group A

January 13, 2021

Version 0.1

Status

Reviewed		
Approved		

Project Identity

Orderer: Rickard Armiento, Linköping University

E-mail: rickard.armiento@liu.se

Supervisor: Abijith S Parackal

E-mail: abijith.s.parackal@liu.se

Project members

Name	**Responsibility**	**E-mail**
Nicholas Sepp Löfgren (NSL)	Team leader	nicse725@student.liu.se
Emil Jivtegen (EJ)	Document responsible	emiji263@student.liu.se
Linda Le (LL)	Scrum master	linle336@student.liu.se
William Stenlund (WS)	Backlogging	wilst106@student.liu.se
Roger Önçü (RÖ)	Nutrition/Group dynamics	rogon956@student.liu.se
Petter Cyrén (PC)	Group dynamics/Nutrition	petcy342@student.liu.se

Contents

List of Figures

List of Tables

Listings

Document history

Version	Date	Changes	Made by	Reviewed
0.1	2021-01-13	First draft of the report.	All	EJ

1 Introduction

The following section contains information describing the background and purpose of the project. Furthermore, this document will provide a detailed description of the product and some of the most fundamental knowledge required in this field of work. The project group will be addressed as PG throughout the document.

All products and documents in this project are free to use according to the MIT open license specified on the projects Github repository[1].

1.1 Background and Purpose

To make new and innovative technologies viable, design of materials with desired properties is crucial. It is a typical challenge to to find materials that fulfill requirements on cost, environmental impact, length of life and safety among other properties. A tool used for this challenge is theoretical investigation of hypothetical materials and system configurations by computational materials simulations. An understanding of the physics of materials can be reached through these simulations and continued progression of computational power, storage capability and improved methodology makes it more available. Molecular dynamics is such a simulation that has been utilized where one can extract valuable information about materials behavior to unlock new, desirable materials.

"The purpose of this project is to use modern software engineering practices for conceiving, designing, and implementing computational physics software and to operate this software in a study aligned with present trends in database-driven materials design" [1].

1 https://github.com/obsqyr/TFYA92-group-A/blob/master/LICENSE

2 Theory of Molecular Dynamics

Molecular dynamics (MD) is a simulation of, as the name suggests, movements of atoms. The implementation of MD varies but the main idea is to calculate the energy and forces corresponding to the simulated system for a time step using Newton's equations of motion. This is repeated for each desirable time step and together they show how the atoms interact with each other and what forces are acting on them. This chapter will describe the underlying theory of MD in more detail.

2.1 Atoms and Integration

During a simulation, atoms in an unit cell are moved by some small time-step using the interatomic forces and an integrator, which depends on the system's ensemble. The ensembles can be microcanonical (NVE), canonical (NVT) and isobaric-thermal (NPT). What this means is that different properties are kept constant during the simulations in order to get the results for other properties. Number of moles is kept constant in all ensembles and it is common to keep the volume constant as well.

2.2 Periodic Boundary Conditions

Large systems, such as bulk gasses, liquids et cetera, are often approximated by a periodic boundary condition (PBC) in which a unit cell represents the whole system. If you exit a unit cells northern side, you would enter the same cell on the southern side with the same velocity and direction. This is a valid simplification since each atom has near negligible size compared to the whole system, which implies that the vast majority of atoms will be in a unit cell.

2.3 Forces and Potentials

From the potential of the atoms one can calculate the force with which the atoms are disturbed. These forces are derived from the interatomic potentials of materials. To do so we use Lennard-Jones potential which is a simple way of calculating interatomic pair potential. Through the Lennard-Jones potential we took the negative gradient of it to get the force. Lennard-Jones is a fairly easy potential to implement but it still yields sufficiently good values.

3 Software Implementation

The main task for this project was programming. To get the code working some libraries and packages were needed. To get the program running the user need to enter some inputs and how the program works is described below.

3.1 Libraries

In order to gain access to libraries and dependencies needed to execute this project the Anaconda package manager was used along with the Python package installer, *pip* [2]. Relevant packages are listed in *requirements.txt* in the GitHub repository[2]. These worked as tools for enabling software and functions needed to run a simulation or to make data available. To implement the general physics needed to present and calculate the molecular dynamics we manage the code libraries ASAP and ASE [3], [4]. The libraries manage several aspects of the molecular dynamics such as the potentials, atom structure, ensemble et cetera.

3.1.1 ASE

The Atmoic Environement Simulation (ASE) contains Python modules which manipulates the atomic simulations. The ASE library offers the fundamental molecular dynamics building blocks to simplify the coding process of the project. Through the ASE the body of the molecular dynamics structure were built to apply the simulations upon the created atoms object. The ASE features are fundamental to conduct the simulations, although ASE is not used to do the simulations per se, ASE functions as the building blocks for the MD simulations.

3.1.2 ASAP

The ASAP library is a calculator for doing large-scale classical molecular dynamics through ASE. This implies that ASAP calculates the important physical entities of molecular dynamics, forces and energies on atoms which has been implemented through ASE. How these calculations are executed by an implementation of the effective medium theory [3]. In the project the ASAP has been implemented when calculating the forces between the interatomic systems and the energies of the systems.

2 https://github.com/obsqyr/TFYA92-group-A/blob/master/requirements.txt

3.2 Used Databases

The databases used for this project were OpenKIM and Materials Project [5], [6]. OpenKIM is an online resource for standardized testing, warehousing and retrieval of interatomic models and data. It was used to retrieve a potential for the simulation. By using the OpenKIM API the extraction of the potential from the database is easily implemented with the ASE calculators. Materials Project provides open web-based access to computed information on known and predicted materials as well as powerful analysis tools to inspire and design novel materials. From this, information about the different materials were gathered.

Much of what will be visible as examples in documentations produced by the project group will be results from usage of a certain subset of materials from *materialsproject.org*, named *test_120_materials.json*. However, there are no restrictions on what database of hypothetical materials to use. The materials the user wish to run the simulation on can be specified in the configuration file, where the user simply needs to input the path to the json-file containing the material. The only requirements placed on this file is that the json-object contains a key called *response* and that inside it is a key called *cif*, which lists all materials in their CIF-format. In other words, the only real requirement on the database which can be used is that the materials are or can easily be converted into CIF-format.

3.3 Database from Simulation Results

Different properties averages such as self-diffusion coefficient, specific heat and more are retained from the simulation. What counts as the results of the project are the values of these properties (in addition to the graphs generated). These results are stored as a database and made available via a REST API, OPTIMADE API. Figure 1 and figure 2 below display the appearance of such a database. The figures display the appearance of material 15, where 15 is simply how each materials in the database are ordered as they are entered, with no significant meaning. Instead what uniquely defines the material is given by the field *task_id*, see table 2.

Note that figure 1 and figure 2 are for demonstrative purposes and don't fully display the values of all the fields. The figures give rough sketches of the database. Each material are ordered and contains a number of fields which describes it. Each material are described by 34 different fields in total where 14 of these are required by OPTIMADE API, and is called below as mandatory fields. The rest 20 fields starts with prefix _groupa_ and give the specific results resulted from the simulation. The types and contents of these fields vary and range from single floats, strings, to list structures of such data types et cetera.

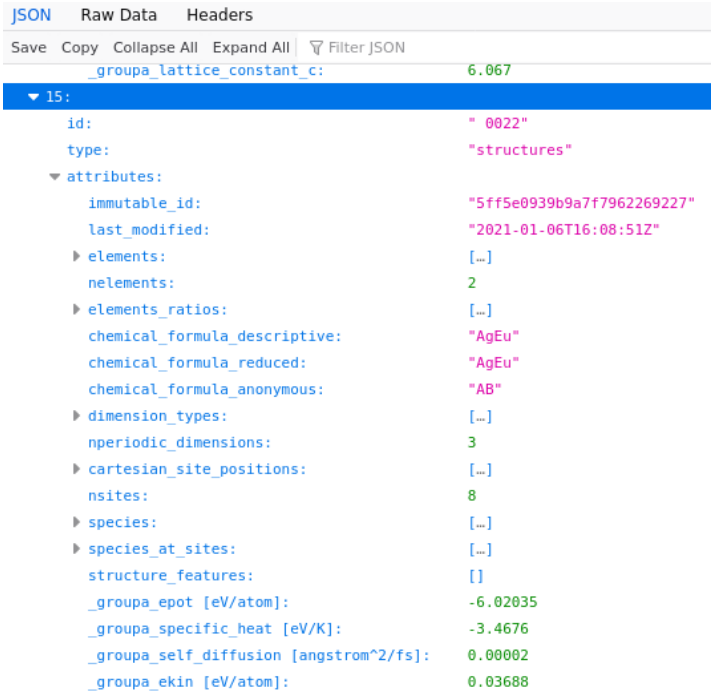


Figure 1: Appearance of the database.

Figure 2: Appearance of the database.

3.3.1 Results in MD_structures.json

The file responsible for making data accessible via REST API is given by *make_mongodb.py*. After running this script a json-object will be created, whose name is by default set to *MD_structures.json*. The data of this json-file is essentially what will be published. By including this data file in the repository *https://github.com/attlevafritt/tfya92-groupa-optimade-python-tools.git* under folder *tfya92-groupa-optimade-python-tools/optimade-python-tools/optimade/server/data/* this data file will be made available via the REST API.

3.3.2 OPTIMADE API

Each material has certain predefined fields which describes it, these are for instance *nelements*, *chemsys* and *cartesian_site_positions*. These fields are made apparent under table 1. The properties specific for the PG's work are described in table 2 below. Note that the listed fields in table 2 all start with a prefix *_groupa_* that is not written out explicitly. This is to indicate that they are specific for this project (see figure 1 and figure 2). The following table lists the mandatory, by OPTIMADE, field entries in the database. For further information see the documentation[3].

Table 1: List of mandatory fields in the database.

Name	Description
pretty_formula	The empirical formula of a material. String.
last_modified	The date last modified. String.
nperiodic_dimensions	How many dimensions are periodic. For this project it's always three (bulk 3D system). Numerical value.
dimension_types	The dimensions material is periodic. For this project it's always [1,1,1]. A three component list of 0 or 1.
elements	Lists the element symbols of a material. A list of strings.
elements_ratios	The ratios of each elements of material. A list of numerical values.
nelements	The number of elements of a material. Numerical value.
species_at_sites	Lists element symbols corresponding to each site in the unit cell. A list of strings.
formula_anonymous	A formula with letters and numbers giving the proportions of each element of the material. Here the elements are not shown but replaced with an number. String.

3 https://github.com/Materials-Consortia/OPTIMADE/blob/develop/optimade.rst?fbclid=IwAR3Oe1_phVsrHxn5XsDrAKOk_Hcgj6DY9I-HlZthctXLZYnx3X9U7uWVKOc#properties-used-by-multiple-entry-types

structure_features	Tells if special features are used, such as disorder simulation et cetera. For this project it is always an empty list. List of strings or empty list.
task_id	A string which contains a series of numbers which uniquely identifies the material. For this project it is the same as *Material ID* (see figure 3), which are unique for each material MD runs on. String.
cartesian_site_positions	The Cartesian coordinates for each site of the unit cell. A list of lists of numerical values.
species	Describes the species of the sites of material. A list of dictionaries.
nsites	The number of sites. Numerical value.

The following table lists all properties derived from the simulation. Note that if a field doesn't have a unit, then column "Unit" is left empty.

Table 2: List of fields in the database specific for this project.

Name	**Unit**	**Description**
epot	eV/atom	The potential energy for the simulated system. Numerical value.
ekin	eV/atom	The kinetic energy for the simulated system. Numerical value.
etot	eV/atom	The total energy for the simulated system. Numerical value.
temp	K	The temperature for the simulated system. Numerical value.
MSD	$Å^2$	The mean square displacement of the atoms in the simulated system. Numerical value.
self_diffusion	$Å^2$/fs	The self diffusion of the simulated system. Numerical value.
pressure	Pa	The pressure of the simulated system. Numerical value.
specific_heat	J/(K*kg)	The specific heat for the simulated system. Numerical value.

unit_cell_composition		The unit cell composition, when it's a supercell this will field will show. String.
lattice_constant_a	Å	The lattice constant in one direction. Numerical value.
lattice_constant_b	Å	The lattice constant in one direction. Numerical value.
lattice_constant_c	Å	The lattice constant in one direction. Numerical value.
interpolated_lattice_constant_a	Å	The interpolated lattice constant in one direction. Numerical value.
interpolated_lattice_constant_b	Å	The lattice constant in one direction. Numerical value.
interpolated_lattice_constant_c	Å	The lattice constant in one direction. Numerical value.
bulk_modulus	GPa	The bulk modulus of the material. Only assumes a value if volume relaxation is specified in configuration file for the simulation. Numerical value.
cohesive_energy	eV/atom	The cohesive energy of a material. Numerical value.
debye	K	The Debye temperature of the system. Numerical value.
lindemann		The melting temperature of the material according to the Lindemann criteria. Numerical value.

3.4 User Input

The user can input desired parameters for how they wish to run their MD simulations. This is done in *settings.json*. The user can for example choose which ensemble (and consequently which integrator) will be used for the simulations, which materials they wish to simulate, at what temperature the simulations should be done in and more. An example of such a settings file is given below.

```
{
    "time_step": 3,
    "max_steps": 1000,
    "temperature": 200,
    "ensemble": "NVE",
```

```
    "friction": 0.001,
    "decimals": 5,
    "interval": 100,
    "vol_relax": true,
    "LC_mod": 0.01,
    "LC_steps": 2,
    "tolerance": 0.0005,
    "materials": "test_120_materials.json",
    "supercell_size": 8,
    "search_interval" : 10,
    "threshold" : 6,
    "cubic_only" : true
}
```

Listing 1: Example of a settings file.

3.5 Molecular Dynamics Program

The MD implementation formed the core of the software. The MD calculations were made in a python script, *md.py* through with its content being the majority of the functionality of a MD simulation. The file contains the necessary functions the MD simulations need to be able to perform material calculations. Desired parameters to calibrate the MD calculations are stored in a settings file to easily tune values to get a desired result of the simulations. The MD simulations are constructed through the *md.py* file. The file uses the extraction of potential through OpenKIM API and integrators from the ASE library. Depending on which ensemble we desire to occur during the running of the simulation, the code runs the proper integrator through the implementation illustrated below.

```
    # Select integrator
    if settings['ensemble'] == "NVE":
        from ase.md.verlet import VelocityVerlet
        dyn = VelocityVerlet(atoms, settings['time_step'] * units.fs)

    elif settings['ensemble'] == "NVT":
        from ase.md.langevin import Langevin
        dyn = Langevin(atoms, settings['time_step'] * units.fs,
            settings['temperature'] * units.kB, settings['friction'])
```

Listing 2: Selection of integrator

Since the MD simulations are constituted by a certain time interval an implementation of a convergence of time was necessary. The important step of checking whether a simulation reached thermal equilibrium or not during the allotted time was made in the *md.py* file as follows

```
    counter = 0
    equilibrium = False
    for i in range(round(settings['max_steps'] / settings['search_interval'
    ])): # hyperparameter
        epot, ekin_pre, etot, t = properties.energies_and_temp(atoms)
        dyn.run(settings['search_interval']) # hyperparamter
        epot, ekin_post, etot, t = properties.energies_and_temp(atoms)
```

```
        if (abs(ekin_pre-ekin_post) / math.sqrt(N)) < settings['tolerance']:
            counter += 1
        else:
            counter = 0
        if counter > settings['threshold']: # hyperparameter
            print("reached equilibrium")
            equilibrium = True
            break

    if equilibrium:
        dyn.run(settings['max_steps'])
        properties.finalize_properties_file(atoms, id, decimals, monoatomic)
    else:
        properties.delete_properties_file(id)
        raise RuntimeError("MD did not find equilibrium")
    return atoms

if __name__ == "__main__":
    run_md()
```

Listing 3: Equilibrium management of the MD simulations

The hyperparameters to consider for the thermal equilibrium in the settings file are: tolerance, threshold and search_interval.

3.6 Main Program

The main program *main.py* handles running MD over a larger set of materials. The user can specify for which set of materials they wish to run their simulations over in the *settings.json* file by specifying a Materials Projects json-file which contains the desired materials. The main program will unpack these desired materials from cif-files and construct a large list of ASE atoms objects, as follows

```
    # create one list of all atom objects in data-file
    atoms_list = []
    for cif in mp_properties['cif']:
        f = open('tmp'+str(rank)+'.cif', 'w+')
        f.write(cif)
        f.close()
        atoms = ase.io.read('tmp'+str(rank)+'.cif')
        if settings['cubic_only']:
            lengths = atoms.get_cell_lengths_and_angles()[0:3]
            angles = atoms.get_cell_lengths_and_angles()[3:6]
            if len(set(lengths)) == 1 and len(set(angles)) == 1 and angles[0] == 90:
                #print("cubic")
                #print(atoms.get_cell_lengths_and_angles())
                if settings['vol_relax']:
                    cell = np.array(atoms.get_cell())
```

```
                    P = settings['LC_steps']
                    for i in range(-P,1+P):
                        atoms_v = copy.deepcopy(atoms)
                        atoms_v.set_cell(cell*(1+i*settings['LC_mod']))
                        atoms_list.append(atoms_v)
                else:
                    atoms_list.append(atoms)
        else:
            atoms_list.append(atoms)
    print("Created atoms list of length " + str(len(atoms_list)))
```

Listing 4: Generating a list of atoms objects to simulate

The main program then simply continues to run MD simulations over all of these atoms objects. If the program is run on a parallel computer, the atoms list will be subdivided and those sublists will be distributed evenly to available computing nodes, see 4.1.

3.7 Tests

To make certain a level of quality is upheld from the generated software several unit tests and system test were created. Unit test were written for relevant modules - python scripts - the project produced. This was done by usage of the unit test framework provided by module *unittest* from python. These tests are named with prefix *test_* followed by the module name they test. These tests are run by running the testing scripts in terminal directly. For verification and with the purpose of ensuring the quality of the software produced by the project, a final system test was conducted. Here, the molecular simulation was run for Ar (Argon) and the results of the simulation was compared with reference values.

4 High throughput calculations and supercomputer use

To be able to run several simulations in parallel and to speed up the process, the PG had access to the supercomputers at NSC.

4.1 Sigma and Parallel Processing

Throughout this project the PG has had access to the supercomputers at NSC at Linköping University, specifically Sigma. Running MD simulations for many materials requires the power of supercomputers, as the time required to run the simulations would be too large to reasonably run on personal computers. The strength of running simulations on for example Sigma is the possibility to run several simulations in parallel. The main program uses a simple parallelization, as described earlier in 3.6. The large list of atoms objects is subdivided and distributed among computing nodes, as follows

```
    # Run the molecular dynamics in parallell
    if rank == 0:
        jobs = np.arange(0, len(atoms_list), dtype=np.int)
        job_array = np.array_split(jobs, size)
        for i in range(0, size):
            comm.isend(len(job_array[i]), dest=i, tag=i)
            comm.Isend([job_array[i],MPI.INT], dest=i, tag=i)

    l = comm.recv(source=0, tag=rank)
    data = np.ones(l,dtype=np.int)
    comm.Recv([data,MPI.INT],source=0, tag=rank)
    for id in data:
        try:
            md.run_md(atoms_list[id], str(id).zfill(4))
        except Exception as e:
            print("Run broke!:"+str(e))
    comm.Barrier()

if __name__ == "__main__":
    main()
```

Listing 5: Subdividing and distributing atoms list to computing nodes.

By running the simulations in parallel a great quantity of simulations can be done in reasonable time.

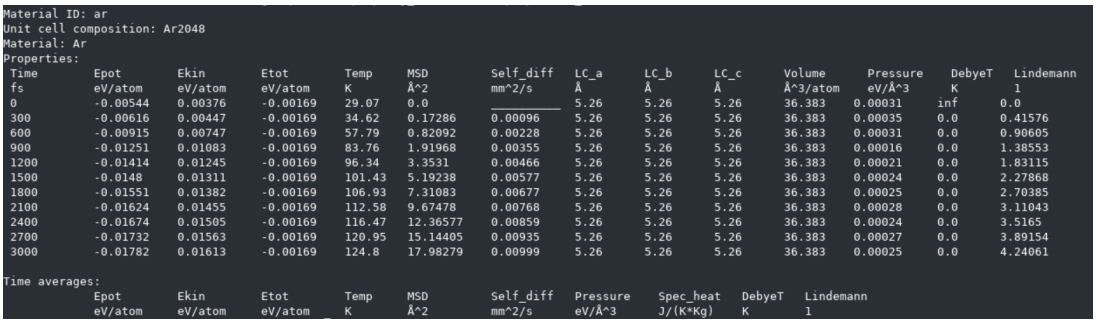
5 Materials Properties

One of the main goals of this project was to analyze several properties of different materials. Some of the properties the PG took interest in is self diffusion and lattice constant.

5.1 Sampling and Calculation

The project was narrowed down to only run MD on orthorhombic structures (i.e. the lattice vectors are orthogonal). On these atomic structures potential, kinetic and total energy, temperature, lattice constant, volume, pressure, self diffusion, mean square displacement and specific heat are calculated. For monoatomic materials the Debye temperature and the Lindemann criterion is also calculated.

Most of these properties can be directly retrieved from the atoms object. The more advanced properties like Debye temperature and specific heat require some more calculation based on the other properties, these calculations are shown in *properties.py*. These properties were printed in a *properties_ID.txt* file for each simulated material, where the *ID* is an identification of the material. An example of such a properties file is given in figure 3.



Material ID: ar
Unit cell composition: Ar2048
Material: Ar
Properties:

Time	Epot	Ekin	Etot	Temp	MSD	Self_diff	LC_a	LC_b	LC_c	Volume	Pressure	DebyeT	Lindemann
fs	eV/atom	eV/atom	eV/atom	K	Å^2	mm^2/s	Å	Å	Å	Å^3/atom	eV/Å^3	K	1
0	-0.00544	0.00376	-0.00169	29.07	0.0	_____	5.26	5.26	5.26	36.383	0.00031	inf	0.0
300	-0.00616	0.00447	-0.00169	34.62	0.17286	0.00096	5.26	5.26	5.26	36.383	0.00035	0.0	0.41576
600	-0.00915	0.00747	-0.00169	57.79	0.82092	0.00228	5.26	5.26	5.26	36.383	0.00031	0.0	0.90605
900	-0.01251	0.01083	-0.00169	83.76	1.91968	0.00355	5.26	5.26	5.26	36.383	0.00016	0.0	1.38553
1200	-0.01414	0.01245	-0.00169	96.34	3.3531	0.00466	5.26	5.26	5.26	36.383	0.00021	0.0	1.83115
1500	-0.0148	0.01311	-0.00169	101.43	5.19238	0.00577	5.26	5.26	5.26	36.383	0.00024	0.0	2.27868
1800	-0.01551	0.01382	-0.00169	106.93	7.31083	0.00677	5.26	5.26	5.26	36.383	0.00025	0.0	2.70385
2100	-0.01624	0.01455	-0.00169	112.58	9.67478	0.00768	5.26	5.26	5.26	36.383	0.00028	0.0	3.11043
2400	-0.01674	0.01505	-0.00169	116.47	12.36577	0.00859	5.26	5.26	5.26	36.383	0.00024	0.0	3.5165
2700	-0.01732	0.01563	-0.00169	120.95	15.14405	0.00935	5.26	5.26	5.26	36.383	0.00027	0.0	3.89154
3000	-0.01782	0.01613	-0.00169	124.8	17.98279	0.00999	5.26	5.26	5.26	36.383	0.00025	0.0	4.24061

Time averages:

Epot	Ekin	Etot	Temp	MSD	Self_diff	Pressure	Spec_heat	DebyeT	Lindemann
eV/atom	eV/atom	eV/atom	K	Å^2	mm^2/s	eV/Å^3	J/(K*Kg)	K	1

Figure 3: Properties file for Argon.

At the end of each properties file are time averages of the properties, if the simulation reached thermal equilibrium. The time averages are calculated for the time steps done after reaching thermal equilibrium. Specific heat is also calculated and written to the end the file.

To calculate the equilibrium lattice constant and the bulk modulus, several MD simulations are required since the simulations are run at constant volume and lattice constants. Multiple simulations can be run for each material with all lattice constants multiplied by a scaling factor (e.g. 0.96, 0.98, 1.00, 1.02 or 1.04), given that the simulations reach thermal equilibrium, a polynomial interpolation of the Energy - Lattice Constant curve gives the scaling factor that gives the lowest energy. Then from a polynomial interpolation of the Energy - Volume curve the bulk modulus is calculated.

After a set of MD calculations has been run the extract() function can be run to gather the time averages of the properties from all the materials that reached equilibrium in a text file called *collected_data.txt*. It is also in this stage lattice constant and bulk modulus is calculated, how these are calculated can be seen in *analysis.py*. These calculations require multiple runs for each material, but only the time averages from the run with lowest energy is written to the text file.

6 Results

The development of the PG lead to MD software that can run simulations. To test the validity of the simulations, an acceptance test was done which yielded values of properties that could be compared to reference values from the literature. In table 3 such a comparison has been done for a simulation of Argon. The simulation was done at 40 Kelvin with the ensemble NVT and with the LJ potential.

Table 3: Comparison of properties for a simulation of Argon at 40 Kelvin.

Property	**Simulation**	**Reference**
Specific heat [J/K/kg]	312.720	312 (at 20°C) [7]
Bulk modulus [GPa]	3.340	2.88 [8]
Debye temperature [K]	91.32	92 (at 0 K) [9]
Equilibrium lattice constants [Å]	5.324	5.26 [10]
Cohesive energy [eV/atom]	0.0586	0.080 [11]

From table 3 it is observed that the simulated values for specific heat, Debye temperature and equilibrium lattice constant are comparable with the reference values. Bulk modulus differs with a factor of 1.15 and cohesive energy differs with a factor of 1.37, which seem to be reasonable differences. In conclusion the results yielded from the simulation are highly reasonable.

6.1 Analysis

Part of the high-throughput data analysis was constructing scatter plots using the data of calculated properties, an example of such a plot is shown in figure 4. One should be careful when interpreting the lattice constant since a general material is determined by three lattice constants, and the angles between basis vectors may not be 90 degrees. Be aware of the structure of the materials that are fetched from materials project. To be safe, set *cubic_only* to true in the settings-file.

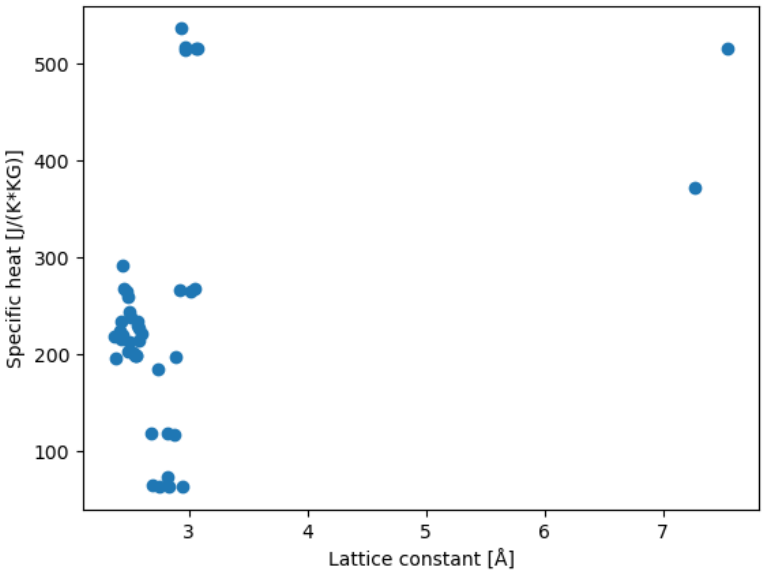


Figure 4: Specific heat plotted against lattice constant for a set of materials.

As is observed from table 3 our simulation can yield reasonable property values for some materials and some potentials. However, in our analysis of mainly pure metals we realized that our chosen LJ potential does not really work well for most of them. A lot of the calculated properties did not, and some did, compare well to reference values. For example we noticed that our values for the specific heat were consistently around half of the reference values. We realized that before starting development a more careful consideration of matching potentials and materials should have been done.

References

[1] Rickard Armiento. Project directive.

[2] Anaconda Inc. Individual edition. [Online]. Available: https://www.anaconda.com/products/individual. [Visited: 20201223].

[3] University Corporation for Atmospheric Research. ASAP tools. [Online]. Available: https://asappytools.readthedocs.io/en/latest/. [Visited: 20201211].

[4] ASE-developers. Atomic Simulation Environment. [Online]. Available: https://wiki.fysik.dtu.dk/ase/. [Visited: 20201211].

[5] E. B. Tadmor and R. S. Elliott and J. P. Sethna and R. E. Miller and C. A. Becker. The Potential of Atomistic Simulations and the Knowledgebase of Interatomic Models. [Online]. Available: https://openkim.org/. [Visited: 20201211].

[6] Materials project. The Materials Project. [Online]. Available: https://next-gen.materialsproject.org/. [Visited: 20201211].

[7] The Engineering Toolbox. Specific Heat and Individual Gas Constants of Gases . [Online]. Available: https://www.engineeringtoolbox.com/specific-heat-capacity-gases-d_159.html. [Visited: 2021-01-11].

[8] Ronald Wayne Wilkins. Low temperature bulk modulus of solid argon.

[9] Knowledge Door. Debye Temperature. [Online]. Available: http://www.knowledgedoor.com/2/elements_handbook/debye_temperature.html. [Visited: 2021-01-11].

[10] Rickard Armiento. Hands-on exercise 3. [Online]. Available: https://liuonline.sharepoint.com/sites/Lisam_TFYA92_2020HT_GU/CourseDocuments/Computational%20physics%20project/Hands-on%20Exercises/hands-on%20exercise%203.pdf?CT=1610379388731&OR=ItemsView. [Visited: 2021-01-11].

[11] Knowledge Door. Cohesive Energy. [Online]. Available: http://www.knowledgedoor.com/2/elements_handbook/cohesive_energy.html. [Visited: 2021-01-11].

E. Users guide

Users guide

Group A

January 13, 2021

Version 1.0

Status

Reviewed		
Approved		

Project Identity

Orderer: Rickard Armiento, Linköping University

E-mail: rickard.armiento@liu.se

Supervisor: Abijith S Parackal

E-mail: abijith.s.parackal@liu.se

Project members

Name	**Responsibility**	**E-mail**
Nicholas Sepp Löfgren (NSL)	Project leader	nicse725@student.liu.se
Emil Jivtegen (EJ)	Documentation	emiji263@student.liu.se
Linda Le (LL)	Scrum master	linle336@student.liu.se
William Stenlund (WS)	Backlogging	wilst106@student.liu.se
Roger Önçü (RÖ)	Nutrition/Group dynamics	rogon956@student.liu.se
Petter Cyrén (PC)	Group dynamics/Nutrition	petcy342@student.liu.se

Contents

List of Tables

Listings

Document history

Version	Date	Changes	Made by	Reviewed
1.0	2021-01-13	Finished document.	All	EJ

1 Introduction

This manual will go through the steps needed to run a molecular dynamics simulation on the software developed by the project group. Some miscellaneous tips will be given in the end.

All products and documents in this project are free to use according to the MIT open license specified on the projects Github repository[1].

2 Getting Started

The software is, as mentioned, available as an open source through given Github repository. The underlying code of the molecular dynamics simulations are available there. To get started with the simulations, do the following steps:

1. Copy the master branch of the Github repository[2].
2. Open a terminal.
3. Install required packages[3].
4. Run "python3 main.py" to run the MD simulation.
5. Run "python3 analysis.py" to make plots and to create collected_data.txt file, where information like bulk modulus and other properties are made.
6. Run "python3 make_mongodb.py" to finally make the database containing the results of the simulation.

These steps runs a simple simulation with the standard inputs in the settings file which have the desired parameters for the molecular dynamics. In the following section the containment of the settings file will be highlighted as the file is important for further simulations. The settings file is meant to be modified depending on the types of simulations the user desires to run.

1 https://github.com/obsqyr/TFYA92-group-A/blob/master/LICENSE
2 https://github.com/obsqyr/TFYA92-group-A/
3 https://github.com/obsqyr/TFYA92-group-A/blob/master/requirements.txt

3 User Input

This section describes which inputs the user can make to run a simulation. The inputs dictates the conditions in which the simulations will be run. Described are the settings file, its fields and materials.

3.1 Settings File

In listing 1 is an example of a *settings.json* file, which is the user's main way of controlling the MD program.

```
{
    "time_step": 3,
    "max_steps": 1000,
    "temperature": 200,
    "ensemble": "NVE",
    "friction": 0.001,
    "decimals": 5,
    "interval": 100,
    "vol_relax": true,
    "LC_mod": 0.01,
    "LC_steps": 2,
    "tolerance": 0.0005,
    "materials": "test_120_materials.json",
    "supercell_size": 8,
    "search_interval" : 10,
    "threshold" : 6,
    "cubic_only" : true
}
```

Listing 1: Example settings.json file.

In table 1 a description is given for each field in the settings file.

Table 1: Table describing all fields in settings file.

Field name	**Data type**	**Description**
time_step	integer	Defines how large time steps the integrators take.
max_steps	integer	Defines for how many steps each simulation is run.
temperature	integer	Initial temperature of the system.
ensemble	string	Defines which ensemble should be used in simulation.
friction	float	Friction coefficient used if NVT is used.
decimals	integer	How many decimals are written for most properties.

interval	integer	Determines how often properties are written to files during a MD run.
vol_relax	boolean	Determines if volume relaxation is run.
LC_mod	float	Scaling factor difference between steps for volume relaxation.
LC_steps	steps	Determines number of steps run for volume relaxation.
tolerance	float	Determines whether the difference between the kinetic energy before and after search_interval amount of steps is small enough to approach equilibrium.
materials	string	The name of a json-object to extract materials from.
supercell_size	integer	Super cell side length in terms of unit cell side length.
search_interval	integer	The amount of steps taken before checking the tolerance.
threshold	integer	The amount of times the kinetic energy difference has to be smaller than the tolerance before equilibrium is achieved.
cubic_only	boolean	Determines if only cubic materials are considered.

3.2 Getting Materials

The software can run MD simulation on materials given in CIF-format. Therefore, the software provides the opportunity to extract data via a database available via *materialsproject.org*. This is however not necessary and the user can at any time choose other databases. In the settings file a json-file can be specified under "materials". What shall be written is the path to the json-file with the data. This can simply be the file name if the file is in the same directory as the script *settings.json*. The only requirements placed on this file is that the json-object contains a key called *response* and that inside it is a key called *cif*, which lists all materials in their CIF-format. In other words, the only real requirement on the database which can be used is that the materials are or can easily be converted into CIF-format. For the user interested in knowing the features of material extraction from *materialsproject.org* the following subsection will go through it in more details.

3.2.1 Getting Materials from materialsproject.org

The extraction of materials from *materialsproject.org* will result in a json-file, which later can be specified in the configuration file if MD simulation is wished to be done on them. To extract materials from *materialsproject.org* simply run the bash script *\query_mp_project.sh <element1> <element2> – <element3> > filename.json*. Here, <element1>, <element2> and <element3> are place holders for different arguments. The example above will result in a query with element3 and at least one of element1 and element2. The results will be put into *<filename>.json*.

4 Produced Data

The data produced can be located in a directory called "property_calculations". When the code is run, a txt-file will be created for each material. The files are named "properties_XXXX" with XXXX being the simulation id number. Each file will contain information about the material name, id number and unit cell composition together with the potential energy, kinetic energy, total energy, temperature, mean square displacement, self diffusion coefficient, lattice constants, volume of the cell, pressure, Debye temperature and Lindemann criterion for each time step of the simulation. It will also contain time averages for properties mentioned above.

4.1 Analysis

To analyse the results, the *analysis.py* file should be run. It will create a file called "collected_data" containing information about the properties for each material. More specific, it will contain values for the time averages of the cohesive energy, mean square displacement, self diffusion coefficient, specific heat, lattice constant, interpolated lattice constant, bulk modulus, Debye temperature and the Lindemann criterion.

When the *analysis.py* file is run, some relevant plots will also be created. As of now, the mean square displacement will be plotted against the self diffusion coefficient and the lattice constant will be plotted against the cohesive energy, interpolated lattice constant and specific heat. If the user wishes to plot other properties, changes in *analysis.py* has to be made.

5 Supercomputer

This section refers specifically to NSC. If the user wishes to run the main program on a parallel computer there are of course many ways to do so. An example of a script to run the main program on the supercomputer Sigma at NSC is given in listing 2.

```
#!/bin/bash
#
#SBATCH -J testjob
#SBATCH -A liu-compute-2020-20
#SBATCH -t 03:00:00
#SBATCH -N 1
#SBATCH --exclusive
#SBATCH -n 32
#
module load Python/3.8.3-anaconda-2020.07-extras-nsc1
module load impi/.2018.1.163-eb
source activate tfya92
time mpirun ~/.conda/envs/tfya92/bin/python3 main.py

echo "job completed"
```

Listing 2: Example script for running main.py on Sigma.

In listing 2 the loaded module "Python/3.8.3-anaconda-2020.07-extras-nsc1" is an Anaconda virtual environment that contains packages required to run the main program. The other loaded module, "impi/.2018.1.163-eb", is a module for loading mpi4py, which is used in *main.py* for parallelization. If these dependencies are considered, the parallell computer should be able to run the main program and do simulations for the user inputted materials.

6 Tips

Here are a few tips on running this program.

1. When trying to run OPTIMADE remember that instead of manually dowloading all relating packages, run the *setup.py* file. Do this by running "python3 setup.py build" followed by "python3 setup.py install" to install all dependencies.

2. If you run in to the error "Illegal instruction (core dumped)", try running another virtual desktop environment. One example could be "rdpklienter".

F. Group contract

Written: 2020-09-03, Linda Le
Revised by: Petter Cyrén, Emil Jivtegen, Nicholas Sepp, William Stenlund and Roger Öncu.

Group Contract

The following contract is written and valid as long as the CDIO project, course TFYA92 at the University of Linköping, Sweden, is ongoing. Hence, lasts till the end of the course.

1. The group is structured with the following roles:
   Project leader: Nicholas Sepp Löfgren
   Responsible to oversee that the project achieves the primary objectives within the budges.
   Scrum master: Linda Le
   Reponsible for maintaining the Scrum process.
   Product owner: William Stenlund
   The main responsible person for maintaining the product backlog.
   Documenation keeper: Emil Jivtegen
   Person mainly responsible for overseeing that documentation is produced, delivered/archieved and up to standards.
   Development team members: Petter Cyrén
   The rest of the group members. Responsible for fika.
2. All group members are to respect each others opinion and be willing to listen to perspectives that they themselves might not have.
3. Only in case of absolute necessity should majority vote be used. Otherwise, the group should strive for a dynamic where all members feel they can be heard and discussion and diplomacy should be used.
4. The group should strive towards delivering necessary documents the day before. Near this time members should be reachable.
5. The group will foremost deliver the results asked for, and when resources allow, do the best they can to get better results.
6. All tasks should on average be assigned so each member work roughly the same amount time wise.
7. Meeting times should be respected, but some lateness can be tolerated if done with moderation, late for 1 minute, or with special reasons.
8. Latex will be used for documentation. Discord (and teams) will be used for communication. During this time only near deadlines or absolutely necessary should the group require that a member is available during the weekends. Otherwise, the group should strive for only asking their members to work on weekdays.
9. The group should strive towards working/be reachable only during working hours 8-17:00. Only during absolute urgency, or the member themselves willing to, will they work after these hours.
10. If internal conflicts should arise, then the project leader with primary responsibility should make sure it's discussed. In the worse case scenario, the mentor should be involved.

Linda Le, Linda Le
Nicholas Sepp Löfgren
Petter Cyrén
Roger Öncu
Emil Jivtegen
William Stenlund