Diffusion Transport in nanoporous SiO_2

Fys2880

Joseph Knutson
github.com/mathhat

January 10, 2017

Abstract

Contents

Chapter 1

Introduction

sdgsg

Chapter 2

Theory

Molecular Dynamics (MD)

Molecular Dynamics is a computer simulation method for estimating the motion of atoms and molecules. The method creates N-body simulations that show how dynamic systems of particles evolve over time. To dictate how the particles move, we apply interatomic potentials. These potentials simulate energies and forces that puts our system in motion. By mathematically integrating Newton's equations of motion, we can find properties such as velocity, local temperatures and diffusion. The catch of solving N-body problems through MD integration is that errors will accumulate over time. To minimize errors, one has to choose the proper algorithm and parameters for the system.

The MD Program

A Molecular Dynamics program begins with initiating a state, then simulating it over time through mathematical means. This section is a simple overview of how an MD program is set up.

Initiating a state

In order to begin an MD program, you first need an initial state:

- **Initial Positioning**

  Particles are initially placed in either ordered or random positions. Another way to initiate the particles' positions is to load a previous state from an earlier simulation. We're going to use all three methods.

- **Initial Velocity**

  The particles' initial velocities follow a random (gaussian) distribution related to the initial temperature.

- **Potential** In order to make our particles act realistically, we apply an interatomic potential. This potential dictates how the particles behave. A potential creates forces between the particles, making them attracted to each other, or the opposite. Potentials between different particles usually spawns different behaviour. Hydrogen atoms will for example interact differently with oxygen than silicat.

Bringing the System to Life

After creating an initial state, you want movement. The initial velocity and the interatomic potential will from here on decide where our particles will travel. In order to simulate the particles' paths, we need to integrate Newton's equations of motion over a time step. A time step is a small interval of time in which our system goes from its present state to a new one.

Imagine an apple falling from a tree. Initially, it is connected to a tree branch, its velocity equals zero. Gravity then does its work. A second later, the apple is disconnected from the branch. The apple now has a new position and a new velocity. Another second passes and the apple is closer to the ground, while the velocity keeps increasing.

This is an example of an MD program where the particle is an apple. When the apple was connected to the tree branch phase, the system was in its initial state. The gravity acted as the potential and the seconds as time steps. In order to foresee where the apple would be a second in the future, one would simply need the apple's present position, velocity and the gravitational force, relative to the Earth.

From the falling apple example, we understand that position and velocity, as well as gravity is essential for knowing where the apple would end up in the next time step.

Here is a technical formula for bringing a system to life:

- **Gather Present Properties** Gather information such as the position and velocity of the particles. Calculate the forces working on each particle by looking at positioning and potential.
- **Use Present Properties to Simulate Future State**

  Possessing the present forces at work, we can numerically integrate these forces over a time step (fragment of a second). When we do this, we're given the particles' positions and velocities a step forward in time.
- **Repeat**

  New positions, means new forces. These forces can again be integrated and give birth to more changes in position and velocity. The amount of repetitions, or *runs*, usually depends on your problem. In our apple example, if your timestep is as big as a second and your goal is for the apple to reach the ground, you would only need 5 runs, 5 seconds. The simulation time will depend on the amount of runs, the size of your system and your computational power.

Bear in mind that for each run you simulate, more error will accumulate.

- **Sample**

  After your program has finished running, dump/sample the data for later simulations and analysis.

Structure of Code

I have chosen to use LAMMPS as my code language. LAMMPS is a classical molecular dynamics code, and an acronym for Large-scale Atomic/Molecular Massively Parallel Simulator.[1] This section aims to give you a basic intuition of how LAMMPS algorithms are structured. For a random simulation, the structure of a program file will typically look like this:

```
# ---------------- Init Section ----------------

include "system.init"
//Initializes units and simulation box


# ---------------- Atom Definition Section ----------------

read_data "system.data"
//Loads presaved state

# ---------------- Settings Section ----------------

include "system.settings"
//Defines particles and potentials between them


# ---------------- Run Section ----------------

run   5000
//Integrates over 5000 time steps

write_data filename.data
//Sample/save the resulting state
```

The first line of code includes a file called "*system.init*". Including is merely adding all the code written inside the file you include. Inside this file lies the code that defines what units were using. SI units are not often practical since we're dealing with relatively small values. The initialization process also defines the volume and shape of the system. Our simulations will be done inside a box.

The second line of code has a command called "read_data". This command looks for a data file, typically containing information about a state from a previous simulation. Sometimes the data file contains much less than a system of particles, e.g. a single water molecule. The molecule can then be multiplied and viola, you have a system of water.

The third line of code includes a file called "*system.settings*". After having loaded a state of particles, they have already been assigned types. This part of

[1] http://lammps.sandia.gov/

the program tells particles how to interact with each other and how much each individual particle type is supposed to weigh. The last part of setting up your system is usually assigning a temperature.

The last lines of code brings the system in motion. The line orders the system to move one time step towards the future, 5000 times. The "write_data" command, saves the final state inside a datafile.

Chapter 3

The Problem

The goal of this project is to analyze the behaviour of water molecules floating around in nanoporous SiO_2. Water, as any other fluid, is in constant motion. This inner fluidic motion is known as self diffusion. Our goal is to determine wether or not water's self diffusion changes as a result of changes to its surroundings. We are going to look at 2 types of nanopores. Spherical and cylindrical. The pores will be placed inside of silicate, or SiO_2, and filled with water. What you know as glass, crystall, quarts and many other materials are all made up of silicate.

Our model of nanopores starts off with a block of betacristobalite, see figure 3.1.

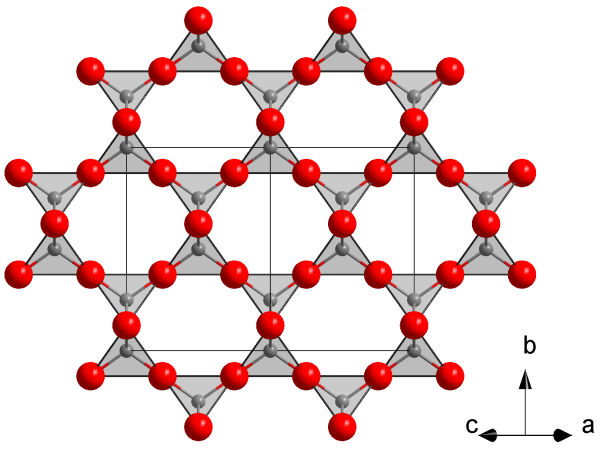

Figure 3.1: Silicat in the form of β cristobalite.

chn

To look at water inside of SiO_2 chambers, we first need to create the chambers.

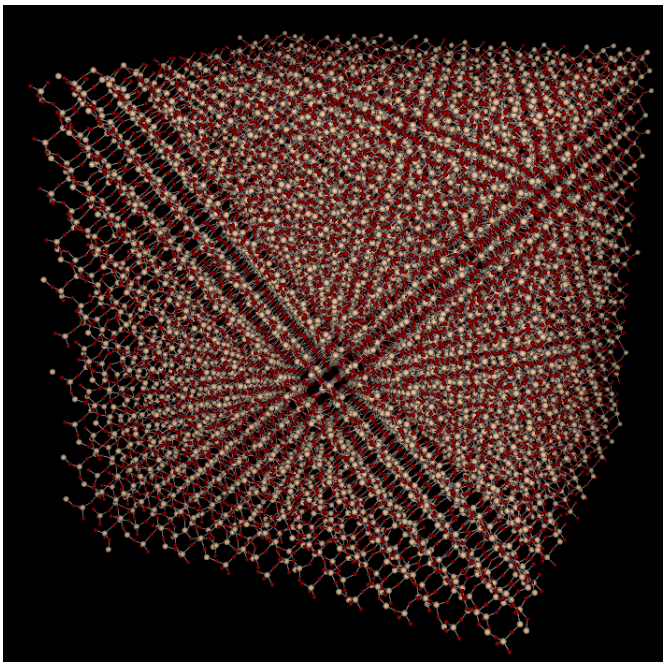

Figure 3.2: Block of Silicat in the form of betacristobalite.

Chapter 4

Methods

This chapter discusses our general methods of Molecular Dynamics in further detail and the other methods relevant to the problem.

Bibliography

[1] Hiorth-Jensen, M., *Project 2, Computational Physics I FYS3150/FYS4150*, University of Oslo, 2016.

[2] Abel, N.H., Memoire sur le équations algébriques, ou l'on démontre l'impossibilité de la résolution de l'équation générale de cinquiéme degré, In Sylow, L. and Lie, S., *Œuvres Complètes de Niels Henrik Abel*, 2nd ed., Grøndahl & Søn, pp. 28-33, 1881.

.1 Laguerre Polynomials

Generally, the name Laguerre polynomials is used for solutions to

$$x\frac{d^2y}{dx^2} + (\alpha + 1 - x)\frac{dy}{dx} + ny = 0. \tag{1}$$

These polynomials, usually denoted, L_0, L_1, L_2 etc are a polynomial sequence which may be defined by the Rodriguez formula

$$L_n(x) = \frac{e^x}{n!}\frac{d^n}{dx^n}(e^{-x}x^n) = \frac{1}{n!}\left(\frac{d}{dx} - 1\right)x^n \tag{2}$$

The first few Laguerre polynomials are shown in table 1.

Table 1: The first few Laguerre polynomials

n	$L_n(x)$
0	1
1	$-x + 1$
2	$\frac{1}{2}(x^2 - 4x + 2)$
3	$\frac{1}{6}(-x^3 + 9x^2 - 18x + 6)$
4	$\frac{1}{24}(x^4 - 16x^3 + 72x^2 - 96x + 24$
5	$\frac{1}{120}(-x^5 + 25x^4 - 200x^3 + 600x^2 - 600x + 120)$
6	$\frac{1}{720}(x^6 - 36x^5 + 450x^4 - 2400x^3 + 5400x^2 - 4320x + 720)$