Vienna University of Technology

Computational Mathematics - Seminary

Institute of Analysis and Scientific Computing

Shape Optimization

Authors:
Camilo Tello Fachin
12127084

Paul Genest
12131124

Supervisor:
Dr. KevinSturm

July 4, 2022

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Abstract in Foreign Language

Here you put in your nice and smooth home country language abstract or summary! In Abbildung **??** sehen sie blablabla

Abstract

Here you put your English Abstract!

Acknowledgement

Here you should thank Me, maybe your mum, your profs and supervisors and whoever decided to help your poor soul.

Contents

Post-Processing

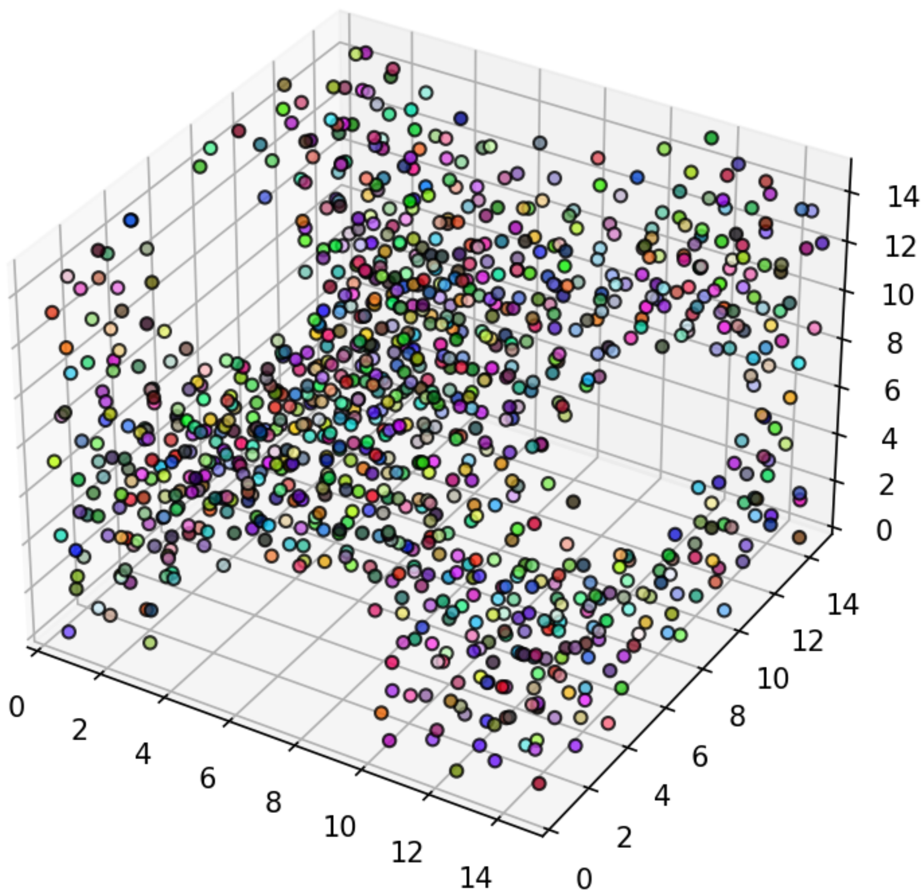

Figure 1: Initial Condition for Molecular Dynamics Simulation with 1000 Particles at 300K

Figure 1 shows the initial condition which is generated from `task2.py`. The positions are at a local minimum which is numerically obtained by the `scipy.optimize.minimize()` funtcion with the conjugate gradient method. The CG method needs the gradient to obtain the minimum which is passed as an argument to the function. Not visible in the figure above, are the initial velocites which are also randomly assigned to every particle with the `numpy.random.multivariate.normal()` function. This initial condition, namely positions and velocities of each particle, are written to a txt file which is interpreted by pythonfile `task3.py`. In said file the trajectories of all thsee particles are calculated by integration of Newtons equations of motion, this part is called the velocity verlet algorithm

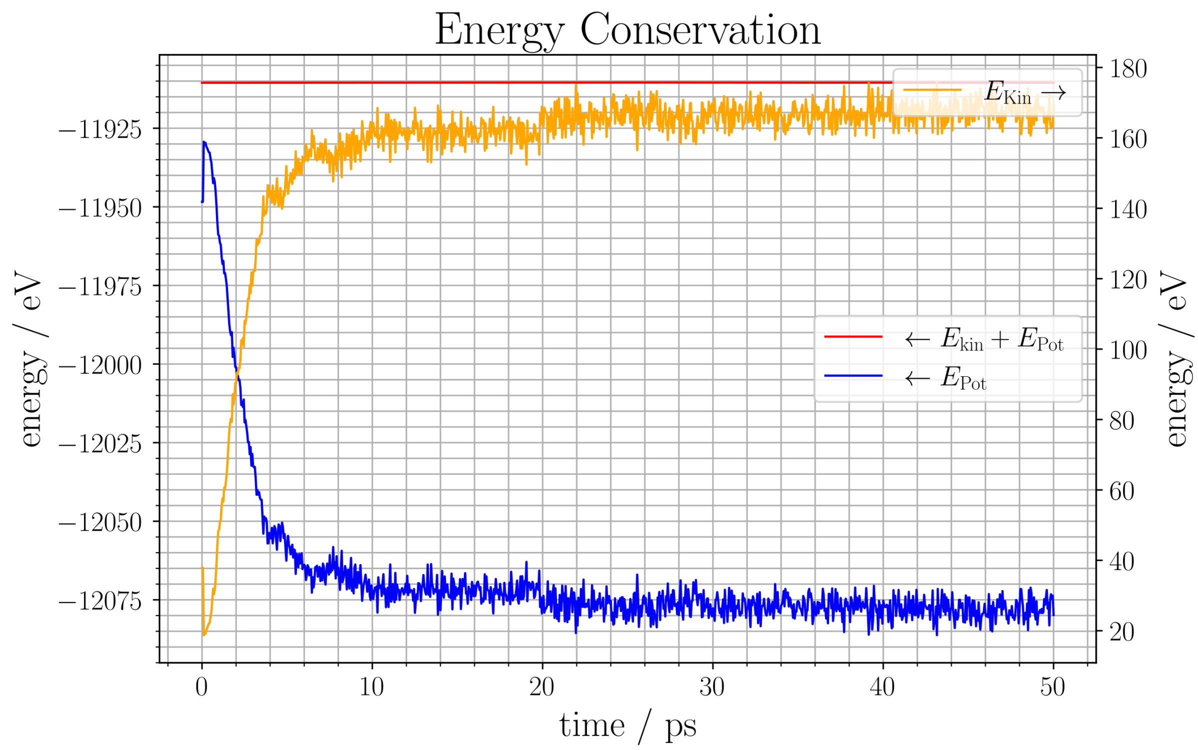

Figure 2: Energy Conservation Plot for 10^6 timesteps of 0.5 femtoseconds

In figure 2 above, the energy conservation is shown. Since this molecular dynamics simulation result is obtained numerically, errors are expected. The main error source being the velocity verlet algorithm, for the chosen timestep, the error is negligible. The timestep in this simulation was reduced until a reasonable energy conservation was observed. The timestep chosen was 0.05 femtoseconds.

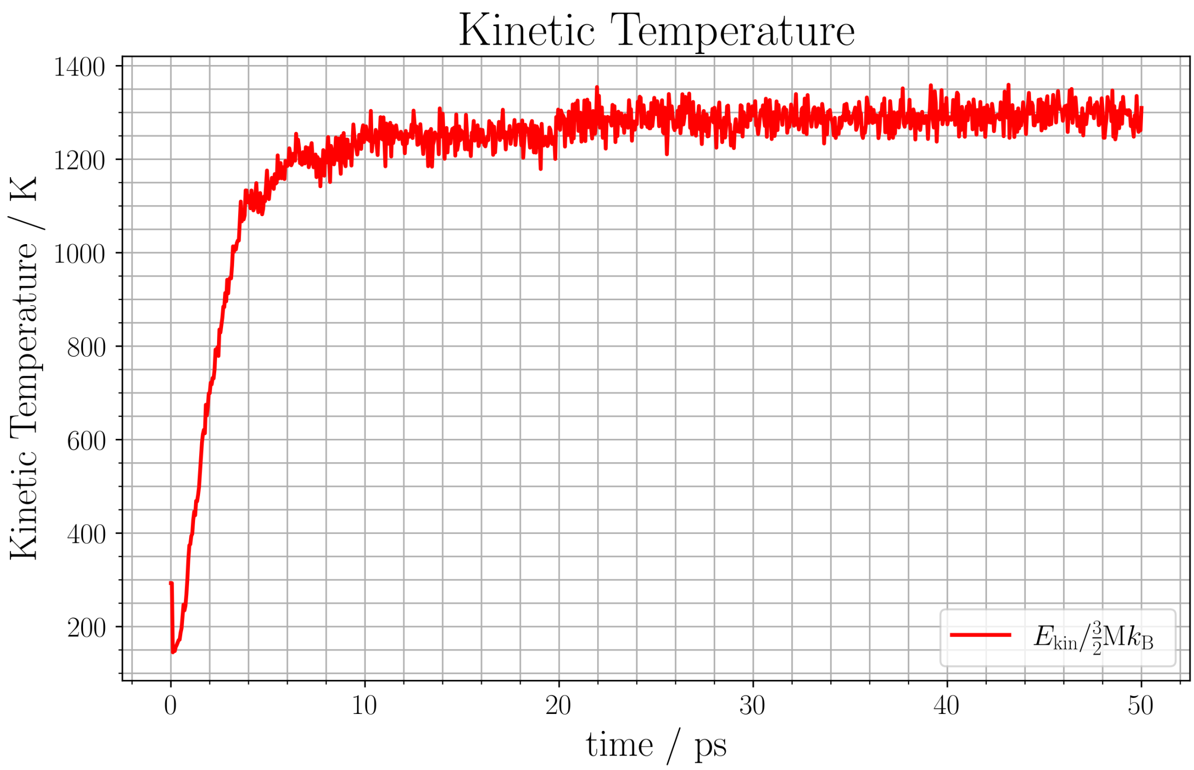

Figure 3: Evolution of Kinetic Temperature for 10^6 timesteps of 0.05 femtoseconds

In figure 3 the evolution of the kinetic temperature over the simulated time is plotted. At $t = 0$, The kinetic temperature is at 300K, which is in agreement with the velocities assigned in the initial conditions. The final state where potential and kinetic energy are not changing each other anymore, it reaches a kinetic temperature of approximately 1300 Kelvin.

In order to obtain a specific heat, an approximation formula was provided, where 25 % of the trajectory data, timewise, had to be neglected. Said approximation formula yielded a quantity in J/K. If this quantity is divided by the mass of the system, the weight specifc heat capacity can be obtained. This coefficient is used to model heat transport in solids as well as in fluids, an example would be the poisson equation.The calculated specific heat capacity was 17.23887287974392 $\mathrm{J}/(\mathrm{kg}\cdot\mathrm{K})$. A comparable material would be hydrogen with a specific heat capacity of 10.16 $\mathrm{J}/(\mathrm{kg}\cdot\mathrm{K})$. However, other than the equations of motion on the shape of the morse potential function, these simulation results do not have physical meaning.

1 To Do's

- What's still to you do make your supervisor/prof happy?

A Python Code Listing

Here is an example of a python listing, you can change appearance of comments, strings, numbering, known commands and variables in the package settings in packages.tex. You can obviously use the listings environment in the rest of the document. The same procedure applies for listings in other languages.

Python Listing Title

```python
# Python Script, API Version = V18

import math

#    DELETE EVERYTHING --------------------------------

ClearAll()

#    PARAMETERS ------------------------------------------

w = float(Parameters.w)      # side length of one element or half of a unit cell
e = float(Parameters.e)      # rectangle ratio e
b = w/(1+e)
rho = float(Parameters.rho) # relative density
f = float(Parameters.f)      # number of layers = folds+1
h = 2*w/f          # layer height = size of a unit cell divided by the number of layers
f = int(Parameters.f)

# Calculation of wall thickness t
t1 = ((math.sqrt(1-rho)+1)*math.sqrt(2)*w)/2
t2 = -((math.sqrt(1-rho)-1)*math.sqrt(2)*w)/2
if t1 <t2:
  t=t1
else:
   t=t2

# auxiliary variable to build up rectangle
m = math.sqrt(pow(t,2)*2)/2
```

B XMl Code Listing

Here is an example for XML code listing.

XML Listing Title

```xml
<extension version="1" name="EnergyIntegral" loadasdefault="True">
  <guid shortid="EnergyIntegral">8005c624-8869-4c74-b32b-97ac59c200b2</guid>
  <script src="energy_integral.py" />
  <interface context="Mechanical">
```

C MATLAB Code Listing

Here is an example for MATLAB code listing

MATLAB Listing Title

```matlab
%% Linear model Poly44 from MATLAB Curve Fit App:

%Polynomial Coefficients (with 95\% confidence bounds):
        p00 =       13.79;  %(13.22, 14.36)
        p10 =      -2.897;  %(-3.454, -2.34)
        p01 =       3.752;  %(3.163, 4.34)
        p20 =       3.279;  %(2.231, 4.327)
        p11 =      0.5404;  %(-0.2001, 1.281)
        p02 =      0.8638;  %(-0.4624, 2.19)
        p30 =       0.299;  %(0.01281, 0.5851)
        p21 =     -0.5091;  %(-0.7299, -0.2884)
        p12 =      0.4973;  %(0.2716, 0.7229)
        p03 =      0.3595;  %(0.04484, 0.6741)
        p40 =     -0.8495;  %(-1.291, -0.4084)
        p31 =    -0.02258;  %(-0.3136, 0.2685)
        p22 =     -0.2819;  %(-0.5502, -0.01351)
        p13 =      0.2674;  %(-0.05265, 0.5874)
        p04 =      0.2019;  %(-0.3968, 0.8006)

   f(x,y) = p00 + p10*x + p01*y + p20*x^2 + p11*x*y + p02*y^2 + p30*x^3 + p21*x^2*y
   + p12*x*y^2 + p03*y^3 + p40*x^4 + p31*x^3*y + p22*x^2*y^2
   + p13*x*y^3 + p04*y^4

  %Goodness of fit:
  %SSE: 3.189
  %R-square: 0.9949
  %Adjusted R-square: 0.9902
  %RMSE: 0.4611
```