NTNU – Trondheim
Norwegian University of
Science and Technology

Polymer failure modeled by Molecular Dynamics

Kristian Berg Keilen

Materials Science and Engineering
Submission date: June 2014
Supervisor: Andreas Echtermeyer, IPM

Norwegian University of Science and Technology
Department of Engineering Design and Materials

Til minne om Bjørn Andersen
16.2.1951 – 28.1.2014

Hjelp til med kreftforskning ved å lese appendiks 2!

THE NORWEGIAN UNIVERSITY

OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF ENGINEERING DESIGN
AND MATERIALS

MASTER THESIS SPRING 2014

FOR

STUD.TECHN. KRISTIAN KEILEN

Polymer failure modeled by Molecular Dynamics
Brudd av polymere modellert med Molecular Dynamics

Long-term properties of polymers are important to understand well when designing critical applications that shall have long lifetimes. Molecular Dynamics (MD) is a method that can simulate polymer behavior based on first principles. This method can potentially be used to replace long lasting experiments.

It shall be investigated how polymer chain breaking mechanisms of a typical semi crystalline polymer can be modeled with MD. The simulations shall be based on atomic potential calculations for the selected polymer. A qualitative comparison to experimental results from the literature shall be made.

Three weeks after start of the thesis work, an A3 sheet illustrating the work is to be handed in. A template for this presentation is available on the IPM's web site under the menu "Masteroppgave" (http://www.ntnu.no/ipm/masteroppgave). This sheet should be updated one week before the Master's thesis is submitted.

Performing a risk assessment of the planned work is obligatory. Known main activities must be risk assessed before they start, and the form must be handed in within 3 weeks of receiving the problem text. The form must be signed by your supervisor. All projects are to be assessed, even theoretical and virtual. Risk assessment is a running activity, and must be carried out before starting any activity that might lead to injury to humans or damage to materials/equipment or the external environment. Copies of signed risk assessments should also be included as an appendix of the finished project report.

The thesis should include the signed problem text, and be written as a research report with summary both in English and Norwegian, conclusion, literature references, table of contents, etc. During preparation of the text, the candidate should make efforts to create a well arranged and well written report. To ease the evaluation of the thesis, it is important to cross-reference text, tables and figures. For evaluation of the work a thorough discussion of results is appreciated.

The thesis shall be submitted electronically via DAIM, NTNU's system for Digital Archiving and Submission of Master's thesis.

Torgeir Welo	Andreas Echtermeyer
Head of Division	Professor/Supervisor

NTNU
Norges teknisk-
naturvitenskapelige universitet
Institutt for produktutvikling
og materialer

ACKNOWLEDGEMENTS

I will of course first think my mum and dad for all the support during my years in higher education. Further thanks go to Iwan Sahputra for having occasional meetings with me and giving me tips on how to proceed if I were stuck.

Before taking this project, I asked for a six week summer work, which was obliged me by Andreas Echtermeyer and Jon Harald Lambert Grave. I am sorry that it took so long for me to pull myself together! Further thanks go to Nils Petter Vedvik, Qazi Sohail Ahmad, Jorunn Hvalby and Halvard Støwer at the faculty and institute for helping me with some technical issues, and finally to the other master students both on IPM and IMT, whom have run the same 5-6 year race with me.

Faust.
Wohin soll es nun gehn?

Mephistopheles.
Wohin es dir gefällt.
Wir sehen die kleine, dann die große Welt.
Mit welcher Freude, welchem Nutzen,
Wirst du den Cursum durchschmarutzen!

Faust I, Vers 2051-2054

SAMMENDRAG PÅ NORSK

Denne oppgaven er forfatted som en obligatorisk del av masterstudieprogrammet Materialteknologi ved NTNU, våren 2014. Oppgaven har blitt gjort for Institutt for Produktutvikling og Materialer.

Molekyldynamikk (MD) er et forskningsfelt som studerer vekselvirkninger mellom atomer. MDs størrelsesorden er veldig liten, på størrelse med Ångstrøm i rom og nanosekund i tid. Allikevel kan vi med bruk av resultater fra termodynamikk og statistisk mekanikk bruke MD til å spå hendelser på makroskopisk nivå.

MD er mye brukt til å modellere molekyler, og denne oppgaven tar sikte på å studere et spesielt fenomen: båndbryting. Båndbryting kan skje i mange tilfeller, men i denne oppgaven ønsker vi å studere irreversibel båndbryting ved stor båndforlengelse for polyetylen. Ved hvilken båndlengde kan man si at båndet har brukket?

Modellen vi skal bruke er en forent atom (UA) polyetylenmodell med kovalente bånd som har form av Morse-potensialet. Vi skal forsøke å finne en båndbrytingsavstand som gir resultater som kan sammenliknes med eksperimentelle resultater, slik at denne modellen kan forhåpentligvis brukes til videre forskning senere.

MD er et fagfelt som krever en del forkunnskap i flere felt som kvantemekanikk, statistisk mekanikk og termodynamikk. I denne oppgaven har jeg skrevet en omfattende introduksjon til emnet, både teoretisk og bruk av dataprogram som LAMMPS, slik at en som ikke kjenner så godt til MD forhåpentligvis skulle kunne lese seg opp på det grunnleggende, nok til å kunne gjøre noen MD-simuleringer selv.

ABSTRACT IN ENGLISH

This thesis is written as a mandatory part of the master degree program Materials Science and Engineering at NTNU, spring 2014. This thesis has been done for the Departement of Engineering Design and Materials.

Molecular dynamics (MD) is a field of science that studies interactions between atoms. MD operates on very small order of magnitudes, in space with Angstroms and in time with nanoseconds. Even with these small scales it is possible to reproduce events on a macroscopic level, due to some results of thermodynamics and statistical dynamics.

MD is often used to model molecules, and in this thesis we will look at a distinct phenomenon: Bond scission. The breaking of bonds can happen in many cases, but in this thesis we wish to study irreversible bond breaking as a result of large strains for polyethylene. At what bond length can we say that the bond is broken?

The model that we will use is a united atom (UA) polyethylene model with covalent bonds that have bond energy given by the Morse potential. We shall attempt to find bond breaking lengths that give strains that may be comparable with experimental results, so that this model can be used in further research down the line.

MD is a science that builds upon a lot of other sciences, like quantum mechanics, statistical mechanics and thermodynamics. In this thesis, I have written a comprehensive introduction to the field, both in theory and in application of computer programs like LAMMPS, in hopefully such a way that one who is not familiar with MD can get a good enough understanding of the field to do some simulations themselves.

CONVERSION TABLES

Many units in this thesis, in both text and figures, are given as the units that LAMMPS uses in its output calculations (*"LAMMPS WWW site," 2013; Plimpton, 1995*). A unit that can be particularly hard to grasp is the one used for force: kcal/mol/Å. The tables underneath give conversion factors between common units for quick referencing (*Wolfram|Alpha, 2013*).

Energy	**kcal/mol**	**eV**	**kJ/mol**	**E_H**
1 kcal/mol	1	0.0434	4.184	0.00159
1 eV	23.1	1	96.5	0.0367
1 kJ/mol	0.239	0.0104	1	$3.81 \cdot 10^{-4}$
1 E_H (hartree)	627	27.2	2 630	1

Distance	**Å**	**nm**	**a_0**
1 Å	1	0.1	18.9
1 nm	10	1	189
1 a_0 (bohr)	0.529	0.0529	1

Force	**kcal/mol/Å**	**eV/Å**	**N**	**E_H/a_0**
1 kcal/mol/Å	1	0.0434	$6.95 \cdot 10^{-11}$	$8.43 \cdot 10^{-4}$
1 eV/Å	23.1	1	$1.60 \cdot 10^{-9}$	0.0194
1 N	$1.44 \cdot 10^{10}$	$6.242 \cdot 10^{8}$	1	$1.21 \cdot 10^{7}$
1 E_H/a_0	1190	51.4	$8.24 \cdot 10^{-8}$	1

Pressure	**atm**	**bar**	**Pa**
1 atmosphere	1	1.013	101 325
1 bar	0.987	1	$1 \cdot 10^{5}$
1 Pa	$9.87 \cdot 10^{-6}$	$1 \cdot 10^{-5}$	1

Note on Figures

Figures are mostly self-made, with some exceptions where sources are listed in the figure text. I have used programs such as Avogadro (*Hanwell et al., 2012*) for simple molecule drawings, OVITO (*Stukowski, 2010*) (see also Part C, section 3.7) for visualization of the simulations, GeoGebra[1] for geometrical drawings and MATLAB for graphs.

List of Abbreviations

Abbreviation	Meaning	See page
µCE	Microcanonical ensemble	36
AA	All-atom model	44
BOA	Born-Oppenheimer Approximation	25
CE	Canonical ensemble	37
DFT	Density functional theory	28
EA	Explicit atom model, same as AA	44
EAM	Embedded atom method	65
F@H	Folding@Home	99
HDPE	High-density polyethylene	11
LAMMPS	Large-scale Atomic/Molecular Massively Parallel Simulator	58
LCAO	Linear combination of atomic orbitals	28
LDPE	Low-density polyethylene	11
LJ	Lennard-Jones (potential)	46
MD	Molecular dynamics	1
MM	Molecular mechanics	1
MS	Molecular statics	78
OVITO	The Open Visualization Tool	72
PBS	Portable Batch System	98
PE	Polyethylene	7
UA	United atom model	44
UHMWPE	Ultra-high molecular weight polyethylene	16
VV	Velocity Verlet	71

List of Symbols and Operators

Symbol	Meaning
A	Thermodynamic dummy variable
A	Helmholtz free energy
A(t)	Gaussian noise
a,b,c	Lattice constants
A,B,C,…	A thermodynamic system
a_0	Bohr radius ≈ 0.529 Å for hydrogen
c	Speed of light, speed of a wave
$Đ_M$	Molar mass dispersity

[1] http://www.geogebra.org

E	Total energy
E	Modulus of elasticity
F, f, P	Force
G	Gibbs free energy
h	Planck constant ≈ $6.6261 \cdot 10^{-34}$ J · s
H	The Hamiltonian operator, The classical Hamiltonian
H	Enthalpy
$\hbar$	The reduced Planck constant = h/2π
i	The imaginary unit $= \sqrt{-1}$
J	Creep compliance
k	The wave vector
K, θ^0	Parameters for the harmonic bond angle potential
k^b	Bond stiffness constant
k^b, K, r^b	Parameters for the harmonic bond length potential
K_i, A_i	Parameters for the dihedral angle potential
M	Number of degrees of freedom
m, m_e, M	Mass of a particle, either an electron (m_e), nucleus or quasiatom, total mass
n	Normal vector
n	Number of particles
n, l, m, s	Quantum numbers
N_A	Avogadro's number $\approx 6.022 \cdot 10^{23}$
p	Momentum of a particle, generalized momentum
p	Pressure
$\tilde{P}$,	Dimensionless force
q	Generalized position
q	Heat
r	Position of a particle
r, l	Bond length
r^0	Reference equilibrium spacing
r^b	Reference equilibrium bond length
$r^{bondbreak}$	Theoretical 1D bond breaking length
r_{break}	Bond breaking length
r_c, r_{cutoff}	The cutoff radius for the pair potential
R_E	Rydberg energy ≈ 13.6 eV
r^{eq}	Equilibrium bond length
r_{ij}	Distance between atom i and atom j
r^{tr}	Bond length at transition state
S	Vector space basis
s	Step direction for optimization algorithm
T_G	Class transition temperature
U	Potential energy
U	Internal energy
V	Vector space
V	Volume
W	Work

w	Weight fraction
$\mathbf{x}$	System vector for optimization algorithm
α, D, r^b	Parameters for the Morse potential
β	Unit cell angle for the monoclinic unit cell
β^n	Conjugate gradient scalar coefficient
Γ	Drag coefficient
ΔE_b	Activation energy for bond breaking
Δt	Time step
ε, ε_{ij}, σ, σ_{ij}	Parameters for the Lennard-Jones potential
η	Viscosity
θ^0	Reference angle
λ	Wavelength
λ	Step length for optimization algorithm
μ	Reduced mass of nucleus-electron system
μ_i	Chemical potential
ν	Frequency
ρ	Electron density
σ_f, ε_f	Fracture stress and strain
σ_m	Ultimate tensile strength
σ_y, $\sigma_{y,0.002}$	Yield stress, yield stress with 0.2 % offset
σ_{yu}, σ_{yl}	Upper and lower yield point
φ	The dihedral angle
Φ, φ	Potential energy function
φ_{nlm}	Hydrogen-like orbital
χ	The improper dihedral angle
χ	Spin orbitals
$\psi(\mathbf{r},t)$	The wave function
Ω	Grand canonical potential

Operator	**Meaning**
∇	Nabla operator
∇^2	The Laplace operator
*	The complex conjugate
^	The Fourier transform
δ	Inexact (path-dependent) differential
$\delta(t-0)$	Dirac delta distribution
^{T}	Matrix transpose

TABLE OF CONTENTS

PART A: INTRODUCTION

Molecular mechanics (MM) is a method for simulating the mechanical interactions of atoms. MM can trace its roots back to 1891, when William Sutherland, an Australian physicist put 5 colored marbles and 100 other marbles in a box, and measuring the diffusion of the colored marbles after shaking the box.

By the 1950s, computer research had come so far that one could seriously do calculations with more than a few atoms. In 1957 the scientists Berni J. Alder and Thomas E. Wainwright did a simulation of 32 particles in a box with periodic boundary conditions. The calculations were done using Monte Carlo simulations on a UNIVAC. Two years later, a simulation of 108 particles were done by the same authors on an IBM 704. From these simulations, the authors could calculate some thermodynamics of the system.

The idea behind molecular mechanics is that the forces between atoms can be described by contributions from interatomic energy, bond stretching, bond angle bending and twisting and so on. The interatomic energy is caused by electrostatic forces for polar molecules, and van der Waals forces for nonpolar molecules. These forces would be expressed as potential energy fields, which would penalize any atomic conformation that strayed from the reference configuration (a bond too long or too short, etc.). Taking this one step further, one can calculate the force for a single atom by differentiating the potential energy: $\boldsymbol{f} = \frac{\partial}{\partial \boldsymbol{r}} U$, and using Newton's second law $\boldsymbol{f} = m\ddot{\boldsymbol{r}}$ to calculate the trajectories of atoms. This is called molecular dynamics (MD).

Using this, J. B. Gibson, A. N. Goland, M. Milgram and G. H. Vineyard (1960) developed a model for radiation attacking metallic cobber. The novel idea was to numerically integrate Newton's second law, and updating the positions of atoms like that. From that time, the field has continued to grow and develop, and with the growing computer power it is in our days possible to calculate millions of atoms in a single system. A problem with this method is that the time step for doing numerical integration is so small ($\sim$1fs), that simulations with millions of time steps will only catch the fastest phenomena. Mr. Sutherland's diffusion experiments would be difficult to do.

Nevertheless, molecular mechanics is still used for studying not only molecules, but also metals and medicine. MM works best when one wants to test something out of the ordinary, like how a crack in the metal will affect its mechanical properties or when an amino acid in a protein is misplaced and so on.

In this thesis we will perform molecular dynamics simulations on short and long n-alkane chains (normal alkane, i.e. unbranched alkane), and try to develop a model for bond breaking of polymers under tensile testing.

It is assumed that the reader is familiar with concepts with fundamental mechanics, calculus and statistics. We will also refer to Taylor expansions several time in this thesis.

Note on citations. I will occasionally cite books and book sections in this thesis. Due to a peculiarity in the citation software, a book will be cited as *author, year + a*, and sections from that book will be cited as *author, year + b* (or *+ c*, *+ d* and so on). For example, a book that we have often used is (Ellad B. Tadmor & Miller, 2011b), and a section from it on cluster potentials is (Ellad B. Tadmor & Miller, 2011a). The page numbers are listed in the bibliography. In addition, to papers published by the same author in the same year will be formatted in the same way: (*Jones, 1924a*) and (*Jones, 1924b*).

Part B: Theory

This section will introduce the theoretical foundation of molecular mechanics. First, we need to invoke some key results from thermodynamics for later use. Afterwards, we will move on to quantum mechanics, which we will increasingly simplify to get a workable model for atomistic simulations. Lastly, we will invoke some results from statistical mechanics, which we will use to relate the microscopic results from molecular mechanics to the macroscopic results from thermodynamics. This section is mainly taken from (Leach, 2001a) (approaches MM for molecules) and (Ellad B. Tadmor & Miller, 2011b) (approaching MM for metals), and (*Frenkel & Smit, 2001*) (for beginners) all excellent books on this topic.

The reason for this part's existence is three-fold. Molecular mechanics is a relatively new field of research, where computers have only recently gotten powerful enough to model systems of a proper size. For this reason, not many people are familiar with molecular mechanics. This thesis will serve as an informal introduction to molecular mechanics for readers new to the field, coming in from physics or chemistry. Secondly, the discussion in part D will often refer back to terms from the theoretical world. Some knowledge of the basics of molecular mechanics is needed to follow the discussion laid out there.

Over this little tour of the elements of molecular mechanics, we will visit several related scientific fields. While some sections below highlight useful applications of the theories we are discussing, they are nevertheless not necessary for an understanding molecular mechanics at a basic level, and can be skipped by the reader. I have prefixed these sections with 'optional'.

All **semi-bolded** symbols below are vectors. These vectors may not necessarily be three-dimensional, for example, we will later discuss abstract vectors with lengths equal to that of the number of degrees of freedom for the system, typically several thousands.

1 Thermodynamics

This section will introduce the thermodynamics necessary for our work with statistical mechanics below, as well as introducing functions such as heat capacity for later use. There is a lot to cover, so we will leave out a lot of important details, instead pointing to other texts on the subject (*Ellad B Tadmor, Miller, & Elliott, 2012*).

1.1 The laws of thermodynamics

First, we will describe systems in thermal equilibrium. Suppose that two systems A and B are in contact such that neither system can exchange particles (n), or energy (W). Heat (q), however can travel between systems somehow. We know intuitively that net heat will transfer from the hotter system to the cooler one. The hot system A will transfer heat q_{AB} to the cold system B, which will transfer the heat q_{BA} back. When $q_{AB} = q_{BA}$, we say that the two systems are in thermal equilibrium with each other, and write $T_A = T_B$. Let us now try to formalize these intuitions.

The *zeroth law of thermodynamics* states that thermal equilibrium is a transitive relation. If two systems are in thermal equilibrium, we write $A \sim B$.[2] Then the first law of thermodynamics state:

$$\text{If } A \sim B \text{ and } B \sim C \text{ then } A \sim C$$

(1) The zeroth law of thermodynamics

This simple and obvious law tacitly introduces us to the nature of temperature, thermal equilibrium and heat flow. While this law does not exactly tell us what temperature is, it tells us some properties of it. If two widely different systems were put in thermal equilibrium with a reference system with a known temperature, then the temperature for those two systems would be equal. You may say that there is only one type of temperature, even for different systems.

The *first law of thermodynamics* implies the existence of an internal energy of the system U, that is to say an energy independent of any external factors, like the kinetic energy from the momentum of the system, or the potential energy of the gravitational field. Consider a system in a state called 1. Now add work (W) and heat (q) to the system, so that it reaches a state 2. Next, do it over again, in a different way, so that the system once again reaches the first state. This is called a thermodynamic cycle. Say that we required more work to get a system from state 1 to 2, rather than vice versa. We would have a net work input of W_{cycle}. Experiments done by James Joule showed that the net heat from the cycle is the same: $W_{cycle} = q_{cycle}$, independent on what paths on takes between 1-2 and 2-1[3]. Since energy is conserved, this leads us to note that the state of the system must be represented by an energy function, which we will call U, the internal energy. Changes in the system will change the internal energy by:

$$\mathrm{d}U = \delta q - \delta W$$

(2) The first law of thermodynamics

That is, an infinitesimal change of state is caused by a change in heat and work. We are using the d,δ notation to show that when we integrate the function for a change of state from 1 to 2, then U is path independent (and only dependent on state) and q and W are generally path dependent. Said in another way: U are state functions, q and W are process functions. We use the sign convention that w is negative when work is done *on* the system, and positive when the work is done *by* the system[4]. q is positive when heat flows into the system.

Temperature is a state function, that was shown by the zeroth law of thermodynamics. Pressure (p), volume (V) and the number of particles (n) in the system are also state functions.

---

[2] This notation is motivated from set theory, where relations are written $a \sim b$. For the mathematically inclined: Thermal equilibrium is an equivalence relation but not a total order (in that $A \sim B$ and $B \sim A$ does not imply A = B!).

[3] Joule's work is the reason we have two units for energy, joule and calories. Work (Nm) and heat (cal) were thought to be separate, but Joule proved the equivalency of these two quantities. Eventually, joule became the universal energy unit, but some still hold on to calories. This is why you see kcal as the unit of energy in this thesis.

[4] Imagine an expanding gas doing pV work on the surroundings. The work would be written $w = \int p\mathrm{d}V$, which would be positive, since the gas expands. There do however exist conventions where the work done on a system is positive.

The pressure-volume work (pV work) of gas caused by changing the volume of the gas can be described as $\delta W = p\,\mathrm{d}V$. While the work is path dependent, if we say that the process is reversible (more on that soon), then we can deduce the path taken.

$$\mathrm{d}U = \delta q - p\mathrm{d}V$$ *(3) The same as (2), but with pV work*

The path dependence is transferred over to the heat. We can go one step further.

The *second law of thermodynamics* implies the existence of another state function S, which governs which way a spontaneous change of state will go. The entropy for a system will increase when a spontaneous change takes place, and decrease whenever outside forces changes the system against the spontaneous direction. Let us restate this: A reversible process is a change of state that is equally possible to go both ways 1 → 2 and 1 ← 2. This causes no change in entropy, as entropy changes implies a favored direction. Reversible processes tend to be infinitesimal in length. An irreversible process is a change of state where one direction is preferred: Either 1 → 2 or 1 ← 2. Mathematically, entropy can be stated as:

$$dS = \frac{\delta q}{T}$$ *(4) The thermodynamic definition of entropy*

Again, the change of states must be reversible. This may sound paradoxical, as we just said that entropy would not change for a reversible process. Equation (4), however refer to entropy change as function of the heat flow. Say a system (in a reversible way) accepts heat from its surroundings. Its entropy will increase with ΔS and the surroundings will decrease with $-\Delta S$, thus the total entropy will not change. We can now revisit the first law, and rewrite it with only state functions:

$$\mathrm{d}U = T\mathrm{d}S - p\mathrm{d}V$$ *(5) Internal energy for a reversible process*

We will not explain why S is a state function, but we will note that any attempts to decrease the entropy of a system by its surrounding will increase the entropy of the surroundings more than it will decrease the entropy of the system. Hence the popular statement of the second law: *The entropy of the universe (or any isolated system) tends to a maximum*. We will return to entropy in section 5.4.1.

With (5), we have gotten our first way of describing temperature. Since this equation is written with only exact diffentials, it can be written in the manner of:

$$\mathrm{d}f(x,y) = \left(\frac{\partial f}{\partial x}\right)_y \mathrm{d}x + \left(\frac{\partial f}{\partial y}\right)_x \mathrm{d}y$$ *(6) Differential form of f(x,y)*

Writing (5) in this matter gives us the following result:

$$T = \left(\frac{\partial U}{\partial S}\right)_V$$ *(7) Absolute temperature*

This means that temperature is the response of the system to a change in entropy while the volume is kept constant. Likewise, the pressure is:

$$-p = (\partial U/\partial V)_S$$ (8) Pressure in thermodynamic context

The *third law of thermodynamics* is the final piece in the puzzle of understanding temperature. It comes from studies of entropy based on statistical mechanics, and says that:

$$\lim_{T\to 0} S = 0$$ (9) The third law of thermodynamics

This requires our system to be a perfect crystal, as explained by equation (71), since a perfect crystal will only have one microstate.

The third law gives us a reference point for entropy, at approximately $T = -273.15\ ℃$, or $T = 0\ \mathrm{K}$ which we will denote absolute zero. We can use this to integrate (4) over a temperature interval.

1.2 Auxiliary functions

We will in this section define some useful functions for later use.

In the section above, we wrote down the expression (5) for internal energy, showing how U depended on entropy and volume when temperature and pressure were fixed (independent). We say that U is decided by the variables S and V, and write U(S,V). It is possible to transform U into other functions that are dependent on other variables, such as:

$$H(S,p) = U + pV$$ (10) Enthalpy

$$A(T,V) = U - TS$$ (11) Helmholtz free energy

$$G(T,p) = U - TS + pV$$ (12) Gibbs free energy

Here U is what we get when we integrate both sides of (5). These functions are called thermodynamic potentials, and they are all state functions. These are often easier to work with, than the internal energy, depending on what types of values are held fixed by the system. It is not difficult to find the differential forms of these potentials, for enthalpy for example,

$$\mathrm{d}H(S,p) = \mathrm{d}(U + pV) = \mathrm{d}U + \mathrm{d}(pV) = T\mathrm{d}S - p\mathrm{d}V + (p\mathrm{d}V + V\mathrm{d}p) = T\mathrm{d}S - V\mathrm{d}p$$

Finally, we will define some functions that are quite easy to measure. The heat capacities will be used later in this thesis, but the volume expressions are only here for completeness.

$$\alpha = \frac{1}{V}\left(\frac{dV}{dT}\right)_p$$ (13) Thermal expansion coefficient

$$\kappa = -\frac{1}{V}\left(\frac{\partial V}{\partial p}\right)_T$$ (14) Compressibility at constant temperature

$$C_p = \left(\frac{\partial H}{\partial T}\right)_p = \left(\frac{\partial U}{\partial T}\right)_p$$ (15) Heat capacity at constant pressure

$$C_V = \left(\frac{\partial H}{\partial T}\right)_V = \left(\frac{\partial U}{\partial T}\right)_V$$ (16) Heat capacity at constant volume

1.2.1 Optional: The grand canonical potential

It is customary for chemists and metallurgists to add another work term, the chemical work with independent μ (chemical potential) and dependent n. The expression for U reads now:

$$\mathrm{d}U(S,V,n) = T\mathrm{d}S - p\mathrm{d}V + \sum_i \mu_i \mathrm{d}n_i$$ *(17) Same as (5), with added chemical work*

i denotes different species in the system, like different molecules. The chemical potential μ is the rate of increase of U with increasing n, as seen by: $\mu_i = \left(\frac{\partial U}{\partial n_i}\right)_{S,V,n_j}$. The only change to the other thermodynamical potentials is that they inherit the dependence on n as well. We can introduce a new potential with variables T, V and μ_i as follows:

$$\Omega(T,V,\mu_i) = U - TS - \sum_i \mu_i n_i$$ *(18) The grand canonical potential*

This potential is called the Landau potential, grand potential or the grand canonical potential for reasons explained in section 5.4.3. It is useful for when chemical reactions take place in an enclosed volume.

2 An introduction to polyethylene

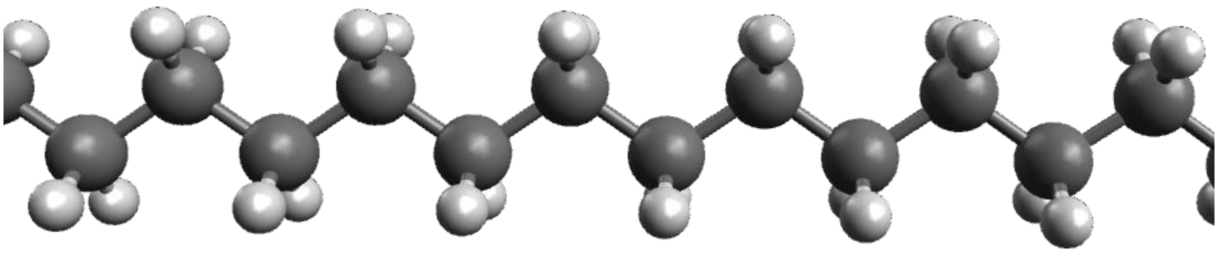


Figure 1: A visualization of polyethylene using the ball-and-stick model. The large black atoms are carbon, the smaller white are hydrogen. The molecule has this zigzag form to avoid overlapping hydrogen orbitals. More on that later.

2.1 Chemistry of polyethylene

In this thesis we will simulate molecules of the form H–[CH_2]$_n$–H. The CH_2 (methyl) group is called a monomer, and when n is relatively small, the molecules are called oligomers, or simply alkanes. If n = 8, the molecules is called octane, for example. On the other hand, when n is large (on the order of n = 10,000 or more), this molecule is called polymethylene, after the repeated methyl group. The length of a polymer chain is often reported as the molecular mass or molar mass of that chain, since this value is easier to measure.

Polymethylene is most often referred to as polyethylene (PE) after the reaction that produces it: n $CH_2=CH_2 \Rightarrow -[CH_2-CH_2]_n-$. Here ethylene ($CH_2=CH_2$) is the gas reactant that gives polyethylene its name (this makes PE a polyolefin as well). The reaction above underestimates the subtleties of PE production. It can be divided into two main types, high-pressure and low-pressure polymerization (Koopmans, Doelder, & Molenaar, 2010a).

High-pressure polymerization is the classical method for producing polyethylene; it is done by adding ethylene gas and peroxides in a high-pressure vessel. The peroxides catalyze the reaction by reacting with ethylene and producing a free radical, starting an addition reaction that eventually produces polyethylene. The polyethylene produced here is of low quality, with many long and short branches (see the section below).

Low-pressure polymerization uses Ziegler-Natta or metallocene catalysts to build polymers. A typical example of a Ziegler-Natta catalyst is a $TiCl_3$ (active site, many transition metals will perform the same role) on an $MgCl_2$ particle (support, must be inert in this environment). The polymerization reaction will take place on the active site, and each site will produce one polymer. The length of the polymer will depend on the diffusion of monomers to the site. To control branching of the polymer, several additives are added to the reaction.

These reactions does not produce polymer chains all with the same molar mass, but rather some distribution of molar masses, see Figure 2. The marked-off columns are various moments of the distribution function, since the distribution is rarely symmetrical, and needs to be described with more than two parameters. The typical parameters describing this distribution are given in equations (19)-(23).

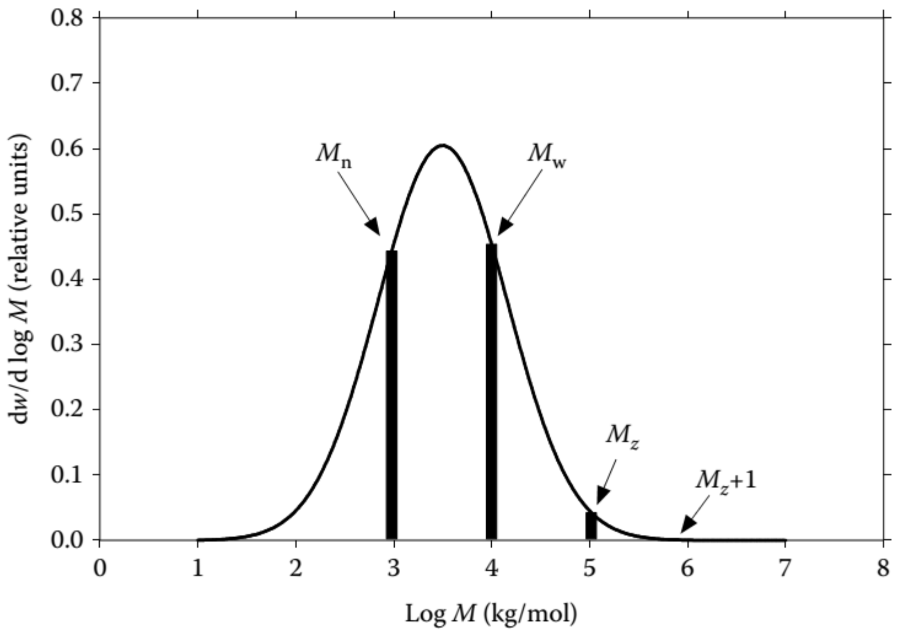


Figure 2: A molar mass distribution for polymers, with some moments marked off. Note the logarithmic scale of masses, from (Koopmans et al., 2010a).

Table 1: List of some moments of the molar mass distribution function, adapted from (Koopmans et al., 2010a).

Entity-based	Weight fraction-based	Name
$M_n = \frac{\sum_i n_i M_i}{\sum_i n_i}$	$M_n = \frac{\sum_i w_i}{\sum_i w_i / M_i}$	*(19) Number average*
$M_w = \frac{\sum_i n_i M_i^2}{\sum_i n_i M_i}$	$M_w = \frac{\sum_i w_i M_i}{\sum_i w_i}$	*(20) Weight average*
$M_z = \frac{\sum_i n_i M_i^3}{\sum_i n_i M_i^2}$	$M_z = \frac{\sum_i w_i M_i^2}{\sum_i w_i M_i}$	*(21) z-statistical weight*
$M_{z+1} = \frac{\sum_i n_i M_i^4}{\sum_i n_i M_i^3}$	$M_{z+1} = \frac{\sum_i w_i M_i^3}{\sum_i w_i M_i^2}$	*(22) (z+1)-statistical weight*

n_i is the number of polymers that have a mass M_i. w_i is the sum weight of all macromolecules with mass M_i. n and w are related by the formula $n_i = \frac{N_A w_i}{M_i}$, where N_A is Avogadro's number. An often-reported parameter describing polymers is the *dispersity* of the distribution given by $Đ_M$, defined as:

$$Đ_M = {M_w}/{M_n} \qquad \textit{(23) Molar-mass dispersity}$$

This is an IUPAC definition, given by (*Stepto, 2009*). The dispersity of Figure 2 for example is 10. Low dispersities ($Đ_M$ close to 1) mean that the distribution is narrow, and we have many polymers of similar lengths. Articles about polymer testing will often include the M_w and $Đ_M$, for the batch, which should be enough information to deduce the molar mass distribution to a reasonable degree.

When it comes to modeling polyethylene for molecular mechanics, we need some parameters as well to describe the geometry for the molecule. Among them is the bond lengths between carbon and hydrogen, and the angles between atoms, if we assume that the bond lengths and angles are the same all over the polyethylene, we can represent these parameters with a simple propane molecule, Figure 3.

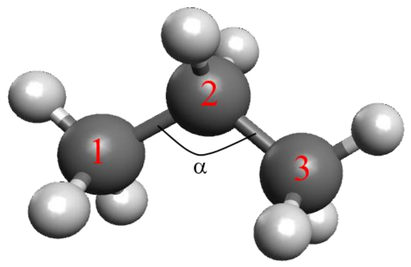


Figure 3: A propane molecule with numbered atoms and a marked-off angle 123.

A bond length is the length between two bonded atoms. If we were to put an atom at position $\boldsymbol{r}_1$ and another at position $\boldsymbol{r}_2$, then the distance between them would be:

$$r_{12} = \|\boldsymbol{r}_1 - \boldsymbol{r}_2\| = \|\boldsymbol{r}_{12}\| \qquad \textit{(24) The bond length}$$

A bond angle is the angle between two bonds. Simple trigonometry gives:

$$\theta_{123} = \cos^{-1}\left(\frac{\boldsymbol{r}_{12}\cdot\boldsymbol{r}_{23}}{r_{12}r_{23}}\right) \qquad \text{(25) The bond angle}$$

For polyethylene, there are two types of bond lengths: C-C and C-H. Likewise, there are three types of angles: C-C-C, C-C-H and H-C-H. Typical values for these parameters are given in Table 2 below.

Table 2: Various parameters for alkanes, adapted from the butane section of table 9 in (Sun, 1998).

Type	Value
Masses	
C	12.011 u
H	1.00794 u
Bond lengths	
C-C	1.53 Å
C-H	1.10 Å
Angles	
C-C-C	113.8°
C-C-H	110.0°
H-C-H	106.6°

Carbon in polyethylene is in theory expected to have sp^3 hybridization (see section 4.6), which causes all bonds to be at an angle of $\cos^{-1}(-1/3) \approx 109.47°$ to each other if they bond to similar atoms (like hydrogen bonded to carbon in CH_4).

There is in practice another parameter that is needed to fully describe the geometry of the molecule, the dihedral angle. We will get back to it in section 2.4.

2.2 Optional: History of plastics up to 1933

This brief history of plastics has been adapted from (*Brydson, 1999*).

Humans have used polymers as far back as we can tell. The Babylonians used bitumen as a mortar for their houses, the Indians used shellac for decoration, and the natives of Central America used natural rubber to manufacture rubber balls for games long before our current era. The use of rubber as a material accelerated in the middle of the nineteenth century, when inventors Thomas Hancock and Charles Goodyear independently of each other treated plastically deformed rubber with sulfur, and found that the rubber retained its elasticity again. This process would later be known as vulcanization. This usually required only a small amount of sulfur. By using larger amount of sulfur, the first synthetic thermoset was born, ebonite.

Another material from this time was Parkesine, or celluloid, which was invented by Alexander Parkes by processing cellulose nitrate (an explosive) to a plastic. Parkesine was easy to mold, and is known as the first synthetic thermoplast. Not long after, John Wesley Hyatt, on his quest to find a material that could replace ivory on billiard balls, came over celluloid, refined the production process, and coated the balls with celluloid so that they looked similar to earlier balls. It is worth mentioning that when a billiard ball collided with another, the celluloid would occasionally explode, producing a loud gun-shot sound.

The next step towards modern plastics came with Bakelite, an early synthetic resin made by phenols and formaldehyde. These reactants were kept at high pressure and temperature, causing them to react with each other and form this early plastic. Bakelite was a hard material but difficult to form, and it would not survive the competition from later plastics.

The Bakelite process would be a model for the production of other plastics, but polyethylene was discovered by accident. The chemists Eric W. Fawcett and Richard O. Gibson carried in 1933 out an experiment with polyethylene and benzaldehyde, but a leak in the pressure vessel caused a trace amount of oxygen to enter the system and work as a free radical, starting the ethylene polymerization process. Soon, the properties of polyethylene were made clear, and production started in 1939, and accelerated by the Second World War. Today polyethylene is the most produced plastic in the world, with a worldwide production of millions of tons every year.

2.3 Branched and unbranched alkanes

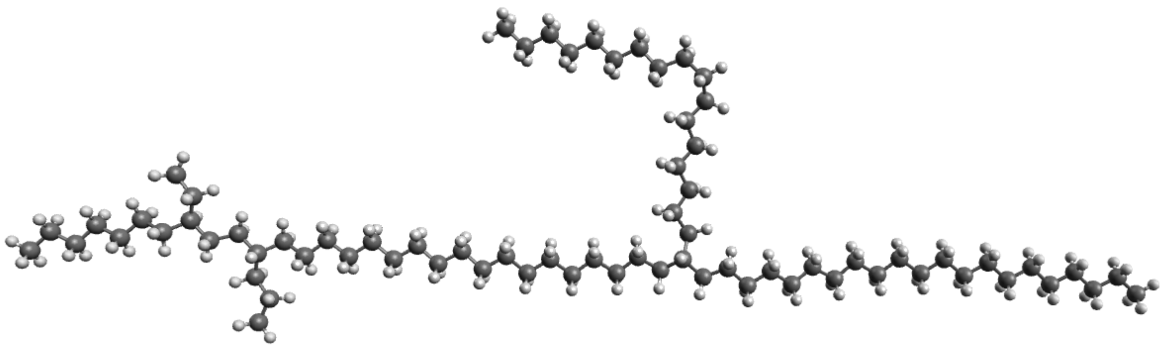


Figure 4: Showing a branched polymer backbone with short branches (left) and a long branch (right).

We mentioned above that the polymerization process could result in branched polymers. These branches are sometimes wanted, but for polyethylene, such branches lead to ineffective packing of the various molecules, giving low densities and bad tensile properties. Polymers with many long branches are designated LDPE (low-density polyethylene)[5].

Alkanes that have few long branches if any, are designated HDPE (high-density polyethylene). Short, unbranched alkanes are designated n-alkanes. We find these interesting; because HDPE can be packed in crystal-like structures (see section 2.5 below). This makes it possible to easily make an initial configuration of polymers using a simple algorithm (see part C, section 4).

2.4 Conformations

In addition to bond lengths and bond angles, a third term is needed to describe the complete geometry of the molecule. If one thinks of the bond length to describe the molecule in one dimension and the bond angle to describe the molecule in two dimensions, then the dihedral angles are the necessary addition to our model to describe the molecule in three dimensions.

[5] Note though, that there exists linear polymers with deliberately low densities. These are designated LLDPE (linear low-density polyethylene), VLDPE (very low-density polyethylene) and ULDPE (ultra ditto), and are made with a process similar to Ziegler.

Technically, a dihedral angle is an angle between two planes. If the two planes have the normal vectors $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$, then the dihedral angle between the two planes would be given by: $\cos(\varphi_{12}) = \boldsymbol{n}_1 \cdot \boldsymbol{n}_2$. The dihedral angle is measured between four consecutively bonded molecules can be found by defining the first three molecules to form a plane, and the last three to form another plane, and measure the angle between those two planes, see Figure 5. In practice, the four atoms are defined by their positions $\boldsymbol{r}_i$, and one needs to implement a fitting algorithm to calculate the angle.

There are three types of dihedral angles in polyethylene that need to be calculated, C-C-C-C, C-C-C-H and H-C-C-H. For the butane molecule in Figure 5, there are 1 C-C-C-C dihedral, 10 C-C-C-H dihedrals and 16 H-C-C-H dihedrals.

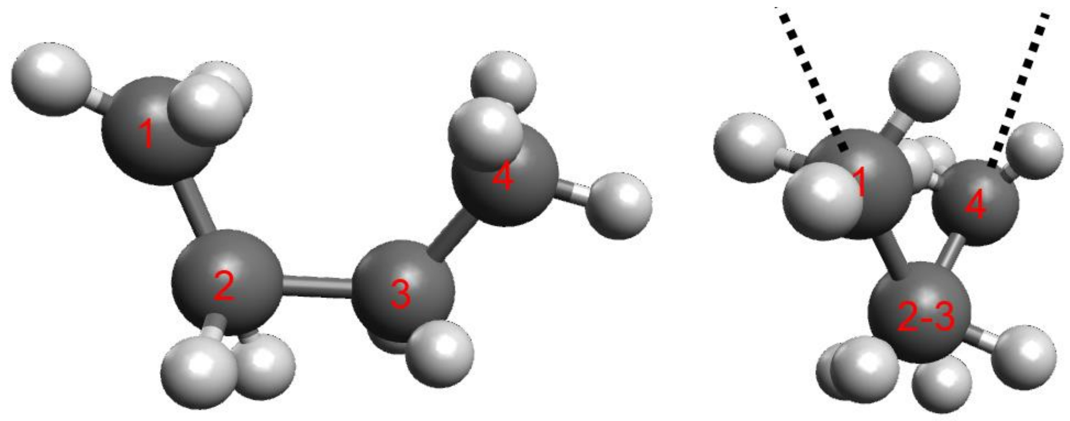


Figure 5: The dihedral angle. (l) A butane molecule 1-2-3-4. (r) The same molecule projected down the axis made by the 2-3 bond, and the 1-2 and 3-4 bonds are extended by dashed lines. The dihedral angle is the angle between these lines.

Some different dihedral angles have names. An angle of 0° is called *cis* or occasionally, *syn*. It is quite unstable. An angle of ±60° is called *gauche*, and it is metastable. An angle of ±180° is called *trans*[6]or sometimes *anti*, and it is often the most stable angle. The reason is thought to be the same as for the bond lengths and angles mentioned above, in that these angles minimize the overlapping of antibonding orbitals.

A state of the molecule described by the dihedral angle is called a conformation. A large molecule with all but one C-C-C-C dihedral is *trans* would have a different conformation than the same molecules where all dihedrals are *trans*. There is no problem with replacing the word "dihedral angle" in the paragraph above with the word "conformation", and it would still make sense. Stable crystalline polyethylene is made up of only trans-conformations, with one exception, see the discussion in section 2.5 below.

The dihedral angle is sometimes referred to as a torsion angle, see Figure 6. This name comes from the fact that if you could rotate the 2-3 bond, it would twist the entire molecule. If you have a ball-and-stick kit, try it out.

[6] Be wary in that some literature may define the *trans* angle at 0° and *gauche* at 120°. It is always a good practice to specify what angle is *trans* and what is *cis*.

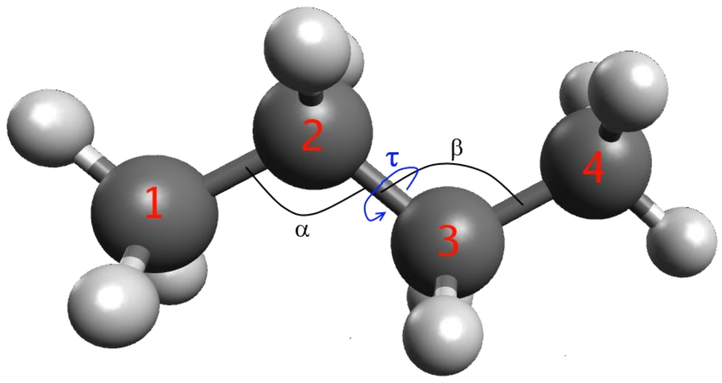


Figure 6: Illustrating the dihedral angle as an angle of torsion τ. If you rotate the bond 2-3, the molecule will change shape. α and β are the angles 1-2-3 and 2-3-4 respectively.

Now look at Figure 35, page 54. That graph shows the potential energy field used to orientate our molecule. The *gauche* and *trans* conformations are stable, with energy barriers between *trans* and *gauche*, and an energy barrier at *cis*. With quantum chemical calculations, like DFT (*Parr, 1983*), one can calculate these barriers and construct a model with parameters that closely fits the experimental data.

With the geometry of a single molecule laid down, let us now turn our attention to crystalline packing of polyethylene.

2.5 Crystals of linear polyethylene

In the simulations below, we will study the tensile behavior of crystal-like polymers. Eventually, a hexagonal unit cell was used to build the crystals (see Part C, section 4.1). Here we will look at some experimental data for HDPE, and save any discussion for its proper place (that would be Part D).

Crystalline polymer phase is often characterized by a few polymer chains that starts to fold themselves back and forth to a structure called lamella (see Figure 10). The lamellae are in bulk crystalline, as has been shown by X-ray crystallography, and they have a thickness of typically 70-200 Å (*Flory, 1962*). The non-crystalline morphology of polymers is amorphous. There are also polymers that forgo the lammelar structure, and are simply long chains of polymers stacked into lattices. These are called Ultra-high molecular weight polyethylene (UHMWPE), and they are perhaps the closest counterpart reality has to what we will simulate.

My reference for the various reported lattices is (*Brandrup, Immergut, Grulke, Abe, & Bloch, 1999*), particularly pp. 493-507, a section on linear HDPE. This compendium presents three different lattices, two stable and one metastable. The lattice is built up of CH_2 units.

The first lattice is orthorhombic, Pnam space group, the most stable configuration at 1 atmosphere and presumably at standard temperature as well. (Bunn, 1939) found the crystal structure using X-ray crystallography. The lattice constants are a = 7.40 Å, b = 4.93 Å and c = 2.53 Å.

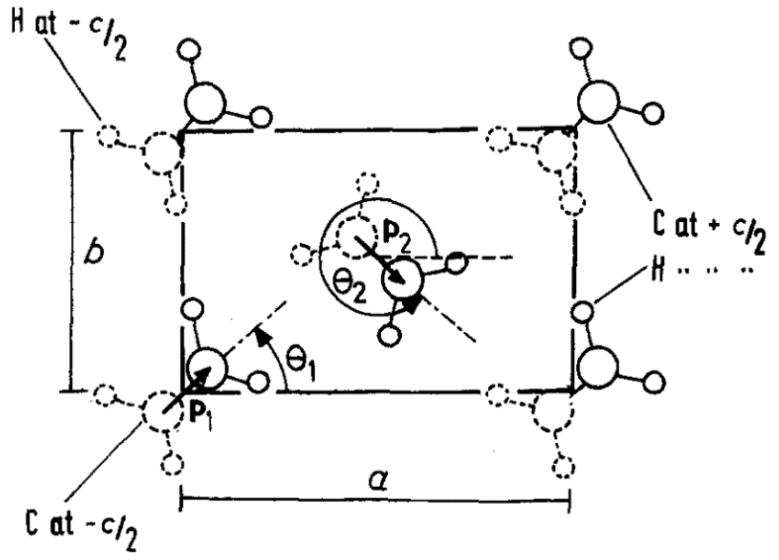


Figure 7: The orthorhombic unit cell (from (Bacon & Geary, 1983)) projected down the [001]-direction (the chain axis). This is to say that the polymer chain points out of the figure plane. Big atoms are carbon and smaller atoms are hydrogen. The dashed atoms are behind the (001)-plane, the solid atoms are in front of it. $\theta_1 \approx 43°$ *and* $\theta_2 \approx 137°$. *Arrows point along the C-C bond out of the plane.*

The second lattice is monoclinic, C2/m space group. It is metastable at normal pressures, and is found by deforming the orthorhombic lattice (*Tsuneo, Tetsuhiko, & Kenzo, 1968*). The lattice parameters are a = 8.09 Å, b = 4.79 Å, c = 2.53 Å and β = 107.9°. A P2/m monoclinic metastable lattice is also known to exist at high pressures (*Fontana et al., 2007*).

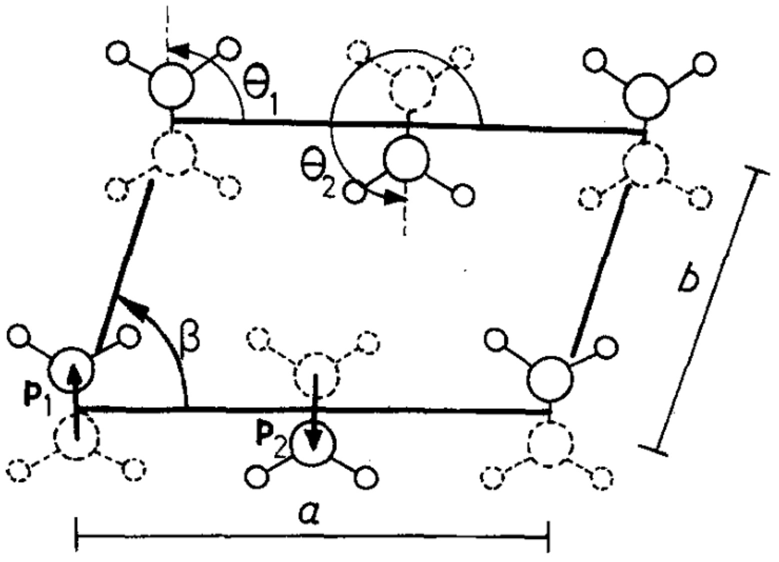


Figure 8: The monoclinic unit cell (figure from (Bacon & Geary, 1983)), looking down the [001]-direction. See Figure 7 for an explanation of the drawing. $\theta_1 \approx 84°$ *and* $\theta_2 \approx 264°$

The last unit cell considered is hexagonal (Bassett, Block, & Piermarini, 1974), with less ordering among the polymer chains, whereas the other lattices have all-trans conformations, the hexagonal phase seems to have a few gauche conformations as well. This is why we cannot simply draw a figure of the hexagonal unit cell (compare the 2D-hexagonal unit cell figure Figure 52), as we have done with the unit cells above. The lattice parameters however are known, and they are a = 4.88 Å and c = 2.45 Å.

The hexagonal unit cell is known to be stable, but at high pressures and temperatures close to the melting point, see Figure 9.

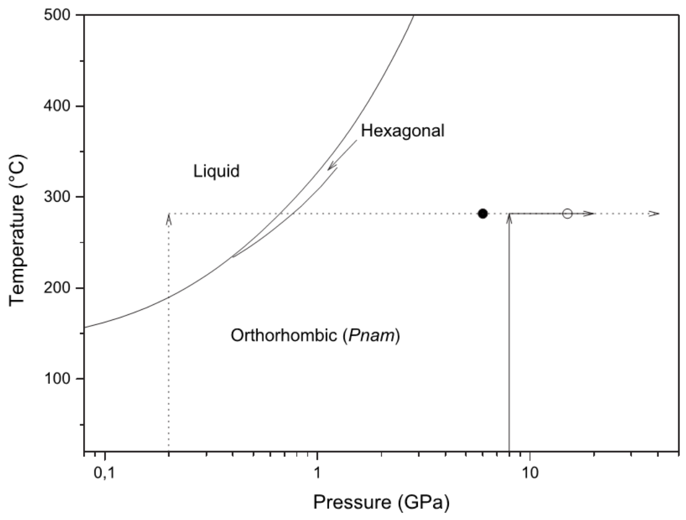


Figure 9: Phase diagram for crystalline polyethylene (figure from (Fontana et al., 2007), based on (Hikosaka, Tsukijima, Rastogi, & Keller, 1992)). The arrows show two experiments carried out by Fontana and the circles marks events where a monoclinic phase is observed (P2/m or C2/m). The authors suppose that the C2/m phase is stable at high pressures. The molecule mass is on the order of thousand monomers.

By looking through a microscope, a crystalline polymer phase could appear as a spherical, and the morphology is called spherulite (Figure 10) (Barham, 1986). The spherulite is grown by nucleation from a supercooled polymer melt, where the polymer chain forms lamellae, causing local crystal-like behavior (Keith & Padden, 1963). Another morphology that can be found is the so-called shish-kebab crystal, where a central rod (shish) that can grow up to several micrometers in length has several perpendicular discs of lamellae (kebab) growing out of it (Xia, Zhang, Wang, Feng, & Yang, 2014).

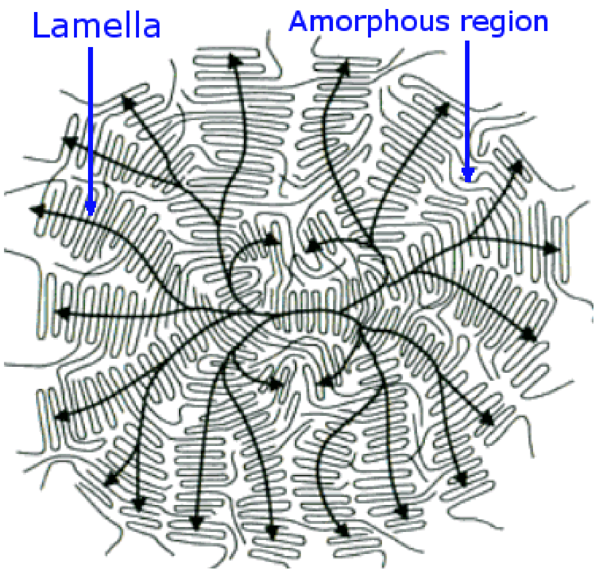


Figure 10: Schematic presentation of spherulite. The polymers (grey) fold in on themselves, forming crystalline packing. The black arrows show the directions of the molecules (Figure from Wikimedia Commons, named Spherulite2.png. This work is created by and given the CC BY-SA 3.0 license by User:Materialscientist).

3 Mechanics of polyethylene

We will here quickly repeat the material mechanics of a simple tensile test. An often-encountered material response is the elastic-plastic one, where a material behaves elastic at small strains and plastic at larger strains. The elastic response causes a system to deform under applied stress, and go back to normal once the stress is removed, it can be seen as the stretching of chemical bonds. A plastic response destroys the material somewhat, since after applied stress, the material will go back to some deformed state.

3.1 Time-independent behavior

A tensile test is often used to gauge a material response to external stresses. It is done by pulling a rod-shaped specimen by a machine, and measuring the pulling force P and rod elongation ΔL. These values can be converted into geometry-independent values σ and ϵ by the formulas underneath, and plotted like in Figure 11.

$$\sigma_n = {}^{P}/_{A_0} \qquad \textit{(26) Nominal stress}$$

$$\varepsilon_n = {}^{\Delta L}/_{L_0} \qquad \textit{(27) Nominal strain}$$

Here A_0 and L_0 are respectively the original cross-section and length of the specimen (Dowling, 2013). Nominal stress is also called engineering stress, and can in 3D be generalized to the First Piola-Kirchhoff stress tensor (Ellad B Tadmor et al., 2012). The nominal strain is also called engineering strain.

Under a true tensile stress, when the specimen elongates, the cross-section of it will shrink in a similar manner. By assuming that the volume of the specimen is constant $A_0L_0 = AL$, we can work out the following “true” stresses and strains:

$$\sigma_{true} = (1 + \varepsilon_n)\sigma_n$$ (28) True stress

$$\varepsilon_{true} = \ln(1 + \varepsilon_n)$$ (29) True strain

The analogue of the true stress in three dimensions is the Cauchy stress tensor.

For small strains, we have a linear relation between stress and strain, given by:

$$P = k\Delta L$$
$$\sigma = E\varepsilon$$ (30): Hooke's law

Here, $E = k \cdot {L_0}/{A_0}$ is Young's modulus or the modulus of elasticity. This modulus is interpreted as the slope of the elastic region of the stress-strain curve, and for polymers it can be interpreted as the sum of the stretching of primary covalent bonds inside molecules, and the secondary bonds that arise from the interaction between molecules.

The point where the elastic region ends and the plastic region begins is called the yield stress, σ_y. For many materials one can place the yield stress where the linear elastic region ends, but often this is not so obvious. For these ambiguous tests, one employs an offset yield stress $\sigma_{y,0.002}$, which is constructed by drawing a line starting at $\varepsilon = 0.002 = 0.2\ \%$ with the same slope as the elastic behavior of the material (E), and find the yield stress where this line crosses the stress-strain curve.

This figure below shows that yielding starts with a sudden drop in stress necessary to deform the material. The upper point before the drop and the lower point after the drop are known as respectively the upper and lower yield points (σ_{yu} and σ_{yl}). For a polymer material, there are two ways to report the yield strength, either as $\sigma_{y,0.002}$ or σ_{yu}.

On the right side of the figure, the stress drops rapidly to zero. The specimen has fractured, and the instruments register a pulling force of 0. The stress and strain at the point of fracture is written as σ_f and ϵ_f respectively. While this figure appears to show that the final strain of the specimen is larger than the fracture strain, it is in fact smaller. The elastic deformation reverses, a situation that can be described by drawing a line with a slope E from the fracture point to the x-axis, from which the final strain can be read.

There are not many resources on the tensile properties of polyethylene nanofibers out there, but (Papkov et al., 2013) however is a very good resource.

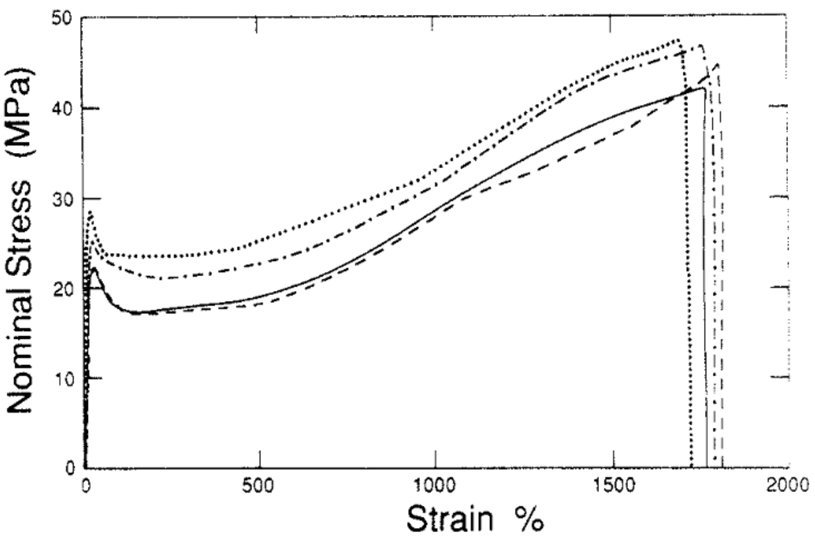


Figure 11: Stress-strain curve of HDPE with various degrees of crystallinities. From top to bottom: 64 %, 55 %, 44 % and 46 %, from (Kennedy, Peacock, & Mandelkern, 1994).

The figure above shows representative results from tensile testing of polymers with different degrees of crystallinities. The degree of crystallinity is the volume fraction of crystalline morphology in the material. In general, it seems that the higher crystal content there is in the material, the higher the tensile strength, and correspondingly lower fracture strain.

To understand this, we need to know how polymers behave under stress. Polymer chains are usually curled up in crystalline (see section 2.5) or amorphous morphologies. Under stress, these morphologies will start to unravel into longitudinal strands of polymers. Since these strands pack more efficiently than other random morphologies, the volume of the specimen will decrease, giving a smaller surface area (necking). Since the area decreases, the stress increases, giving a feedback effect that continues onto some minimal surface area that has the best packing (see figure 4.10 in (*Dowling, 2013*)). This new structure is quite anisotropic, where the material is strong along the axial length, and weaker in the radial direction. It is these structures that cause the material to get harder as the strain increases.

It is possible to produce polymer fibers that consist of many aligned filaments packed in crystalline patterns. These are called ultra-high-strength polyethylene filaments (*Smith & Lemstra, 1980*) or high-performance polyethylene (*Jacobs & Van Dingenen, 2001*), and are often produced by spinning and drawing extremely long polyethylene fibers (UHMWPE, ultra-high molecular weight polyethylene). Spinning is a process where a polymer solution (2-10 wt % PE in an organic solvent, like decalin or tetralin) is extruded at temperatures about 130-175 °C to thin filaments. These filaments are then quenched in cold water, followed by being drawn through a die at temperatures about 120-143 °C to unravel and orient the polymer chains, similar to rolling of steel sheets (*Rein et al., 2007; Smith & Lemstra, 1980*).

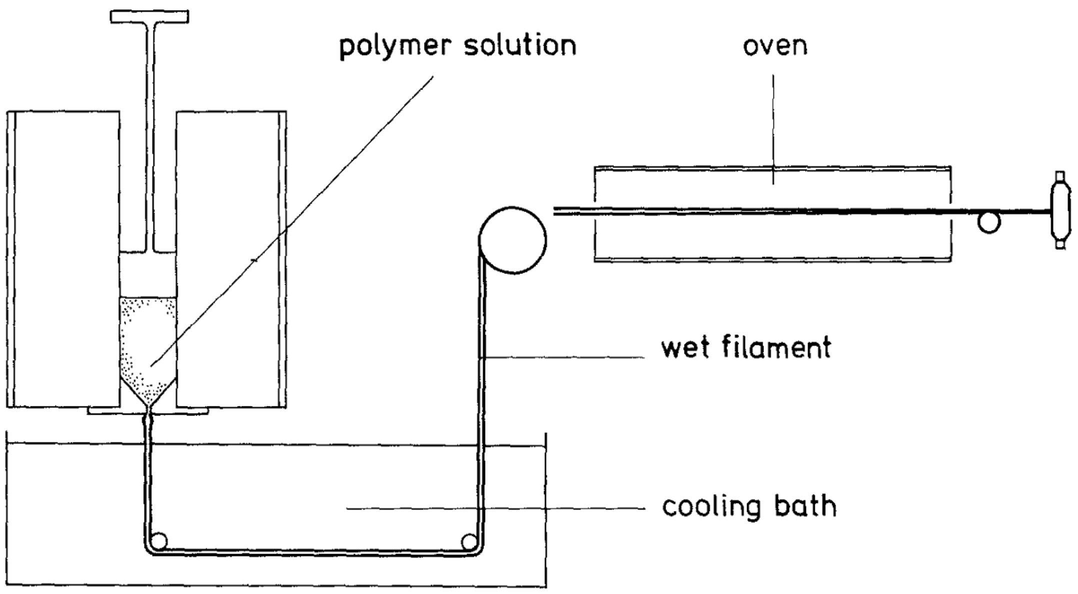


Figure 12: The production of ultra-high strength polyethylene filaments (from (Smith & Lemstra, 1980))

It is worth mentioning that the theoretical Young's modulus and fracture strength is about 300 GPa (compared to steel's 210 GPa) and 10-20 GPa respectively (Barham, 1986).

3.2 Optional: Creep

Up until now, we have ignored the time-dependent properties of polyethylene, by assuming that the material strains immediately when exposed to stress. This is not the case, and we need to add an extra parameter, the strain rate $\dot{\varepsilon} = \frac{d\varepsilon}{dt}$ to our model. To better describe creep, we need to make more assumption of the material-time dependence. We can for example say that it is viscoelastic, or behaving like a viscous fluid, with a time-independent elastic component. A common stress-strain relationship for this model is the Kelvin-Voigt model.

$$\sigma(t) = E\varepsilon(t) + \eta\frac{d\varepsilon(t)}{dt}$$

(31) The constitutive relation of a Kelvin-Voigt material

$$J(t) = \frac{1}{E}\left[1 - e^{-\frac{E}{\eta}t}\right]$$

(32) The creep compliance of a Kelvin-Voigt material

Here, η is the viscosity of the material, and $J(t)$ is the compliance (Mainardi & Spada, 2011). If we for example set a constant stress σ_0 on the system, then the evolution of the strain would be

$$\varepsilon(t) = \varepsilon_0 + \sigma_0 \cdot J(t)$$

(33) Constant stress in a Kelvin-Voigt material

Here the first term is the instantaneous strain $\varepsilon_0 = \frac{\sigma_0}{E_0}$ (elastic strain) and the second term is the time-dependent strain (viscous strain), see the figure below. The increase in stress from time t0 is called creep, and the decrease in stress from time t1 is called recovery.

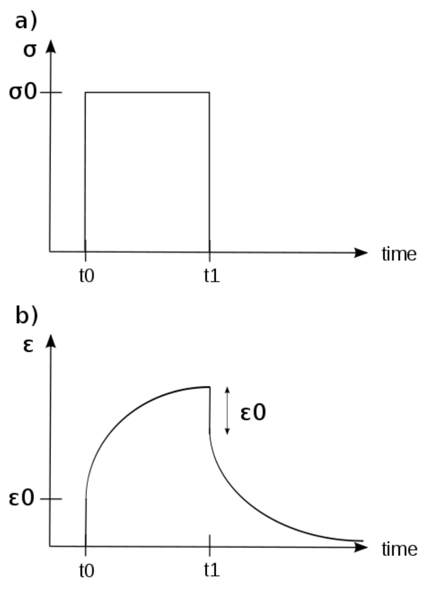


Figure 13: Viscoelastic material strain response to an induced stress state. The linear parts of size ε_0 are the immediate elastic response ($\varepsilon_0 = \sigma_0/E_0$), while the curved parts are the viscous time-dependent response. Picture from Wikimedia Commons, released into public domain.

We can also keep the strain constant, where the stress time-dependence is called stress relaxation. For the Kelvin-Voigt model, this gives a stress of $\sigma(t) = [E + \eta\delta(t - t_0)]\varepsilon_0$, where $\delta(t-t_0)$ is the Dirac delta function. Think of this stress state as $[E + \eta]\varepsilon_0$ at time $t = t_0$ and $E\varepsilon_0$ afterwards. This is quite inaccurate. A better constitutive relation for relaxation is the Maxwell model, but we will not discuss it here, see (*Dowling, 2013*) or (*Mainardi & Spada, 2011*) instead.

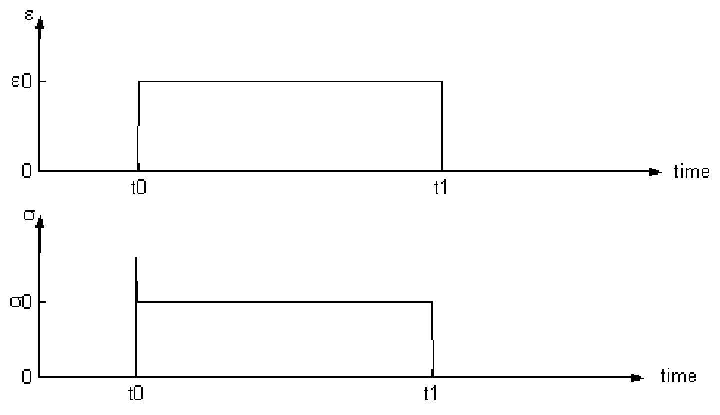


Figure 14: Viscoelastic stress response to a constant strain. The overshoot at the beginning is the instantaneous viscous stress, see text. Self-made picture.

For metals and similar materials, creep will only become relevant, when the temperature becomes sufficiently high. In the same manner, a polymer loses its viscoelasticity when the temperature becomes low, see the section below. We also recommend (Koopmans, Doelder, & Molenaar, 2010b) for a discussion of polymer as a viscous fluid, where shear stress is used for developing the model instead of normal stress.

3.3 Optional: Glass transition

As established above, the polymer consists of amorphous and crystalline regions. When the temperature is high, these molecules are allowed to move, giving the polymer a high ductility. Conversely, when the temperature is low, these molecules are locked into place, giving a hard and brittle material. The range of transition between these two states is quite narrow, and can be represented by a glass transition temperature T_G.

Glass transition, like viscoelasticity, of a polymer arises from its microstructure, which is a few orders of magnitude higher than what we can model in molecular dynamics so we doubt that we will see any big temperature effects from our simulations.

4 Quantum mechanics

Molecular mechanics belong to the realm of atoms, which is governed by the laws of quantum mechanics. While the rules of quantum mechanics can be used by computers to calculate simple systems that can be transferred to larger systems, the computational complexity blows up with more particles added to the system. Compare to an ideal gas (section 5.4.2), which are governed by classical mechanics.

This section will introduce some basics of quantum mechanics, and introduce the assumptions necessary for MM. We will also take a look at the molecular orbital model for quantum bonding.

4.1 Historical development

For people who would like to read the following section in a graphic novel form, I will recommend (*Ottaviani, 2009*) as an accessible introduction to the foundations of quantum mechanics and its founders.

Quantum mechanics arose around the 1900's, with Max Planck's work on Wien's displacement law, which describes radiation from a blackbody. Planck was interested in deriving this law from entropy rather than temperature, but struggles with resolving the law with experimental data on short wavelengths (*v*). Eventually, he has to introduce discrete energy levels and a constant *h* from formula (34), to his model[7], giving Planck's law, from which Wien's law can be derived (*Planck, 1901*).

$$E = h\nu$$

(34) Planck relation for a photon

In 1905, working on the photoelectric effect, Albert Einstein introduces the concept of a light quanta, that is to say photons that can only attain certain energy levels (*Einstein, 1905*). The photons are light, with frequency and energy related by (34). From earlier experiments, light proved to be hard to figure out, as light propagated through space like a wave, but it had

[7] These innovations came from Boltzmann, who had used similar ideas for his probabilistic statistics. Planck was a scientist of the old school, and personally resented the implications of Boltzmann's theories. Nevertheless, the theory worked out with these alterations, and Planck added them at the very least as a mathematical convenience. In fact, the constant that bears his name has its *h* from the word Hilfsgröße (Auxiliary factor)

momentum like a particle. Einstein derived an expression for the momentum of a photon ($p = h\nu/c$), and concluded that light was both a particle and a wave.

By 1913, the following facts about the atom were clear: 1. An atom consists of a positively charged core, and negatively charged electrons. 2. Light excited from atoms were quanta with energies *hv*. From this, Bohr (*Bohr, 1913*) develops his model for the hydrogen atom, where the electrons orbit around the nucleus, and can only have specific energies.

$$E_n = -\frac{1}{n^2} R_E$$

(35): Energy of an electron orbiting an atom.

Here, n is the principal quantum number (takes on values n = 1, 2, 3…) here a metric on how far an electron is removed from the nucleus, and $R_E = hcR_\infty \approx 13.6\,\mathrm{eV}$ and $R_\infty = \frac{m_e e^4}{8\varepsilon_0^2 h^3 c} \approx 10973732\,\mathrm{m}^{-1}$ are respectively the Rydberg energy, and the Rydberg constant for a nucleus with infinite mass. If an electron were to lose energy from E_2 to E_1, the difference would be released as a photon with an energy $\Delta E = E_2 - E_1$, and frequency given by (34). That the energy released would be represented by a photon was not Bohr's idea, but John Slater's (*Bohr, Kramers, & Slater, 1924*).

We will skip some of the other things that were going on with quantum theory at this time, but we will note Louis de Broglie's introduction of the particle-wave duality (*De Broglie, 1924*). de Broglie took up Einstein's particle-wave duality for photons, and took it one step further. He proposed that any particle with a given momentum could behave like a particle:

$$\nu = \frac{pc}{h}, \quad \omega = \frac{pc}{\hbar}$$

(36) de Broglie's relation in frequency form

This was eventually verified in 1928 (*Davisson, 1928; Thomson, 1928*). Here we have introduced $\hbar$, which is the reduced Planck constant[8] $\hbar = \frac{h}{2\pi} \approx 1.055 \cdot 10^{-34}\,\mathrm{J \cdot s}$ and $\omega = 2\pi \cdot \nu$ is the angular frequency. The second relation is often written in vector form:

$$\boldsymbol{p} = \hbar \boldsymbol{k}$$

(37) de Broglie's relation in wave vector form

Here, ***k*** is the wave vector, a variable that determines how a wave propagates. Its magnitude is $\|\boldsymbol{k}\| = \frac{\omega}{2\pi c} = \frac{1}{\lambda}$. The description of particles as waves put a severe limit to our ability to measure the position and momentum of a particle simultaneously. This is formalized in the uncertainty principle.

4.2 The Schrödinger equation

Now that it has been established that particles can be described as waves, we can continue to describe the mechanics of these waves. Our tool will be the wave function Ψ(**r**,t), a complex-valued function that contain the information about the system's ***r*** and ***k*** and how they evolve over time. We require that the wave function is normalized, that is:

[8] $\hbar$ was originally introduced as the quantum of angular momentums.

$$\int \Psi^{*}\Psi \, \mathrm{d}\boldsymbol{r} = 1$$ (38) *Normalization criterion for a wave function*

The integration takes place over all space. An unnormalized wave function Ψ_a can be divided by $\int \Psi_a^{*}\Psi_a d\boldsymbol{r}$ to be properly normalized. The reason we do this is that the real-valued function $(\Psi^{*}\Psi)(\boldsymbol{r},t)$ can be interpreted as the probability density of a particle's position at a given time. The integral of this probability density function must be 1, implying (38). We will also need that the solutions are orthogonal, i.e.

$$\int \Psi_a^{*}\Psi_b \mathrm{d}\boldsymbol{r} = 0\,, \qquad a \neq b$$ (39) *Orthogonality criterion for a wave function*

Here Ψ_a and Ψ_b are two different wave functions that describe the same system. If a = b, then the integral takes the value 1 as long as the Ψ_a is normalized.

The behavior of the wave function is governed by the famous Schrödinger equation:

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + U(\boldsymbol{r},t)\right]\Psi(\boldsymbol{r},t) = i\hbar\frac{\partial\Psi(\boldsymbol{r},t)}{\partial t}$$ (40) *The time-dependent Schrödinger equation for a single particle*

The bracketed expression is called the Hamiltonian operator, *H*. We shall discuss it further in section 5, but for now, we may note that the first term is an expression of the kinetic energy and the second term is equal to the potential energy of the system. The equation above shows the behavior of a single particle with mass *m* in a potential energy field *U*. It is possible to design Hamiltonians for a collection of different particles, and theoretically solve the equation to find a wave function that governs the particles. An example of this is two bonded hydrogen atoms (a proton and an electron each), but even this simple-sounding problem is tough to calculate, and we need to rely on other methods.

Optional: The reason for why the Schrödinger equation is complex-valued is that it contains information on both a particles position ***r***, and its momentum ***p*** (or more exactly its wave vector). Just like the square of the wave function Ψ gives the probability density function of finding a particle at a position ***r***, the Fourier transform $\widehat{\Psi}$ will give information about the particle's momentum. $\widehat{\Psi}$ is given by:

$$\widehat{\Psi}(\boldsymbol{k}) = \frac{1}{(2\pi)^{3/2}}\int \Psi(\boldsymbol{r})e^{-i\boldsymbol{k}\cdot\boldsymbol{r}}d\boldsymbol{r}$$ (41) *The Fourier transform of a wave function (3-dimensional)*

The square of this wave function $\widehat{\Psi^{*}}\widehat{\Psi}$ gives the probability of finding a particle with a wave vector ***k***, that is to say, its momentum.

Equation (40) can be simplified by assuming that the potential energy and wave function can be separated in time-dependent and space-dependent functions: $U(\boldsymbol{r},t) = U_\tau(t)U_\rho(\boldsymbol{r})$ and $\Psi(\boldsymbol{r},t) = \tau(t)\psi(\boldsymbol{r})$, one can write down a time-independent Schrödinger equation:

$$\left[-\frac{\hbar}{2m}\nabla^2 + U(\boldsymbol{r})\right]\psi(\boldsymbol{r}) = E\psi(\boldsymbol{r})$$ (42) *The time-independent Schrödinger equation for a single particle*

The bracketed term is once again our Hamiltonian. A common way to write (42) is as the eigenvalue problem $H\psi = E\psi$. E is the energy of our system: $E = \hbar\omega$ from equations (34) and (36).

4.3 Optional: A single electron orbiting a nucleus

Equation (40) can be solved for a single particle in a presence of several potential fields, for example the zero field $U(\boldsymbol{r}) \equiv 0$ or the particle-in-a-box field:

$$U(r) = \begin{cases} 0, & r_1 < r < r_2 \\ \infty, & \text{otherwise} \end{cases}$$

What we will look at here, is the field that represents the electrostatic field of an atomic nucleus, that is to say:

$$U(r) = -\frac{Ze^2}{4\pi\epsilon_0 r}$$ *(43) Coulomb's law for a nucleus of Z protons*

We can solve (42) with this potential for one single electron. To do this, we must rewrite ψ into polar coordinates, and we need to suppose that it can be decomposed into three components only dependent on one of these parameters ($\psi(r,\theta,\phi) = \rho(r)\Theta(\theta)\Phi(\phi)$). We will skip the derivation of the solution; it can be found in many textbooks on quantum mechanics. The exact solution is:

$$\varphi_{nlm}(r,\theta,\phi;Z) = \rho_{nl}(r;Z)\Theta_{lm}(\theta)\Phi_m(\phi)$$ *(44) Hydrogen-like orbitals*

$$\begin{gathered} n \in \{1,2,\dots\} \\ l \in \{0,1,\dots,(n-1)\} \\ m \in \{-l,(-l+1),\dots,0,\dots,(l-1),l\} \\ s \in \left\{-\frac{1}{2},\frac{1}{2}\right\} \end{gathered}$$ *(45) Quantum numbers*

$$E_n = -\frac{m^e e^4}{2\hbar^2 n^2}$$ *(46) Energy*

$$\rho_{nl}(r;Z) = -\left[\left(\frac{2Z}{na_0}\right)^3 \frac{(n-l-1)!}{2n[(n-1)!]^3}\right]^{\frac{1}{2}} \exp\left(-\frac{Zr}{na_0}\right)\left(\frac{2Zr}{na_0}\right)^l L_{n-l-1}^{2l+1}\left(\frac{2Zr}{na_0}\right)$$ *(47) r-function*

$$\Theta_{lm}(\theta) = \left[\frac{(2l+1)}{2}\frac{(l-|m|)!}{(l+|m|)!}\right]^{\frac{1}{2}} P_l^{|m|}(\cos(\theta))$$ *(48) θ-function*

$$\Phi_m(\phi) = \left[\frac{1}{\sqrt{2\pi}}\right] e^{im\phi}$$ *(49) ϕ-function*

Let us go through the equations in order.

(44): We have here introduced the notation φ_{nlm} for the eigenfunction solution. The reason is that wave function is encountered so often, that it is convenient to denote it by a specific symbol. The letters n, l and m are the quantum numbers, defined in (45), and are subscripted in the functions that call for them. These cause the eigenfunctions to only take discrete values, causing the discrete states introduced by Planck and Bohr.

(45): Shows how the quantum numbers are constrained.
n is, as previously mentioned, the *principal quantum number*. It states which electron shell the electron is in, i.e. the electron's energy.

l is the *azimuthal quantum number*, and specifies what shape the electron orbital will take. An electron orbital is a wave function that contains an electron.

m is the *magnetic quantum number*. It divides the orbitals into different directions.

s is the *spin quantum number*. All elementary particles have a value called spin, and for electrons this takes the value ½. This gives us two possible values for s, giving the interpretation that it is possible to contain two electrons in each orbital. Dealing with spin would require another factor into (44).

Aside from integers, some quantum numbers are often represented in other ways. The *principal quantum numbers* are often represented as numbers, but they can sometimes be referred to as the shells K, L, M, N… in the atom.
The *azimuthal quantum numbers* can be represented by the mapping $0 \rightarrow s, 1 \rightarrow p, 2 \rightarrow d, 3 \rightarrow f, 4 \rightarrow g, 5 \rightarrow h, 6 \rightarrow j, 7 \rightarrow k$ etc.).

The *spin quantum number* can be represented by the states $\frac{1}{2} \rightarrow |\uparrow\rangle$ and $-\frac{1}{2} \rightarrow |\downarrow\rangle$, respectively "spin up" and "spin down". For an orbital to have two electrons, they must have a different spin.

(46) Is the eigenvalue of the solution, as given by (42). It represents the total energy of the system, and is the quantum mechanical foundation for (35). Two orbitals that have different quantum numbers, but the same energy are called degenerate (indistinguishable), and from this energy function it is easy to see that any collection of orbitals n fixed and l,m varying are degenerate.

(47) – (49) are the different components of the wave function. The values inside the square brackets are necessary for normalizing each component function, thus normalizing the entire wave function. $a_0 = \frac{\hbar^2}{\mu e}$ is the Bohr radius, for hydrogen equal to 0.529 Å. It can be seen as the length scale of the atom radius. $\mu = \frac{m_e M}{m_e + M}$ is the reduced mass of the atom. M is the mass of the nucleus. $L_{n-l-1}^{2l+1}(x)$ and $P_l^{|m|}(x)$ are respectively the associated Laguerre polynomials and the associated Legendre polynomials.

The orbitals are also denoted after the first two quantum numbers. A 1s orbital corresponds to the quantum numbers n = 1, l = 0, m = 0, and a 2p orbital corresponds to quantum numbers n= 2, l = 1, m = one of -1, 0 or 1, in other words 2p is the collection of three degenerate orbitals. The number of electrons currently occupying these degenerate orbitals are denoted as a raised number at the end. The electron configuration for oxygen is for example is $1s^2 2s^2 2p^4$. See an illustration (i.e. a 3D plot of $\varphi_{nlm}^* \varphi_{nlm}$) of the first few orbitals in (Leach, 2001a), page 34.

4.4 The Born-Oppenheimer approximation

Solving the Schrödinger equation analytically can be done for simple systems containing one electron, and possibly a proton. Any more particles added, gives us problems similar to the three-body problem, for which it is known that there is no general analytical solution for it. From this point on, (40) and (42) must be solved approximately.

For a mixed system with nuclei and electrons, one could replace each nucleus with a Coulomb potential (like equation (43)), and get a Hamiltonian consisting of the kinetic energy of electrons, and the potential energy field from the nuclei. The nuclei may move as well, but on a timescale much slower than the electrons, since a proton alone is 1836 times more massive than an electron. In this case we can assume that the electrons find their new states instantly. The Born-Oppenheimer approximation (BOA) (*Born & Oppenheimer, 1927*) maintains that the mixed system can be divided into two systems, one of nuclei and one of electrons, each with their own Hamiltonian and wave function:

$$\Psi_{total}(\boldsymbol{r}^{nuc}, \boldsymbol{r}^{el}, t) = \Psi_{nuc}(\boldsymbol{r}^{nuc}, t) \cdot \Psi_{el}(\boldsymbol{r}^{nuc}, \boldsymbol{r}^{el})$$

(50) The Born-Oppenheimer approximation

nuc stands for nuclei and *el* means electrons. **This is a milestone in our way towards molecular dynamics**. The Hamiltonian corresponding to the nuclei wave function is a mixture of the kinetic energy from the nuclei, the potential Coulombic energy from nucleus-nucleus relations and a potential energy field emerging from interactions concerning electrons. Since the BOA expects the electrons to instantly reach their ground state, then this energy is only a function of the positions of the nuclei.

$$H_{nuc} = -\frac{\hbar}{2}\sum_{i}^{N}\frac{\nabla_i^2}{m_i} + U_{nuc}^{coul}(\boldsymbol{r}, t) + \epsilon_0(\boldsymbol{r}, t)$$

(51) Hamiltonian operator for the nuclear wave function

The known Coulombic interaction and the unknown ϵ_0 can be summed up to a single unknown potential U(r). If we could somehow approximate this field, we could find the eigenfunction to it.

We evoke the particle-wave duality (equation (36)) again, this time in order to turn the matter waves back to classical particles again. Consider a box of volume V containing N identical particles. The average distance between particles is $(V/N)^{1/3}$. Next we will need to introduce the de Broglie thermal wavelength (also called the mean thermal wavelength):

$$\Lambda = \frac{h}{\sqrt{2\pi m k_b T}}$$

(52) de Broglie thermal wavelength

If this wavelength is much smaller than the average distance between atoms, then the wave functions are all localized in space and can be approximated as classical particles (Ellad B. Tadmor & Miller, 2011b) pages 240-1 and (*Pathria & Beale, 2011*) pages 135-7.

$$\Lambda \ll \left(\frac{V}{N}\right)^{\frac{1}{3}}$$

(53) Criterion for approximating the system as classical (the classical limit)

This criterion holds for all systems at sufficiently high temperatures, and indeed for most systems that are of interest to us in this thesis. For a classical system, we can use Newtonian mechanics $\boldsymbol{F} = m\boldsymbol{a}$ to calculate the movement of the particles, a problem that we will get back to in part C.

Table 3: de Broglie thermal wavelength for carbon atoms at various temperatures

Temperature	Thermal wavelength
5 K	2.26 Å
20 K	1.13 Å
50 K	0.7 Å
100 K	0.5 Å
200 K	0.35 Å
300 K	0.29 Å
400 K	0.25 Å
500 K	0.23 Å

In comparison, the C-C bond length in graphene is 1.42 Å and in polyethylene it is 1.535 Å.

4.5 Optional: Several electrons orbiting a nucleus

In order to discuss a model for quantum chemistry, we will first need to construct a model for more than one electrons orbiting a nucleus. At this point the Schrödinger equation cannot be solved as one would need to take electron-electron repulsion into consideration.

First we will mention the Aufbau principle. This model sees the orbitals as bins that one can fit two electrons in, two electrons with different spin in each orbital. The electrons are added to fill the lowest energy states first, then working themselves up. An electron in an orbital will resist having another electron added to that orbital due to electron-electron repulsion, so that adding electrons to empty orbitals is preferable to half-filled orbitals. One might say that adding an electron to an orbital raises its energy. In some cases, it is favorable to add an electron to a higher shell than fill up an orbital.

The second model we will mention is the Slater determinant. We could think of a wave function for several electrons as $\Psi(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N) = \chi_1(\boldsymbol{x}_1)\chi_2(\boldsymbol{x}_2)\cdots\chi_N(\boldsymbol{x}_N)$, where $\mathbf{x}_i$ are the positions of the electrons and χ are functions of these positions. χ contains information about the electrons position and its spin state, and is called a spin orbital. This function form does not work, because the Pauli exclusion principle states that by exchanging the position of two electrons, the sign of the wave function will change[9]. A mathematical structure that satisfies this condition is a determinant, so we can write the wave function as following:

$$\Psi(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \chi_1(\boldsymbol{x}_1) & \chi_2(\boldsymbol{x}_1) & \cdots & \chi_N(\boldsymbol{x}_1) \\ \chi_1(\boldsymbol{x}_2) & \chi_2(\boldsymbol{x}_2) & \cdots & \chi_N(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \chi_1(\boldsymbol{x}_N) & \chi_2(\boldsymbol{x}_N) & \cdots & \chi_N(\boldsymbol{x}_N) \end{vmatrix}$$ (54) Slater determinant

The $1/\sqrt{N!}$ term is a normalization factor.

[9] A corollary of this where two electrons occupy the same place, where any exchange of the positions will give the same wavefunction, since electrons are indistinguishable will have only one solution: Ψ = 0. This means that it is impossible to put two electrons or indeed any half-spin particles (such as electrons, protons and neutrons in the same position).

4.6 Optional: Quantum chemistry and molecular orbitals

This section is not immediately useful for most molecular mechanics, as we will model a bond as a force that ties molecules together. We will however use it to motivate the Morse potential in Part C, section 2.5. The Morse potential will be used to model bond stretching for most of our research.

There are several ways to model the quantum mechanics of atomic bonding, all of whom require the BOA to work. We will here look briefly at two models, Linear Combinations of Atomic Orbitals (LCAO) and Density Functional Theory (DFT).

Covalent bonding in chemistry assumes that two atoms share electrons, for our purposes this means that the orbitals containing these electrons "overlap". We write overlap in scare quotes, since simply laying an orbital on top of another violates the Pauli principle. Instead a new orbital is formed, called a molecular orbital. LCAO assumes that we can write this orbital as some linear combination of the participating orbitals:

$$\psi_i = \sum_{\mu} c_{\mu i}\, \varphi_{\mu}$$

(55) LCAO approach

μ represents the quantum numbers nlm, c is a coefficient and φ are basis functions, in this case the atomic orbitals obtained in equation (44). The coefficients are set so that energy is minimized, and the way to find these coefficients is to use Hartree-Fock theory in general and the Roothaan-Hall equations in special (Leach, 2001a).

For the H_2 molecule, the molecular orbital is a linear combination of two $1s^1$ orbitals. Finding the coefficients give us two different orbitals,

$$\psi = \varphi_{1s}^{1} + \varphi_{1s}^{2}$$

(56) Hydrogen bonding orbital

$$\psi_{*} = \varphi_{1s}^{1} - \varphi_{1s}^{2}$$

(57) Hydrogen antibonding orbital

The two solutions have different energies, the bonding orbital has lower energies than the two non-bonded hydrogen atoms. The other solution called the antibonding orbital has higher energies than the separate atoms. These two statements are shown schematically in Figure 15.

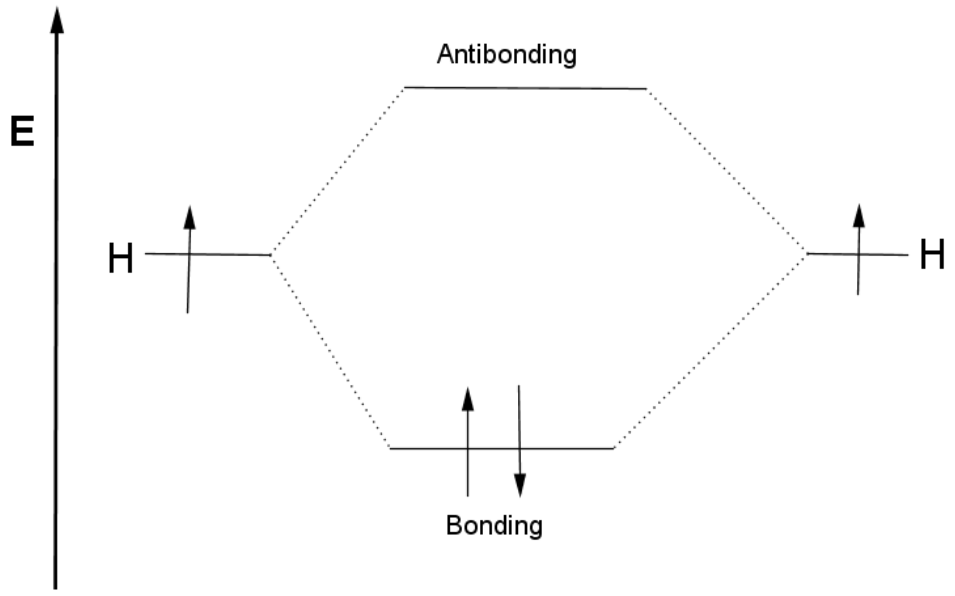


Figure 15: Schematic illustration of a molecular orbital in a bond. Each line is an orbital, and the higher up the line is, the higher energy the orbital has. Individual atoms are at each end, and the molecule is in the middle. Each arrow corresponds to an electron. The number of electrons for the individual atoms must be the same as for the molecule.

A bond is considered unstable if there are equal or more electrons in antibonding orbitals than the bonding orbitals.

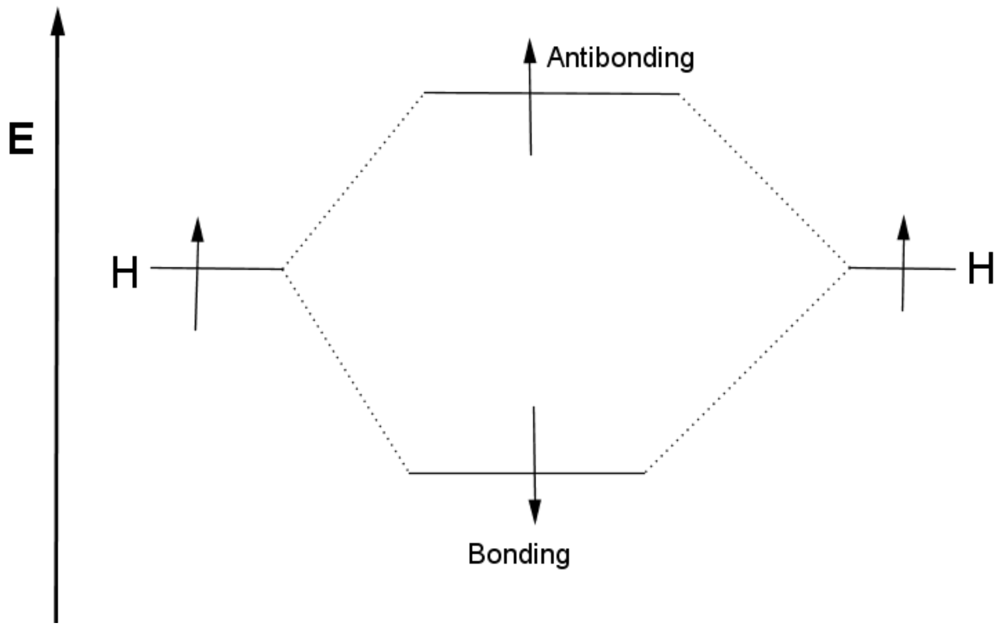


Figure 16: An electron is in the bonding orbital and the other is in the antibonding orbital. The molecule is unstable and the bond breaks.

This method can be generalized to other molecules as well. The energy of the bonding orbital for H_2 as a function of the interatomic distance is plotted below. This model predicts the interatomic separation quite well, but overestimates the depth of the potential well (energy at minimum compared to infinite separation) by a factor of 4.

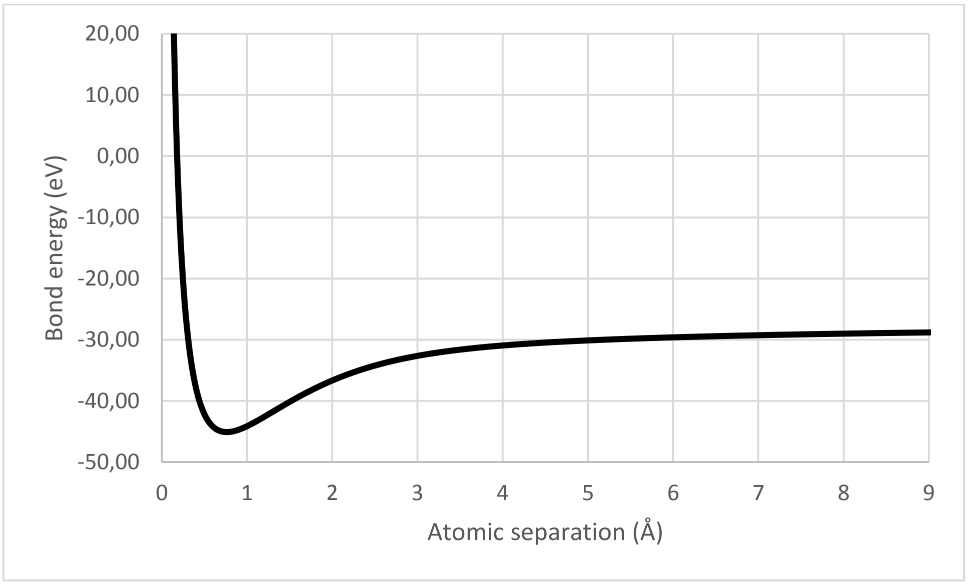


Figure 17: Plotting the hydrogen bond energy from molecular orbitals. Compare the shape of this energy with Figure 31.

LCAO and Hartree-Fock theory uses many assumptions to model the molecular orbital. A more modern method that is often used in quantum chemistry (ab initio) calculations is called Density Functional Theory (DFT). In simple terms, DFT replaces the different electrons with a density term $\rho(\boldsymbol{r})$ which can be used to determine the energy of the system. This density can be found from the position of atoms and their orbitals. We have now a potential energy field that we can place a single particle in, and solve for its movement. From this we can make quite good calculations on molecules that are too large for the Schrödinger equation to solve directly.

5 Statistical mechanics

Now that the foundations for molecular mechanics have been laid out, we will next look at ways to relate the microscopic world with the macroscopic. The key lies in statistical mechanics, and we will breezily present key models and results.

5.1 Hamiltonian mechanics

The theoretical groundwork for statistical mechanics is given by the Hamiltonian formulation of dynamics. Let's say we have a system of N particles $\{\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \dots, \boldsymbol{r}_N\}$ with associated momentums $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3, \dots, \boldsymbol{p}_N\}$. If some of the particles' positions are dependent on others, e.g. if one particle is kept at a constant distance from another, then we may represent the positions by generalized coordinates q_i, so that $\boldsymbol{r}_i = f_i(q_1, q_2, \dots, q_M)$, where $M \leq 3N$ is called the number of degrees of freedom in the system. These coordinates have associated generalized momentums p_i. The generalized coordinates does not need to have the units of distance, and the generalized momentums does not need to have the units of momentum, but the product of these two must have the unit of energy.

Hamilton's equations are usually written with dot notation, but since this dot may be hard to see, we have instead used the d/dt notation instead:

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}, \qquad \frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}$$ *(58) Hamilton's equations*

These equations give us a complete picture of the dynamics of the system, which we will in short denote as the *trajectory*. A trajectory from one set of coordinates $\{q_1^1, q_2^1, \ldots, q_M^1,\ p_1^1, p_2^1, \ldots, p_M^1\}$ to another at a later or earlier time $\{q_1^2, q_2^2, \ldots, q_M^2,\ p_1^2, p_2^2, \ldots, p_M^2\}$ is called a canonical transformation in time, where the new Hamiltonian may not be the same as the old one.

The (classical) Hamiltonian is here given as the total energy or the sum of kinetic and potential energy.

$$H = E = K + U$$ *(59) The total energy*

The total potential energy is a sum of internal (i.e. atomic interactions) and external (e.g. gravitation, electric field, walls) potentials.

$$U = U^{int} + U^{ext}$$ *(60) The potential energy*

We will get back on how to calculate this energy in Part C, section 2.1.

The total kinetic energy can likewise be predicted from the sum of each particle's kinetic energy (here written in particle form, rather than degrees of freedom form):

$$K = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i \cdot \boldsymbol{p}_i}{2m_i}$$ *(61) Kinetic energy from momentum*

The classical Hamiltonian is then written as (compare with equation (51)):

$$H = \sum_{i=1}^{N} \frac{\boldsymbol{p}_i^2}{2m_i} + U^{int}(\boldsymbol{r}) + U^{ext}(\boldsymbol{r})$$ *(62) The classical Hamiltonian*

5.2 Phase space

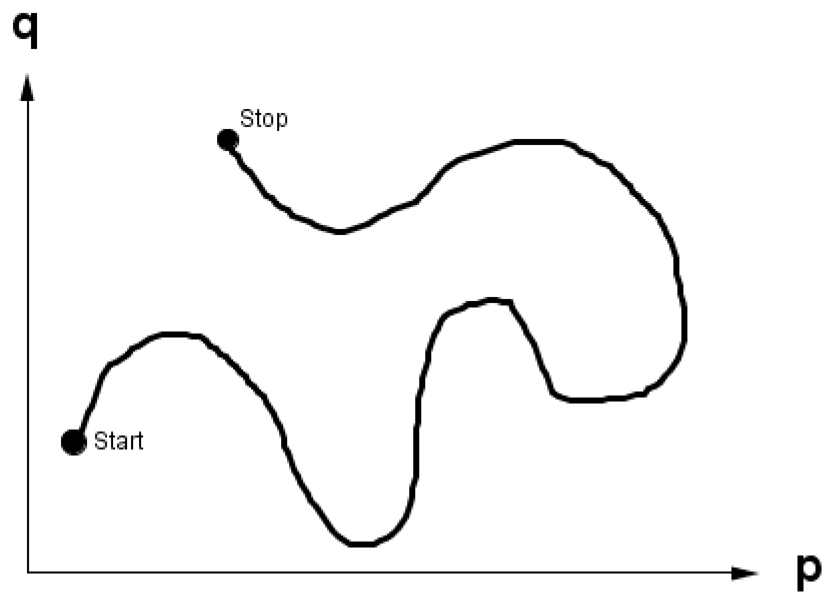


Figure 18: A schematic representation of a system's trajectory through phase space

We will now introduce the concept of phase space, a theoretical used to understand the dynamics of a system. A phase space is a 2M-dimensional space where a single point has the coordinates $\boldsymbol{x} = (q_1, q_2, q_3, \dots q_M, p_1, p_2, \dots p_M)$. Each point represents a state of some system. As the system changes state over time (governed by (58)), then it will follow a continuous and smooth line, the trajectory. This trajectory gives us a picture about how the system will look like in the future.

We can pick multiple independent systems in the phase space. If the choice of systems is constrained, for example in a way that all systems have the same number of atoms, volume and temperature, then this collection of points is called an *ensemble*, and the example is called an *NVT ensemble* or *canonical ensemble*. This ensemble would be contained in some subset of the phase space, on which we can describe a point density ρ as the probability of finding a picking a point at a given place.

$$P(\boldsymbol{x}) = \rho(q_1, q_2, \dots, q_M, p_1, p_2, \dots, p_M)$$ *(63) The distribution function*

Since we require the probability of picking a point in the entire region to be 1, this distribution function needs to be normalized in the same way as the wave functions of quantum mechanics.

To illustrate the distribution function, consider figures Figure 19 and Figure 20. In both figures we pick 15 points from a region (in red), each corresponding to a system.

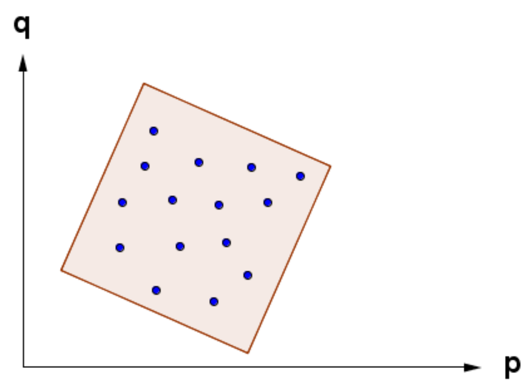


Figure 19: Points are distributed uniformly over the phase space.

In Figure 19, the points are evenly distributed, so that the distribution function is simply 1/V_R, where V_R is the volume of the region. In Figure 20, there is a slight bias towards the lower left corner, so that the points are more often picked from there than the rest of the region.

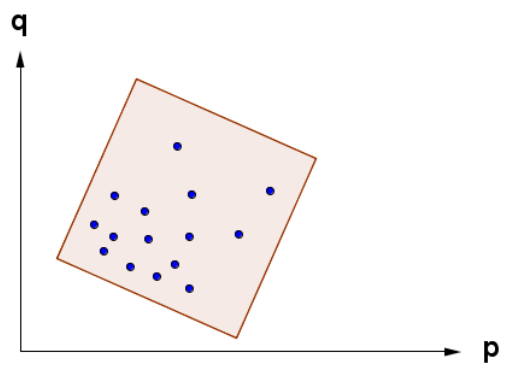


Figure 20: There is a higher probability of picking a point in the lower left corner than the rest of the space.

We will get back to the distribution function later, but for now, we present the following theorem:

$$\frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + \sum_{i}^{M} \left[\dot{q}_i \frac{\partial \rho}{\partial q_i} + \dot{p}_i \frac{\partial \rho}{\partial p_i} \right] = 0 \qquad \text{(64) Liouville's theorem}$$

This equation has the same form as the continuity equation from fluid mechanics, and it represents a similar idea (Figure 21). As the systems move through phase space, the area of the regions containing the system is kept constant at all times, so that the velocity $(\dot{q}_i, \dot{p}_i)$ of each system is constrained in such a way to keep the area of the region constant. If needed be, the velocities are slow, for example.

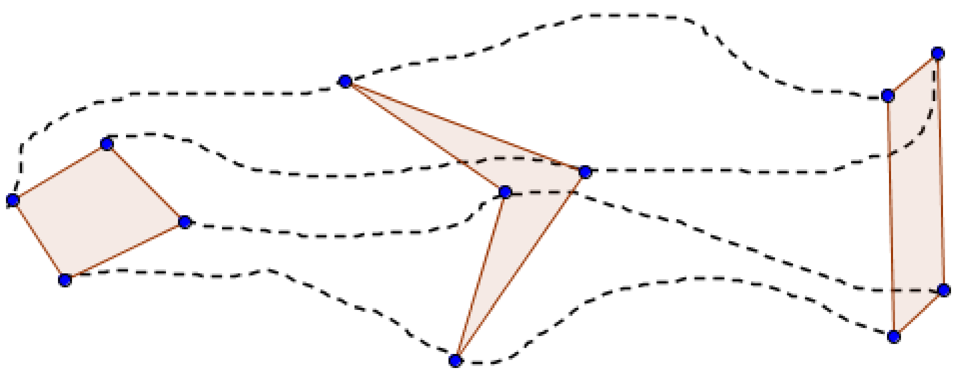


Figure 21: A common interpretation of Liouville's theorem. Four points form the vertices of a polygon, and as they move through phase space, the area/volume of the polygon does not change, but its shape may.

5.3 Calculating macroscopic observables: The time average approach

When we run observe a system, we usually do this over time, it may be over seconds or over hours. On the macroscopic, level we observe values such as temperature, pressure and energy, which we will collectively denote by the variable A, and on the microscopic level we can calculate these values by some function which we will denote as $A(q_i, p_i)$. The relation between these two are given by:

$$\bar{A} = \lim_{\Delta t \to \infty} \frac{1}{\Delta t} \int_0^{\Delta t} A(q_i(t), p_i(t))\, \mathrm{d}t \qquad \text{(65) Time average}$$

The two figures below are molecular dynamics simulations done on 200 united atom (see part C, section 2.2) octane molecules in contact with a heat bath (see part C, section 3.8.2). For every thousandth time step, the total energy of the system has been calculated and plotted, along with its running average, computed in MATLAB as `cumsum(E)./(1:length(E))` (this is actually equation (68) which describes ensemble averages, but we will use these figures as a starting point for our discussion).

First, note that the initial energy is quite high. Both simulations start with the same initial structure (a system of octane molecules held at 900 K to get a semi-random initial structure)with the same initial potential energy. When the simulations start, they are quenched to a temperature 250 K (Figure 22) or 700 K (Figure 23) with different kinetic energies. Over time, the energy of the simulation drops, as the systems eventually reach equilibrium. This is called *relaxation*.

For Figure 23 the system relaxes almost immediately, but Figure 22 shows that the system does not reach equilibrium over the simulation run. As explained in Part D, section 3.1, this is because equilibrium is reached by solidifying the system, and this takes time. In opposition, equilibrium at 700 K happens quickly because the system is fluidic at this temperature.

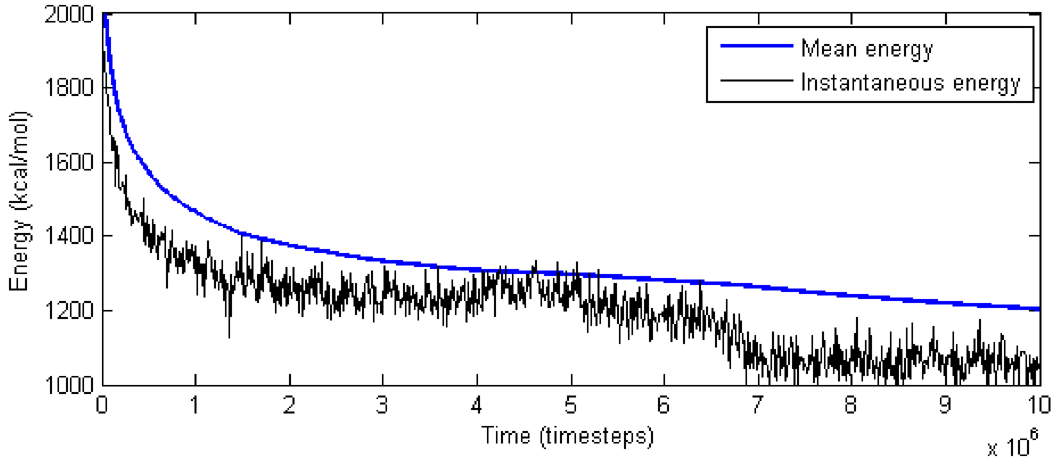


Figure 22: Energy of a liquid of octane (with temperature 900 K) quenched to a solid at temperature 250 K. Note that the mean energy does not reach an equilibrium during the simulation.

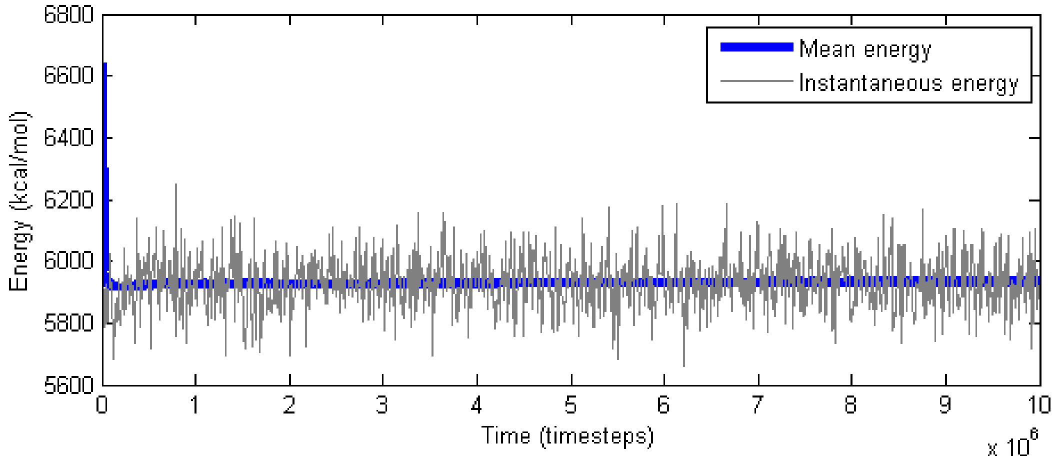


Figure 23: The same initial structure is kept at 700 K, and reaches equilibrium quickly.

Finally note the violent fluctuations of energy, and the reason we average the energy in the system. The standard deviations are respectively $\sigma_{250} = 57.4$ kcal/mol K, and $\sigma_{700} = 91.1$ kcal/mol for 700 K.

Calculating $\bar{A}$ is difficult in practice, since we require the system trajectory to pass over all possible points of phase space, which would take a long time. Time averaging can only properly be used for the smallest of systems.

5.4 Calculating macroscopic observables: The ensemble approach

An ensemble is a large collection of systems that have the same properties, see figure 5.1 in (*Metzger, 2012*). The ensemble average is thought of as calculating A(q_i,p_i) for all systems in the ensemble, and averaging them, resulting in A. For the phase space, this procedure is summed up as:

$$\langle A \rangle = \int A\big(q_i(t), p_i(t)\big)\, \rho(q_i, p_i) \mathrm{d}q_1 \mathrm{d}q_2 \cdots \mathrm{d}q_M \mathrm{d}p_1 \mathrm{d}p_2 \cdots \mathrm{d}p_M$$

(66) Theoretical ensemble average

Here we have again used the distribution function (63) to weigh the outcome A(q_i,p_i) with its probability of finding it. The ensemble average is much more flexible than the time average, and the relation between these two is a point of contention in the theory behind statistical mechanics. It is thought that:

$$\bar{A} = \langle A \rangle$$

(67) Ergodic hypothesis

This equation **does not** in general hold. It requires that the system's trajectory passes over all points in phase space that is consistent with the ensemble. It is however believed that this equality becomes valid when the trajectory visits a reasonable amount of points, so that (67) may be taken as an axiom for statistical mechanics (Ellad B. Tadmor & Miller, 2011c).

What we are doing is essentially using statistical methods to infer properties about the whole region from a set of samples. We can rewrite the ensemble average as such:

$$\langle A \rangle = \lim_{N\to\infty} \frac{1}{N} \sum_{Measurements}^{N} A(q_i(t), p_i(t))$$

(68) Practical ensemble average

Figure 22 and Figure 23 were made using this method. If one only measures the evolution of one system over time as we will do in this thesis, then equations (65) and (68) fall together. Think of the ensemble average as a speed-up of the time average.

We will finally describe some popular ensembles and their associated distribution functions.

5.4.1 The microcanonical ensemble

The microcanonical ensemble (µCE or NVE ensemble) is the set of all systems with a given number of atoms (N), system volume (V) and energy (E). Thermodynamically it's equivalent to an isolated system. Its distribution function is given as

$$\rho(q_i, p_i) = \frac{1}{Q_{NVE}}$$

(69) The microcanonical distribution function

This is a constant, so the particles are spread evenly over the region (compare Figure 19)Q_{NVE}, sometimes written Ω is the volume of the region in phase space, often called the number of microstates in the system, formally defined for a single-component system as (*Frenkel & Smit, 2001*):

$$Q_{NVE} = \frac{1}{h^{3N} N!} \int \delta[H(q_i, p_i) - E]\, (\mathrm{d}q_i)^{3N} (\mathrm{d}p_i)^{3N}$$

(70) The microcanonical partition function

δ is the Dirac delta distribution, and $h^{3N}N!$ is a correction term. The first component aims to make Q_{NVE} dimensionless (*h* as dimensions momentum · distance) and the second component, the "correct Boltzmann counting" removes overcounting when the integral counts identical states where two or more indistinguishable atoms have switched places. The most-known application of Q_{NVE} is the famous relation after Boltzmann:

$$S(N,V,E) = k_b \log Q_{NVE}$$ *(71) Boltzmann's equation*

The microcanonical ensemble can be used to derive the Maxwell-Boltzmann velocity distribution (81), which is used when one wants to assign velocities to atoms consistent with a chosen temperature.

For an ideal gas with a large number of atoms, it is possible to work out this form for the entropy:

$$S = k_b N \ln\left[\frac{e^{\frac{5}{2}} m^3 V}{N h^3}\left(\frac{4\pi U}{3mN}\right)^{\frac{3}{2}}\right]$$ *(72) Sackur-Tetrode equation, entropy for an ideal gas*

Solving for U (internal energy) and using equatiosn (7) and (8), we get the temperature and pressure:

$$T = \frac{2U}{3Nk_b}, \qquad p = \frac{2U}{3V}$$

Comparing the two equations, one will end up with the ideal gas law: $pV = nk_bT$.

5.4.2 The canonical ensemble

The canonical ensemble (CE or NVT ensemble) has a fixed number of atoms (N), fixed volume (V) and fixed temperature (T). Thermodynamically, the canonical ensemble is equivalent to a closed system. The distribution function is given as:

$$\rho(q_i, p_i) = \frac{1}{h^{3N} N!\, Q_{NVT}} \exp\left(-\frac{H(q_i, p_i)}{k_b T}\right)$$ *(73) Canonical distribution function*

Q_{NVT} is again a sum-over-states that normalizes the distribution function:

$$Q_{NVT} = \frac{1}{h^{3N} N!} \int \exp\left(-\frac{H(q_i, p_i)}{k_b T}\right) (dq_i)^{3N} (dp_i)^{3N}$$ *(74) Canonical partition function*

Once again the partition function generates a thermodynamic potential. In this case the Helmholtz free energy:

$$A(N,V,T) = -k_b T \ln(Q_{NVT})$$ *(75) Helmholtz free energy*

If we want to study an ideal gas in the canonical ensemble, then we can represent the Hamiltonian (62) by only its kinetic part. The partition function is thus:

$$Q_{NVT}^{ideal} = \frac{1}{N!}\left[\frac{V}{h^3}(2\pi m k_b T)^{\frac{3}{2}}\right]^N$$ *(76) Partition function for an ideal gas*

It is possible to derive the ideal gas law from this equation without using the large N approximation.

5.4.3 Optional: Other ensembles

Two other occasionally used ensembles are the grand canonical ensemble (GCE, µVT) and the isobaric-isothermal ensemble (IIE, NPT). They generate the grand canonical function (18) and the Gibbs free energy respectively.

5.5 Some important results

Finally, we will move on to some useful microscopic computations that can be related to macroscopic functions. Our sources have been (Leach, 2001c; Ellad B. Tadmor & Miller, 2011c), and the documentation for LAMMPS (*"LAMMPS WWW site," 2013*).

The **total energy** is given as:

$$E = \langle H \rangle = \langle K \rangle + \langle U \rangle$$

(77) Total energy and Hamiltonian

The kinetic energy is given by equation (61), and we will get back to the potential energy in Part C. The total energy that we can measure in our system is related to the **internal energy** in the following way:

$$U_{internal} = \langle H \rangle$$

(78) Internal energy and measured energy

We have given the internal energy a subscript to differentiate it from the potential energy U.

Likewise, the **enthalpy** for a system with a constant volume:

$$H_{ent} = U + pV = \langle H_{Ham} \rangle + \langle p \rangle V$$

(79) Enthalpy and measured energy and pressure

The subscripts differentiate the Hs for enthalpy and Hamiltonian.

The **temperature** of the system is related to the kinetic energy by the following formula:

$$\langle K \rangle = \frac{3}{2} N k_b T$$

(80) Equipartition theorem

A useful **velocity distribution** for classical particles:

$$p(v) = \left(\frac{m}{2\pi k_b T}\right)^{\frac{3}{2}} 4\pi \boldsymbol{v}^2 \exp\left(-\frac{m\boldsymbol{v}^2}{2k_b T}\right)$$

(81) Maxwell-Boltzmann distribution

Some typical speeds are:

$$v_{max} = \sqrt{\frac{2k_b T}{m}}$$

(82) Most probable speed

$$\langle v \rangle = \frac{2}{\sqrt{\pi}} v_{max}$$

(83) Expected value

$$\langle v^2 \rangle^{\frac{1}{2}} = \sqrt{\frac{3}{2}} v_{max}$$

(84) Root-mean-square speed

The Cauchy **stress tensor** (similar to the true stress, equation (28)) is given as:

$$\sigma_{ij} = \frac{1}{V}\left[-\sum_a m^a v_i^a v_j^a + \sum_a r_i^a f_j^a\right]$$

(85) Stress tensor for the entire system

Here a is an index for atoms, and ij are indexes for the directions of the tensor *x*, *y* and *z*. *f* is the force on each atom, given by (91). V is the volume of the system. The first term is the contribution from the kinetic energy, and the second term is the contribution from the potential energy, also called the *virial*. Both forces, velocities and positions are routinely kept track of in molecular dynamics simulations, so that the pressure is easy to calculate.

The scalar **pressure** of the system can be calculated as:

$$p = -\frac{1}{V}\left[-Nk_bT_{inst} + \frac{1}{3}\sum_a \boldsymbol{r}^a \cdot \boldsymbol{f}^a\right] \qquad \text{(86) Pressure scalar for the entire system}$$

Note that if we exclude the virial, then the equation above reduces to the ideal gas law. The virial term is thus a correction to the ideal gas approximation.

5.6 Optional: Equilibrium

We know intuitively that a system that minimizes its potential energy will be the most stable one, and that nature will seek this state spontaneously. Mathematically this is quite tricky to prove, but it can be done for continuum mechanics ((Ellad B. Tadmor & Miller, 2011b) page 107) and the thermodynamical equivalent can be derived from the second law of thermodynamics.

Here we will look at an illustration of this principle.

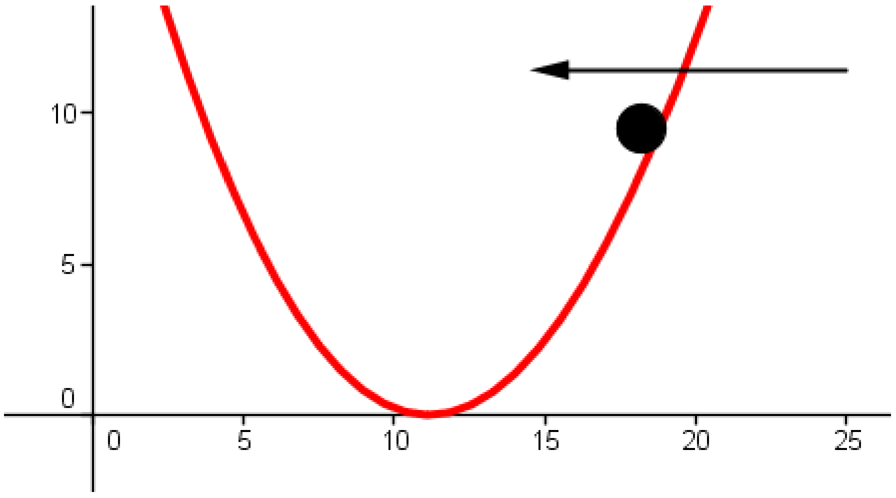


Figure 24: System seeking towards equilibrium

This figure illustrates a system with a ball in the presence of a potential energy field (red). The force from the potential energy is given by $F = -\nabla U$, so that the ball will be pushed towards the direction where the potential energy decreases the fastest.

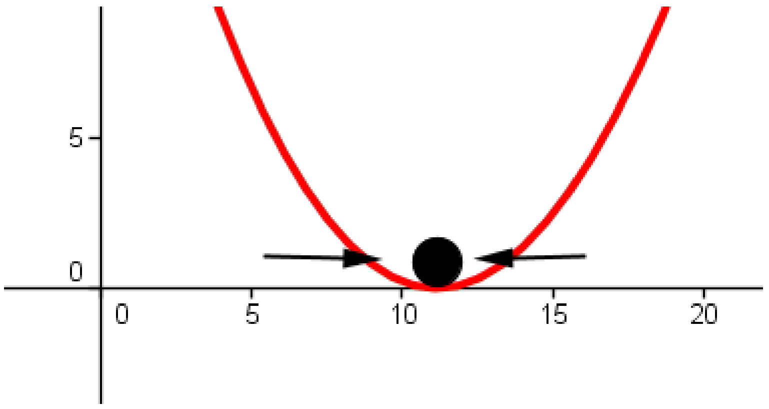


Figure 25: System in stable equilibrium

This figure shows the ball in a stable equilibrium. It is in equilibrium because there is no force that pushes it in either direction, and it is stable because any fluctuation away from equilibrium will result in a force that pushes the ball back into equilibrium (arrows).

One can have unstable equilibrium in the same way, illustrated as the maximum of a potential energy curve. Any fluctuation from that point will result in a force that pushes the ball away from the equilibrium.

Lastly, we have the metastable equilibrium. It is locally stable, but a large enough energy fluctuation may push it over some barrier to a more stable state.

PART C: APPLICATIONS

We will now turn our heads to the application of molecular mechanics. We will here concern us with the questions regarding the simulation, like how to move atoms and how to set up an initial configuration.

1 An introduction to LAMMPS

LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) (*Plimpton, 1995*) is the molecular mechanics software that we will use for our simulations.

LAMMPS is written in C++, open source and developed by Sandia National Laboratories. It can be found on the web at http://lammps.sandia.gov/index.html

It is primarily built for running in systems with multiple processors. This works as follows: A system containing atoms is split up into several subsystems that each processor needs to work with. This is called parallel processing, as opposed to serial processing. When a processor needs information about the system on another processor (to calculate the interaction energy between two atoms on different processors, for example), then it can send a request to the processor in question to do that calculation. LAMMPS uses MPI (*Aoyama & Nakano, 1999*) to do this. This might sound like a lot of extra work, but it is actually much faster than running the simulation on a single processor, see Appendix 1.

Alternatives to LAMMPS include Gromacs (*Berendsen, van der Spoel, & van Drunen, 1995; Hess, Kutzner, Van Der Spoel, & Lindahl, 2008; Lindahl, Hess, & Van Der Spoel, 2001; Van Der Spoel et al., 2005; van der Spoel et al., 2008*) (see also appendix 2) and NAMD (*Phillips et al., 2005*).

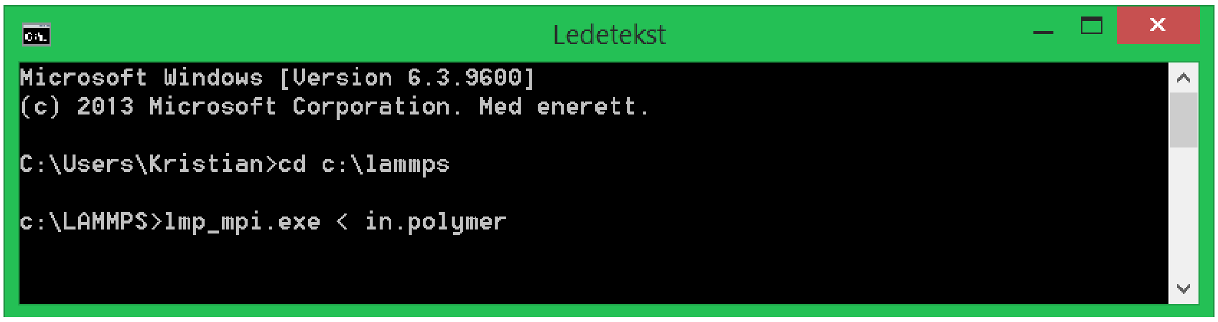
1.1 *Installing LAMMPS*

LAMMPS can be downloaded from http://lammps.sandia.gov/download.html, via Github or Homebrew (for Mac), a personal package archive (for Ubuntu) or executables from http://rpm.lammps.org/windows.html (for Windows). If one runs on Windows and has more than one processor core, one will need additional programs to implement MPI, these are given in the link above, and require a little extra work.

The windows installation will pack out to a folder called `/LAMMPS + version number`. The excecutables `lmp_mpi.exe` and `lmp_serial.exe` are both in the `/bin` subfolder. What I do is that I copy one executable (if you run serial, then there is no difference between them) over to the folder containing run-files.

1.2 *Running LAMMPS*

LAMMPS is text-based, and accepts commands in the same way as for example MATLAB does. Just as for MATLAB, we can write scripts that do the work for us, called run-files. In order for LAMMPS to read the script, one should (in Windows) open the PowerShell/command prompt and navigate to the folder where the executables lie, from here called the home folder. From there on, one writes one of the following commands: `lmp_mpi.exe < in.whatever`, which means that we open the executable lmp-mpi.exe and asks it to read the file in.whatever that lies in the home folder.



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. Med enerett.

C:\Users\Kristian>cd c:\lammps

c:\LAMMPS>lmp_mpi.exe < in.polymer
```

Figure 26: How to start LAMMPS. We first navigate to the folder where LAMMPS lies, on my computer defined as c:\lammps. It could well be in any other folder. From that folder we run LAMMPS (lmp_mpi.exe) and asks it to read the script called in.polymer.

If for some reason the process needs to stop immediately, one can press ctrl+c to kill the LAMMPS process.

We will get back to what other files and folders should be in the home folder in Part D, section 1, and we will get back to LAMMPS in Part C, section 3, but first we will need to specify the potential energy field for the atoms.

2 Empirical force field models

We will now describe the potential energy field U that we will be using in these simulations. We will use a family of potentials (*Paul, Yoon, & Smith, 1995*) that together describe U and can be use to predict macroscopic properties, and tweak it so that it fits our tensile tests.

2.1 Modeling the potential energy

A force field is simply a way to push a molecule into a desired configuration. As an example, imagine two atoms bonded with each other, with some bond length r. From experiments or ab initio simulations (like (*Aljibury, Snyder, Strauss, & Raghavachari, 1986*)), we can tell that the bond length is typically of some length r^b, where the forces on the bond are in equilibrium. We want to model these forces in a simple non-quantum mechanical matter, where any bond length away from equilibrium is penalized in some manner.

The solution is to use potential energies, defined to be 0 when the bond is in stable or metastable equilibrium (see Part B, section 5.6) and there is no outside influence on the bond. This is called the reference, where r^b is the reference bond length. We set the potential to zero, but in reality it is only zero in reference to the so-called zero-point energy.[10] Any deviation from the stable or metastable equilibrium (viz. pushing atoms together, or pulling them apart) will result in increased potential energy, resulting in a force that pushes the bond back to equilibrium. As shown in the sections below, this can be used to control all sorts of molecular geometry, like bonds, angles and dihedrals. We call these bonded terms.

For interactions between molecules, things get more complicated. The individual atoms in the molecule will attract and repulse each other with van der Waals forces or Coulomb forces. In

[10] This convention will not make a difference in our calculations, but the outputted values for total potential energy and total energy will be meaningless unless in context.

opposition to the bonded terms above, these forces interact with all other atoms in the neighborhood, except the ones that are bonded to it. To better illustrate this, consider a gas of N atoms, with no bonds. The potential energy field is described by a function Φ.

$$U(x,y,z) = \Phi_{\mathrm{N}}(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \dots, \boldsymbol{r}_N)$$

(87) The potential energy field

This field can always be decomposed in some way into contributions from one-body terms, two-body terms and so on (J. Martin, 1975; Ellad B. Tadmor & Miller, 2011a).

$$\Phi_N = \varphi_0 + \sum_i \varphi_1(\boldsymbol{r}_i) + \frac{1}{2!}\sum_{i,j} \varphi_2(\boldsymbol{r}_i, \boldsymbol{r}_j) + \frac{1}{3!}\sum_{i,j,k} \varphi_3(\boldsymbol{r}_i, \boldsymbol{r}_j, \boldsymbol{r}_k) + \cdots$$

(88) Force field decomposition

The requirement is that that $i \neq j \neq k \neq \cdots$ in the summation (meaning that no two particles can be the same in an evaluation of φ_m, that the multiple-body terms satisfy permutation symmetry ($\varphi(\boldsymbol{r}_i, \boldsymbol{r}_j, \boldsymbol{r}_k) = \varphi(\boldsymbol{r}_j, \boldsymbol{r}_i, \boldsymbol{r}_k)$, for example) and that if at least one particle in the term is infinitely removed from the others, the term reduces to zero.

The φ_0 term is the zero energy term mentioned above and is customarily set to zero. The φ_1 term likewise represents an external potential energy field (compare equation (60)), and may also be dropped. The really interesting term here is the φ_2 term, which we will call the pair potential. This function computes the potential energy between pairs of atoms, the internal potential energy. This term covers van der Waals and Coulomb forces induced from an atom *i* on its neighbors *j*, and is the one most encountered in literature.

Other terms such as the 3-body and higher count as corrections to the pair potential. Note that the angle and dihedral terms as defined below are *not strictly* the same as the three-body and four-body terms in equation (88) above. This is because by changing the order of atoms, the angles will change as well. Nevertheless, for convenience we will write these down as multi-body potentials. We stop at four-body potentials, since a collection of four atoms is the least amount you need to describe non-planar molecules, something that is a common occurrence in three dimensions. We will now denote the potential energy as:

$$U(x,y,z) = \sum_{i,j,k,l} U_{ijkl}$$

(89) Potential energy capped off at four-body potentials

The U_{ijkl} term contains the two-body, three-body and four-body terms from (88). By fixing one atom, and summing over all of its neighbors, one gets the potential energy for that atom.

$$U_i = \sum_{j,k,l \neq i} U_{ijkl}$$

(90) Potential energy for a single atom

We can calculate the force on a single atom from the potential energy by $\boldsymbol{F}_i = -\nabla_{\mathrm{i}} U_i$ (thus our alternative name for the potential energy field: Force field), and from Newton's second law can we work out the trajectory of each atom. Here $\nabla_i = \frac{\partial}{\partial \boldsymbol{r}_i}$ is the gradient operator acting on a single particle. This is why we write the potentials as simple analytical functions. For such a function, one can find the analytical derivative and work out the exact force, instead of doing a numerical integration with errors.

$$m_i \ddot{\boldsymbol{r}}_i = \boldsymbol{F}_i = -\nabla_i \sum_{j,k,l \neq i} U_{ijkl}$$ *(91) Newton's second law in potential field form. Newtonian dynamics*

Some additional terms may be added on the right side, like a friction term, see section 3.8.2 for an example. See section 3.6 on how to solve this equation. The potential energy term can be seen as a rewriting of (88):

$$\sum_{j,k,l \neq i} U_{ijkl} = \sum_{j} U_{ij}^{nb} + \sum_{j} U_{ij}^{b} + \sum_{j,k} U_{ijk}^{a} + \sum_{j,k,l} U_{ijkl}^{d}$$ *(92) Contributions to the force field*

The first expression on the right side is the nonbonded potential, and the three last expressions are the bonded interactions, respectively bond length, angle and dihedral potentials, all of them described in sections 2.3-2.7 below.

The bonded interactions can also be grouped into 1-2, 1-3 and 1-4 interactions. Here 1-2 is the term for two atoms separated by a bond, 1-3 is by two atoms separated by two bonds and 1-4 is by two atoms separated by three bonds and so on. There may be even more terms on the right hand side, but we will only use these four. Some additional terms are mentioned in sections 2.8 and 2.9.

Any atoms belonging to the same molecule, but with a coordination 1-5 or higher also count as non-bonded atoms, so the non-bonded interactions are calculated for them as well.

All parameters are from (*Sahputra & Echtermeyer, 2013*), unless otherwise noted. Other popular force field families, include TraPPE (*M. G. Martin & Siepmann, 1998*), which was built around modeling unbranched alkanes and expanded from there, and MMFF94 (*Halgren, 1996*), built as an "universal" organic molecule force field and OPLS *(Jorgensen, Maxwell, & Tirado-Rives, 1996*).

2.2 United atom representation

Before we begin, let us discuss a way to simplify our calculations. Let us illustrate this by counting the number of interactions of a simple butane molecule (Figure 27).

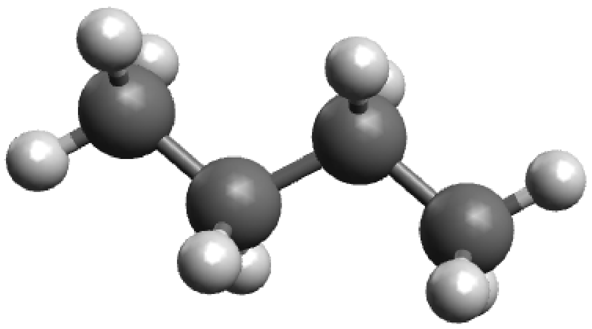


Figure 27: Butane

This simple-looking molecule requires many interactions to describe. By assuming that 1-5 coordinations and higher count as non-bonded interactions, we get the following:

Table 4: Number of interactions in a butane molecule, explicit hydrogen

Type of interaction	Number of interactions	
1-2 (bonds)	13	3 C-C, 10 C-H
1-3 (angles)	24	2 C-C-C, 14 C-C-H, 8 H-C-H
1-4 (dihedrals)	27	1 C-C-C-C, 10 C-C-C-H, 16 H-C-C-H
1-5 or higher	15	6 H-C-C-C-C, 9 H-C-C-C-C-H

This adds up to 69 interactions, or 54 if one only want to count up to the 1-4 interactions. If two butane molecules interact, then there will be 196 non-bonded interactions between them as well.[11] Let us now see what happens when we remove the hydrogen atoms (Figure 28).

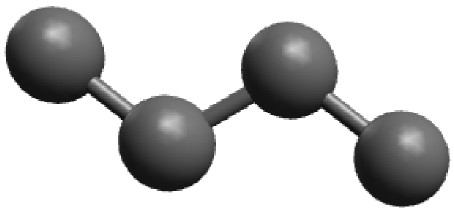


Figure 28: Butane, but with no hydrogens drawn.

All the hydrogens are now made implicit, and the geometry of the molecule is decided by the different potentials.

Table 5: Number of interactions in a butane molecule, implicit hydrogen.

Type of interaction	Number of interactions	
1-2 (bonds)	3	C-C
1-3 (angles)	2	C-C-C
1-4 (dihedrals)	1	C-C-C-C

6 interactions in all plus 16 non-bonded terms if two butane molecules interact, that is less than a tenth of what we got with explicit hydrogen atoms. If we were to only model these interactions, we could save a lot of time. This model is called the United Atom (UA) model, as opposed to the explicit atom (EA) or All-atom model (AA). The point of UA is to remove all atoms not necessary for the simulation. For organic molecules, we can formulate some rules for UA:

1. Hydrogens are excluded from the simulation, unless they are bonded to a polar molecule such as O or N (This is to model hydrogen bonds better).
2. Hydrogens count as joined on the parent atom, an H-C-H monomer becomes a CH_2 *pseudoatom*. The masses add up, so that the mass of this pseudoatom is 14.03 u.
3. For computational simplicity, this pseudoatom should be placed where the parent atom stood

[11] The individual cases are 16 $\mathrm{C} \sim \mathrm{C}$, 80 $\mathrm{C} \sim \mathrm{H}$ and 100 $\mathrm{H} \sim \mathrm{H}$

We will slightly violate rule 2 in our simulations. Both CH_2 and CH_3 will be represented by the same pseudoatom, with a mass of 14.03 u. This is to save some computation time. The error is less than 7 %, so this simplification should be acceptable.

Rule 3 is the source of another subtle systematic error. A pseudoatom is isotropic, but CH_2 monomer isn't. By isotropic we mean that the non-bonded potentials are measured equally in all directions. There are models that corrects this by placing the pseudoatom in the geometrical center of a CH_2 triangle, though they are not widely in use. Look for example at the Anisotropic United Atom model of (*Toxvaerd, 1990*)).

There has been made comparisons between AA and UA, and while earlier models may have not been up to snuff (*Yoon, Smith, & Matsuda, 1993*), more modern models do significantly better (*Chen et al., 2006*). The question is if one wants to trade off the shorter simulation time of UA with the less systematic errors of AA.

It is of course possible to take this is a step further, and represent several atoms as a single particle. These are called coarse-grained molecular mechanics, and represent a compromise between the atomistic level and continuum mechanics, a so-called multiscale model *(Rudd & Broughton, 1998*). This coarse-grained approach has been used to model polymers (with $C_{20}H_{40}$-particles) on a larger scale than is viable with ordinary UA models *(Padding & Briels, 2002*).

2.3 Nonbonded potential, Lennard-Jones

The most important component of the potential field, the unbonded potential decides how atoms that are unbonded, or more than three bonds away from each other will interact. The unbonded potential will determine how molecules will attract and repulse each other.

We will use the Lennard-Jones potential to calculate this potential. This potential was developed during the twenties by Sir John Edward Lennard-Jones, who was working on experimental data for Argon. Lennard-Jones wrote it down in a rather general form *(Jones, 1924a, 1924b*) for an ideal gas (equation (93)).

$$U(r) = \frac{A_m}{r^m} - \frac{C_n}{r^n}$$

(93) The Lennard-Jones potential (LJ potential)

The equation consists of two parts, an attractive force $\sim r^{-n}$, and a repulsive force $\sim r^{-m}$. The attractive force comes from London dispersive forces, where electrons between two neighboring atoms move in such a way that a dipole is temporarily created, causing the atoms to attract (Leach, 2001b). The repulsive force is easier to explain. Due to the Pauli exclusion principle (Part B, section 4.5, especially footnote 9), there are strong forces preventing electron clouds from overlapping. This force is better expressed as an exponential than a power (Leach, 2001b).

Later work showed that the attractive force between two noble gas atoms is on the order $\sim r^{-6}$ (*Eisenschitz & London, 1930; London, 1930*). For the sake of convenience, the power of the repulsive force is set to m = 12. This is so that when computers calculate the r^6 term, it can easily be squared to get the r^{12} term. In this form the so-called Lennard-Jones 12-6 potential has become one of the most used and abused pair potentials in molecular dynamics, with a vast literature of applications on many diverse systems far removed from its initial use in studying noble gases (Ellad B. Tadmor & Miller, 2011a).

An example of a potential that uses an r^{-6} term for attracting atoms, and an e^{-Cr} term for repulsing them is the Buckingham potential (*Buckingham, 1938*). Another one is the Born-Mayer potential, see (*Gibson, Goland, Milgram, & Vineyard, 1960*).

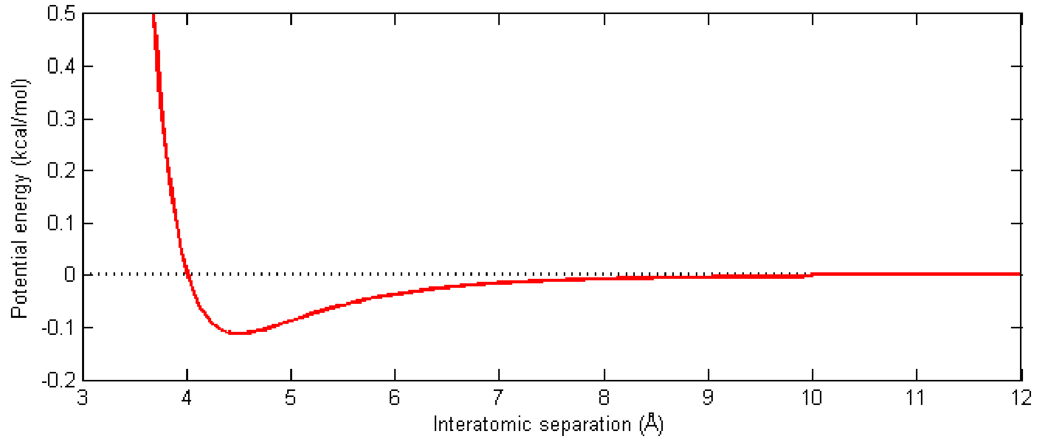


Figure 29: The Lennard-Jones 12-6 potential with the parameters from Table 6. The cutoff distance is 10 Å.

Figure 29 shows the Lennard-Jones potential that we will use in our simulations. The standard form of the potential is given below, equation (94), with the parameters given in table Table 6. ε is the depth of the potential well, or equivalently: The global minimum of the potential at interatomic separation r^0 is –ε. r^0 is called the equilibrium distance. The potential crosses the x-axis at the interatomic separation σ. σ and r^0 are related by the equation $r^0 = 2^{1/6}\sigma$, meaning that we need only two parameters to determine this potential.

As explained in section 3.3, calculating the pair potential between an atom and all other atoms in the simulation box is a huge and ineffective task, so a cutoff radius r_c is introduced, calculating pair potentials within this radius and disregarding all other atoms. One can say that the potential energy for two atoms with a distance larger than the cutoff radius is $U^{nb}(r_{ij} \geq r_c) = 0$. The cutoff distance is often set somewhere between 2σ and 2.5σ.

$$U^{nb}(r_{ij}) = 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right], r < r_c$$

(94) The Lennard-Jones 12-6 potential

$$U^{nb}(r_{ij}) = \varepsilon_{ij}\left[\left(\frac{r_{ij}^0}{r_{ij}}\right)^{12} - 2\left(\frac{r_{ij}^0}{r_{ij}}\right)^{6}\right], r < r_c$$
$$r_{ij}^0 = 2^{1/6}\sigma_{ij}$$

(95) An equivalent variant of (94), using the equilibrium distance as a parameter

The *ij*-indices stand for different atoms. Using our united atom models they could stand for CH_2 and CH_3 pseudoatoms, for example. LAMMPS accepts parameters on the form ε_{ii} and σ_{ii}, which shows how an atom interacts with similar atoms, and ε_{ij} and σ_{ij} which shows how an atom *i* interacts with an atom *j*.

Many nonbonded potentials include a Coulumbic term $\frac{q_i q_j}{4\pi\epsilon_0 r_{ij}}$, where *q* is the charge of the atom, and ε_0 is the permittivity of free space. This term will not be modelled, since we will only look at nonpolar hydrocarbons.

If LAMMPS searches for a potential for an atom pair *ij* whose coefficients have not been defined, one can set up a mixing rule using `pair_modify`. For example the Lorentz-Berthelot mixing rules (*Berthelot, 1898; Lorentz, 1881*) are simple, not very accurate (*Delhommelle & Millié, 2001*), but popular (see for example *(M. G. Martin & Siepmann, 1998*)).

$$\sigma_{ij} = \frac{\sigma_{ii} + \sigma_{jj}}{2}$$ *(96) Lorentz' rule*

$$\varepsilon_{ij} = \sqrt{\varepsilon_{ii} \cdot \varepsilon_{jj}}$$ *(97) Berthelot's rule*

These rules are respectively the arithmetic and geometric means of the Lennard-Jones parameters, and the `pair_modify` keyword is called `arithmetic`. If the user does not specify anything else, the `pair_modify` default is that the geometric mean is taken on both parameters. We will not use these rules in the simulations below since we will operate with only one atom type, CH_2.

Table 6: Parameters for the unbonded potential (Lennard-Jones)

Symbol	Name	Value
σ	Zero-crossing distance	4.01 Å
r^0	Equilibrium distance	4.50 Å
ε	Depth of the potential well	0.112 kcal/mol
r_c	Cutoff distance	10.0 Å

2.4 Bond length potential, Harmonic

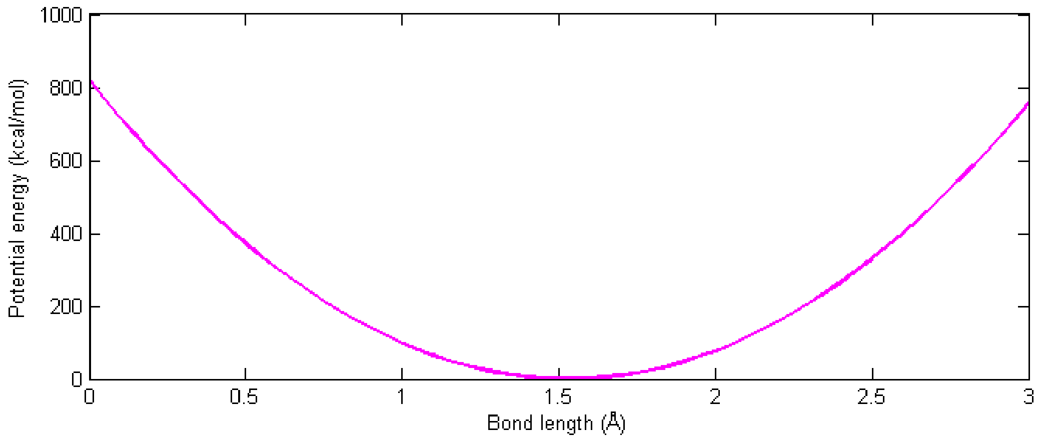


Figure 30: The harmonic bond length potential with parameters from Table 7.

The bond length potential models the potential energy for a stretched bond. We will here look at two related potentials, the harmonic and the Morse potential. The harmonic potential is quite simple, and is commonly used in many MD simulations. The harmonic potential was used in our thermodynamic simulations, for example. It is given by two equivalent definitions:

$$U^{b,H}(r) = \frac{k^b}{2}(r - r^b)^2$$
$$- - -$$
$$U^{b,H}(r) = K(r - r^b)^2$$

(98) The harmonic bond length potential and an equivalent definition, where $k^b/2 = K$

Here r^b is the reference bond length. The second definition is the one used by LAMMPS.

k^b is a significant parameter, seen by the interatomic force given by the harmonic potential:

$$F_{int}(r) = \frac{d}{dr}U^{b,H}(r) = k^b(r - r^b)$$

(99) Intern force from the harmonic potential

This is Hooke's law, and the reason why we prefer to write the potential with a $k^b/2$ term. One might say that this potential gives a linear elastic material behavior. A chain with these bonds could be seen mechanically as linear elastic springs connected in series.

This potential is taken from (*Sahputra & Echtermeyer, 2013*), who got their force field from (*Lavine, Waheed, & Rutledge, 2003*), who got it from (*Paul et al., 1995*). While Lavine operates with a harmonic bond length potential, Paul constrains the bond length to r^b = 1.53 Å, compare this with Table 2. Where Lavine got the bond stiffness parameter from is, as far as I can see, not explained, but they mention that the parameters have been found by *ab initio* calculations, possibly DFT or similar.

Table 7: Parameters for the bonded potential (Harmonic)

Symbol	**Name**	**Value**
r^b	Reference bond length	1.54 Å
k^b	Bond stiffness	700 kcal/mol
K	Bond stiffness parameter	350 kcal/mol

2.5 Bond length potential, Morse

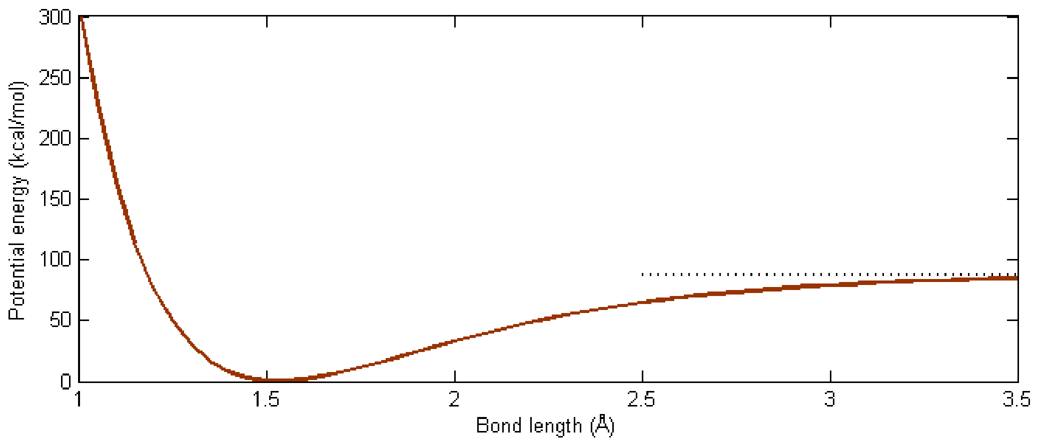


Figure 31: The Morse potential with the potentials from Table 8. Dotted line at the right of the figure is U = 88 kcal/mol, or the potential energy at infinite separation.

The harmonic potential does a good job when the bond lengths are not far removed from the reference bond length. However, we know that the potential energy between two atoms does not go to infinity when the bond is stretched (see Figure 17). There have been ways of trying to fit curves such as Figure 17 to a simple potential. The Lennard-Jones potential works, but a much better fit is the Morse potential (*Morse, 1929*):

$$U^{b,M} = D\left[1 - e^{-\alpha(r-r^b)}\right]^2$$

(100) The Morse potential

D is the potential energy at infinite bond length, and thus traditionally seen as the bond dissociation energy. Various sources agree that the C–C bond dissociation energy in alkanes is about 88 kcal/mol. (*Hageman, de Wijs, de Groot, & Meier, 2000*) has 4 eV = 92 kcal/mol. Note that we will not exactly use the Morse potential in our simulations, we will instead use a tabulated approximation (section 2.10), in order to break bonds (section 2.12).

Note that the repulsion for short bond lengths is an exponential term; compare the discussion in section 2.3.

The harmonic potential (section 2.4) can be related to the Morse potential by a Taylor expansion. The second-order expansion around $r = r^b$ is:

$$U(r^b) + \frac{(r-r^b)}{1!}U'(r^b) + \frac{(r-r^b)^2}{2!}U''(r^b)$$

$$= 0 + \frac{r-r^b}{1}\cdot 0 + \frac{(r-r^b)^2}{2}\cdot 2\alpha^2 D$$

(101) Second-order Taylor expansion for the Morse potential

$$= \alpha^2 D(r-r^b)^2$$

U is here a shorthand for $U^{b,M}$. We have now arrived to a potential, reminiscent of the harmonic bond length potential. Comparing terms from (98) and (101), and solving for α:

$$\alpha = \sqrt{\frac{k^b}{2D}} = \sqrt{\frac{K}{D}}$$

(102) Relating the harmonic and Morse potentials

Setting for example $K = 350$ kcal/mol/Å^2 and $D = 88.0$ kcal/mol, gives $\alpha = 1.99$ Å^{-1}. It is possible that the author of this force field started with a known α = 2 Å^{-1}, and constructed the harmonic bond length potential from there. In Figure 32 below, both potentials with the variables above are plotted in the same figure.

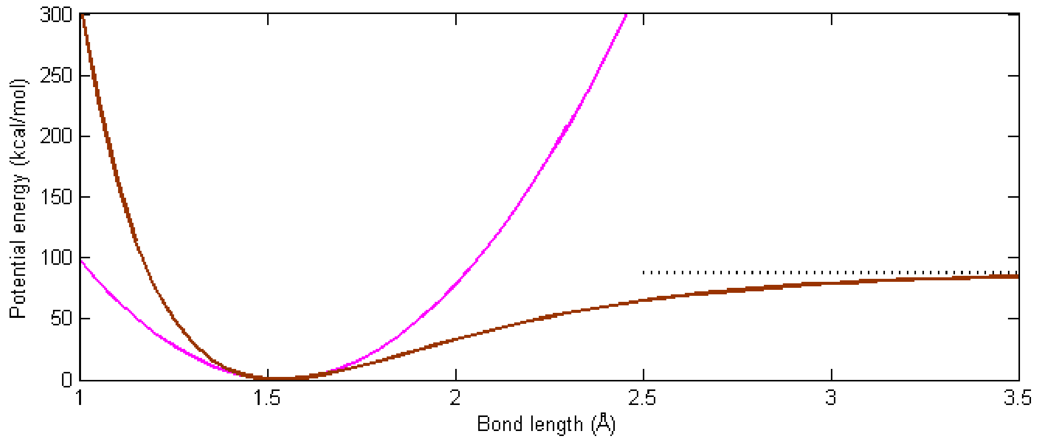


Figure 32: The harmonic bond length potential and the Morse potential with D = 88 kcal/mol plotted together.

As mentioned above, the harmonic potential gives a linear elastic behavior of the chain. In comparison, the Morse potential would give a non-linear elastic behavior. Let us now try to deform a bond between two atoms mentally. If a bond is stretched by a force P to a new length r_{deform}, and this force is balanced by the force emerging from the force field, then sum of forces on that bond gives:

$$P - \frac{d}{dr}U^b(r)\bigg|_{r=r_{deform}} = 0$$

$$P = \frac{d}{dr}U^b(r)\bigg|_{r=r_{deform}}$$

(103) A bond in mechanical equilibrium

Figure 33 shows the differentiated curves of both the harmonic and Morse potential (given by equations (99) and (114) respectively). They can essentially be seen as load-displacement curves for a single bond. Calculating a similar load-displacement curve for an entire molecule is not easy, since the displacement contains angle opening as well (see section 2.11 below).

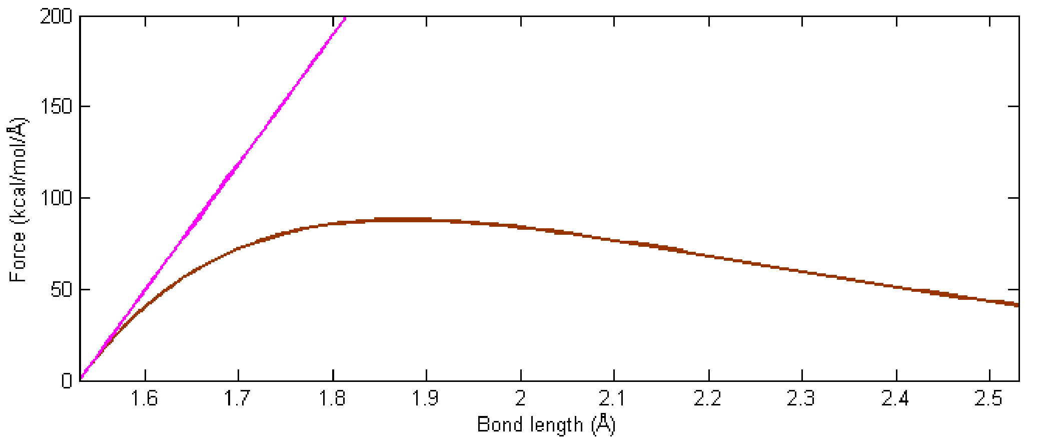


Figure 33: Differentiating the curves in Figure 32 gives the magnitude of the force that pushes the molecules back to the reference configuration. The linear plot is the harmonic potential.

There are very few force fields that use the Morse potential, for the simple reason that one needs three parameters to specify the potential as opposed to two for a harmonic potential. Many simulations done on organic molecules assume that a bond will only deform minutely, so that the short-range harmonic potential is sufficient to model this interaction. Some force fields go even further by constraining the bond length, as described in section 2.4. As our goal is to describe bond breaking by stretching bonds, such models will not do. Some force fields designed for non-equilibrium bond lengths use a quartic expansion of the Morse potential (*Allinger, Yuh, & Lii, 1989*).[12]

Looking for parameters motivated by the Morse potential through article searches at Web of Science and similar databases has proven difficult, but two parameter sets are presented in Table 9. Both provide parameters close to what we have, but with a dissociation energy of 83 kcal/mol.

Table 8: Parameters for the bonded potential (Morse) from formula (102)

Symbol	**Name**	**Value**
α	Shape parameter	2.0 Å^{-1}
D	Bond dissociation energy	88.0 kcal/mol
r^b	Reference bond length	1.53 Å

Table 9: Examples of Morse parameters used by others

Symbol	**(*Nyden & Noid, 1991*)**	**(*Weaver & Madix, 1999*)**
α	1.924 Å^{-1}	2.0 Å^{-1}
D	83.08 kcal/mol	347 kJ/mol ≈ 82.9 kcal/mol
r^b	1.529 Å	1.7 Å

[12] A cubic expansion would be a bad idea, as this potential goes to negative infinity for large bond lengths, causing such molecules to fly apart!

2.6 Angle potential, cosine/squared

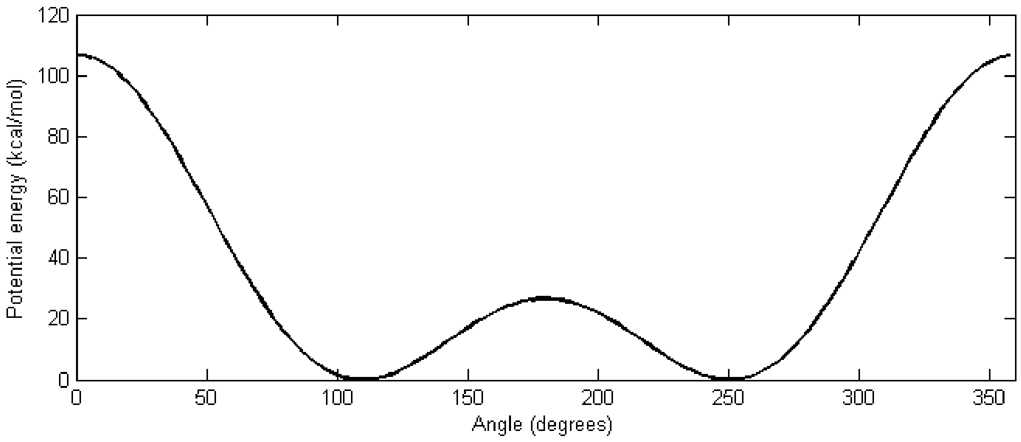


Figure 34: The angle potential with parameters from Table 10.

The angle potential is introduced to keep the bonds at proper angles compared to each other. These angles arise from quantum mechanics, namely the electron configuration of the atom. While quantum chemistry supplies the whole picture, interested newcomers may find more use in VSEPR theory, outlined in many general chemistry textbooks (like (*Zumdahl, 2009*)).

$$U^a(\theta) = K(\cos(\theta) - \cos(\theta^0))^2$$ *(104) The cosine/squared potential*

The potential used is similar to the harmonic bond potential, but uses cosines as a basis. The inspiration for this potential comes from (*Paul et al., 1995*). The force given by the derivative of this potential is given as equation (115). One of several problems with this potential is that there is another stable angle at $2\pi - 2\theta^0$, see Figure 34, but as shown in section 2.10, an axial force cannot deform the bond to larger than $\theta_p = \pi$.

In (*Sahputra & Echtermeyer, 2013*), the angle potential is given as a harmonic potential, like (98) with K = 60.0 kcal/mol/deg^2, which corresponds to $2.0 \cdot 10^5$ kcal/mol/rad^2. This is a very stiff potential, and one requires a lot of force to deform the angle away from the reference position, making it unsuitable for our purposes.

Table 10: Parameters for the angle potential

Symbol	**Name**	**Value**
K	Angle stiffness parameter	60 kcal/mol
θ^0	Reference angle	109.5°, $\cos(\theta^0) = -0.334$

2.7 Dihedral potential, OPLS

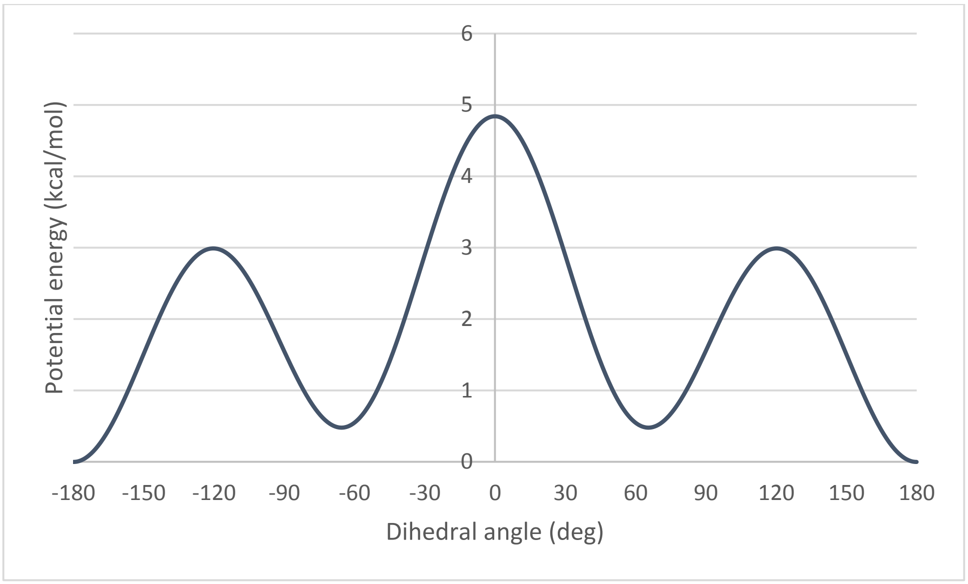


Figure 35: The OPLS dihedral potential with parameters from Table 11. The graph is centered at the cis *conformation, with the* trans *conformation at each end.* Gauche *is at ± 60°.*

Dihedral geometry has been explained back in Part B, section 2.4. The (proper) dihedral potential used in these simulations is plotted above in Figure 35. A good dihedral potential should have local minima at *gauche* and *trans*,[13] and be fitted to incorporate known *gauche-trans* and *cis* energy barriers.

The form of the potential that we will be using, is from a force field family called OPLS (Optimized Potentials for Liquid Simulations) (Jorgensen et al., 1996)

$$U^d(\varphi) = \sum_{n=1}^{4} \frac{1}{2} K_n [1 + (-1)^{n+1} \cos(n\varphi)]$$

(105) The OPLS potential as defined by LAMMPS

This potential with parameters is modified from (Paul et al., 1995), which uses a potential where the *trans*-conformations are defined at a 0° dihedral angle (see warning in footnote 6, page 12), which means that it cannot be input directly into LAMMPS. It is given as equation (106) and illustrated in Figure 36 as an orange dotted line.

$$U^{d,trans=0}(\varphi) = \sum_{n=1}^{4} \frac{1}{2} K'_n [1 - \cos(n\varphi)]$$

(106) The potential as given by Paul et al.

To modify this potential to a trans-180°-potential is quite simple: Use the same parameters from (106) and put them into (105) (viz. $K'_i = K_i$), and the potential turns into the one shown in Figure 35, or the black solid line in Figure 36. We have not looked closely at the mathematics behind this transformation, but it works out for our case!

[13] Gauche may occasionally be more stable than trans. See figure 4.8 in (Leach, 2001a), for the O-C-C-O dihedral in deoxyribose.

Table 11: Parameters for the dihedral potential (OPLS)

Symbol	Value
K_1	1.6 kcal/mol
K_2	-0.867 kcal/mol
K_3	3.24 kcal/mol
K_4	0.0 kcal/mol

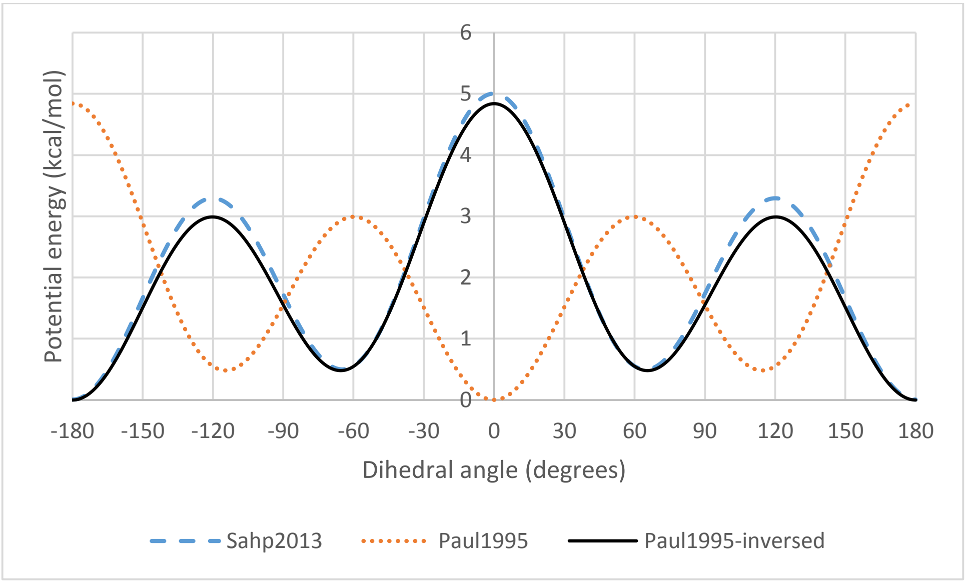


Figure 36: Comparisons of two dihedral terms given in literature (dotted and dashed lines) and the one we will be using in our simulations (solid line).

Our main source for potentials, (*Sahputra & Echtermeyer, 2013*) uses another definition, with parameters given by Table 12. This is called the `multi/harmonic` potential, and is illustrated as the blue dashed line in Figure 36.

$$U^d(\varphi) = \sum_{n=1}^{5} A_n \cos^{n-1}(\varphi)$$

(107) The multi/harmonic dihedral energy as defined by LAMMPS.

The parameters for this potential appears to have been fitted from (*Paul et al., 1995*), in a way to fit the form (107) where the function minimum is at *trans*. As Figure 36 shows, this form is not a bad fit, but it has a subtle drawback. When LAMMPS produces a data file (see section 4 below), it will include the parameters for many force field terms, except `multi/harmonic` and possibly others. This may be corrected in a later build, but for now, if one has to carry a data file from a simulation run (like energy minimization) to another (like a tensile test), one would have to copy and paste the relevant parameters from the input data file to the output file. This is not a problem with the OPLS `dihedral_style`.

The initial thermodynamic simulations were run with the `multi/harmonic` potential.

Table 12: Parameters for the variant dihedral potential (multi/harmonic)

Symbol	Value
A_1	1.73 kcal/mol
A_2	-4.49 kcal/mol
A_3	0.776 kcal/mol
A_4	6.99 kcal/mol
A_5	0.0 kcal/mol

2.8 Optional: Improper potential

There do exist a second type of dihedrals, called improper dihedrals or simply impropers. They are introduced to keep cyclic molecules (like benzene) planar. Consider, like (Leach, 2001a) does, the molecule cyclobutanone C_4H_6O.

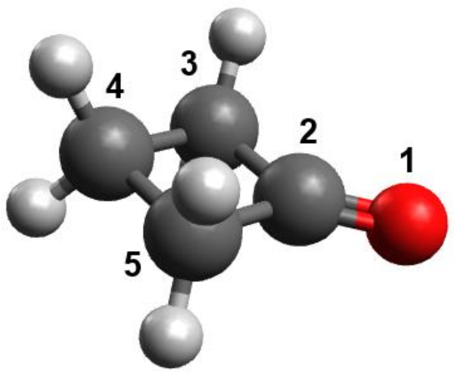


Figure 37: Cyclobutanone, with numbered atoms

The improper dihedral that keeps the oxygen atom in plane with the rest of the molecule is 1-3-5-2. That is to say, the angle between planes 1-3-5 and 3-5-2. Note that the atoms do not need to be bonded in the order 3-5-2 to compute the improper dihedral. In other words, the improper dihedral angle χ_{ijkl} is a measure for how out-of-plane the atom *i* is, compared to the plane *jkl*. A typical improper potential would look like this: $U^{id}(\chi) = K[1 - 2\cos(2\chi)]$, which has a minimum when χ = 0, or *i* is in the same plan as *jkl*.

2.9 Optional: Cross-terms

Consider this problem, illustrated in Figure 38: A propane molecule where the carbon atoms are numbered 1-2-3 is subjected to some external force that reduces the central angle. The atoms 1 and 3 would repulse each other, as explained in section 2.3. A UA model does not calculate the Lennard-Jones repulsion between the atoms, and the only force keeping them apart would be the angle potential.

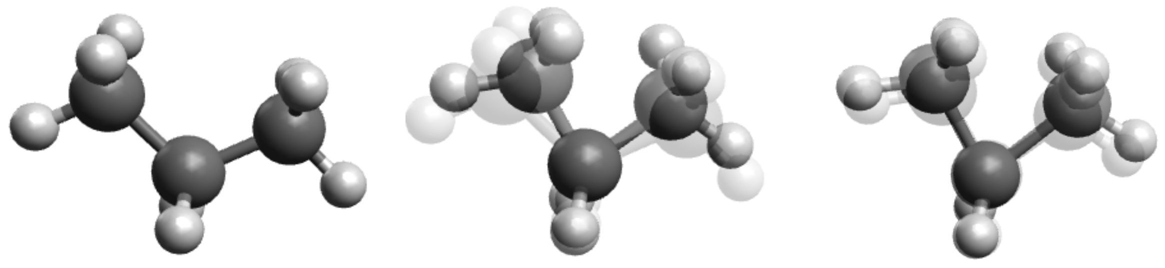


Figure 38: Illustrating the bond stretch/angle cross-term
(l) A propane molecule in equilibrium.
(m) The same molecule, where the C-C-C bond angle has been decreased.
(r) The molecule responds to the decreased angle by increasing C-C bond lengths to push apart the atoms.
The semitransparent configuration in (m) and (r) are the figures (l) and (m) respectively.

Figure 38 shows an example of interplay between angle and bond lengths. When the angle decreases, the bond length increases to push apart 1-3 atoms.

While LAMMPS does not support cross-terms by adding commands to most potentials, a few of them come packaged with such cross-terms. The CHARMM angle potential, for example is written as $U^a = K(\theta - \theta^0)^2 + K_{UB}(r_{13} - r_{13}^0)^2$, where the last term is the Urey-Bradley term which maintains a fixed distance between atoms 1 and 3 in an angle. Another potential with implicit cross-terms is the Compass Class2 family (*Sun, 1998*), which requires a special package to work.

2.10 Potential energy tables

If all else fails and one cannot find a potential form that exactly matches one of the forms that LAMMPS is prepackaged with, one could use a tabulated potential instead *(Wolff & Rudd, 1999*). A table is an external file that can be read by LAMMPS to compute the potential energy between atoms.

The file is formatted so that each line couples a parameter (like atom separation) to a potential energy and a force. For a non-bonded pair potential for example, each line looks like this:

```
index  r     potential energy  force
120    2.92  47.5              21.1
```

The upper line is the general format, and the lower line is an example. The table will also contain a header with the name of the table, so that a file may contain many tables. Interpolation between two points is usually with a cubic spline for both the energy and force values. Similar tables can of course be created for the bond length, angle and dihedral potentials as well. A script that can be used to produce a special table (bond length, with special properties) is given in appendix 4.3.

2.11 2D Chain deformation

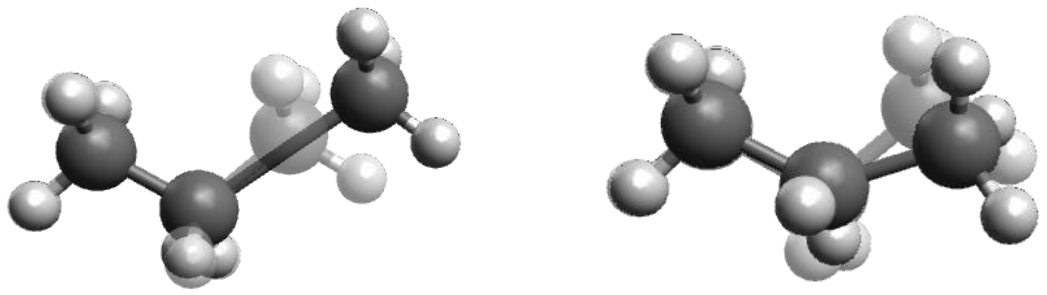


Figure 39: The two ways of increasing the distance between 1-3 atoms. (l): Stretching the bond, (r): Increasing the bond angle. The molecule with ordinary bond lengths and angle is semitransparent and in the background.

We can now apply the force fields above to the mechanics of deformation. Consider a linear polyethylene chain (like in Figure 1) with N carbon atoms, subjected to an axial force P. We can divide the chain into subunits consisting of two atoms and a bond (Figure 40) and do a force analysis on this unit. If we assume that the deformation is spread evenly over all subunits (in fact, the outer bonds are deformed most), then the rest of the chain follows by symmetry. From the figure below, we can derive equation (108), and some related terms.

$$l = r \cdot \cos\left(\frac{\theta}{2}\right)$$ (108) Length of a polymer subunit

$$L = (N-1) \cdot l = (N-1) \cdot r \cdot \cos\left(\frac{\theta}{2}\right)$$ (109) Length of a chain with N atoms

$$L_0 = (N-1) \cdot r^b \cdot cos\left(\frac{\theta^0}{2}\right)$$ (110) Reference chain length

$$\varepsilon = \frac{L - L_0}{L_0} = \frac{r}{r^b}\sqrt{\frac{1+\cos(\theta)}{1+\cos(\theta^0)}} - 1$$ (111) Nominal strain

Here, the reference chain length is when no force is imposed on the chain, and the chain length and angle is at their reference levels.

If we exclude the energy terms from pair potentials and dihedrals, then the deformation of the molecule can be described by the bond length and angle potentials. A bond length is deformed by stretching, and an angle is deformed by opening, see Figure 39. If we assume that the subunit is in mechanical equilibrium, then the pulling force is counteracted by the potential energy field, so that $\frac{dU}{dr} = \boldsymbol{P}$, roughly, compare equation (103).

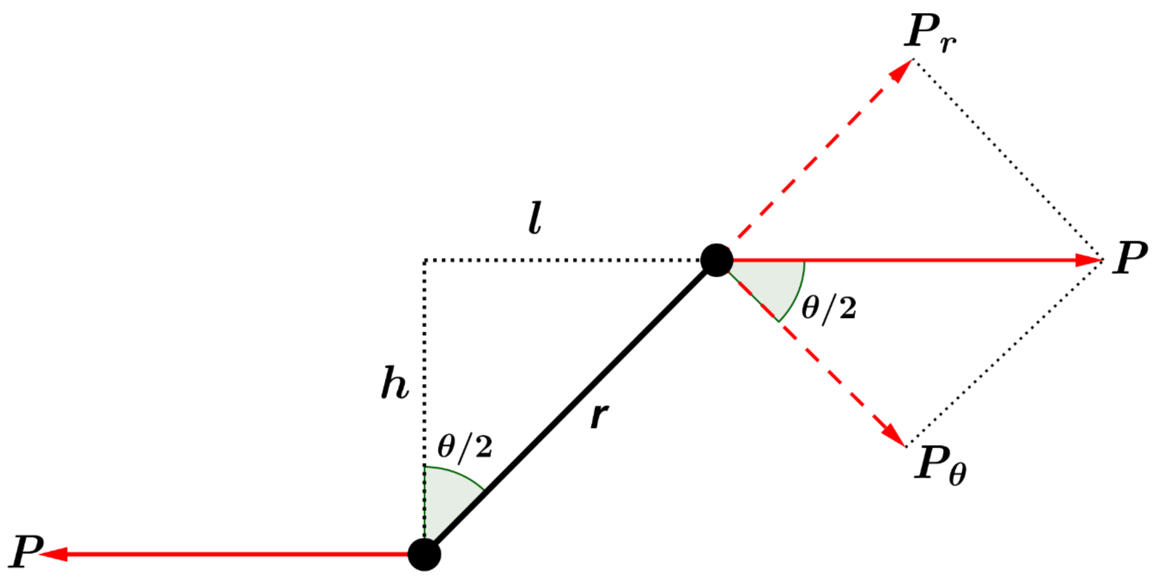


Figure 40: Decomposition of a force on a single bond. The rest of the molecule follows by symmetry.

Here we have also decomposed the force vector **P** acting on an atom into a force $\mathbf{P}_r$ along the bond, and its orthogonal component $\mathbf{P}_\theta$. Since $\mathbf{P}_\theta$ acts orthogonal to the bond, it can in no way stretch the bond, and its sole contribution to the subunit deformation is angle opening, likewise $\mathbf{P}_r$ will only stretch the bond. If we assume mechanical equilibrium, then the relations between the force **P** and the deformed state is:

$$P_r = \boxed{P\sin\left(\frac{\theta_p}{2}\right)} = P_{int}^b(r_p) = \boxed{\left.\frac{dU^b(r)}{dr}\right|_{r=r_p}}$$ (112) *Bond stretch deformation*

$$P_\theta = \boxed{P\cos\left(\frac{\theta_p}{2}\right)} = P_{int}^a(\theta_p) = \boxed{\left.\frac{dU^a(\theta)}{d\theta}\right|_{\theta=\theta_p}}$$ (113) *Angle opening deformation*

The framed equations are the left and right side of the equality that we need to solve. Here r_p and θ_p are respectively the bond length and angle of the deformed structure, and P_{int} is the internal force, or the system's response to external deformations. These equations need to solved for r_p and θ_p and these quantities are inserted into (109) to get the complete chain length. We need to solve (113) first, to find the angle necessary for force decomposition.

For the harmonic potentials (99), we can always solve (112) and (113) for a given P. The force given by the Morse potential, see equation (114) and illustrated in Figure 33, has a maximum at $r_{max} = r^b + \frac{\ln(2)}{\alpha}$ where $P_{int}^{b,Morse}(r_{max}) = \frac{1}{2}\alpha D$. Any P_r higher than that will give a system that is not in static equilibrium, with some time-dependent bond stretching, as **P** continues to pull the chain apart.

$$P_{int}^{b,Morse} = 2\alpha D e^{-\alpha(r-r^b)}\left[1 - e^{-\alpha(r-r^b)}\right]$$ (114) *Internal force from the Morse potential*

The force from the angle potential is:

$$P^a_{int} = 2K\sin(\theta)\,[\cos(\theta^0) - \cos(\theta)]$$ *(115) Internal force from the angle potential*

Using harmonic potentials for both bond stretching and angle opening, one can derive the following graph, by first solving (113) numerically for θ_p, and then plug that angle into (112). Script is included in Appendix 4.6.

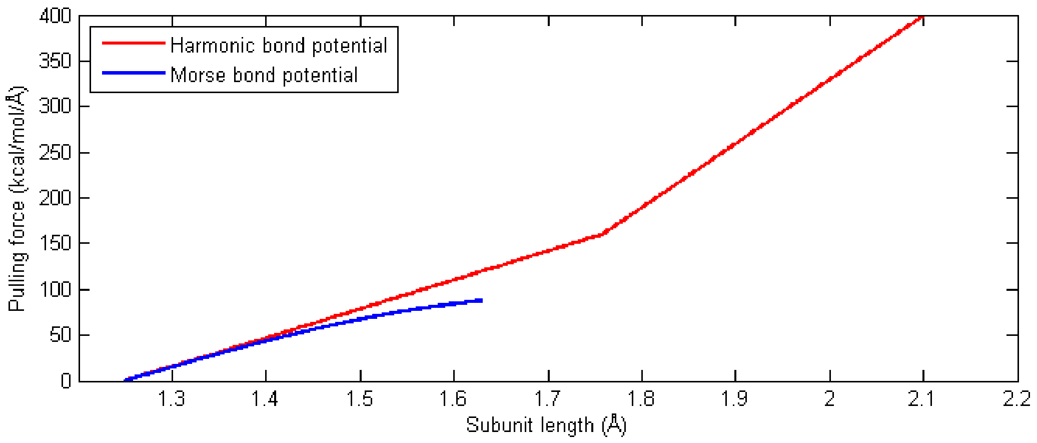


Figure 41: Theoretical load-displacement curve for a chain subunit with different bond length potentials, but same angle potential. The Morse potential plot is only taken as far as static equilibrium can be applied. Note that the graph plots l versus P, not r versus P.

Compare Figure 41 and Figure 33. The change in stiffness at $l_p \approx 1.73$ Å is caused by a change from a mixed bond length and angle deformation to a pure bond length deformation. As θ gets higher, P_θ gets smaller, and the angle will eventually reach a limit, causing it to stop deforming. At this point, one can read the bond length off Figure 41, since $l = r$ when the angle is completely opened.

The curve for the Morse potential stops where mechanical equilibrium ends, as the numerical solver cannot solve (112) for large P.

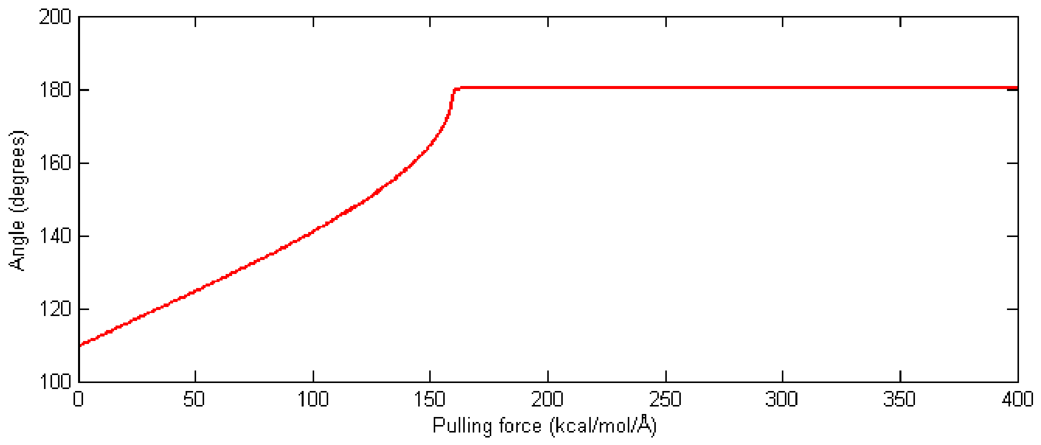


Figure 42: Deforming the bond angle as a function of the axial pulling force. Note that the x-axis in this figure is the y-axis of Figure 41.

The curve in Figure 42 is the solution of the equation:

$$2K sin(\theta)[\cos(\theta) - \cos(\theta^0)] - P\cos\left(\frac{\theta}{2}\right) = 0,$$

where K and θ^0 are given in Table 10.

2.12 Breaking bonds

We are now at a point where we can worry about how to break bonds.

2.12.1 1D Activation energy

By looking from a quantum chemical point of view, breaking a bond is simple enough. Simply excite the enough electrons in the bonded orbitals, so that more electrons are antibonding than bonding (see Figure 16). This requires some activation energy, usually the bond dissociation energy D, when compared to the unstressed length r^b. By increasing the separation between atoms, the electrons in a sigma bond will in average be longer and longer separated from the nuclei, giving an increased energy for the system (Figure 17). This is the nature behind a good bond length potential.

Researchers on 1-dimensional chains of polymers (*Bolton, Nordholm, & Schranz, 1995; Ghosh, Dimitrov, Rostiashvili, Milchev, & Vilgis, 2010; Puthur & Sebastian, 2002; Stember & Ezra, 2007*) uses the following bond length potential or a variant of it:

$$U_{tot}^{b}(r,P) = \sum_{n=1}^{N} U^{b,Morse}(r_n - r_{n-1}) - Pr_N \qquad \text{(116) Modified Morse potential}$$

Here *n* counts atoms along the chain with *N* atoms and r_n is the position of that atom. This chain is subjected to a force P on the Nth atom. This modified potential can be extended to any bond length potential, getting similar results as below. For positive P, the modified potential will result in an emergent bond energy, given as ΔE_b in Figure 43. We will from this point on consider a chain of two atoms, so that (116) will refer to a single bond.

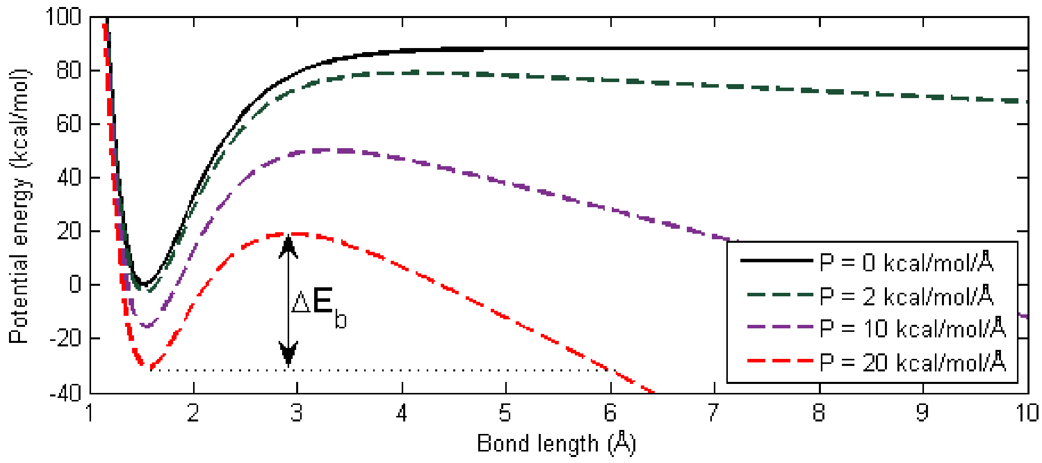


Figure 43: The modified one-bond Morse potential for some values of P, with a marked-off activation energy.

Figure 43 is a plot of (116) over a single bond. Note how the equilibrium distance increases with the force. This is why we denote r^b the reference bond length.

The activation energy is given by:

$$\Delta E_b = D\left[\sqrt{1-\tilde{P}} + \frac{\tilde{P}}{2}\ln\left(\frac{1-\sqrt{1-\tilde{P}}}{1+\sqrt{1-\tilde{P}}}\right)\right] \qquad \text{(117) Activation energy}$$

Here, $\tilde{P}$ is the dimensionless force (*Ghosh et al., 2010; Puthur & Sebastian, 2002*):

$$\tilde{P} = \frac{2}{\alpha D}P \qquad \text{(118) Dimensionless force}$$

The energy barrier disappears when $\tilde{P} = 1$, that is to say $P = \frac{\alpha D}{2} \approx 88\ \mathrm{kcal/mol/Å}$, which we have earlier established as the maximal force where the Morse bond can be in equilibrium (section 2.11).

The two stationary points on this potential are designated r^{eq} (local minimum) and r^{tr} (local maximum). r^{eq} is the equilibrium bond length, and r^{tr} is the bond length at the transition state. As the force increases, r^{eq} increases and r^{tr} diminishes as well.

$$r^{eq} = r^b + \frac{1}{\alpha}\ln\left(\frac{2}{1+\sqrt{1-\tilde{P}}}\right) \qquad \text{(119) Equilibrium bond length}$$

$$r^{tr} = r^b + \frac{1}{\alpha}\ln\left(\frac{2}{1-\sqrt{1-\tilde{P}}}\right) \qquad \text{(120) Transition bond length}$$

These two points join at a saddle point when the dimensionless force reaches 1, and there is no energy barrier to conquer. The local maximum of the force from the Morse potential equation (114) and Figure 33 lies on this point.

$$r^{bondbreak} = r^b + \frac{\ln(2)}{\alpha} \approx 1.88\ \text{Å} \qquad \text{(121) Theoretical 1D bond breaking length}$$

At this point, everything is touch and go. If the bond is slightly extended, the potential energy will irreversibly drive the two atoms from each other, and the bond is considered broken.

The moral of this story is that we can model the bond as broken when enough energy is transferred to the bond, or it reaches a certain length. We will denote this length by r_{cutoff}. We will now look at various methods to implement r_{cutoff} in our model.

2.12.2 First attempt: A special fix

LAMMPS has implemented a fix (see section 3.1) `bond/break` designed to break bonds. It works by deleting bonds that are longer than a certain length r_{cutoff}, with an optional probability of success. Interactions like angles and dihedrals over these bonds are thought of as independent of the bond topology, and are kept in this system even if the bond is deleted. This means that these potentials will still be calculated, giving rise to a possible long-term error in the further evolution of the system. Think of this problem as what happens when a molecule is broken in two. The halves cannot move independently of each other as one would expect, since the angle potentials constrain such movements.

To first break bonds, the simulation needs to have been set up with the `special_bonds` command with arguments `lj/coul 0 1 1`. This command turns on or off the calculation of Lennard-Jones forces on bonded atoms; they are by default off. `0 1 1` means that the calculation is not done for 1-2 atoms, but it is done for 1-3 and 1-4 atoms. We need to invoke the `special_bonds` command due to a restriction in the LAMMPS code. It is possible that this restriction will be lifted in later versions.

Calculating the Lennard-Jones force over 1-3 atoms is a problem, since these atoms have a distance of $2l_0 \approx 1.77$ Å to each other. Calculating the LJ potential for this distance gives $U^{nb} \approx 8700$ kcal/mol, and that is for a single 1-3 relation! This potential makes the atoms fly apart during molecular dynamics making `special_bonds` unsuitable for our simulation, and by extension `bond/break` as well. A possible way to circumvent this is to use the `neigh_modify` command to exclude nonbonded interactions between atoms in molecules, an exclusion that will be maintained even when the bond is broken.

2.12.3 Second attempt: A tabulated potential

The other method is to explicitly write the bond breaking in the bond length potential. As of this writing, there is only one built-in potential for LAMMPS that does this, the `harmonic/shift/cut` potential. We will instead use tables to build a bond that breaks at some point.

Figure 44 illustrates this concept. Here, the bond potential is given by (100) up to the cutoff distance. The bond is not topologically broken, the bond energy is still calculated, but the potential energy is 0, so that interatomic separation is taken as two completely uninteracting. This model is employed by studies on irreversible bond breaking in polymers (*Doerr & Taylor, 1994; Oliveira & Taylor, 1994*).

$$U^b(r) = \begin{cases} U^{b,Morse}(r), r < r_{cutoff} \\ 0, \qquad\qquad r \geq r_{cutoff} \end{cases}$$ *(122) Bond breaking potential of the first type*

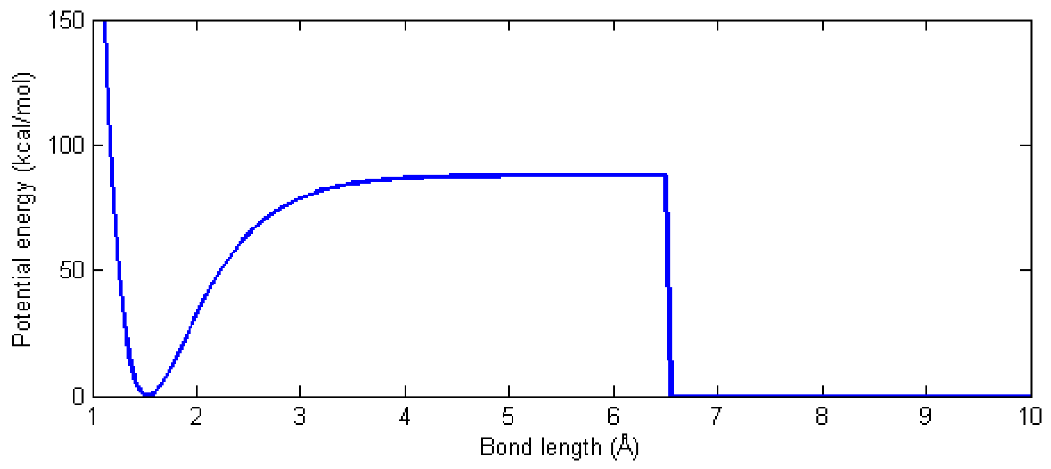


Figure 44: A simple bond break table with a Morse potential and a cutoff distance.

With this model, we can avoid implementing the `special_bonds` command, which gave us trouble before.

The next step in refining this model is if we suppose that the simulation is done in a low-pH environment, so that a hydrogen atom immediately bonds to the free radical carbon atoms on each end of the broken bond. This means that we may use the nonbonded potential for "bond lengths" larger than cutoff. This potential looks like this:

$$U^b(r) = \begin{cases} U^{b,Morse}(r), & r < r_{cutoff} \\ U^{nb}(r), & r \geq r_{cutoff} \end{cases}$$ *(123) Bond breaking potential of the second type*

This is the actual bond length potential that we will implement in our simulations. The only question now is what r_{cutoff} is. We will need to do a parameter search and compare simulation output with real data, but we can narrow the search down somewhat.

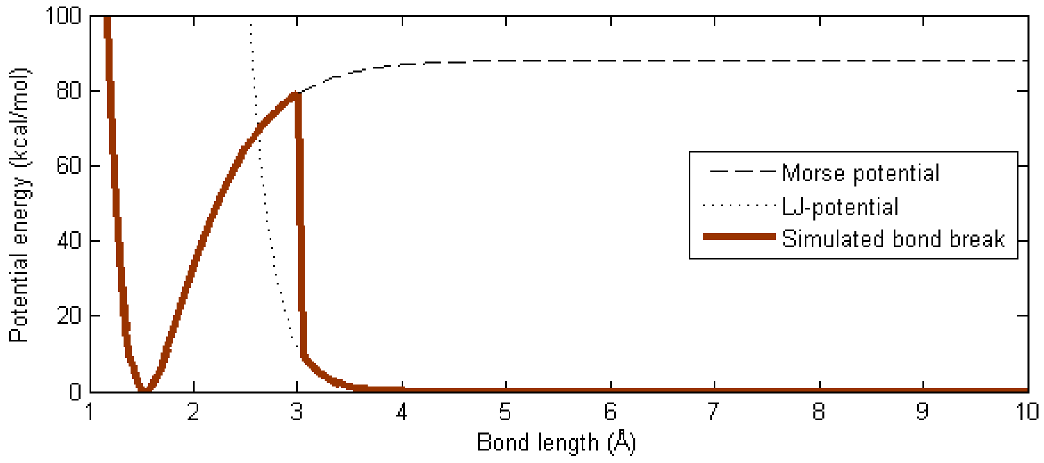


Figure 45: A Morse potential and a Lennard-Jones potential plotted together, with an overlayed simulated bond with cutoff length 3.05 Å. Bond break happens when the table jumps from the Morse potential to the LJ-potential.

The point where the Morse and Lennard-Jones potentials cross each other is denoted (r^*_{cutoff}, U^*). Here $r^*_{cutoff} \approx 2.6174$ is the smallest Morse cutoff distance that will not increase the energy of the system when the bond breaks. This means that we can concentrate our search for a cutoff distance on the interval $[r^*_{cutoff}, r_c]$, where the upper limit is the non-bonded cutoff distance, given by Table 6 as $r_c = 10.0$ Å.

Since we know from experiments that the bond dissociation energy for C-C bonds is about D = 88 kcal/mol, ergo $D = U^{b,morse}(r^D_{cutoff}) - U^{nb}(r^D_{cutoff}) = 88$ kcal/mol, this would give us a cutoff of about $r^D_{cutoff} = 5.5$ Å or larger. If the parameter search shows that the cutoff distance that best fits experiments gives smaller bond dissociation energy, then the next generation of Morse parameters (equation (100)) should include a simulated large-r energy $D' > D$, which gives a bond-breaking energy of 88 kcal/mol where the bond breaks.

$$U^{b,Morse}(r) = D'\left[1 - exp\left(-\alpha(r - r^b)\right)\right]^2, \quad r < r_{cutoff}$$ *(124) The Morse potential with a new parameter D'*

$$U^{b,Morse}(r_{cutoff}) - U^{nb}(r_{cutoff}) = D$$ *(125) The new meaning of the old D*

A third type of bond breaking would include a healing pathway. In other way the potential for pushing together a broken bond would be different from the potential that breaks it. This model could be used when there is no hydrogen to stop the bond healing. It is schematically given in Figure 46. Getting LAMMPS to interpret this would be a nightmare, as we would need to identify the atoms between a healing bond as a new atom type, and input new potentials for this type.

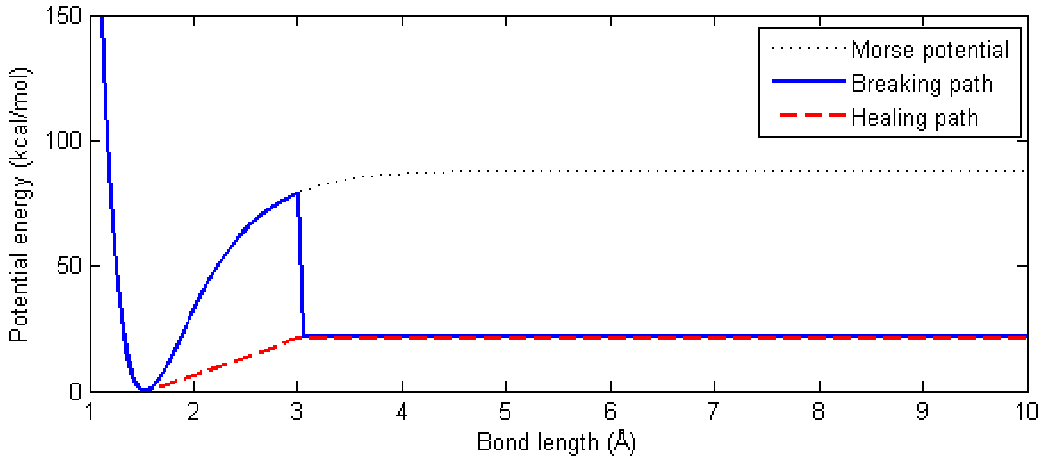


Figure 46: A way to represent the healing of a bond. The breaking path follows the Morse potential up to the breaking point, after which the potential has a nonzero energy (not necessarily flat as this figure shows, it should go to zero as the bond length approaches 10 Å). If this bond is pushed back, it will follow the healing point, here represented as a linear interpolation between the broken bond energy and the zero point.

2.13 Optional: EAM

Far away from the realm of molecules, in the science of modeling solids, such as salts, metals and semiconductors, will the simple Lennard-Jones pair potential not give very good results, no matter how good one tries to fit parameters. For metals, a good model rooted in a different form of pair potentials is the Finnis-Sinclair Embedded Atom Method (EAM) *(Finnis & Sinclair, 1984)*. It is written like this:

$$U_i = F_i(\rho_{ij}) + \frac{1}{2}\sum_{i\neq j} \varphi_{ij}(\boldsymbol{r}_i, \boldsymbol{r}_j) \qquad \text{(126) Finnis-Sinclair EAM pair potential}$$

Here, φ_{ij} is an ordinary pair potential and F(ρ) is a function of the electron density ρ(**r**), effectively the energy given by embedding an atom *i* into a sea of electrons ρ. This electron density is computed by the positions of the surrounding atoms, and does not require any more computation time than any other non-bonded potential. The ways of computing the pair potential, ρ and F(ρ) is usually tabulated in DYNAMO files, and interpolated by cubic splines. Some of these files follows the LAMMPS package, and others can be downloaded from the internet: http://www.ctcms.nist.gov/potentials/.

2.14 Recapitulation

With the length of this chapter, it would be best to recapitulate exactly what we will do in our simulations, in case some of the text above were unclear. We will run a united-atom simulation with the following potential energy field (equation (92)):

$$\sum_{j,k,l\neq i} U_{ijkl} = \sum_j U_{ij}^{nb} + \sum_j U_{ij}^{b} + \sum_{j,k} U_{ijk}^{a} + \sum_{j,k,l} U_{ijkl}^{d}$$

Here, each term are in turn and order:

1. U_{ijkl} the total potential energy for an atom i
2. U_{ij}^{nb}, the non-bonded potential energy between atom i and all atoms j not bonded to i. Equation (94) and Table 6.
3. U_{ij}^{b}, the bonded potential energy between atom i and all atoms j bonded to i. For the thermodynamic simulations, use (98) with Table 7, and for the mechanical simulations, use a tabulation of (123) with Table 6 and Table 8 and a given r_{cutoff}, not yet known.
4. U_{ijk}^{a}, the angle potential between i, and atoms j and k that form an angle at i. Given by (104) and Table 10.
5. U_{ijkl}^{d}, the dihedral potential on i. Given by the OPLS potential (105) with Table 11.

We do not still know the value of r_{cutoff}, the parameter search for it is our first priority, and it is shown in Part D, section **Feil! Fant ikke referansekilden.**.

3 Working with LAMMPS

LAMMPS is text-based, which means that it accepts input either as typed in, or by scripts. In this section, we will present how scripts work, and what many of the commands mean. For those who are not familiar with LAMMPS, I recommend reading this section and read some of the example files packaged with LAMMPS (in.peptide, in.micelle and so on) armed with the digital LAMMPS documentation that can be found at ("LAMMPS WWW site," 2013), before going over the scripts that follow.

3.1 Scripts

LAMMPS input scripts are usually named as in.*, where * is the name of the file. They are divided into commands, variables, fixes and computes. Variables are formulae for generating variable that can be used by fixes and computes. Fixes are in short a collection of rules for how the simulation will behave. Computes calculates properties of either the entirety of the system or a subset of it. Commands are everything else.

A typical script may begin like this:

```
dimension 3
boundary  p p p
units          real
atom_style     molecular
timestep  1
thermo    1000
```

These lines are all commands and mean in order that our simulation will be 3-dimensional. All boundary conditions will be periodic (see next section). Outputted units will be built on the `real` unit set (for example distances are measured in Å, energy in kcal/mol, time in fs). The atoms are expected to behave like molecules (with bonds, angles and dihedrals) as opposed to metals for example. The length of the timestep is 1 time unit (in this case fs), and properties will be output every 1000 timesteps.

The next step is to set up the initial configuration of atoms, which is detailed in section 4, and then define the parameters for the potential energy field. It may look like this:

```
pair_style     lj/cut 10.0
bond_style     table spline 1000
angle_style    cosine/squared
dihedral_style opls

read_data      data.polymer
bond_coeff     1 morse.table BREAK
bond_coeff     2 morse.table ZERO
```

The first four lines set up the different models for the potential energy, and the last three contain the parameters for those. This is a good time mention the anatomy of a line in LAMMPS. The first word is a command, the second (in case of computes and fixes) are IDs of fixes, groups or regions, which can be used to work on parts of the simulation that we are interested in. After that comes the name of the command that we are invoking, followed by arguments.

The first line above is `command name, style name, argument`. It tells us that we will use a Lennard-Jones pairwise potential with a cutoff radius (section 3.3) of 10.0 length units in our simulation. All commands have a syntax that are detailed in LAMMPS' documentation.

The line that begins with `read_data` imports the data file `data.polymer`. This file contains the position of each atom, the definition of the simulation box that will contain this system, a definition of their bonds, angles and dihedrals, as well as the parameters for `pair_style`, `angle_style` and `dihedral_style`, so that we do not need to repeat them in the in-file. The `bond_coeff` lines reads one table each from the file `morse.table`, one for the bond type 1, which is named `BREAK`, and one for the bond type 2, named `ZERO`. The `BREAK` table is used to model the breaking of polymers, see section 2.12.3. The `ZERO` table is useful if we need some non-interacting atoms, see section 4.3.

From this point we will need to specify the rules for how the atoms will move (fixes), and what type of outputs we want (computes and variables). This is a large topic, and is best learnt by reading good input files. We will look at some lines of interest.

```
compute      poteng all pe/atom
thermo_style custom step temp etotal pe ke press vol enthalpy cpu cpuremain
dump         1 all custom 20000 dumps\dump.polymer type xs ys zs vx vy vz &
             fx fy fz c_poteng
log          logs\polymer.txt

write_data   dumps/data.polymer
```

The first line is a compute. It computes the per-atom potential energy to be stored in the dump file, these values can later be used to visualize various properties of the simulation (see Figure 66). This compute is stored as `c_poteng` for later use.

The second line decides what variables to output on screen and in the log file. In our case these variables are `step` (current timestep), `temp` (temperature given by (80)), `etotal` (total energy given as pe + ke), `pe` (total potential energy), `ke` (total kinetic energy), `pressure` (given by equation (86)), `vol` (volume of the simulation box), `enthalpy` (equation (79)), `cpu` (elapsed simulation time) and `cpuleft` (estimated simulation time to the end of the current run). These values are outputted at a frequency suggested by the `thermo` command (in our case it's every 1000 time steps).

The third line generates a dump file called `dump.polymer` in the folder `dumps`. It contains information about the atom type, position (`xs`, `ys`, `zs`), velocity (`vx`, `vy`, `vz`), per-atom force (`fx`, `fy`, `fz`) and per-atom potential energy (`c_poteng`). This dump file can be read by visualization programs like Ovito later (section 3.7).

The fourth line generates the log file `polymer.txt` in the folder `logs`. This file will contain the outputted thermodynamic values. Otherwise, the output would be saved as the file `log.lammps`.

The last line generates the data file `data.polymer` in the folder `dumps`. This data file contains information about the system at the moment LAMMPS read that line, and that file can be read by the `read_data` command in case one wants to restart the simulation.

```
fix        1 all nve
velocity   all create 298.15 3278410
```

These two lines are for moving atoms. The first one assign an integrator to the system (section 3.6), and the second line adds velocities to the atoms so that the kinetic energy of the system is consistent of a temperature of 298.15 K (section 3.5). The last argument is a random seed that LAMMPS needs in order to generate random numbers. Two equal in files with different random seeds may produce quite different phase space trajectories.

```
run        1000000
```

Finally, the `run` command starts the simulation, and tells it to run for 1,000,000 time steps, outputting dumps and thermodynamical data at the specified frequencies. We will now look closer at how LAMMPS works.

3.2 Boundary conditions

LAMMPS does simulations in a contained space that we call a simulation box. This box has usually the shape of a rectangular prism, with some lengths `lx`, `ly` and `lz`. When a particle reaches the edge of the box, LAMMPS will need a rule to decide what will happen to it. This is a boundary condition, and we can assign one to each edge. We will concern us with three types of boundary conditions here: `p`, `s` and `f`, as defined by LAMMPS.

The `p` boundary condition is periodic, and means that a particle that crosses the boundary will reappear on the other side. The remapping is simple: If a particle crosses over the high x boundary, then the new coordinate is $x_{wrapped} = x_{un-wrapped} - n \cdot l_x$. Here the symbols are respectively: The recorded coordinate of the particle, the actual coordinate of the particle, the number of times that particle has crossed the boundary (*n* starts as 0. Add one when the particle crosses a higher boundary and subtract one when the particle crosses a lower boundary) and the box length in the x dimension. We store the value of *n*, because it is often useful to know how far a particle has travelled away from its initial position. Let us do a numerical example. A one-dimensional box has periodic boundaries and length 5. At one point, a particle crosses the boundary and has a position 6. This means that the particle has reappeared on the other side of the box on position 1.

Another feature with the periodic boundary condition is that each particle has an image on the other side of the boundary that can interact with the particles inside the box, see Figure 47. Additionally, if a particle can interact with both a particle inside the box and an image outside the box, the interaction is done with the nearest choice. For example, in the figure below, the particle A can interact with the particle E or its image E'$_1$ on the left of the box. E'$_1$ is closer to A than E, so the interaction is calculated between these two particles. This rarely happens because the cut-off radius (section 3.3) is usually quite smaller than the size of the simulation box.

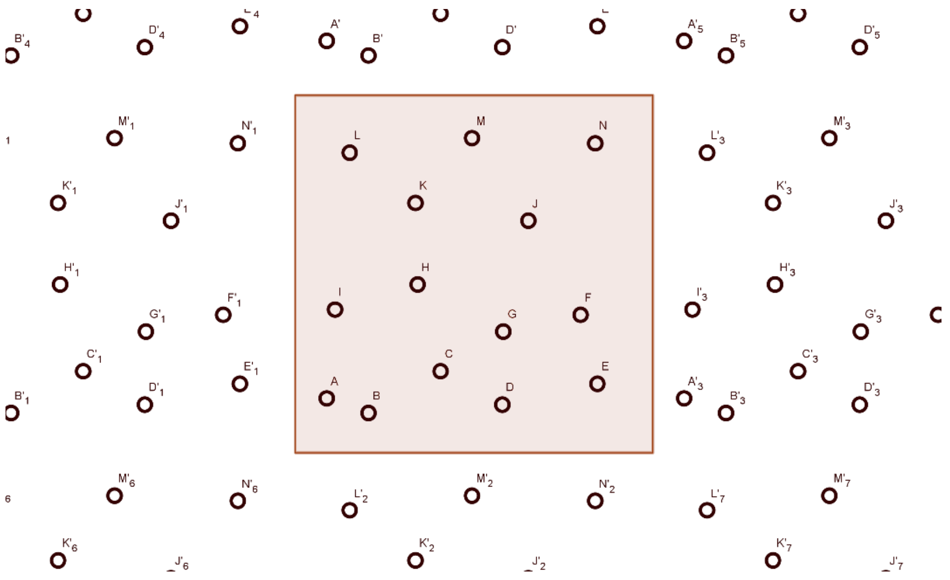


Figure 47: A simulation box (shaded) containing particles A-N surrounded by periodic images of the particles.

The s boundary condition means shrink-wrapped edge, here meaning that the edges are defined by the maximum extent of all atoms. Imagine a one-dimensional simulation box, where the atoms lie scattered over some interval. Then the edges of this simulation box would be at the lower end be on the atom with the lowest coordinates and at the upper end on the atom with the highest coordinates. The box is "shrink-wrapped". Atoms do not interact over the boundaries.

The f boundary condition means fixed edge. A particle that passes over the edge will disappear. Atoms do not interact over the boundaries.

While not strictly a boundary condition, one could use fixes to define walls at the boundaries, on which the particles can collide.

3.3 Cutoff radius

Calculating the unbonded potential for atoms is not a trivial task. If our simulation box contains 3200 atoms, then that potential would be calculated 3199 times for each atom per time step, an atom-atom distance calculation, and computing the Lennard-Jones potential gives a total of more than 20 million calculations ($O(N^2)$)! In this section and the next, we will look at two common ways to simplify these calculations.

The first is to add a non-bonded cutoff distance, not to be confused with the bond breaking cutoff of section 2.12.3. If any atom is closer to our target atom than this distance, then we can proceed with calculating the potential. If the atom-atom distance is larger than the cut-off, then the potential calculation is ignored. This cuts the number of calculations roughly in half, but it is still $O(N^2)$.

Some LJ-potentials with cutoffs have a function that gives the potential a smooth curve towards zero instead of abruptly cutting the potential off at that point (see (Leach, 2001a) pp. 330-4). As far as we understand, LAMMPS does not support these functions and simply set the potentials to zero for distances above cutoff (analogous to equation (122)).

3.4 Neighbor lists

The second way to decrease pair energies is to build neighbor lists. Each atom in the simulations receives such a list. A neighbor list contains the IDs of all atoms in within the cutoff distance of that atom. This means that we can simply look up the list of neighboring atoms to calculate pair potentials instead of going through every single atom for every single time step to calculate these energies. This gets the number of computations down to O(N).

Atoms move of course, and they will sometimes be within cutoff and sometimes not. These neighbor lists are recalculated occasionally, but this method is still much faster than the simple pair search of the section above.

3.5 Assigning velocities

Velocities are usually given to atoms at the beginning of a simulation run, in order to make the system consistent with a given temperature. This works by assigning random velocities to each atom so that the overall velocity distribution resembles the Maxwell-Boltzmann distribution (81). This distribution resembles the Gaussian distribution, and the velocities are often picked by a Gaussian random number generator, and fitted into the Maxwell-Boltzmann distribution.

A problem when picking velocities is that we can end up with a total linear and angular momentum of the system given the momentums of each atom.

$$\boldsymbol{p_{tot}} = \sum_i m_i \boldsymbol{v}_i$$ *(127) Total linear momentum*

$$\boldsymbol{L}_{tot} = \sum_i m_i \boldsymbol{v}_i \times \boldsymbol{r}_i$$ *(128) Total angular momentum*

These momentums may cause the system to move or rotate involuntarily. These values may be zeroed initially by the `velocity` command, or repeatedly during the run by the `momentum` fix by adjusting the velocities slightly. This adjustment causes the kinetic energy to be the same, but the momentums are zeroed out. Removing linear momentum for example is done by the formula $\boldsymbol{v}_{i,new} = \boldsymbol{v}_{i,old} - \frac{\boldsymbol{p}_{tot}}{M}$, where M is the total mass of the system.

3.6 Integrators

We now turn to the question of solving the equation $m\ddot{\boldsymbol{r}}_i = -\nabla_i U_i$. While simple-looking, we can not use any numerical integration scheme to solve this equation, see Figure 48.

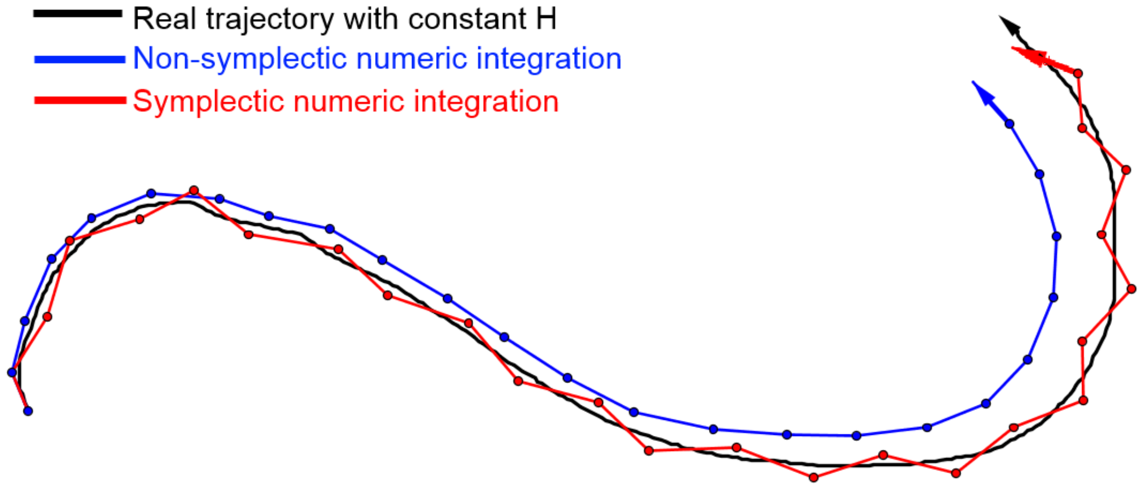

Real trajectory with constant H
Non-symplectic numeric integration
Symplectic numeric integration

Figure 48: Schematic illustration of numerical integration of a microcanonical system traveling through phase space. A non-symplectic integrator will gradually drifts off track due to round-off errors, while a symplectic integrator never will drift far away from the real system.

Ordinary numerical integration schemes like Euler's method and Runge-Kutta methods can solve this equation quite accurately, but when several hundreds of thousands of time steps are run, small numerical errors will eventually creep into the system, causing the numeric solution to drift away from the real trajectory.

Symplectic means volume-preserving in phase space (see equation (64) and its accompanying discussion), and can in this case be seen as a numerical integration schema that keeps the factors that we want to keep fixed (for example energy in a microcanonical ensemble). LAMMPS supports two integrators: The Velocity Verlet Method, the rRESPA integrator (*Tuckerman, Berne, & Martyna, 1992*). We will look at the former.

The NVE integrator is invoked by the command `run_style` and the fix `nve`. It is given by the Velocity Verlet Algorithm (*Swope, Andersen, Berens, & Wilson, 1982*) and it aims to keep the classical Hamiltonian constant.

$$H = \sum \frac{\boldsymbol{p}_i^2}{2m_i} + U$$

Since the Hamiltonian is kept constant, this integration is consistent with the microcanonical/NVE ensemble, thus the name.

The algorithm is given by the following equations:

$$\boldsymbol{r}(t+\Delta t) = \boldsymbol{r}(t) + \Delta t\, \boldsymbol{v}(t) + \frac{1}{2}\Delta t^2 \boldsymbol{a}(t)$$ *(129) VV Updating positions*

$$\boldsymbol{v}\left(t+\frac{1}{2}\Delta t\right) = \boldsymbol{v}(t) + \frac{1}{2}\Delta t \boldsymbol{a}(t)$$ *(130) VV Updating velocities, part 1*

$$\boldsymbol{v}(t+\Delta t) = \boldsymbol{v}\left(t+\frac{1}{2}\Delta t\right) + \frac{1}{2}\Delta t\, \boldsymbol{a}(t+\Delta t)$$ *(131) VV Updating velocities, part 2*

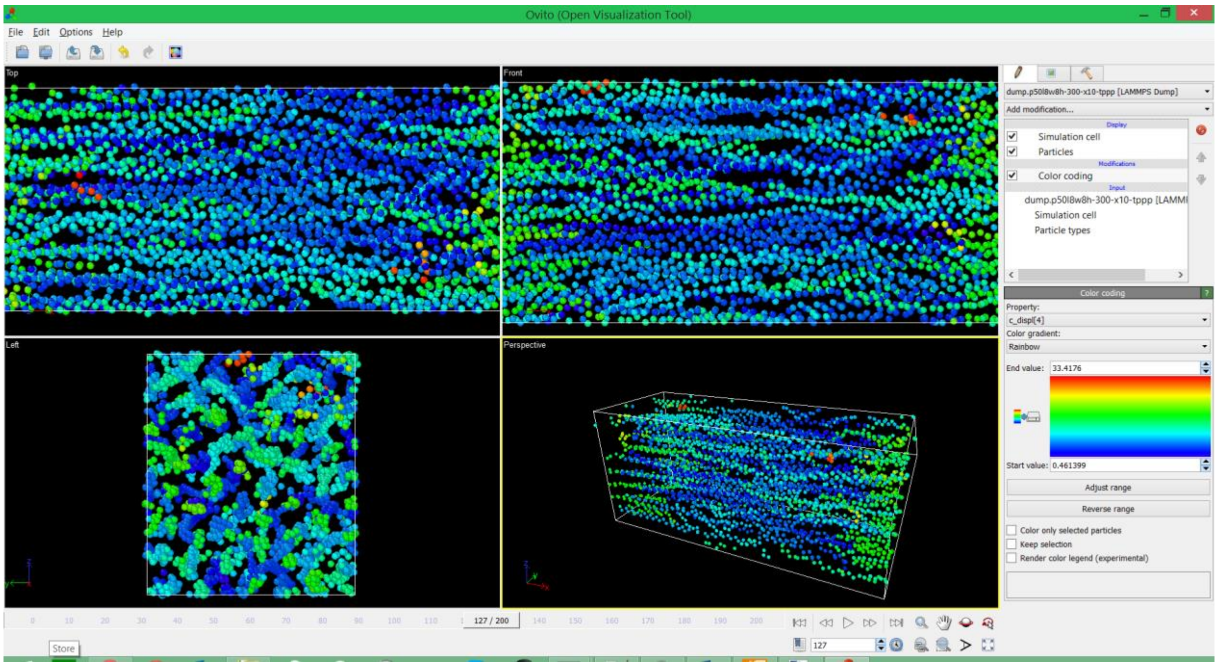
Δt is the time step. $\boldsymbol{a}(t+\Delta t)$ comes from the per-atom force calculated from Newton's law, possibly with Brownian dynamics (see section 3.8.2) on the positions $\boldsymbol{r}(t+\Delta t)$.

For more information on how to keep the temperature constant, see section 3.8.

Velocity Verlet is a symplectic algorithm (Ellad B. Tadmor & Miller, 2011b).

3.7 Ovito

Ovito (The Open Visualization Tool) (*Stukowski, 2010*) is a tool for visualizing output data from LAMMPS. It reads dump files output from LAMMPS, and plots these dumps as 3D figures where each atom is given a position in space. It can also read per-atom values such as forces and potential energy if one choses to output these into the dump files. These values may be shown over each atom as a color coding (see for example Figure 66). While Ovito works best for metal and other solids, it does a good job at outputting results for molecules as well.



Property: c_displ[4]
Color gradient: Rainbow
End value: 33.4176
Start value: 0.461399

Figure 49: A screenshot of Ovito in action. The four big pictures are: Top left: The simulation box as seen from the top (xy-plane, looking down z). Top right: The simulation box as seen from the side (xz-plane, looking down y). Bottom left: The box looked at from the left (yz looking down x). Bottom right: A screen where one can pan and rotate the "camera". The color coding here is a measure of how far the atoms have traveled from their initial positions.

3.8 Thermostats

If we want to run simulations in the NVT ensemble, we will need to use thermostats. Thermostats are fixes that attempt to keep the temperature of the microcanonical system constant, usually by changing each individual atom's dynamics somewhat.

There are also barostats, which change the velocities of atoms to match that of an external stress tensor. We will not look at those here, but they work as the same principle as the Nosé-Hoover thermostat below.

3.8.1 Velocity rescaling

This is the simplest way to keep the temperature constant. If we calculate the temperature every N timesteps, and compare it to a wanted temperature, then we need to multiply all velocities with the factor $\sqrt{T_{wanted}/T_{calculated}}$ to get the temperature we want. This may results in some total momentums for the systems, so they are removed as well.

Velocity rescaling is easy to understand, but not without its problems. First and foremost does in not correspond to the canonical ensemble exactly. In other words, we introduce errors into the simulation.

3.8.2 Langevin thermostat

This thermostat corresponds to bathing the system in a heat bath consisting of small invisible particles. This bath introduces a friction term $\Gamma\boldsymbol{v}$ and a Gaussian noise term $\boldsymbol{A}_i$ to the system. We get the following dynamical equation:

$$m_i\boldsymbol{a}_i = \frac{\partial U(\boldsymbol{r})}{\partial \boldsymbol{r}_i} - m_i\Gamma\boldsymbol{v}_i + \boldsymbol{A}_i(t) \qquad \text{(132) Brownian dynamics}$$

Solving this equation with the Velocity Verlet retains the symplecticity of that algorithm, meaning that we can run simulations for many time steps without introducing large errors.

The dampening constant Γ cannot be too small, or else the system will take too long to equilibriate. If it is too large, then the dynamics will drown in noise. A common value is to set Γ = 100 fs^{-1}.

The noise term has the mean $\langle A_i \rangle = 0$ and the variance $\mathrm{Var}(A_i) = 2\Gamma m_i k_b T/\Delta t$, where Δt is the time step. This specific variance makes the Langevin thermostat properly canonical. Since the noise term needs random numbers to work, this thermostat will need a random number seed as well.

3.8.3 Nosé-Hoover thermostat

This thermostat is built upon adding an invisible particle with mass M to the system (think of it as an extra degree of freedom). The properties of this particle is that it alone acts as a heat reservoir, which turns the microcanonical system to a system where the temperature is kept. As a consequence, the dynamics of the system is given as:

$$m_i\boldsymbol{a}_i = \frac{\partial U(\boldsymbol{r})}{\partial \boldsymbol{r}_i} - m_i\Gamma\boldsymbol{v}_i \qquad \text{(133) Nosé-Hoover dynamics}$$

Where the damping coefficient is given as:

$$\frac{d}{dt}\Gamma = \frac{N}{3k_b M}(T_{calculated} - T_{wanted})$$

(134) Nosé-Hoover damping coefficient

Once again, the value of the parameter M must be set carefully.

This thermostat has the advantage of being deterministic (non-random), but it has the major disadvantage of being non-ergodic, meaning that the system can be trapped in bounded region of the phase space, making the output meaningless. This can be remedied by adding more particles to the system, each one is a thermostat that thermostats the previous thermostat. This is called a *Nosé-Hoover chain*, and LAMMPS uses a chain of length 3 as default, where all particles have the same mass.

4 Building an initial configuration

LAMMPS needs an initial configuration of atoms to work with. There are several ways to generate such initial structure: Adding each atoms or molecules randomly, setting up a lattice to fill with atoms or molecules, using a data file or restarting a previous simulation. For the first two methods, LAMMPS has commands in place, the last two methods are used by reading external files.

The `create_atoms` command fills the simulation box with either atoms or molecules (see Appendix 4.4) by filling a lattice (itself defined before by the command `lattice`) or filling the box randomly with N atoms or molecules. The lattice variant is useful if one wants to set up an initial configuration of metal, as lattices such as `fcc`, `bcc` and `hcp` are supported.

The `read_data` command reads a text file containing the initial structure of atoms with coordinates, bond topology, angle definitions and more. This file is created by a script (see the section below, and Appendixes 4.1 and 4.2) or LAMMPS can write it by a command.

The `read_restart` command reads a binary file containing information about where the atoms are, their velocities, what pair-styles are applied and so on. It basically contains a lot of data one needs to restart a simulation where one left off. This is useful in case a simulation would suddenly quit, the task would hit the wall time limit (if the task was submitted to a supercomputer for example, see Appendix 1) or the number of time steps defined in the run were not enough to finish the task. These binaries need to be defined in the script in order to be produced. See section 6.1 in the LAMMPS documentation for more details.

4.1 Building crystals

Our approach to building initial structures of polymers is motivated by the fact that polymers form crystalline phases (see Part B, section 2.5). The crystals given there are possible to build, but quite difficult, so we will use simpler lattices instead. We will also not model the lamellar structure of polyethylene, but instead stack several long polymer chains in a crystalline way. Let us start with the basics.

A Bravais lattice refers to an infinite array of points in space generated by a basis of vectors, called *primitive vectors*. Three vectors generate a three-dimensional lattice, for example.

$$\boldsymbol{r}_{n_1 n_2 n_3} = n_1\boldsymbol{u} + n_2\boldsymbol{v} + n_3\boldsymbol{w},\ \ n_i \in \{0, \pm1, \pm2, \pm3, \dots\}$$

(135): The Bravais lattice in three dimensions

Here, ***r*** is a position vector of a point on the lattice, and ***u***, ***v*** and ***w*** are the primitive vectors, not all in the same plane and not two in the same direction.

Most solids are built up from crystal structures, which are based on the mathematics of lattices. A point on the lattice does not need to be a single atom; in general, each point contains a *basis*, which contains some arrangement of atoms. A basis in this context can be an octane molecule, or when dealing with polyethylene it can be -C_2H_4-.

The strategy is: Build a single layer of molecules, on which the first atoms are located on some 2D-lattice. The molecules will interact with each other isometrically, since the Lennard-Jones potential is not dependent on the direction. It then makes sense to use one of the two 2D-lattices with equal spacing (that is: $\|\boldsymbol{u}\| = \|\boldsymbol{v}\|$). These are the cubic and the hexagonal lattices (See figures Figure 51 and Figure 52 below).

The crystal building script works roughly as follows: Ask for the width and height of the system in number of atoms, and then ask for the length of each chain. From this information, a simulation box is built that is big enough to contain all chains with equilibrium spacing. The yz-plane contains the lattice, and the first atom is placed at the point (0,0,0). From there, the first chain is built along the x-axis in a wavy fashion, as shown by Figure 1. After this, the x-coordinate is reset, and the algorithm moves one spot up the lattice along the z-axis and does this all over again. When the algorithm has reached the end of the simulation box along the z-axis, it resets the z-coordinate and moves one step up the y-coordinate. Repeat until the entire lattice is filled.

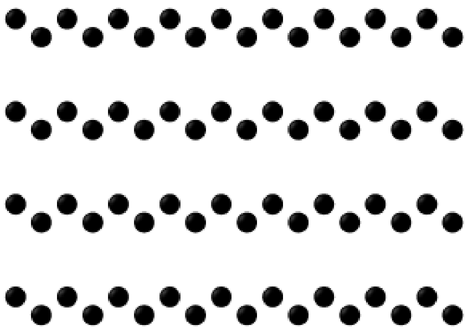


Figure 50: Example of output from the crystal building script. This is an actual output from LAMMPS, drawn in OVITO, so the proportions between nonbonded and bonded equilibrium distance are correct.

This algorithm makes the molecules "swing in phase", as in figure Figure 50. It is also necessary for LAMMPS to define the topology of the system, that is to say which atoms are bonded and which angles and dihedrals are defined. This script does this as well. Users familiar with Python (see for example (Dawson, 2010), or any reference book on the language) can read the script in Appendix 4.1.

4.1.1 *The cubic lattice*

The crystal building script was first implemented with a cubic lattice, since it seemed to be the simplest lattice to work with. The angle between the primitive vectors is 90°, and the length of these is 4.5 Å (see Table 6). Figure 51 shows atoms lying in a cubic lattice, with certain atoms within a cutoff radius (see section 3.3). In that figure, we can count 12 atoms inside the cut-off radius not including the central atom.

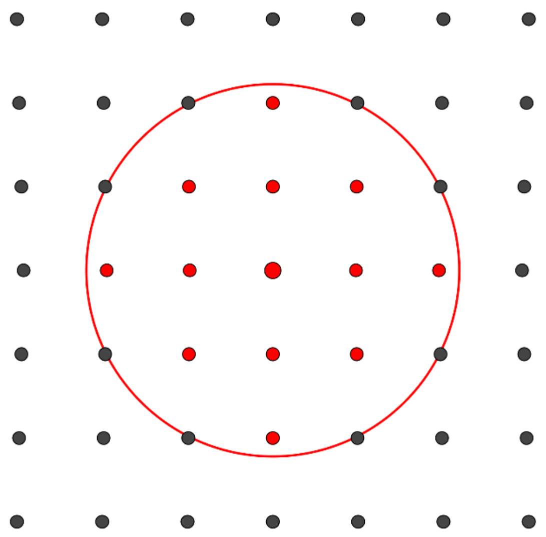


Figure 51: An atom's neighborhood in two dimensions. The red atoms are within in the cut-off radius and count as neighbors for the central atom.

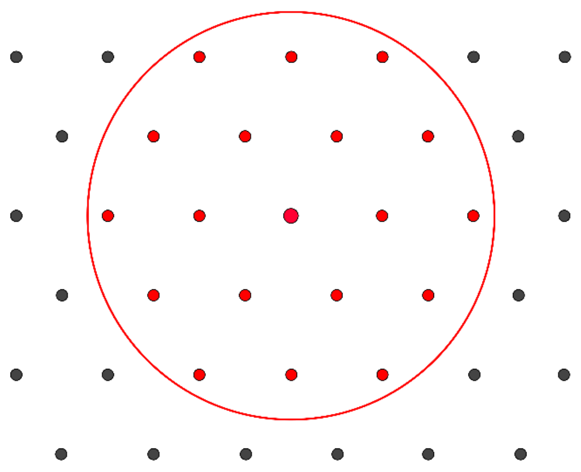


Figure 52: Same as Figure 51, but with a hexagonal lattice. The atom has more neighbors, and the configuration is more stable, see part D, section 3 for more information.

4.1.2 *The hexagonal lattice*

The next generation of the crystal building script used a hexagonal lattice instead of a cubic. The hexagonal lattice is characterized by six nearest neighbors, contra the four of a cubic lattice. The angle between the primitive vectors is 60° or 120° depending on how one wants to look at it, and the length of these vectors is 4.5 Å. The lattice is illustrated in Figure 52.

With the same cut-off radius, an atom has 18 neighbors within it. The hexagonal lattice is simply more close-packed than the cubic lattice.

4.2 *Building semi-random chains*

Polymers are rarely lying in perfect alignment with each other. As stated in Part B, section 2, polymer chains can be curled up into lamellae and amorphous regions. The ways these chains are pulled out to straight chains are one of the principal reasons that polymers have such high ductility. Such unfolding, if done with MD, can take a long time to calculate.

This script, a biased self-avoiding walk is supposed to model such a molecule in its final stage of unfolding, when the molecule is almost, but not entirely, straightened out. The idea is that MD can do the rest of straightening, and we can output load-displacement curves from this. "Self-avoiding walk" means a random walk that will not revisit any of its previously visited spots. "Biased" means that some directions are favored over other. For example given that the walk will go up or down the x-axis, it will go up 70 % of the time and down 30 % of the time.

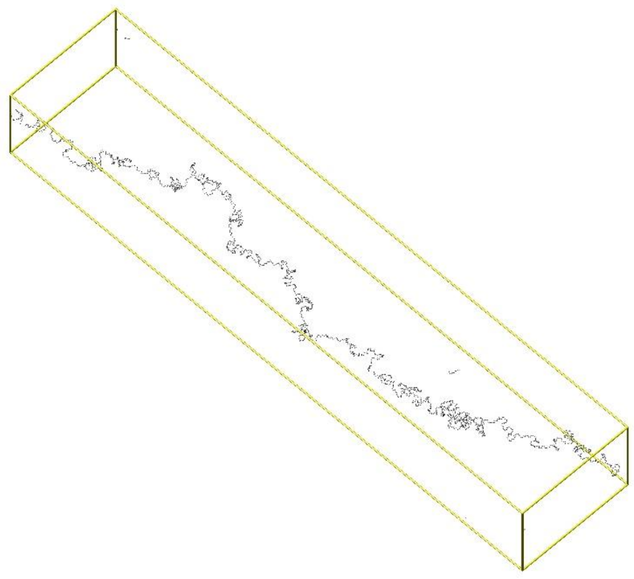


Figure 53: A polymer chain of 2000 pseudoatoms generated with a biased self-avoiding walk algorithm and refined by an energy minimization (see section 5) with periodic boundary conditions. This explains the seemingly isolated chains of atoms at the top of the box, and about 3/4ths of the way down. These are simply part of the polymer poking through a periodic boundary.

The algorithm can be outlined as follows: It will first decide to go along an axis (x, y or z), then choose whether to go up or down that axis. There may be biases to the choices, such as preferring an axis over the others, or preferring to go up rather than down an axis. If the new point found has been visited before, a new attempt is done. If a certain number of attempts have been done without getting anywhere, the algorithm is probably stuck in a “sack” of previously-visited points, and the scripts stops. The algorithm, as it stands today, can only build one single polymer strand at a time. It can be found in Appendix 4.2.

4.3 Simulating chemical attacks

Sometimes a bond breaks even when the bonded atoms are in an equilibrium position. This could be due to radiation kicking out valence electrons of the bond, or a chemical of some sort could attack the bond. Some of these bonds heal immediately afterwards, but for others, the damage is irreversible. We will look at this case here.

The LAMMPS data file is divided into several subsections, the `Atoms` section contain the definition of atoms, by ID, type and position. The ID is unique for each atom, and is used to define bonds, angles and dihedrals (also with their own unique ID). These IDs may not necessarily be in order, a section can begin with the following IDs: 48, 49, 221, 17, 65, 92, … Our approach towards deleting bonds is to pick a random bond from the list of `Bonds`, and change the bond type from 1 to 2. The 2-bond will have a tabulated zero potential, which is done by the same table building algorithm that builds the bond breaking table.

While there is no doubt that an algorithm that changes random bond types from one to another can be written in Python or similar, we have not prioritized to write such a script, and instead have removed bonds manually from a script.

5 Energy minimization

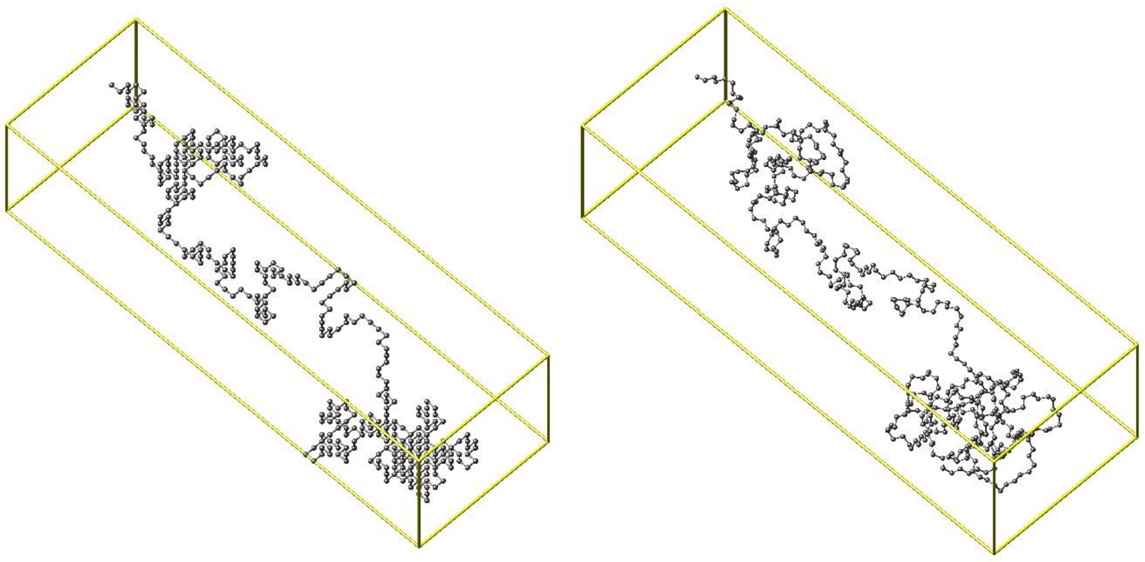


Figure 54: A semi-random of length 421 atoms before (left) and after (right) energy minimization.

Where molecular dynamics focuses on the atomic forces, molecular statics (MS) focuses on the potential energy. The meaning of this statement is that we will with molecular mechanics move atoms based on their potential energy instead of the calculated force. Energy minimization is a way to relax the system before any runs by finding a stable equilibrium for that system (see Part B, section 5.6). Energy minimization can also be used to find the most stable state of molecules, see Figure 55.

If we write the number of degrees of freedom (think bond lengths, angle openings, etc.) as M, then the we can define a M-dimensional vector **x** for the entire system, so that U(**x**) is the potential energy landscape (equation (92)) that we need to minimize. This U(**x**) can be written as a multi-dimensional Taylor expansion:

$$U(\boldsymbol{x}) = U(\boldsymbol{x}^n) + (\boldsymbol{x} - \boldsymbol{x}^n)\boldsymbol{U}'(\boldsymbol{x}^n) + \frac{(\boldsymbol{x} - \boldsymbol{x}^n)^T \boldsymbol{U}''(\boldsymbol{x}^n)(\boldsymbol{x} - \boldsymbol{x}^n)}{2!} + \cdots$$

(136) Taylor expansion

Here, we have expanded the function around the point $\mathbf{x}^n$, $\boldsymbol{U}'(x^n) = \nabla U(\boldsymbol{x} = \boldsymbol{x}^n)$ is the local gradient, $\mathbf{x}^T$ is the vector transpose and $\boldsymbol{U}''(\boldsymbol{x}^n)$ is the Hessian matrix. We will look at two classes of algorithms that attempts to minimize U(**x**).

The first order methods truncate this series after the gradient term, and the second order methods truncate this series after the Hessian term. We will look at the first order methods first.

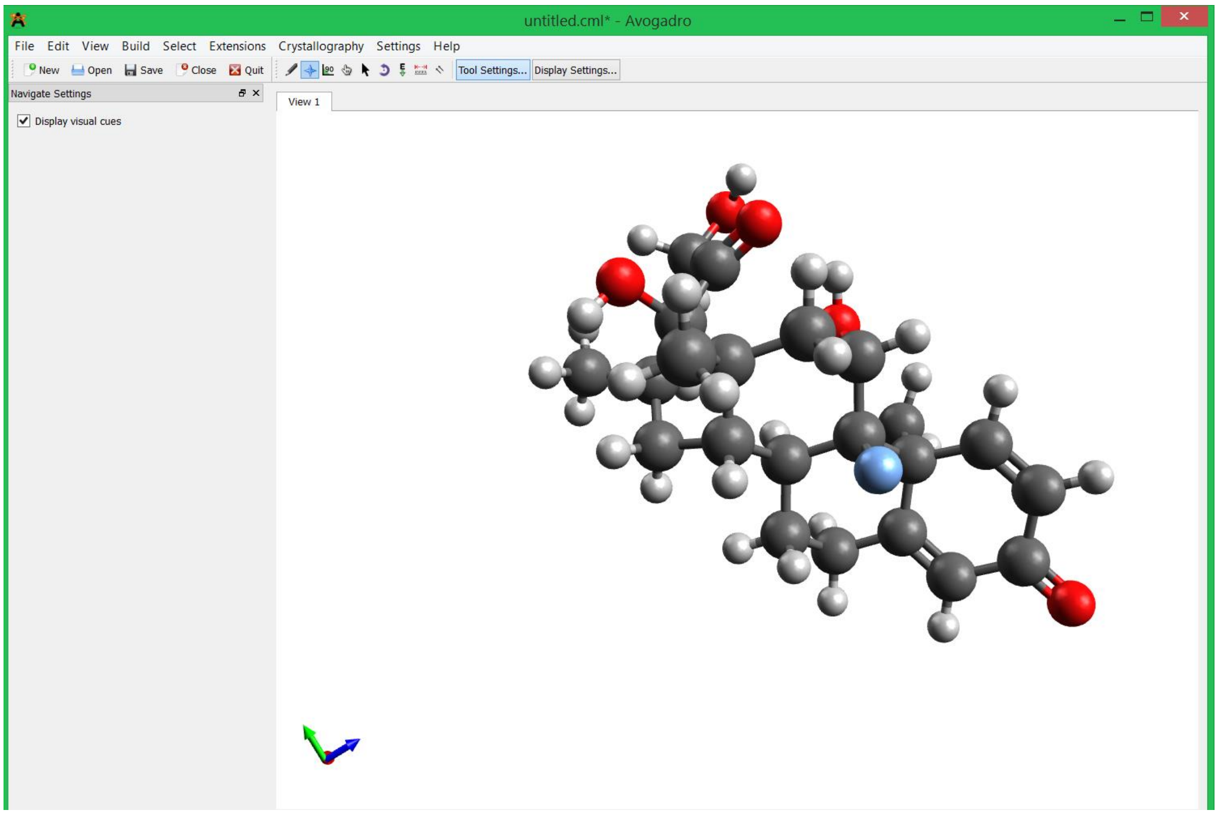

untitled.cml* - Avogadro
File Edit View Build Select Extensions Crystallography Settings Help
New Open Save Close Quit
Tool Settings... Display Settings...
Navigate Settings
Display visual cues
View 1

Figure 55: A screen-shot of Avogadro with a dexamethasone molecule, structure found by energy minimization. The blue atom is fluorine. Among its many uses, dexamethasone can be used to treat external otitis (inflammation of the outer ear).

Our mission is to minimize U(x), and given an initial guess $\mathbf{x}^0$, then the generic algorithm for the first order energy minimization would look like this:

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \lambda^n \boldsymbol{s}^n \qquad \textit{(137) First order searching}$$

λ is the step length and **s** is the search direction, preferably normalized $\|\boldsymbol{s}^n\| = 1$. Now we will only need to determine these two.

5.1 Line search

If we can find a good search direction s, then we can search along this line for a good energy minimum, that is to say that we are trying to minimize the value $U(\boldsymbol{x}^n + \lambda^n \boldsymbol{s}^n)$ for the variable λ^n.

Instead of trying several values of λ, one can find a good value by approximating U as a parabola. We take three points $\lambda_1 = 0$, λ_2 and λ_3, the middle one with somewhat lower potential energy than the others and fit a parabola to these three points. The minimum of this parabola is our estimate for the line minimum, which we will call λ_4. We take this point and the two λ_i around it to find a new parabola and repeat the procedure until it converges. See figure 5.8 in (Leach, 2001a).

5.2 Steepest descent

With a way to find λ, we now turn our attention towards **s**. A seemingly obvious way to do this is to employ the negative gradient of U, which is the local direction where the potential energy decreases the fastest, ergo:

$$\boldsymbol{s}^n = -\nabla U(\boldsymbol{x}^n) = \boldsymbol{f}^n$$

(138) Steepest descent search direction

f is the force arising from our potential energy field. By using this **s**, and repeatedly applying (137) until it converges, and we will have our solution.

The steepest descent algorithm is not the fastest first-order algorithm out there, see figure 5.10 in (Leach, 2001a) for a worst-case scenario. Luckily, there are more adaptive algorithms out there, like the ones we will turn our attention to next.

5.3 Conjugate gradient method

The conjugate-gradient method is a refinement of the steepest descent method, where the new search direction has a "memory" of the previous one. Written in short, the search direction is given as:

$$\boldsymbol{s}^0 = -\nabla U(\boldsymbol{x}^0) = \boldsymbol{f}^0$$

$$\boldsymbol{s}^n = -\nabla U(\boldsymbol{x}^n) + \beta^n \boldsymbol{s}^{n-1} = \boldsymbol{f}^n + \beta^n \boldsymbol{s}^{n-1}$$

(139) The conjugate gradient

β is a scalar factor, that differs between various methods, summed up in the following table:

Table 13: Various formulas for β

Value	Name of method	Article
$\beta^n = \dfrac{\boldsymbol{f}^n \cdot \boldsymbol{f}^n}{\boldsymbol{f}^{n-1} \cdot \boldsymbol{f}^{n-1}}$	Fletcher-Reeves	(*Fletcher & Reeves, 1964*)
$\beta^n = \dfrac{\boldsymbol{f}^n \cdot (\boldsymbol{f}^n - \boldsymbol{f}^{n-1})}{\boldsymbol{f}^{n-1} \cdot \boldsymbol{f}^{n-1}}$	Polak-Ribière	(*Polak & Ribiere, 1969*)
$\beta^n = -\dfrac{\boldsymbol{f}^n \cdot (\boldsymbol{f}^n - \boldsymbol{f}^{n-1})}{\boldsymbol{s}^{n-1} \cdot (\boldsymbol{f}^n - \boldsymbol{f}^{n-1})}$	Hestenes-Stiefel	(*Hestenes & Stiefel, 1952*)

LAMMPS uses the Polak-Ribière β for its conjugate gradient method.

5.4 Hessian-free truncated Newton method

Finally, we will look briefly at a second-order method. The Hessian in (136) can in principle be inverted, but this requires a lot of calculations to do, so we have to make an approximation instead. The minimum for (136) can then be found by Newton-Raphson's method. We can not find the algorithms that lie under this method, so we cannot represent them here. In general Newton's method requires more calculations than the conjugate gradient method, but needs less iterations to converge. Newton's method also is faster when close to an energy minimum.

In LAMMPS we will minimize the system in two steps. The first method is the conjugate gradient method that will roughly find a minimum, and the second is Newton's method that finishes the job.

PART D: RESULTS AND DISCUSSION

This section is quite empty, as there were a haste to get this file finished before the deadline.

1 A survey of scripts

With the various scripts belonging to the simulations, it is in its place to do a taxonomy of them here.

Firstly, we have the data builder files: **builddata.py**, **buildbsaw.py** and **tablebuild.py**. These produce files necessary to run the simulations. Builddata produces crystalline fibres and saves them to files named data.AlBwCh, where A, B, and C are the number of atoms along each edge of the simulation box. Buildbsaw produces a single random chain in a file named data.bsawL, where L is the length of the chain. Tablebuild builds bond energy tables, with varying cutoffs. These files are all read by in-files.

Of the in-files, the file **in.justmini** is special. It takes a data file, does an energy minimization on it an returns a new, optimized data file. That data file is the relaxed state of that system, and it can be used in other in-files. For our simulations, **in.thermo** was used to produce the thermodynamical runs, and **in.tensilesss** and **in.tensileppp** was used to produce the stress-strain data, each with their own way of deforming the chain.

Schematically, the `sss` deformation is done like this:

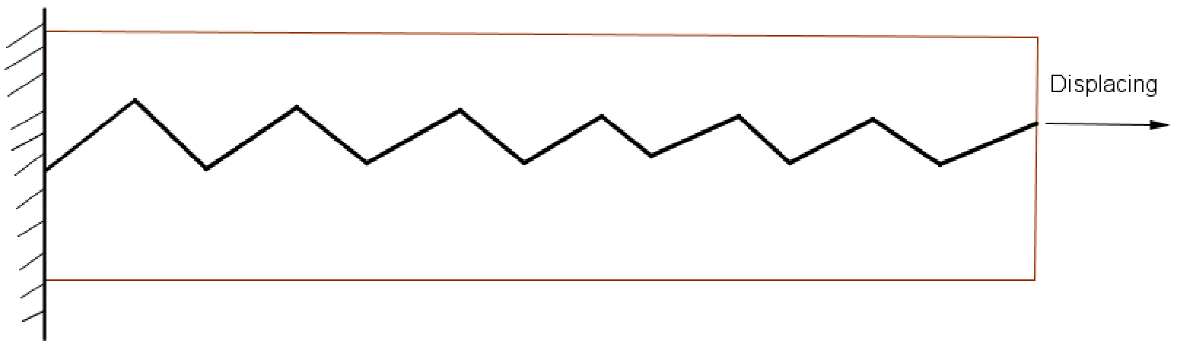


Figure 56: Illustration of the deformation in the sss box. The chains are fixed on one end and moving in the other.

The polymer in the `sss` box can be seen as being fixed on a substrate on the left end and subjected to a displacement on the other. The force that will be used for the force-displacement curves in `sss` are calculated as the force on the fixed atoms.

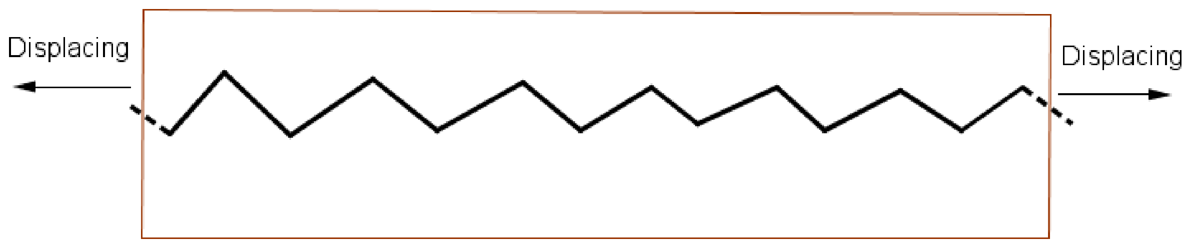


Figure 57: Illustration of the deformation in the ppp box. The chains wrap around the periodic boundaries, whose distances increase over time, giving the chain a certain strain.

For the `ppp` box, we build polymers that wrap over the periodic boundaries. This means that the atom on the left is bonded to the atom on the right over the boundary, and angles and dihedrals follow suit. This results in states where every atom is surrounded by neighbors, which should be ideal, as we could thus connect these results with those on higher length scales. builddata.py can produce these wraparound files as well. Do not attempt to use a file meant for the ppp box in an sss script!

All of this scripts are added as extra material to the thesis on the database, and they can be downloaded from there. Many of them are given in the Appendices as well.

2 Energy minimization

The first order of business was to find a proper initial structure for the hydrocarbons. This was done in two ways, energy minimization (part C, section 5) and freezing a random collection of hydrocarbons, as shown in section 3 below. Energy minimization is also used to prepare some structures for tensile tests (part D, section 4), since this will give an equilibrated system (Part B, section 5.6).

Energy minimization is done in these simulations by first running a conjugate gradient algorithm, followed by a Hessian-free Newton method, as explained in Part C, section 5.

As explained in Part C, section 4.1.1, our first guess for an initial structure was a cubic unit cell (figure Figure 58(l)). Several attempts at minimization were done for different boundary conditions, `sss/pss` and `ppp` unchanged simulation boxes, and `ppp` enlarged box. From simulations regarding freezing octane and icosane systems showed that these spontaneously formed solids would have a hexagonal lattice (see section 3 below), therefore one would expect that an energy minimization would transform a cubic lattice to a hexagonal one.

For the `sss/pss` unchanged box the atoms did barely move, since there would be no way to move without getting closer to any other atom, increasing the energy of the system.

For the `ppp` unchanged box it was expected that every other layer of the cubic crystals would move so that the resulting lattice would be hexagonal. While atoms did move, the entire cubic lattice was kept as a whole. There were probably energy barriers preventing this transformation.

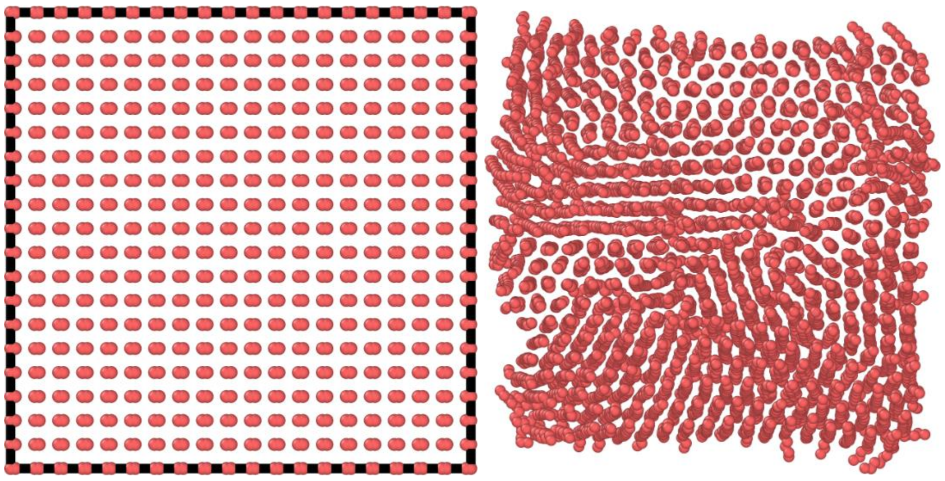


Figure 58: A single-layer icosane crystal before (l) and after (r) energy minimization. The black borders on the left are the boundaries of the simulation box. Both are perspectives down the x-axis. The box boundaries are moved far from the atoms, so that the atoms have room to move during minimization.

The last attempt was done by moving the walls of the simulation boxes by a factor `m`, viz a box with lengths (`lx`, `ly`, `lz`) is increased to a box with lengths (`m*lx`, `m*ly`, `m*lz`) without remapping the atoms within. The size of m is not important, it is merely enough that it gives the atoms some space to move. The result is shown in figure Figure 58(r), while some of the icosane chains make it difficult to make out the figure, it is clear by looking at for example the upper middle that the new crystal structure is hexagonal.

Figure 59 shows the potential energy of an atom in a 2D crystal (like figures Figure 51 and Figure 52) with a Lennard-Jones potential (94), page 47 as a function of the cutoff distance. The energy was calculated with a MATLAB script (an example is given in appendix 4.5, with the underlying mathematics explained in appendix 3) for two equidistant lattices, hexagonal and cubic. The cutoff distance used elsewhere in this thesis is 10 Å. As can be seen from the figure,
$U^{hex} \leq U^{cub}$, and the hexagonal lattice is always more stable than the cubic. This is also made clear by counting the number of neighbors of a single atom. Each immediate neighbor

will give a potential energy of –ε to the atom. An atom in a cubic lattice will have four immediate neighbors, against six neighbors for a hexagonal lattice.

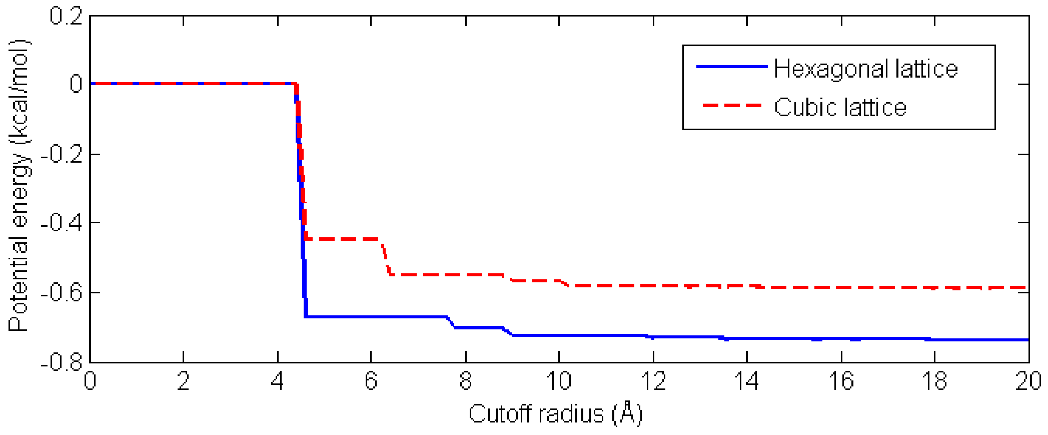


Figure 59: Showing the potential energy of an atom in a 2D-lattice (with lattice constant a = 4.5 Å) as a function of the cutoff radius. The potential energy function is our Lennard-Jones potential from part C, section 2.3.

3 Thermodynamic testing

These test were run by keeping a system at a given temperature for a long time and then outputting some thermodynamic values.

3.1 Octane

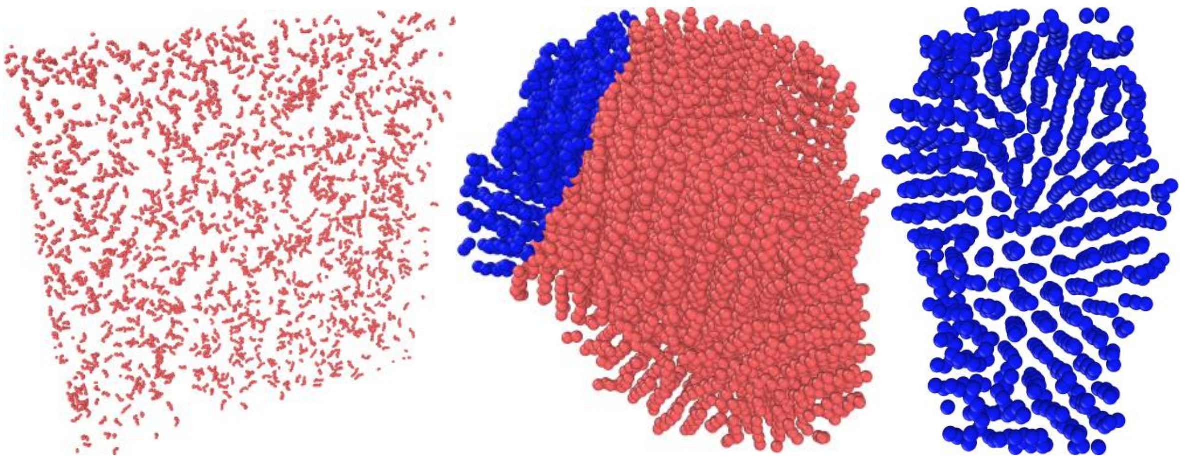


Figure 60: An octane fluid at 900 K (l) frozen to a solid at 150 K (m), where the blue atoms marked are shown on the third picture (r), where one can clearly see the hexagonal crystal structure.

The initial configuration of the octane simulations were done by making a octane structure with builddata.py and melting it at 900 K to get a properly random structure of octane molecules. These initial configurations were then held at various temperatures with the Langevin thermostat in order to produce various outputs.

Of most interest were the cases when this random structure solidified again at small temperatures. The formed solid are shown in Figure 60, with a seemingly random structure, but there is order underneath. The blue atoms on the right lie in a hexagonal order, something that further suggests that the hexagonal unit cell is the most stable one. The reason the orthorombic Pnam cell did not exist in any form for either the energy minimization runs or the thermodynamic simulation could be that the Lennard-Jones cannot reproduce such complicated lattices, or that the united atom model can only form simple lattices or maybe both.

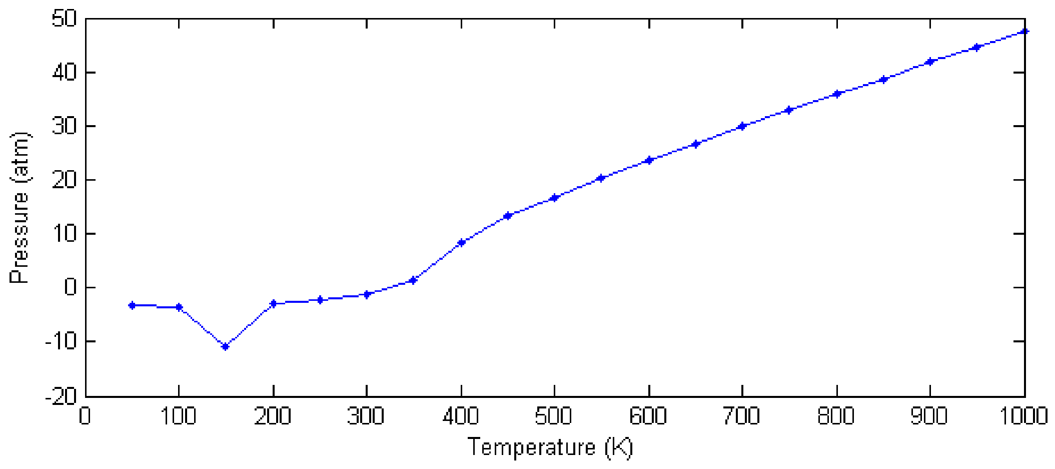


Figure 61: Pressure versus temperature for octane. Each point is a simulation. The results for low temperatures are very uncertain, since the simulations apparently had not come to equilibrium yet.

For low temperatures, the pressures are not exact, as it takes a long time for the octane to solidify and then find equilibrium (see Figure 22), much longer than the usual run times for these simulations. For high temperatures, the data points are much more accurate, and the pressure seems to increase linearly with the temperature. This seems to suggest that the model acts like an ideal gas.

3.2 Icosane

For icosane two different initial configurations were set up in order to counter the weakness of the initial structure of octane. The first configuration is the crystalline one produced by builddata.py (which now was set up to build hexagonal unit cells), and the second is a fluidic configuration done by holding an initial configuration at a high temperature for a long time. The assumption is that each initial state produces results that are better for their native phase (solid and liquid/gas respectively).

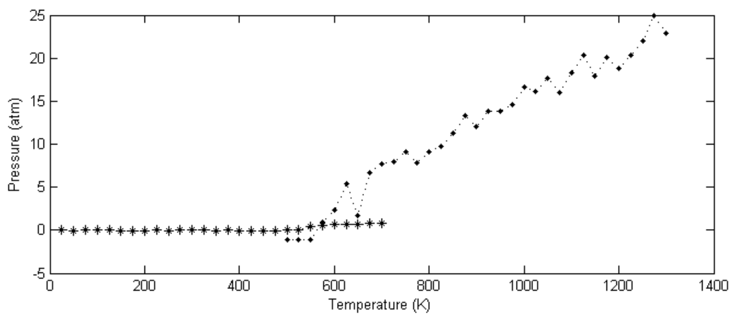


Figure 62: Pressure versus temperature for icosane. Each point is a simulation. The points on the low temperature end of the file started out with the crystalline initial structure. The points on the high temperature end started out with the fluidic initial structure.

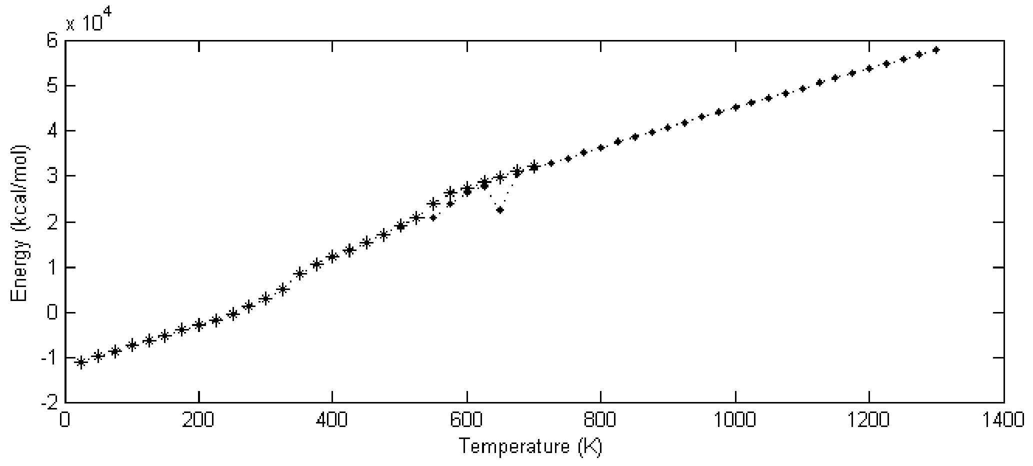


Figure 63: Total energy as a function of temperature. The linearity sections imply, by formula (16) that the isochoric heat capacity is constant.

4 Tensile testing

4.1 ppp boundaries

Our task was to find the bond breaking distance, as given by the cutoff table potential. Many simulations with different bond breaking distances and temperatures were made, and they gave similar results. ***The fracture strain did not depend on the bond breaking length***. This is curious, and goes contrary to our expectations. Compare the two figures underneath. They are made with different bond breaking lengths, yet they have the exact same fracture strain and tensile strength.

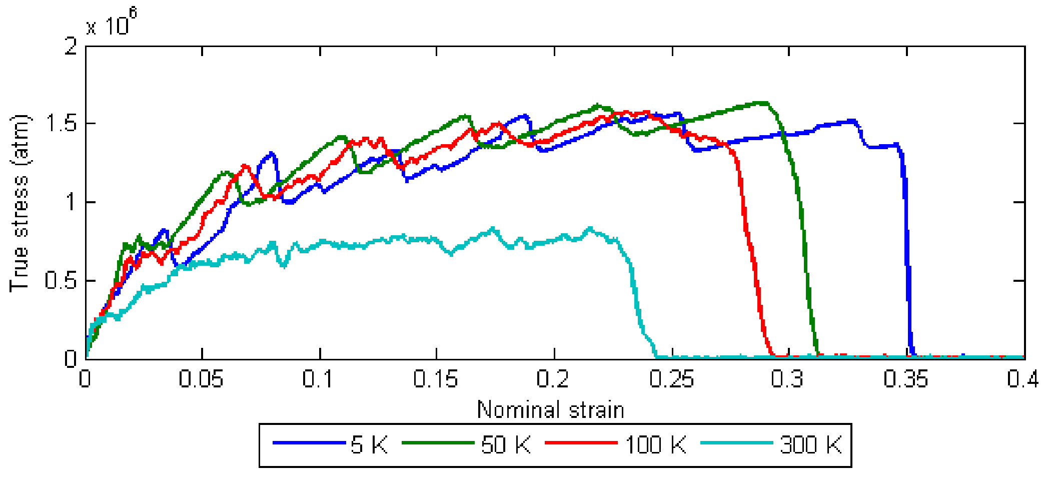


Figure 64: Stress-strain curve for a chain with a table cutoff of 2.6 Å.

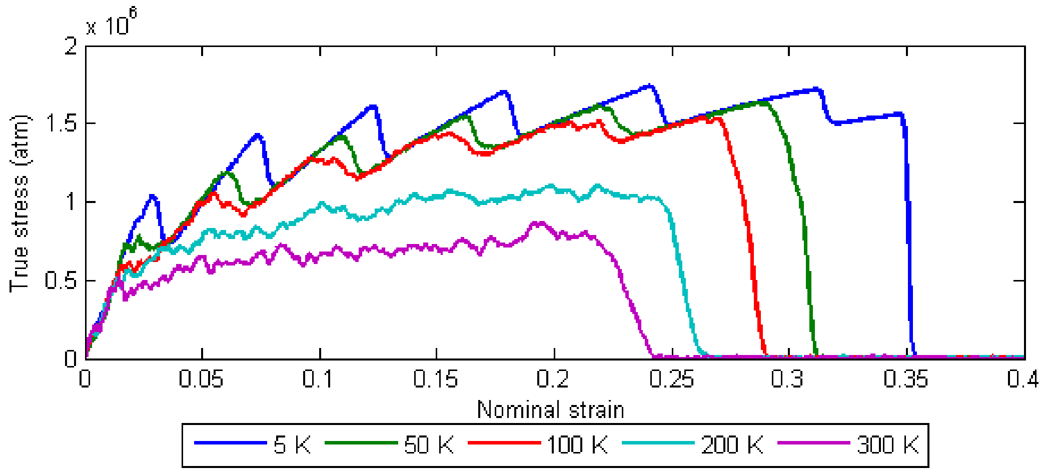


Figure 65: Stress-strain curve for a chain with a cutoff of 5.6 Å.

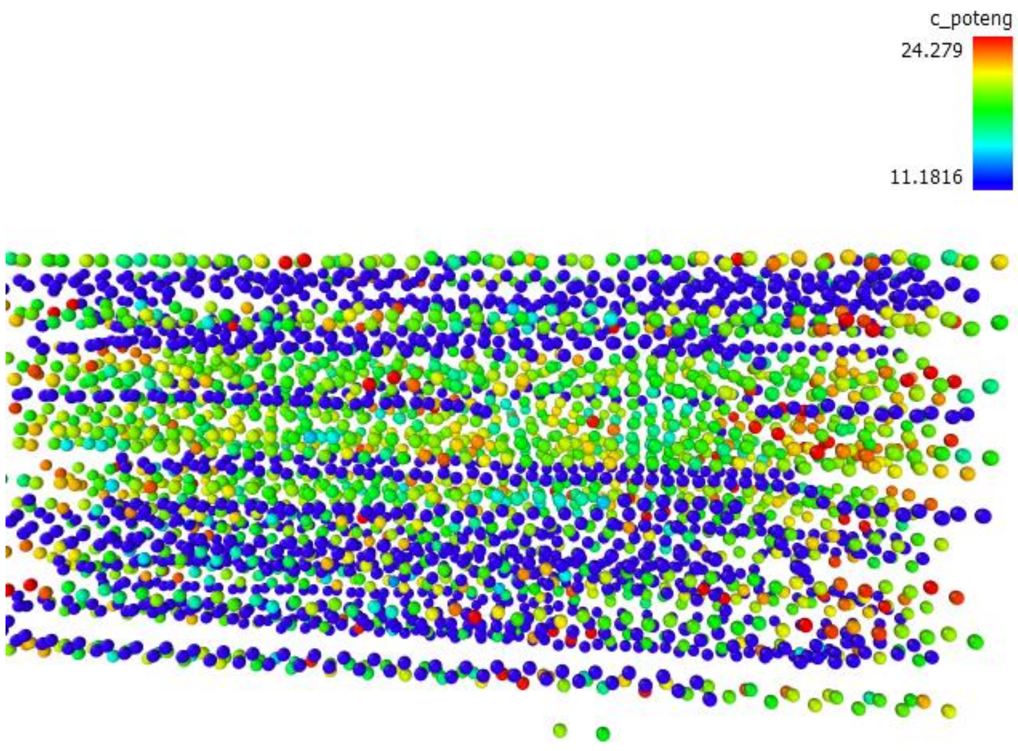


Figure 66: Color coding of the per-atom potential energy. Atoms that are blue have low energy, meaning that blue-colored chains have fractured.

4.2 sss boundaries

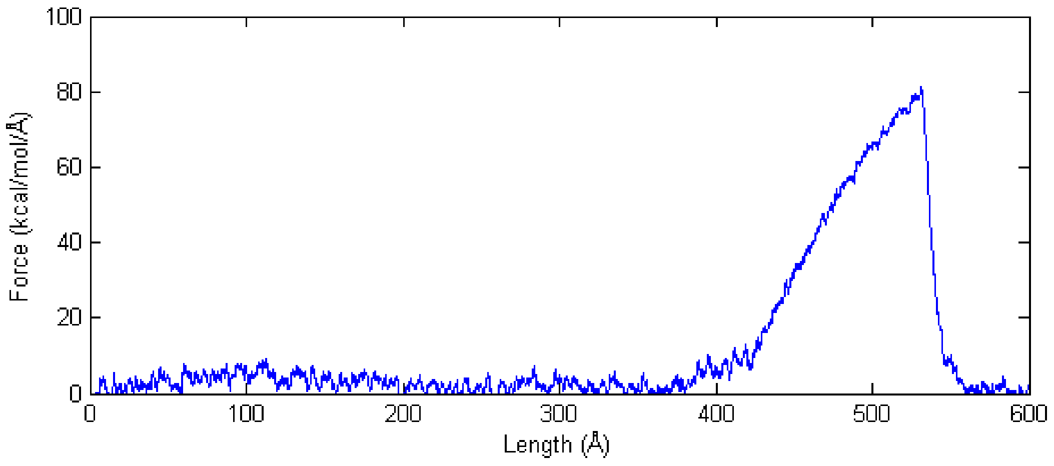


Figure 67: Load-Displacement curve for a semi-random chain of 421 atoms at 50 K. The displacement is relative the length of the curled-up chain, 84 Å

This figure illustrates the load-displacement curve for a single semi-random chain of atoms, the same one as in Figure 54. Much of the early displacement consists of unpacking the molecule to a straight chain. First when there is no more unpacking that can be done, the molecule deforms in the same way as a single straight chain.

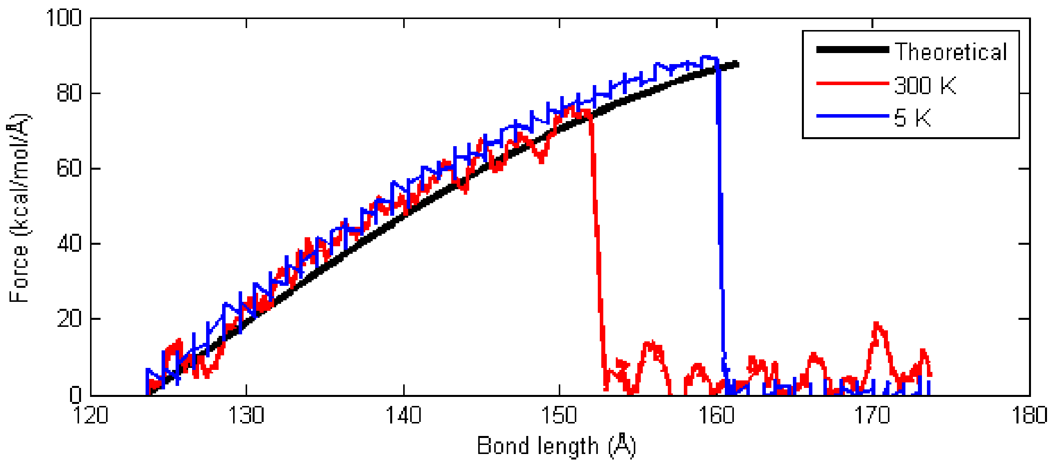


Figure 68: Comparison of the theoretical load-displacement curve (equation (109)) with a simulation run of 100 atoms (table cutoff at 9.0 Å) at 300 K and at 5 K. The theoretical framework does a good job of predicting the simulation results.

At last, we are interested in how the simulation results can be compared to the theoretical results from Part C, section 2.11. It turns out that the theoretical data (worked out for a subunit of 2 atoms) is quite a good prediction for a chain of 100 atoms at the very least, even with different temperatures.

Part F: Conclusions

The search for a bond breaking distance proved unfruitful, in that the same load-displacement curves were produced no matter what sort of cutoff one used. There is obviously a need for improvement here.

The preferred lattice was hexagonal, in opposition to the orthogonal lattice that we know is more stable. We challenge anyone who wants to build the orthogonal Pnam lattice in Python, and attempt to use it as an initial structure for MD simulations.

The theoretical model developed in Part C, section 2.11 seems to agree quite good with the molecular dynamics simulations. This one has potential for further study.

PART G: LIST OF PARAMETERS

Table 14: Potentials

	Unbonded potentials (Lennard-Jones)	
σ	Zero-crossing distance	4.01 Å
r^0	Equilibrium distance	4.50 Å
ε	Depth of the potential well	0.112 kcal/mol
r_c	Cutoff distance	10.0 Å
	Bonded potentials (Morse)	
α	Shape parameter	2.0 Å^{-1}
D	Bond dissociation energy	88.0 kcal/mol
r^b	Reference bond length	1.53 Å
	Bonded potentials (Harmonic)	
r^b	Reference bond length	1.54 Å
k^b	Bond stiffness	700 kcal/mol
K	Bond stiffness parameter	350 kcal/mol
	Angle potentials (cosine/squared)	
K	Angle stiffness parameter	60 kcal/mol
θ^0	Reference angle	109.5°, $\cos(\theta^0) = -0.334$
	Dihedral potentials (OPLS)	
A_1		1.73 kcal/mol
A_2		-4.49 kcal/mol
A_3		0.776 kcal/mol
A_4		6.99 kcal/mol
A_5		0.0 kcal/mol
	Table constants	
r_{cutoff}	Bond breaking distance	? Å

Part H: References

Aljibury, A. L., Snyder, R. G., Strauss, H. L., & Raghavachari, K. (1986). The structure of n-alkanes: High precision abinitio calculation and relation to vibrational spectra. *The Journal of chemical physics, 84*(12), 6872-6878. doi: doi:http://dx.doi.org/10.1063/1.450691

Allinger, N. L., Yuh, Y. H., & Lii, J. H. (1989). Molecular mechanics. The MM3 force field for hydrocarbons. 1. *Journal of the American Chemical Society, 111*(23), 8551-8566.

Aoyama, Y., & Nakano, J. (1999). *Rs/6000 sp: Practical MPI programming*: IBM Corporation.

Bacon, D. J., & Geary, N. A. (1983). Computer simulation of polyethylene crystals. *Journal of Materials Science, 18*(3), 853-863. doi: 10.1007/BF00745585

Barham, P. J. (1986). Strong polymer fibres. *Physics in Technology, 17*(4), 167.

Bassett, D. C., Block, S., & Piermarini, G. J. (1974). A high-pressure phase of polyethylene and chain-extended growth. *Journal of Applied Physics, 45*(10), 4146-4150. doi: doi:http://dx.doi.org/10.1063/1.1663028

Berendsen, H. J., van der Spoel, D., & van Drunen, R. (1995). GROMACS: A message-passing parallel molecular dynamics implementation. *Computer Physics Communications, 91*(1), 43-56.

Berthelot, D. C. (1898). Sur le mélange des gaz. *Comptes rendus hebdomadaires des séances de l'Académie des Sciences, 126*, 1703-1855.

Bohr, N. (1913). I. On the constitution of atoms and molecules. *Philosophical Magazine Series 6, 26*(151), 1-25. doi: 10.1080/14786441308634955

Bohr, N., Kramers, H. A., & Slater, J. C. (1924). LXXVI. The quantum theory of radiation. *Philosophical Magazine Series 6, 47*(281), 785-802. doi: 10.1080/14786442408565262

Bolton, K., Nordholm, S., & Schranz, H. W. (1995). Fragmentation of One-Dimensional Monatomic Chains under Tension: Simulation and Statistical Theory. *The Journal of Physical Chemistry, 99*(9), 2477-2488. doi: 10.1021/j100009a005

Born, M., & Oppenheimer, R. (1927). Zur Quantentheorie der Molekeln. *Annalen der Physik, 389*(20), 457-484. doi: 10.1002/andp.19273892002

Brandrup, J., Immergut, E. H., Grulke, E. A., Abe, A., & Bloch, D. R. (1999). *Polymer handbook* (Vol. 1999): Wiley New York.

Brydson, J. A. (1999). *Plastics Materials* (Seventh ed.): Butterworth-Heinemann.

Buckingham, R. A. (1938). The Classical Equation of State of Gaseous Helium, Neon and Argon. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 168*(933), 264-283. doi: 10.1098/rspa.1938.0173

Bunn, C. W. (1939). The crystal structure of long-chain normal paraffin hydrocarbons. The "shape" of the <CH2 group. *Transactions of the Faraday Society, 35*(0), 482-491. doi: 10.1039/TF9393500482

Chen, C., Depa, P., Sakai, V. G., Maranas, J. K., Lynn, J. W., Peral, I., & Copley, J. R. (2006). A comparison of united atom, explicit atom, and coarse-grained simulation models for poly (ethylene oxide). *The Journal of chemical physics, 124*(23), 234901.

Davisson, C. J. (1928). The diffraction of electrons by a crystal of nickel. *Bell System Technical Journal, The, 7*(1), 90-105. doi: 10.1002/j.1538-7305.1928.tb00342.x

Dawson, M. (2010). *Python Programming for the Absolute Beginner, Third Edition* (3rd ed.): Course Technology.

De Broglie, L. (1924). *Recherches sur la théorie des quanta.* Migration-université en cours d'affectation.

Delhommelle, J., & Millié, P. (2001). Inadequacy of the Lorentz-Berthelot combining rules for accurate predictions of equilibrium properties by molecular simulation. *Molecular Physics, 99*(8), 619-625. doi: 10.1080/00268970010020041

Doerr, T., & Taylor, P. (1994). Breaking in polymer chains. I. The harmonic chain. *The Journal of chemical physics, 101*(11), 10107-10117.

Dowling, N. E. (2013). *Mechanical Behavior of Materials: Engineering Methods for Deformation, Fracture and Fatigue: International Edition* (4th ed.). Essex: Pearson Education Limited.

Einstein, A. (1905). Über einen die Erzeugung und Verwandlung des Lichtes betreffenden heuristischen Gesichtspunkt. *Annalen der Physik, 322*(6), 132-148. doi: 10.1002/andp.19053220607

Eisenschitz, R., & London, F. (1930). Über das Verhältnis der van der Waalsschen Kräfte zu den homöopolaren Bindungskräften. *Zeitschrift für Physik, 60*(7-8), 491-527. doi: 10.1007/BF01341258

Finnis, M. W., & Sinclair, J. E. (1984). A simple empirical N-body potential for transition metals. *Philosophical Magazine A, 50*(1), 45-55. doi: 10.1080/01418618408244210

Fletcher, R., & Reeves, C. M. (1964). Function minimization by conjugate gradients. *The Computer Journal, 7*(2), 149-154. doi: 10.1093/comjnl/7.2.149

Flory, P. J. (1962). On the Morphology of the Crystalline State in Polymers. *Journal of the American Chemical Society, 84*(15), 2857-2867. doi: 10.1021/ja00874a004

Fontana, L., Vinh, D. Q., Santoro, M., Scandolo, S., Gorelli, F., Bini, R., & Hanfland, M. (2007). High-pressure crystalline polyethylene studied by x-ray diffraction and ab initio simulations. *Physical Review B, 75*(17), 174112.

Frenkel, D., & Smit, B. (2001). *Understanding molecular simulation: from algorithms to applications* (Vol. 1): Academic press.

Ghosh, A., Dimitrov, D., Rostiashvili, V., Milchev, A., & Vilgis, T. (2010). Thermal breakage and self-healing of a polymer chain under tensile stress. *The Journal of chemical physics, 132*(20), 204902.

Gibson, J. B., Goland, A. N., Milgram, M., & Vineyard, G. H. (1960). Dynamics of Radiation Damage. *Physical Review, 120*(4), 1229-1253.

Hageman, J., de Wijs, G., de Groot, R., & Meier, R. J. (2000). Bond scission in a perfect polyethylene chain and the consequences for the ultimate strength. *Macromolecules, 33*(24), 9098-9108.

Halgren, T. A. (1996). Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94. *Journal of computational chemistry, 17*(5-6), 490-519. doi: 10.1002/(SICI)1096-987X(199604)17:5/6<490::AID-JCC1>3.0.CO;2-P

Hanwell, M. D., Curtis, D. E., Lonie, D. C., Vandermeersch, T., Zurek, E., & Hutchison, G. R. (2012). Avogadro: an advanced semantic chemical editor, visualization, and analysis platform. *Journal of cheminformatics, 4*(1), 1-17.

Hess, B., Kutzner, C., Van Der Spoel, D., & Lindahl, E. (2008). GROMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation. *Journal of chemical theory and computation, 4*(3), 435-447.

Hestenes, M. R., & Stiefel, E. (1952). *Methods of conjugate gradients for solving linear systems* (Vol. 49): NBS.

Hikosaka, M., Tsukijima, K., Rastogi, S., & Keller, A. (1992). Equilibrium triple point pressure and pressure-temperature phase diagram of polyethylene. *Polymer, 33*(12), 2502-2507. doi: http://dx.doi.org/10.1016/0032-3861(92)91130-T

Jacobs, M. J. N., & Van Dingenen, J. L. J. (2001). Ballistic protection mechanisms in personal armour. *Journal of Materials Science, 36*(13), 3137-3142. doi: 10.1023/A:1017922000090

Jones, J. E. (1924a). On the Determination of Molecular Fields. I. From the Variation of the Viscosity of a Gas with Temperature. *Proceedings of the Royal Society of London. Series A, 106*(738), 441-462. doi: 10.1098/rspa.1924.0081

Jones, J. E. (1924b). On the Determination of Molecular Fields. II. From the Equation of State of a Gas. *Proceedings of the Royal Society of London. Series A, 106*(738), 463-477. doi: 10.1098/rspa.1924.0082

Jorgensen, W. L., Maxwell, D. S., & Tirado-Rives, J. (1996). Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids. *Journal of the American Chemical Society, 118*(45), 11225-11236. doi: 10.1021/ja9621760

Keith, H. D., & Padden, F. J. (1963). A Phenomenological Theory of Spherulitic Crystallization. *Journal of Applied Physics, 34*(8), 2409-2421. doi: doi:http://dx.doi.org/10.1063/1.1702757

Kennedy, M. A., Peacock, A. J., & Mandelkern, L. (1994). Tensile Properties of Crystalline Polymers: Linear Polyethylene. *Macromolecules, 27*(19), 5297-5310. doi: 10.1021/ma00097a009

Koopmans, R., Doelder, J. d., & Molenaar, J. (2010a). Polymer melt fracture (pp. 21-52). Boca Raton: Taylor & Francis.

Koopmans, R., Doelder, J. d., & Molenaar, J. (2010b). Polymer melt fracture (pp. 53-85). Boca Raton: Taylor & Francis.

LAMMPS WWW site. (2013). Retrieved Nov 25, 2013, from http://lammps.sandia.gov

Lavine, M. S., Waheed, N., & Rutledge, G. C. (2003). Molecular dynamics simulation of orientation and crystallization of polyethylene during uniaxial extension. *Polymer, 44*(5), 1771-1779. doi: http://dx.doi.org/10.1016/S0032-3861(03)00017-X

Leach, A. R. (2001a). *Molecular Modelling: Principles and Applications*: Prentice Hall.

Leach, A. R. (2001b). Molecular Modelling: Principles and Applications (pp. 204-212): Prentice Hall.

Leach, A. R. (2001c). Molecular Modelling: Principles and Applications (pp. 303-352): Prentice Hall.

Lindahl, E., Hess, B., & Van Der Spoel, D. (2001). GROMACS 3.0: a package for molecular simulation and trajectory analysis. *Molecular modeling annual, 7*(8), 306-317.

London, F. (1930). Zur Theorie und Systematik der Molekularkräfte. *Zeitschrift für Physik, 63*(3-4), 245-279. doi: 10.1007/BF01421741

Lorentz, H. A. (1881). Ueber die Anwendung des Satzes vom Virial in der kinetischen Theorie der Gase. *Annalen der Physik, 248*(1), 127-136. doi: 10.1002/andp.18812480110

Mainardi, F., & Spada, G. (2011). Creep, relaxation and viscosity properties for basic fractional models in rheology. *The European Physical Journal Special Topics, 193*(1), 133-160.

Martin, J. (1975). Many-body forces in metals and the Brugger elastic constants. *Journal of Physics C: Solid State Physics, 8*(18), 2837.

Martin, M. G., & Siepmann, J. I. (1998). Transferable Potentials for Phase Equilibria. 1. United-Atom Description of n-Alkanes. *The Journal of Physical Chemistry B, 102*(14), 2569-2577. doi: 10.1021/jp972543+

Metzger, R. M. (2012). *The physical chemist's toolbox*. Hoboken, N.J.: Wiley.

Morse, P. M. (1929). Diatomic molecules according to the wave mechanics. II. vibrational levels. *Physical Review, 34*(1), 57.

Nyden, M. R., & Noid, D. W. (1991). Molecular dynamics of initial events in the thermal degradation of polymers. *The Journal of Physical Chemistry, 95*(2), 940-945. doi: 10.1021/j100155a081

Oliveira, F., & Taylor, P. (1994). Breaking in polymer chains. II. The Lennard-Jones chain. *The Journal of chemical physics, 101*(11), 10118-10125.

Ottaviani, J. P., L. (2009). *Suspended in Language: Niels Bohr's Life, Discoveries, and the Century He Shaped* (2 ed.): General Tektronics Labs.

Padding, J. T., & Briels, W. J. (2002). Time and length scales of polymer melts studied by coarse-grained molecular dynamics simulations. *The Journal of chemical physics, 117*(2), 925-943. doi: doi:http://dx.doi.org/10.1063/1.1481859

Papkov, D., Zou, Y., Andalib, M. N., Goponenko, A., Cheng, S. Z. D., & Dzenis, Y. A. (2013). Simultaneously Strong and Tough Ultrafine Continuous Nanofibers. *ACS Nano, 7*(4), 3324-3331. doi: 10.1021/nn400028p

Parr, R. G. (1983). Density functional theory. *Annual Review of Physical Chemistry, 34*(1), 631-656.

Pathria, R. K., & Beale, P. D. (2011). *Statistical mechanics* (3rd ed.). Amsterdam: Elsevier.

Paul, W., Yoon, D. Y., & Smith, G. D. (1995). An optimized united atom model for simulations of polymethylene melts. *The Journal of chemical physics, 103*(4), 1702-1709.

Phillips, J. C., Braun, R., Wang, W., Gumbart, J., Tajkhorshid, E., Villa, E., . . . Schulten, K. (2005). Scalable molecular dynamics with NAMD. *Journal of computational chemistry, 26*(16), 1781-1802. doi: 10.1002/jcc.20289

Planck, M. (1901). Ueber das Gesetz der Energieverteilung im Normalspectrum. *Annalen der Physik, 309*(3), 553-563. doi: 10.1002/andp.19013090310

Plimpton, S. (1995). Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics, 117*(1), 1-19.

Polak, E., & Ribiere, G. (1969). Note sur la convergence de méthodes de directions conjuguées. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique, 3*(R1), 35-43.

Puthur, R., & Sebastian, K. L. (2002). Theory of polymer breaking under tension. *Physical Review B, 66*(2), 024304.

Rein, D. M., Shavit-Hadar, L., Khalfin, R. L., Cohen, Y., Shuster, K., & Zussman, E. (2007). Electrospinning of ultrahigh-molecular-weight polyethylene nanofibers. *Journal of Polymer Science Part B: Polymer Physics, 45*(7), 766-773. doi: 10.1002/polb.21122

Rudd, R. E., & Broughton, J. Q. (1998). Coarse-grained molecular dynamics and the atomic limit of finite elements. *Physical Review B, 58*(10), R5893-R5896.

Sahputra, I. H., & Echtermeyer, A. T. (2013). Effects of temperature and strain rate on the deformation of amorphous polyethylene: a comparison between molecular dynamics simulations and experimental results. *Modelling and Simulation in Materials Science and Engineering, 21*(6), 065016.

Smith, P., & Lemstra, P. J. (1980). Ultra-high-strength polyethylene filaments by solution spinning/drawing. *Journal of Materials Science, 15*(2), 505-514.

Stember, J. N., & Ezra, G. S. (2007). Fragmentation kinetics of a Morse oscillator chain under tension. *Chemical Physics, 337*(1–3), 11-32. doi: http://dx.doi.org/10.1016/j.chemphys.2007.06.019

Stepto, R. F. T. (2009). Dispersity in polymer science (IUPAC Recommendations 2009). *Pure and Applied Chemistry, 81*(2), 351-353. doi: 10.1351/PAC-REC-08-05-02

Stukowski, A. (2010). Visualization and analysis of atomistic simulation data with OVITO–the Open Visualization Tool. *Modelling and Simulation in Materials Science and Engineering, 18*(1), 015012.

Sun, H. (1998). COMPASS: An ab Initio Force-Field Optimized for Condensed-Phase ApplicationsOverview with Details on Alkane and Benzene Compounds. *The Journal of Physical Chemistry B, 102*(38), 7338-7364. doi: 10.1021/jp980939v

Swope, W. C., Andersen, H. C., Berens, P. H., & Wilson, K. R. (1982). A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *The Journal of chemical physics, 76*(1), 637-649. doi: doi:http://dx.doi.org/10.1063/1.442716

Tadmor, E. B., & Miller, R. E. (2011a). Modeling materials : continuum, atomistic and multiscale techniques (pp. 246-262). New York: Cambridge University Press.

Tadmor, E. B., & Miller, R. E. (2011b). *Modeling materials : continuum, atomistic and multiscale techniques*. New York: Cambridge University Press.

Tadmor, E. B., & Miller, R. E. (2011c). Modeling materials : continuum, atomistic and multiscale techniques (pp. 377-439). New York: Cambridge University Press.

Tadmor, E. B., Miller, R. E., & Elliott, R. S. (2012). *Continuum Mechanics and Thermodynamics: From Fundamental Concepts to Governing Equations*. Cambridge: Cambridge University Press.

Thomson, G. (1928). *Experiments on the diffraction of cathode rays.* Paper presented at the Proc. Roy. Soc. A.

Toxvaerd, S. (1990). Molecular dynamics calculation of the equation of state of alkanes. *The Journal of chemical physics, 93*(6), 4290-4295.

Tsuneo, S., Tetsuhiko, H., & Kenzo, T. (1968). Phase Transformation and Deformation Processes in Oriented Polyethylene. *Japanese Journal of Applied Physics, 7*(1), 31.

Tuckerman, M., Berne, B. J., & Martyna, G. J. (1992). Reversible multiple time scale molecular dynamics. *The Journal of chemical physics, 97*(3), 1990-2001. doi: doi:http://dx.doi.org/10.1063/1.463137

Van Der Spoel, D., Lindahl, E., Hess, B., Groenhof, G., Mark, A. E., & Berendsen, H. J. (2005). GROMACS: fast, flexible, and free. *Journal of computational chemistry, 26*(16), 1701-1718.

van der Spoel, D., Lindahl, E., Hess, B., Van Buuren, A., Apol, E., Meulenhoff, P., . . . van Drunen, R. (2008). GROMACS user manual version 3.3.

Weaver, J. F., & Madix, R. J. (1999). Trapping dynamics of isobutane, n-butane, and neopentane on Pt(111): Effects of molecular weight and structure. *The Journal of chemical physics, 110*(21), 10585-10598. doi: doi:http://dx.doi.org/10.1063/1.478990

Wolff, D., & Rudd, W. G. (1999). Tabulated potentials in molecular dynamics simulations. *Computer Physics Communications, 120*(1), 20-32. doi: http://dx.doi.org/10.1016/S0010-4655(99)00217-9

Wolfram|Alpha. (2013). Wolfram Alpha.   Retrieved Nov 26, 2013, from http://www.wolframalpha.com

Xia, X.-C., Zhang, Q.-P., Wang, L., Feng, J.-M., & Yang, M.-B. (2014). The Complex Crystalline Structure of Polyethylene/Polycarbonate Microfibril Blends in a Secondary Flow Field. *Macromolecular Chemistry and Physics.* Retrieved n/a, 215, from http://dx.doi.org/10.1002/macp.201400021

Yoon, D. Y., Smith, G. D., & Matsuda, T. (1993). A comparison of a united atom and an explicit atom model in simulations of polymethylene. *The Journal of chemical physics, 98*, 10037.

Zumdahl, S. S. (2009). *Chemical Principles* (6th ed.): Houghton Mifflin Company.

APPENDICES

1 Connecting to a supercomputer

While I did not use one of the available supercomputers at NTNU for my thesis, it is indeed possible to submit LAMMPS tasks to speed up calculation time significantly. With the capacities of supercomputers, one can simulate larger systems than one could with a personal computer. Additionally, LAMMPS is built and optimized for parallel computing, where the system is divided into several subsystems, one for each processor. Neighbor lists are also divided over processors.

To submit a task to a supercomputer, one needs first to apply for a project. For Vilje[14], one needs to send an application to Notur[15]. This can be done twice a year. If one is a student, writing a master thesis or a PhD thesis, a supervisor employed at the university needs to write the application.

When this is done, one can create a user account, and log on with SSH in terminal (Mac or Linux) or putty/WinSCP (Windows; I recommend installing both programs, WinSCP can be used as a graphical interface to upload and download files). All the files necessary for the simulation are uploaded to a home folder, and all one needs is to write a PBS (Portable Batch System) script. This script is used for queuing and scheduling tasks on the supercomputer, ask the support at NTNU HPC for help writing one.

To see the newest version of LAMMPS installed on Vilje, use the command `$ module avail`, and look for `lammps/yymmdd`, and load the newest version in your PBS script.

[14] See https://www.hpc.ntnu.no/display/hpc/Vilje
[15] https://www.notur.no/allocation

2 Folding@Home

At this point, it is worth noting another application of molecular mechanics. The first few paragraphs are a reworking of a similar discussion from chapter 10 in (Leach, 2001a), the last paragraphs are from Folding@Home's webpage.

Peptides and proteins are large molecules built up of amino acids in a sequence dictated by genes in DNA. These molecules do many, often highly specialized, tasks essential for life. A protein folds itself into a shape determined by which of the myriad possible conformations of the protein will have the lowest energy. For example in a water solution, a protein will fold itself up so that the hydrophilic ends will point outwards and the hydrophobic ends inwards. Other effects that make up the shape of proteins include dihedral energy and hydrogen bonds.

We can experimentally determine the shape of such proteins (its tertiary structure, or quaternary if several proteins work together) using Nuclear Magnetic Resonance (NMR) spectroscopy or X-ray crystallography, this takes time and money. By sequencing genes, we can also determine the sequence of amino acids that make up the protein (its primary structure). The problem is determining how the protein goes from the latter to the former.

When the protein folds right, there will be no problem, but if it for some reason it would fold wrong, we can get diseases that may irreversibly damage cells such as Alzheimer's disease, Multiple sclerosis, cancer and so on. This thesis is dedicated to my uncle Bjørn "Bestamann", who died, weakened by cancer and Epstein-Barr as I was working on this thesis.

A great step toward fighting such diseases is to understand the proteins which may fold right or wrong. There are many proteins out there that we still lack a complete understanding of, and this is where molecular mechanics come in play. By using force fields specialized in protein structures, one can predict the shape of proteins, both behaving and misbehaving ones. Doing this for the vast selection of sequenced proteins is a formidable task, but one that luckily can be divided into subtasks.

This brings us to distributed computing. In principle, it is the same as for parallel computing; each processor is given a task, finishes it, and returns it. The difference is that each task is given to a computer somewhere in the world. When the computer is done with the task, it uploads the results to a server, where he or she who requested the task can post-process the data. Typically, one will receive a task/work unit at simulation time t_1, and receive a request to run it to time t_2. With other computers being able to check the results of your computer, simulations can often go up to the scale of milliseconds, the timescale at which proteins form. Distributed computing does not restrict itself to proteins, but can be used on many projects where large quantities of data are analyzed.

A program named Folding@Home[16] (F@H) is one of the largest distributed computing project as of this writing. I have run two threads (one on my CPU and one on my GPU) for almost a year of this writing, and I have finished 226 work units so far, finishing a work unit about every 24 hours or so. My work has ranged from studying how the enzyme Protein kinase C reacts

[16] Visit http://folding.stanford.edu/home/

with the ligand Bryostatin (this could possibly give a cure to Alzheimer’s) to code optimization. Folding@Home uses only idle processor time, and it will run almost unnoticed on your computer. The calculations are run in the molecular dynamics language Gromacs, which also contains force fields optimized for protein research (that also can be used for polymer research). My user name is Baldrian, named for a scene from the German movie “Die Feuerzangenbowle”.

3 Two-dimensional crystals and the cutoff radius

In Part C, section 4.1 and its subsections, I described a Bravais lattice used for filling up an initial structure and in Part D, sections 2 and 3, discussed the hexagonal lattice, which was shown to be the crystal structure that a random configuration of molecules adopted when solidifying. Figure Figure 59 shows how an atom's potential energy in a two-dimensional lattice changes when we change the cutoff radius. The graph was produced in Matlab (Appendix 4.5), this section will explain how.

On a piece of graphing paper, one could draw a perfect lattice of one's choosing, draw a circle with a radius *r*, and count the number of points inside the circle. For a cubic lattice with a lattice constant of one, this is called Gauss's circle problem. In addition, we need to know each point's distance to a center atom. Drawing a lattice that is big enough to house several cutoff distances, and then writing an algorithm searching over all points to see which are within the cutoff radius and which aren't is a waste of time. I decided on the approach detailed below.

We will need some simple theory on vector spaces for this, any text on linear algebra will do. A *real vector space V* is a set closed under the operations of vector addition and scalar multiplication, such that:

$$\begin{gathered} \boldsymbol{a}, \boldsymbol{b} \in V \ \Rightarrow \boldsymbol{a} + \boldsymbol{b} \in V \\ \boldsymbol{a} \in V, k \in \mathbb{R} \ \Rightarrow k\boldsymbol{a} \in V \end{gathered}$$

(140) Closure under vector addition and scalar multiplication respectively

There is more to the definition of vector space than this. Among other things, vector addition needs to be associative and commutative, and scalar multiplication needs to be distributive. We will not need these definitions, as the usual addition and multiplication have these properties. We will however note the following properties of a vector space:

$$\begin{gathered} \boldsymbol{a} \in V \ \Rightarrow -\boldsymbol{a} \in V \\ \boldsymbol{0} \in V \end{gathered}$$

(141) Existence of a negative element and a zero element

A vector space can be *generated* by a set $S = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_n\}$ of vectors, such that any linear combination of the vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_n$ is in V:

$$c_1\boldsymbol{v}_1 + c_2\boldsymbol{v}_2 + \cdots + c_n\boldsymbol{v}_n = \boldsymbol{v} \in V$$

(142) A linear combination of vectors

If *S* is linearly independent, that is that the equation $c_1\boldsymbol{v}_1 + c_2\boldsymbol{v}_2 + \cdots + c_n\boldsymbol{v}_n = \boldsymbol{0}$ has only one solution: $c_1 = 0, c_2 = 0, \dots, c_n = 0$, and that every vector in V can be expressed by formula (142), S is said to be a basis of V. This has the neat effect that each vector in V can be written in only one way from formula (142), motivating the following notation for vectors:

$$\boldsymbol{v} = c_1\boldsymbol{v}_1 + c_2\boldsymbol{v}_2 + \cdots + c_n\boldsymbol{v}_n = (c_1, c_2, \dots, c_n)$$

(143) Coordinate notation

To allow this, we need that *S* is ordered, i.e. that the order of basis vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_n$ stays fixed.

Now for the point of this discussion: By assigning a point in a Bravais lattice the zero vector, one can see that it corresponds to a vector space generated by the primitive vectors, and coefficients that are integers $c_1, c_2, \dots, c_n \in \{\dots, -2, -1, 0, 1, 2, \dots\} = \mathbb{Z}$. In Part C, Section 4.1, I mentioned that no two primitive vectors should lie along the same direction and not three in the same plane. This is to guarantee linear independence.

Let us now study a two-dimensional Bravais lattice. It is generated by a basis of two vectors: $(\boldsymbol{u}, \boldsymbol{v})$. For simplicity's sake, we will direct the **u**-vector along the x-axis, or more likely, direct the x-axis along the **u**-vector. The **v**-vector will then be directed at an angle θ relative to the **u**-vector, see figure Figure 69.

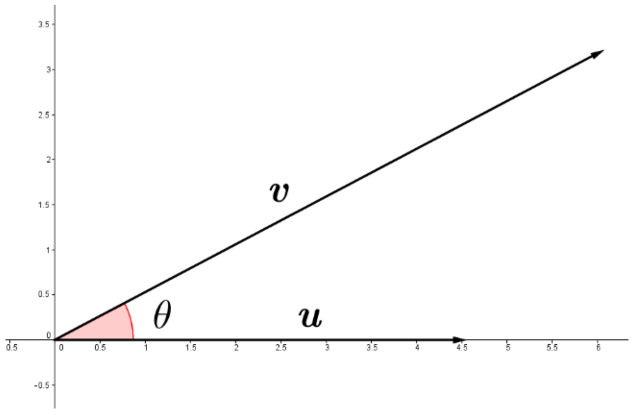


Figure 69: The basis vectors for a 2D Bravais lattice. ***u*** *is defined to go along the x-axis.*

The **u**-vector will have a length (lattice constant) a, and the **v**-vector will have a length b. Any point on the lattice can then be described as:

$$\boldsymbol{r}_{mn} = (m, n) = m\boldsymbol{u} + n\boldsymbol{v}$$
$$\boldsymbol{r}_{mn} = \langle ma + nb\cos(\theta), nb\sin(\theta)\rangle$$

(144) A point on a generic Bravais lattice

The distance from the center is:

$$\|\boldsymbol{r}_{mn}\| = \sqrt{(ma)^2 + 2(ma)(nb)\cos(\theta) + (nb)^2}$$

(145) Distance from point to center

From here on, we will study the two special cases from the thesis, the cubic and the hexagonal lattice. For both cases, the basis vectors are of the same length a, and the angles are 90° and 120° respectively[17]. The distances work out to be:

$$\|\boldsymbol{r}_{mn}^{cub}\| = a\sqrt{m^2 + n^2} \qquad \|\boldsymbol{r}_{mn}^{hex}\| = a\sqrt{m^2 - mn + n^2}$$

(146) Distances on cubic and hexagonal lattices.

We can now proceed to work on the problem stated above. Each pair of (m, n) is an atom (see figures Figure 51Figure 52; part C, section 4.1), giving a distance $\|(m, n)\|$ from a center atom that is easily calculated from (146). Run over all pairs (m, n), such that $\|(m, n)\| = \|r_{mn}\| \leq r_c$, and use all $\|r_{mn}\|$ to calculate the potential energy for that configuration.

Let us try to work out some algorithms. The ***cubic lattice*** is the simplest to work with, since we can use symmetry to cut the work by an eighth, see figure Figure 70. The symmetries are $\|(|m|, n)\| = \|(-|m|, n)\|$ and $\|(m, |n|)\| = \|(m, -|n|)\|$, this means that changing the sign of the coefficients does not change the distance from center. Both taken together, this means that each atom has a mirror image in each of the four quadrants of the coordinate system.

[17] We could also use an angle of 60° for the hexagonal lattice.

This means that we only need to focus our attention on only one quadrant. Another symmetry is $\|(m,n)\| = \|(n,m)\|$, meaning that we need only to look at the atoms in a triangle-shaped space in the lattice, the rest of the atoms follow by this symmetry (Figure 70).

It is however not as simple as counting all the atoms bounded by the triangle and the cutoff radius and multiply by 8. For example, the point $(m,0)$ exists on both the first and fourth quadrants, and has only one image $(-m,0)$ on the second and third quadrants. Likewise, the point (m,m) will be counted twice by this approach. These special cases must be multiplied by a factor of 4 instead of 8.

Finally, we can restrict the coefficients upwards by saying that the largest *m* we will consider is the largest *m* that solves the inequality $ma \leq r_c$, and the largest *n* will be given by $n \leq m$.

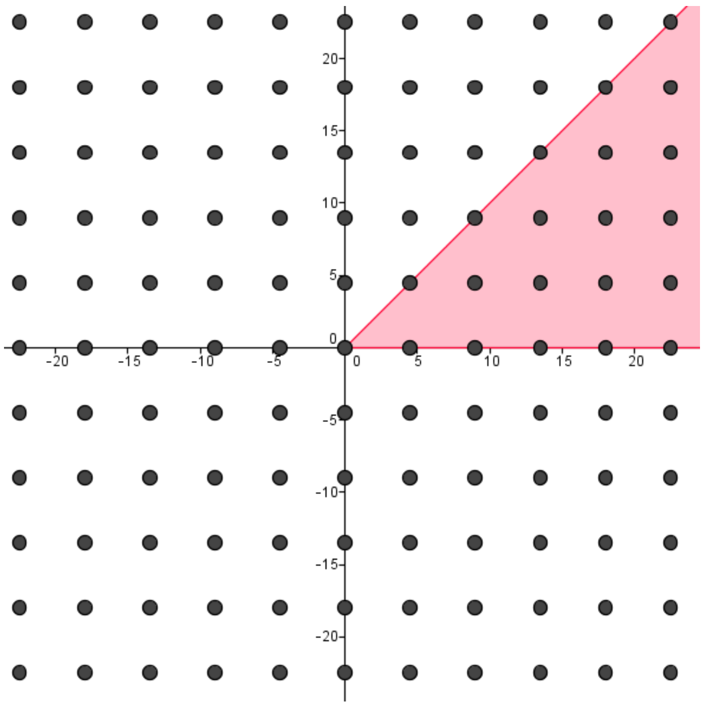


Figure 70: The colored area shows which atoms we will consider in our algorithm. The rest of the lattice follows by symmetry.

Our pseudocode will look like this:

```
function U = cubic_lattice(r_cutoff,a)
m_max = r_cutoff/a

U4 = U8 = 0

count points on the form (m,0) m = 1:m_max, get r, calculate U(r)
and add U(r) to U4

count points on the form (m,m) m = 1:m_max/sqrt(2), get r,
calculate U(r) and add U(r) to U4

i = j = 0

for i = 1:m_max
     j = i + 1
     while |(i,j)| < r_c
          calculate U(r) and add to U8
          increment j
     end
end
U = 4*U4 + 8*U8
```

U(r) is calculated from the Lennard-Jones potential, part C, section 2.3

For the ***hexagonal lattice***, we need to use different symmetries. In the rest of this appendix, I will only skim through the mathematics for this section, but the interested reader may use the formula for a point on the lattice $\boldsymbol{r}_{mn} = a\,\langle m + \frac{1}{2}n, \frac{\sqrt{2}}{2}n\rangle$ to derive the results underneath his- or herself.

Looking at an image of a hexagonal lattice (like Figure 52, page 76), one can easily spot that a point in the first quadrant ($x > 0, y > 0$) has a symmetric image in the three other quadrants. The condition for a point to exist in a quadrant is summed up in the following table:

Table 15: Dividing a hexagonal lattice into four quadrants

Quadrant	**Sign of coordinates**	**Equivalent condition for $(\boldsymbol{m}, \boldsymbol{n})$**
I	$x > 0, \quad y > 0$	$2m > n, \quad n > 0$
II	$x < 0, \quad y > 0$	$2m < n, \quad n > 0$
III	$x < 0, \quad y < 0$	$2m < n, \quad n < 0$
IV	$x > 0, \quad y < 0$	$2m > n, \quad n < 0$

A point existing on one of the two axes has one symmetric image, and are kept as special cases. A point on the x-axis is written on the form $(m, 0)$, and it has an image $(-m, 0)$. Likewise, a point on the y-axis is written as $(n, 2n)$ with image $(-n, -2n)$.

For a point inside the quadrants, there are three images in each of the other quadrants. In other words, a point (m, n) has the three images $(n - m, n)$, $(m - n, -n)$ and $(-m, -n)$. It does not matter which quadrant (m, n) lies in, the other points will lie in the three other quadrants. The images are respectively reflection over x-axis, y-axis and origin.

When we now know that by counting points only in the first quadrant, we will cover the entire space. We need now only to concentrate on the points in the first quadrant, as given by table Table 15 and the points on the positive halves of each axis and multiply by 4 and 2 respectively.

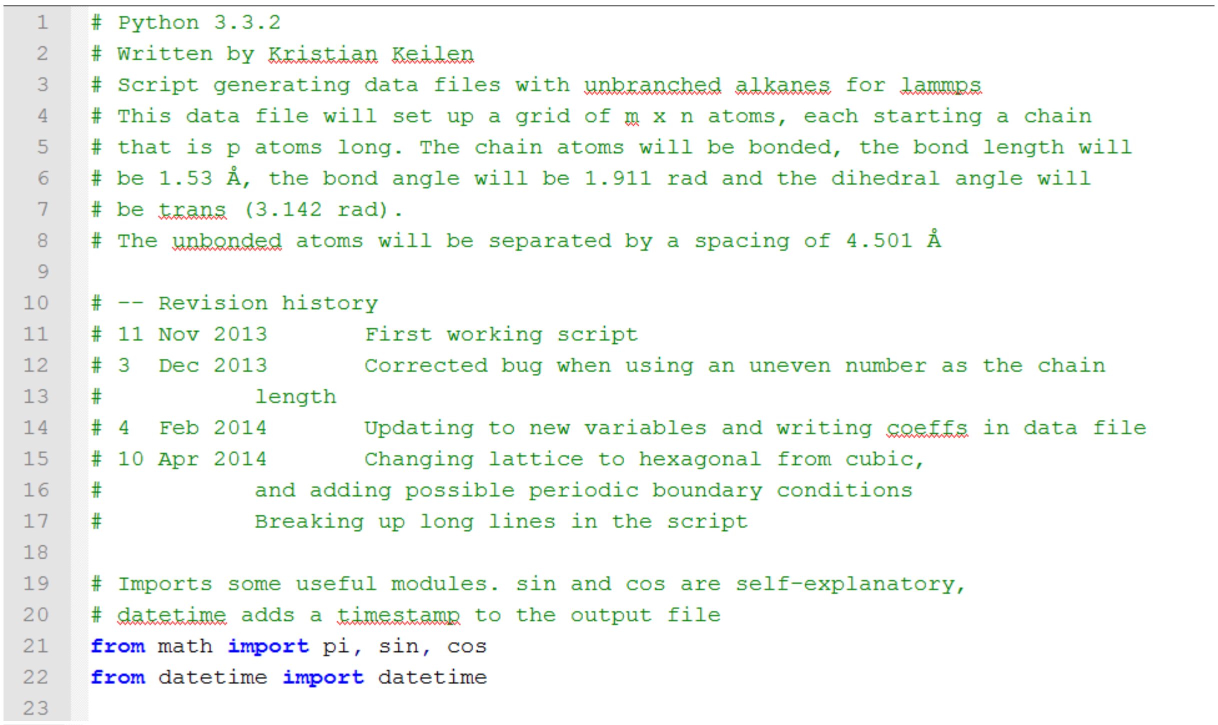
The algorithm for calculating the potential energy for a single atom in an infinite hexagonal lattice is given in appendix 4.5, and visualized in figure Figure 59, page 84.

4 A selection of scripts

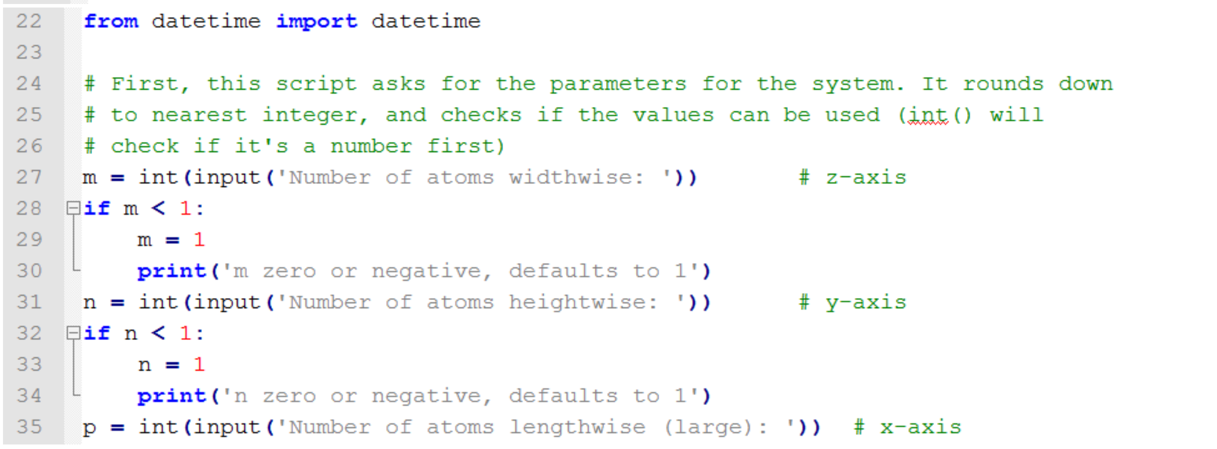
Almost every script written for this thesis has been uploaded to DAIM as supplementary material. The scripts were put into this script as *images*, because many of the scripts used curly brackets, which conflicted with EndNote, the citation software for this thesis. These pictures are not all bad, as we do get nice line numbering and color formatting out of it.

4.1 Crystal-generating script (Python)

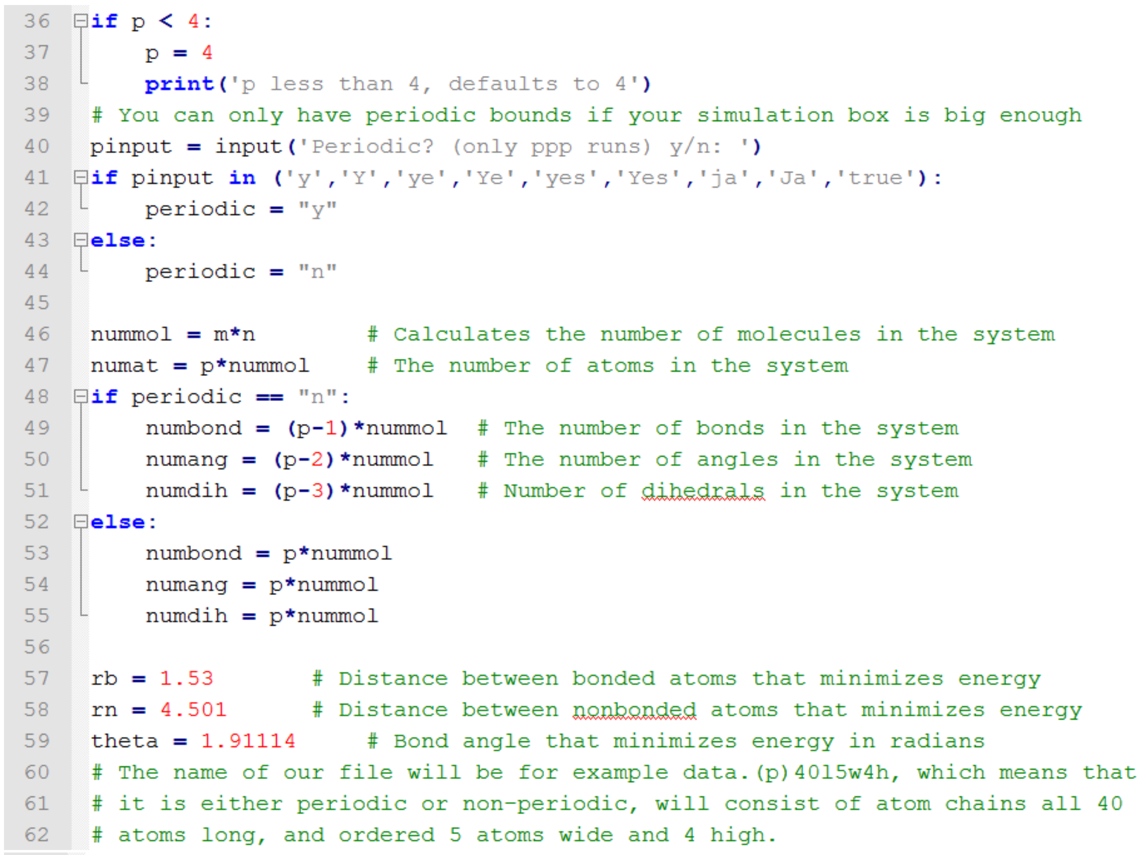
The following script is the oldest part of this thesis, and it was first written for the master project. It writes a data file (Part C, section 3.1 and Part D, section 1) that LAMMPS can interpret as an initial configuration (Part C, section 4). This script asks for the geometry of the system (number of molecules in height and width, length of molecules in number of atoms) as well as if bonds should wrap around over the periodic boundary to produce a closed molecule.



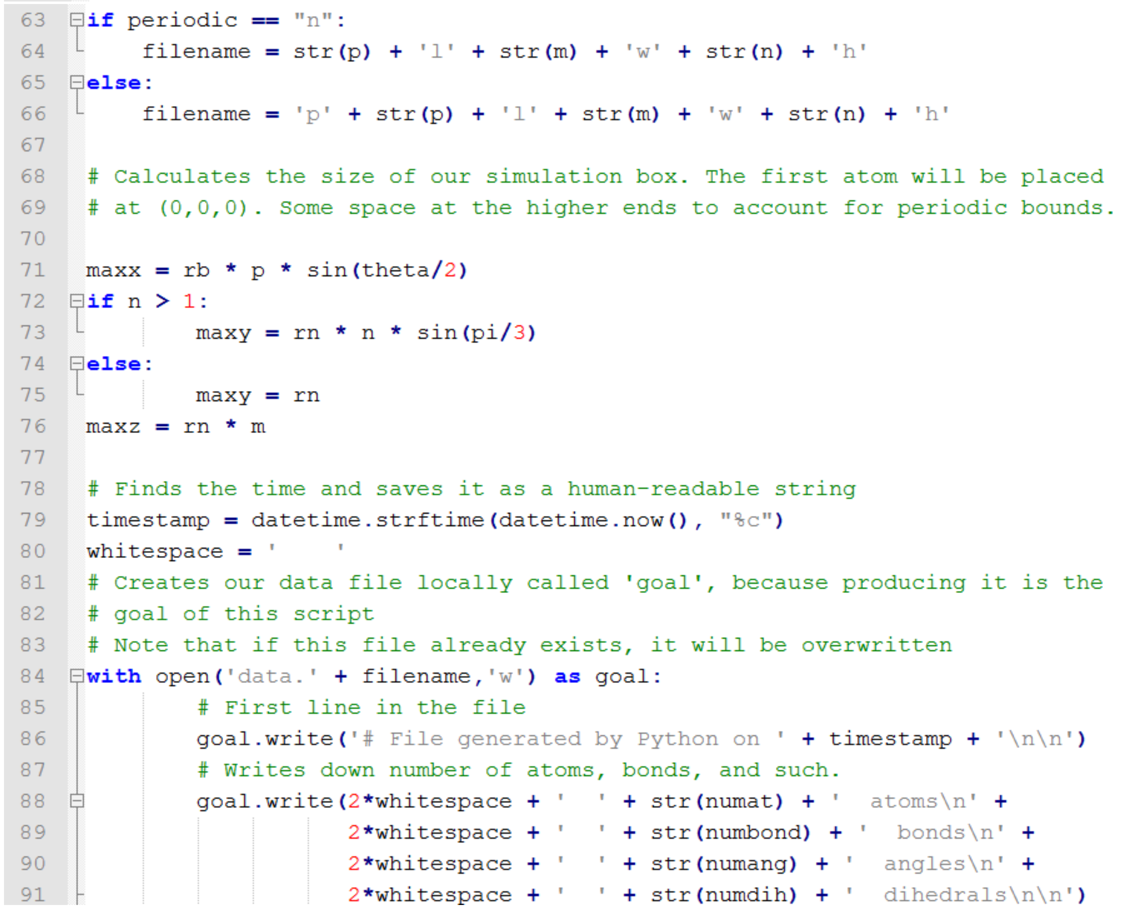
```
# Python 3.3.2
# Written by Kristian Keilen
# Script generating data files with unbranched alkanes for lammps
# This data file will set up a grid of m x n atoms, each starting a chain
# that is p atoms long. The chain atoms will be bonded, the bond length will
# be 1.53 Å, the bond angle will be 1.911 rad and the dihedral angle will
# be trans (3.142 rad).
# The unbonded atoms will be separated by a spacing of 4.501 Å

# -- Revision history
# 11 Nov 2013       First working script
# 3  Dec 2013       Corrected bug when using an uneven number as the chain
#           length
# 4  Feb 2014       Updating to new variables and writing coeffs in data file
# 10 Apr 2014       Changing lattice to hexagonal from cubic,
#           and adding possible periodic boundary conditions
#           Breaking up long lines in the script

# Imports some useful modules. sin and cos are self-explanatory,
# datetime adds a timestamp to the output file
from math import pi, sin, cos
from datetime import datetime

```



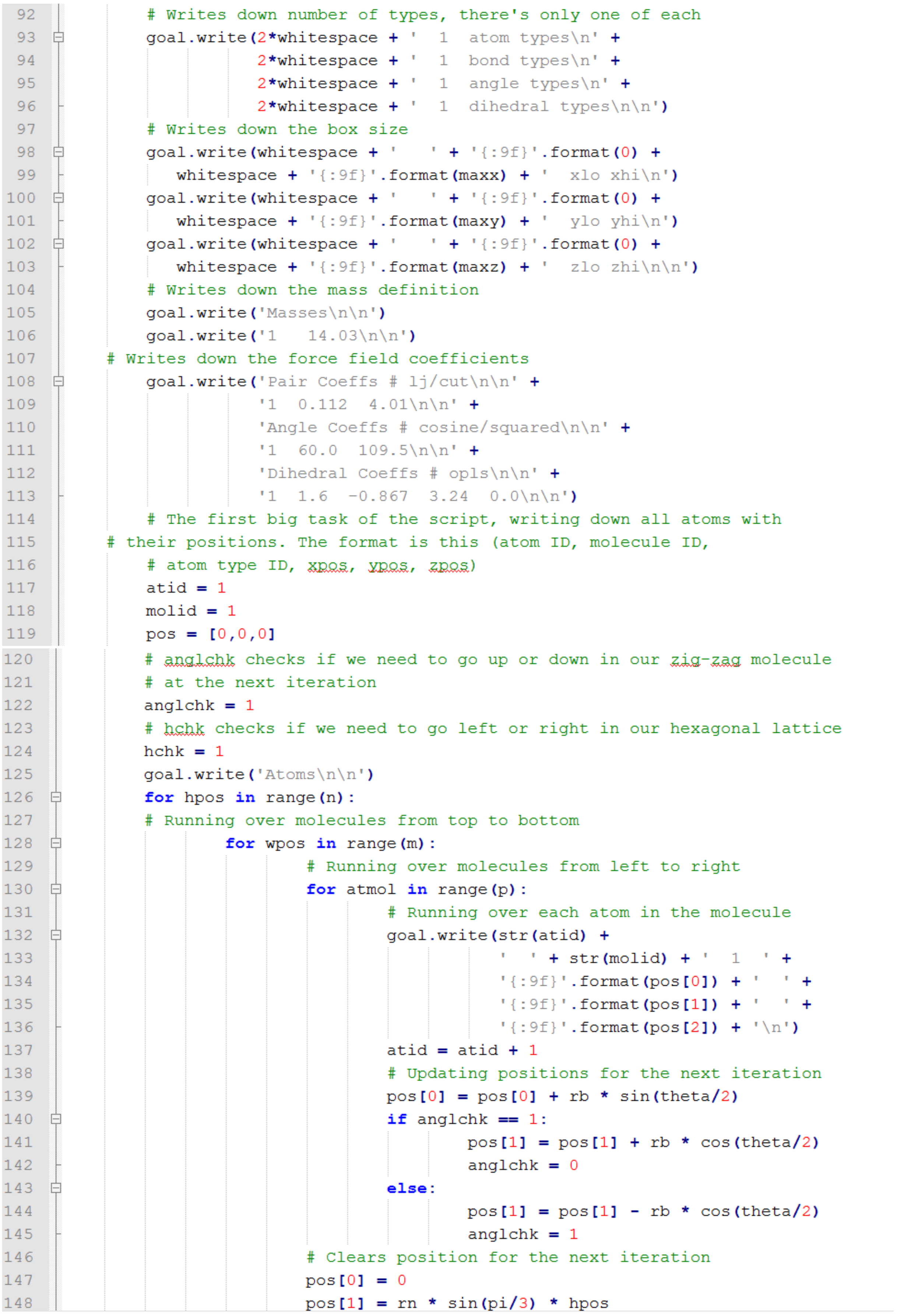
```
from datetime import datetime

# First, this script asks for the parameters for the system. It rounds down
# to nearest integer, and checks if the values can be used (int() will
# check if it's a number first)
m = int(input('Number of atoms widthwise: '))       # z-axis
if m < 1:
    m = 1
    print('m zero or negative, defaults to 1')
n = int(input('Number of atoms heightwise: '))      # y-axis
if n < 1:
    n = 1
    print('n zero or negative, defaults to 1')
p = int(input('Number of atoms lengthwise (large): '))  # x-axis
```



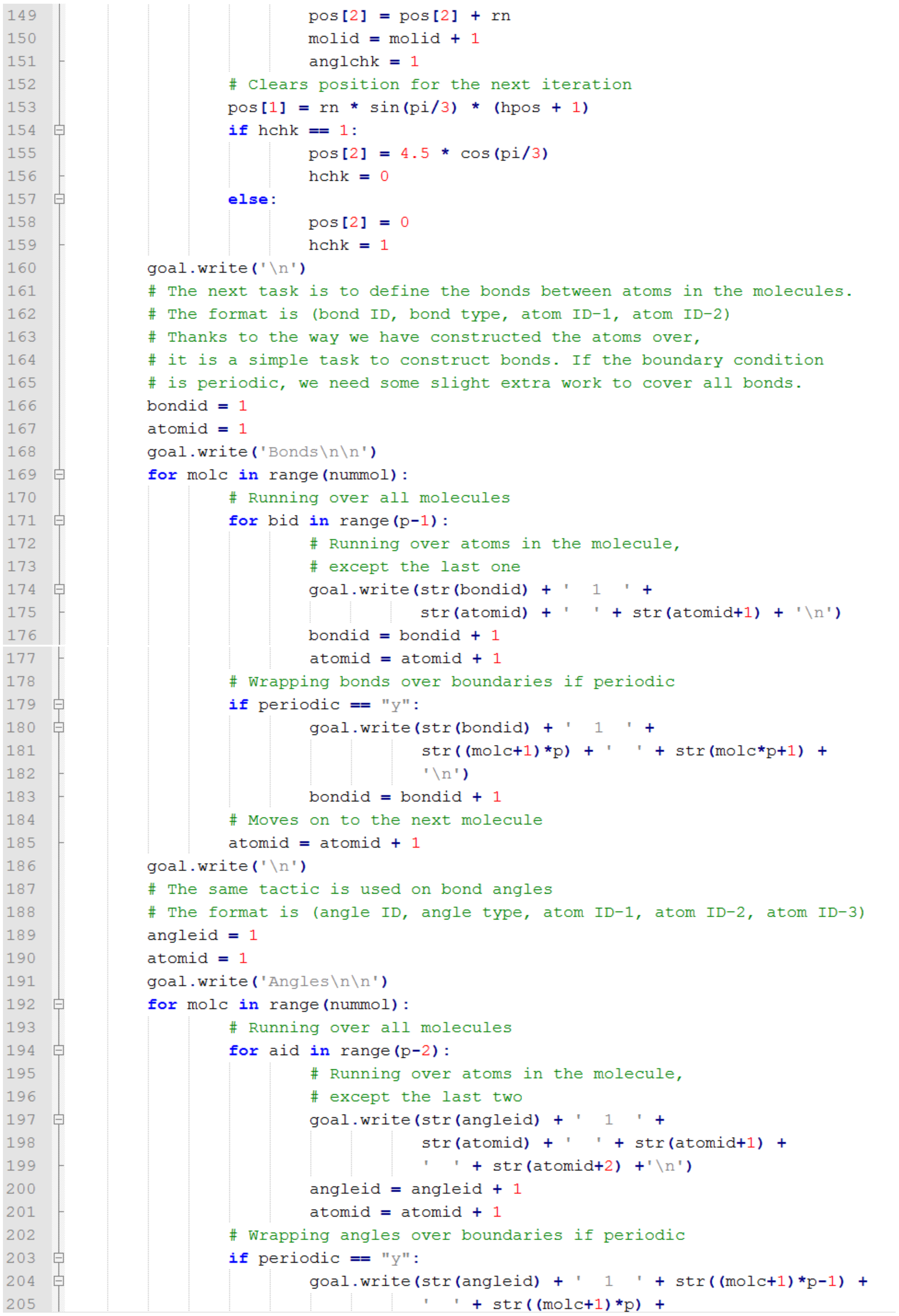
```python
if p < 4:
    p = 4
    print('p less than 4, defaults to 4')
# You can only have periodic bounds if your simulation box is big enough
pinput = input('Periodic? (only ppp runs) y/n: ')
if pinput in ('y','Y','ye','Ye','yes','Yes','ja','Ja','true'):
    periodic = "y"
else:
    periodic = "n"

nummol = m*n        # Calculates the number of molecules in the system
numat = p*nummol    # The number of atoms in the system
if periodic == "n":
    numbond = (p-1)*nummol  # The number of bonds in the system
    numang = (p-2)*nummol   # The number of angles in the system
    numdih = (p-3)*nummol   # Number of dihedrals in the system
else:
    numbond = p*nummol
    numang = p*nummol
    numdih = p*nummol

rb = 1.53       # Distance between bonded atoms that minimizes energy
rn = 4.501      # Distance between nonbonded atoms that minimizes energy
theta = 1.91114     # Bond angle that minimizes energy in radians
# The name of our file will be for example data.(p)40l5w4h, which means that
# it is either periodic or non-periodic, will consist of atom chains all 40
# atoms long, and ordered 5 atoms wide and 4 high.
```



```python
if periodic == "n":
    filename = str(p) + 'l' + str(m) + 'w' + str(n) + 'h'
else:
    filename = 'p' + str(p) + 'l' + str(m) + 'w' + str(n) + 'h'

# Calculates the size of our simulation box. The first atom will be placed
# at (0,0,0). Some space at the higher ends to account for periodic bounds.

maxx = rb * p * sin(theta/2)
if n > 1:
        maxy = rn * n * sin(pi/3)
else:
        maxy = rn
maxz = rn * m

# Finds the time and saves it as a human-readable string
timestamp = datetime.strftime(datetime.now(), "%c")
whitespace = '    '
# Creates our data file locally called 'goal', because producing it is the
# goal of this script
# Note that if this file already exists, it will be overwritten
with open('data.' + filename,'w') as goal:
        # First line in the file
        goal.write('# File generated by Python on ' + timestamp + '\n\n')
        # Writes down number of atoms, bonds, and such.
        goal.write(2*whitespace + '  ' + str(numat) + '  atoms\n' +
                   2*whitespace + '  ' + str(numbond) + '  bonds\n' +
                   2*whitespace + '  ' + str(numang) + '  angles\n' +
                   2*whitespace + '  ' + str(numdih) + '  dihedrals\n\n')
```



```
        # Writes down number of types, there's only one of each
        goal.write(2*whitespace + '  1  atom types\n' +
                   2*whitespace + '  1  bond types\n' +
                   2*whitespace + '  1  angle types\n' +
                   2*whitespace + '  1  dihedral types\n\n')
        # Writes down the box size
        goal.write(whitespace + '    ' + '{:9f}'.format(0) +
           whitespace + '{:9f}'.format(maxx) + '  xlo xhi\n')
        goal.write(whitespace + '    ' + '{:9f}'.format(0) +
           whitespace + '{:9f}'.format(maxy) + '  ylo yhi\n')
        goal.write(whitespace + '    ' + '{:9f}'.format(0) +
           whitespace + '{:9f}'.format(maxz) + '  zlo zhi\n\n')
        # Writes down the mass definition
        goal.write('Masses\n\n')
        goal.write('1   14.03\n\n')
    # Writes down the force field coefficients
        goal.write('Pair Coeffs # lj/cut\n\n' +
                   '1  0.112  4.01\n\n' +
                   'Angle Coeffs # cosine/squared\n\n' +
                   '1  60.0  109.5\n\n' +
                   'Dihedral Coeffs # opls\n\n' +
                   '1  1.6  -0.867  3.24  0.0\n\n')
        # The first big task of the script, writing down all atoms with
    # their positions. The format is this (atom ID, molecule ID,
        # atom type ID, xpos, ypos, zpos)
        atid = 1
        molid = 1
        pos = [0,0,0]
        # anglchk checks if we need to go up or down in our zig-zag molecule
        # at the next iteration
        anglchk = 1
        # hchk checks if we need to go left or right in our hexagonal lattice
        hchk = 1
        goal.write('Atoms\n\n')
        for hpos in range(n):
        # Running over molecules from top to bottom
                for wpos in range(m):
                        # Running over molecules from left to right
                        for atmol in range(p):
                                # Running over each atom in the molecule
                                goal.write(str(atid) +
                                           '  ' + str(molid) + '  1  ' +
                                           '{:9f}'.format(pos[0]) + '  ' +
                                           '{:9f}'.format(pos[1]) + '  ' +
                                           '{:9f}'.format(pos[2]) + '\n')
                                atid = atid + 1
                                # Updating positions for the next iteration
                                pos[0] = pos[0] + rb * sin(theta/2)
                                if anglchk == 1:
                                        pos[1] = pos[1] + rb * cos(theta/2)
                                        anglchk = 0
                                else:
                                        pos[1] = pos[1] - rb * cos(theta/2)
                                        anglchk = 1
                        # Clears position for the next iteration
                        pos[0] = 0
                        pos[1] = rn * sin(pi/3) * hpos
```

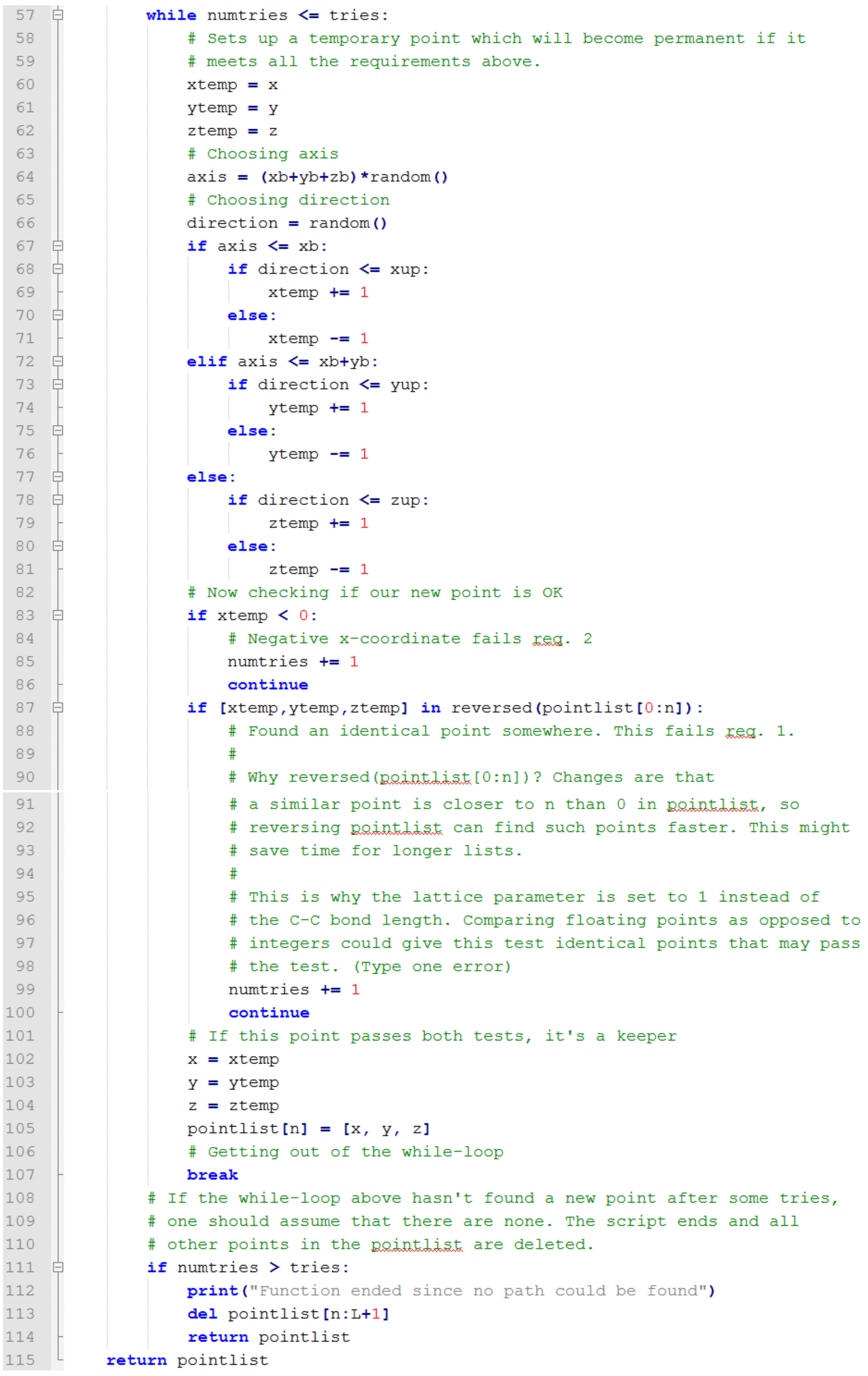


```
                        pos[2] = pos[2] + rn
                        molid = molid + 1
                        anglchk = 1
                # Clears position for the next iteration
                pos[1] = rn * sin(pi/3) * (hpos + 1)
                if hchk == 1:
                        pos[2] = 4.5 * cos(pi/3)
                        hchk = 0
                else:
                        pos[2] = 0
                        hchk = 1
        goal.write('\n')
        # The next task is to define the bonds between atoms in the molecules.
        # The format is (bond ID, bond type, atom ID-1, atom ID-2)
        # Thanks to the way we have constructed the atoms over,
        # it is a simple task to construct bonds. If the boundary condition
        # is periodic, we need some slight extra work to cover all bonds.
        bondid = 1
        atomid = 1
        goal.write('Bonds\n\n')
        for molc in range(nummol):
                # Running over all molecules
                for bid in range(p-1):
                        # Running over atoms in the molecule,
                        # except the last one
                        goal.write(str(bondid) + '  1  ' +
                                   str(atomid) + '   ' + str(atomid+1) + '\n')
                        bondid = bondid + 1
                        atomid = atomid + 1
                # Wrapping bonds over boundaries if periodic
                if periodic == "y":
                        goal.write(str(bondid) + '  1  ' +
                                   str((molc+1)*p) + '   ' + str(molc*p+1) +
                                   '\n')
                        bondid = bondid + 1
                # Moves on to the next molecule
                atomid = atomid + 1
        goal.write('\n')
        # The same tactic is used on bond angles
        # The format is (angle ID, angle type, atom ID-1, atom ID-2, atom ID-3)
        angleid = 1
        atomid = 1
        goal.write('Angles\n\n')
        for molc in range(nummol):
                # Running over all molecules
                for aid in range(p-2):
                        # Running over atoms in the molecule,
                        # except the last two
                        goal.write(str(angleid) + '  1  ' +
                                   str(atomid) + '   ' + str(atomid+1) +
                                   '   ' + str(atomid+2) +'\n')
                        angleid = angleid + 1
                        atomid = atomid + 1
                # Wrapping angles over boundaries if periodic
                if periodic == "y":
                        goal.write(str(angleid) + '  1  ' + str((molc+1)*p-1) +
                                   '   ' + str((molc+1)*p) +
                                   '  ' + str(molc*p+1) +'\n')
                        goal.write(str(angleid+1) + '  1  ' +
                                   str((molc+1)*p) + '  ' +
                                   str(molc*p+1) + '  ' + str(molc*p+2) +'\n')
                        angleid = angleid + 2
                # Moves on to the next molecule
                atomid = atomid + 2
        goal.write('\n')
        # And finally, the dihedrals
        # The format is (dihedral ID, dihedral type,
        # atom ID-1, atom ID-2, atom ID-3, atom ID-4)
        dihedid = 1
        atomid = 1
        goal.write('Dihedrals\n\n')
        for molc in range(nummol):
                # Running over all molecules
                for aid in range(p-3):
                        # Running over atoms in the molecule,
                        # except the last three
                        goal.write(str(dihedid) + '  1  ' +
                                   str(atomid) + '  ' + str(atomid+1) +
                                   '  ' + str(atomid+2) + '  ' +
                                   str(atomid+3) +'\n')
                        dihedid = dihedid + 1
                        atomid = atomid + 1
                # Wrapping dihedrals over boundaries if periodic
                if periodic == "y":
                        goal.write(str(dihedid) + '  1  ' +
                                   str((molc+1)*p-2) + '  ' +
                                   str((molc+1)*p-1) + '  ' + str(molc*p+1) +
                                    '  ' + str(molc*p+2) +'\n')
                        goal.write(str(dihedid+1) + '  1  ' +
                                   str((molc+1)*p-1) + '  ' + str((molc+1)*p) +
                                   '  ' + str(molc*p+1) + '  ' +
                                   str(molc*p+2) +'\n')
                        goal.write(str(dihedid+2) +
                                   '  1  ' + str((molc+1)*p) + '  ' +
                                   str(molc*p+1) + '  ' + str(molc*p+2) + '  ' +
                                   str(molc*p+3) +'\n')
                        dihedid = dihedid + 3
                # Moves on to the next molecule
                atomid = atomid + 3
        goal.write('\n')
# End of script
```

4.2 Monte Carlo polymer (Python)

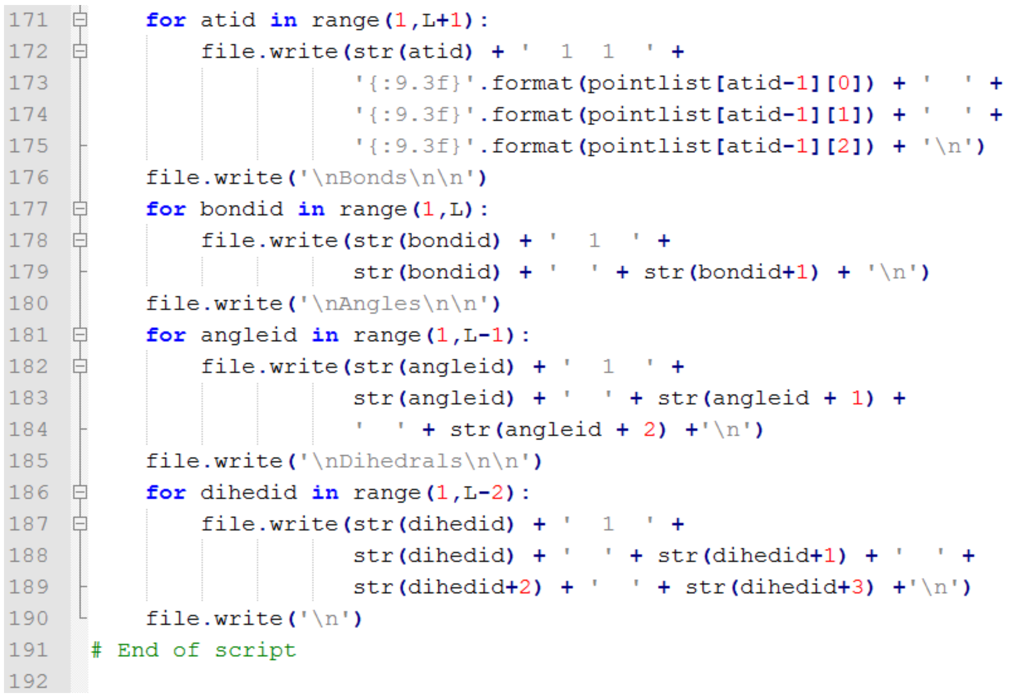
This script build the self-avoiding walk described in Part C, section 4.2, and produces a data file containing a single polymer chain. This is a long script, but a lot of it is commentaries.

```python
# buildbsaw.py
# Kristian Keilen
# February 2014
# Builds a lammps-readable polymer data file based on a biased self-avoiding
# walk, to give some randomness to the polymer configuration. The structure
# is in no way energy minimized, so run it through in.justmini first.

bond_length = 1.53
###### BUILDING A PATH #######
# The first part of the code defines the function that constructs our
# random walk. This one will be called later. The parameter = argument is a list
# of default settings, which means that you can call on the function bsaw(L)
# instead of bsaw(L,xa,ya,za,xb,yb,zb,tries)
def bsaw(L, xa = 0.4, ya = 0, za = 0, xb = 1, yb = 1, zb = 1, tries = 15):
    """Returns a biased self-avoiding walk on a cubic lattice
with a lattice parameter of 1 and maximal length L"""
    # This constructs a list of points of the format (x, y, z),
    # These are the coordinates for the point.
    # For a cubic lattice, the walk is constrained along the three axes.
    # The following restrictions apply
    # 1. Only one point is allowed at each lattice point, if the walk
    #    visits the same point twice, the loop is done over again.
    #    If  the number of retries gets larger than "tries" chances are that
    #    one is stuck in a "bag", and the walk ends.
    # 2. The x-coordinate must be non-negative. If a negative x is spotted,
    #    the loop is done over.
    #
    # The 'a' biases take on values between -1 and 1, except for xa
    # which only takes on values between 0 and 1. A bias of 0 means that
    # the walk has a equal chance of going up or down the lattice. A bias of
    # 0.4 means that the probability of going up is 0.7 and down is 0.3,
    # effectively the difference between going up and down is 0.4.
    #
    # Similarly the 'b' biases which axis will be chosen. They will add up
    # and partition some number. A random number between 0 and xb+yb+zb will
    # be generated. If it falls between xb and xb+yb it will walk along the
    # y-axis and so on.
    # Since this function is built in this code, I will not bother with
    # checking if the parameters are OK.
    from random import random
    n = 0
    x = 0
    y = 0
    z = 0
    pointlist = [[0,0,0] for var in range(L)]
    # For a user familiar with matlab, think of the above line as equivalent to
    # pointlist = zeros(L,3), where python interpretes the matrix as
    # [[0,0,0],[0,0,0],[0,0,0]...,[0,0,0]]. We can at any time extract a single
    # number from pointlist by writing pointlist[45][1] or so.

    # Probabilities of going "up" in the lattice
    xup = 0.5*(1 + xa)
    yup = 0.5*(1 + ya)
    zup = 0.5*(1 + za)
    for n in range(1,L):
        numtries = 0
```



```
        while numtries <= tries:
            # Sets up a temporary point which will become permanent if it
            # meets all the requirements above.
            xtemp = x
            ytemp = y
            ztemp = z
            # Choosing axis
            axis = (xb+yb+zb)*random()
            # Choosing direction
            direction = random()
            if axis <= xb:
                if direction <= xup:
                    xtemp += 1
                else:
                    xtemp -= 1
            elif axis <= xb+yb:
                if direction <= yup:
                    ytemp += 1
                else:
                    ytemp -= 1
            else:
                if direction <= zup:
                    ztemp += 1
                else:
                    ztemp -= 1
            # Now checking if our new point is OK
            if xtemp < 0:
                # Negative x-coordinate fails req. 2
                numtries += 1
                continue
            if [xtemp,ytemp,ztemp] in reversed(pointlist[0:n]):
                # Found an identical point somewhere. This fails req. 1.
                #
                # Why reversed(pointlist[0:n])? Changes are that
                # a similar point is closer to n than 0 in pointlist, so
                # reversing pointlist can find such points faster. This might
                # save time for longer lists.
                #
                # This is why the lattice parameter is set to 1 instead of
                # the C-C bond length. Comparing floating points as opposed to
                # integers could give this test identical points that may pass
                # the test. (Type one error)
                numtries += 1
                continue
            # If this point passes both tests, it's a keeper
            x = xtemp
            y = ytemp
            z = ztemp
            pointlist[n] = [x, y, z]
            # Getting out of the while-loop
            break
        # If the while-loop above hasn't found a new point after some tries,
        # one should assume that there are none. The script ends and all
        # other points in the pointlist are deleted.
        if numtries > tries:
            print("Function ended since no path could be found")
            del pointlist[n:L+1]
            return pointlist
    return pointlist
```

```python
# End of function
from datetime import datetime
L = int(input('Write down path length: '))

# Calling our function
pointlist = bsaw(L)
# In case the program ended before the path was completed, let's redefine
# L to be the length of the pointlist.
L = len(pointlist)
filename = 'bsaw' + str(L)
# Scalar-matrix multiplication is a bit tricky in python. bond_length was
# defined at the top of the script.
pointlist = [[bond_length*pointlist[i][j] for j in range(3)]
             for i in range(L)]
# Axis decomposition, axdeco[0] for example is an ordered list of all the
# x-coordinates. Think of it as a matrix transpose.
axdeco = list(zip(*pointlist))
# Finding the maximal and minimal value of each axis for the box
maxcoords = [max(axdeco[i]) for i in range(3)]
mincoords = [min(axdeco[i]) for i in range(3)]

##### BUILDING THE DATA FILE ####
# This will be built in a similar manner to builddata.py. The differences are
# that we must import the points from our pointlist instead of building, and
# that we are only dealing with one molecule now. This simplifies the
# procedures somewhat
timestamp = datetime.strftime(datetime.now(), "%c")
whitespace = '    '
with open('data.' + filename,'w') as file:
    file.write('# File generated by Python on ' + timestamp + '\n\n')
    file.write(2*whitespace + '  ' + str(L) + '  atoms\n' +
               2*whitespace + '  ' + str(L-1) + '  bonds\n' +
               2*whitespace + '  ' + str(L-2) + '  angles\n' +
               2*whitespace + '  ' + str(L-3) + '  dihedrals\n\n')
    file.write(2*whitespace + '  1  atom types\n' +
               2*whitespace + '  1  bond types\n' +
               2*whitespace + '  1  angle types\n' +
               2*whitespace + '  1  dihedral types\n\n')
    file.write(whitespace + '   ' + '{:9.3f}'.format(mincoords[0]) +
               whitespace + '{:9.3f}'.format(maxcoords[0]) + '  xlo xhi\n')
    file.write(whitespace + '   ' + '{:9.3f}'.format(mincoords[1]) +
               whitespace + '{:9.3f}'.format(maxcoords[1]) + '  ylo yhi\n')
    file.write(whitespace + '   ' + '{:9.3f}'.format(mincoords[2]) +
               whitespace + '{:9.3f}'.format(maxcoords[2]) + '  zlo zhi\n\n')
    file.write('Masses\n\n' +
               '1  14.03\n\n')
    file.write('Pair Coeffs # lj/cut\n\n' +
               '1  0.112  4.01\n\n' +
               'Angle Coeffs # cosine/squared\n\n' +
               '1  60.0  109.5\n\n' +
               'Dihedral Coeffs # opls\n\n' +
               '1  1.6  -0.867  3.24  0.0\n\n')
    # What's left now is writing down the position of atoms and defining bonds,
    # angles and dihedrals. Note how easy this is with only one molecule!
    file.write('Atoms\n\n')
```

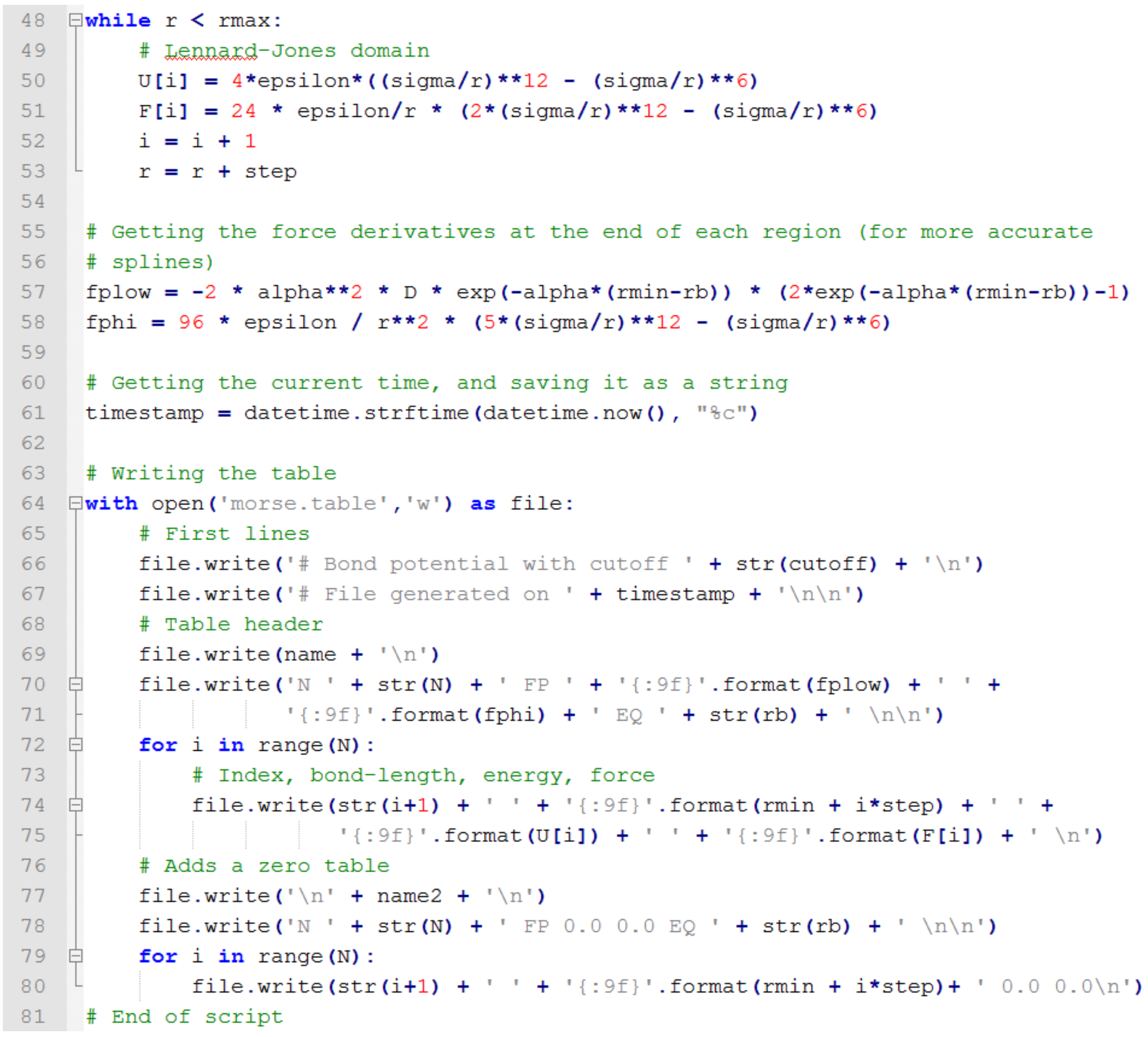


```
    for atid in range(1,L+1):
        file.write(str(atid) + '  1  1  ' +
                   '{:9.3f}'.format(pointlist[atid-1][0]) + '  ' +
                   '{:9.3f}'.format(pointlist[atid-1][1]) + '  ' +
                   '{:9.3f}'.format(pointlist[atid-1][2]) + '\n')
    file.write('\nBonds\n\n')
    for bondid in range(1,L):
        file.write(str(bondid) + '  1  ' +
                   str(bondid) + '  ' + str(bondid+1) + '\n')
    file.write('\nAngles\n\n')
    for angleid in range(1,L-1):
        file.write(str(angleid) + '  1  ' +
                   str(angleid) + '  ' + str(angleid + 1) +
                   '  ' + str(angleid + 2) +'\n')
    file.write('\nDihedrals\n\n')
    for dihedid in range(1,L-2):
        file.write(str(dihedid) + '  1  ' +
                   str(dihedid) + '  ' + str(dihedid+1) + '  ' +
                   str(dihedid+2) + '  ' + str(dihedid+3) +'\n')
    file.write('\n')
# End of script

```

4.3 Potential table builder (Python)

This script builds tabulated values for the bond length potential, as explained in Part C, section 2.12.3. The input is the Morse cutoff (maximal bond length), but other parameters may be changed in the source code.

```python
# Python 3.3.2
# Written by Kristian Keilen

# This script creates a table of N lines with a Morse potential for bond lengths
# less than cutoff and a LJ potential for larger bond lengts. This is meant to
# simulate bond breaking.

# The base parameters are given here, in case one wants to change them
rb = 1.53          # Base bond length

alpha = 2.0        # Morse parameters
D = 88             #

sigma = 4.01       # Lennard-Jones parameters
epsilon = 0.112 #

rmin = 0.0         # Min extent of table
rmax = 10.0        # Max extent of table, cutoff for the pair potential
N = 1000           # Table length, should be similar to one in the run file
                   # The table starts from rmin and goes up steps towards rmax,
                   # which will not actually be in the table.
name  = 'BREAK'    # The name of our table, also referenced from the run file
name2= 'ZERO'

from math import log, exp
from datetime import datetime
cutoff = float(input('Cutoff distance: '))
try:
    # Quick check to see if the input is valid, otherwise, the run is killed
    a = log(cutoff - rb)
except:
    raise RuntimeError("Too small cutoff")

step = (rmax - rmin)/N

# Building the arrays for both potential and force. We start with zero arrays
U = [0] * (N+1)
F = [0] * (N+1)
r = rmin
i = 0

while r < cutoff:
    # Morse domain
    U[i] = D*(1-exp(-alpha*(r-rb)))**2
    F[i] = -2*alpha*D*exp(-alpha*(r-rb))*(1-exp(-alpha*(r-rb)))
    i = i + 1
    r = r + step
```

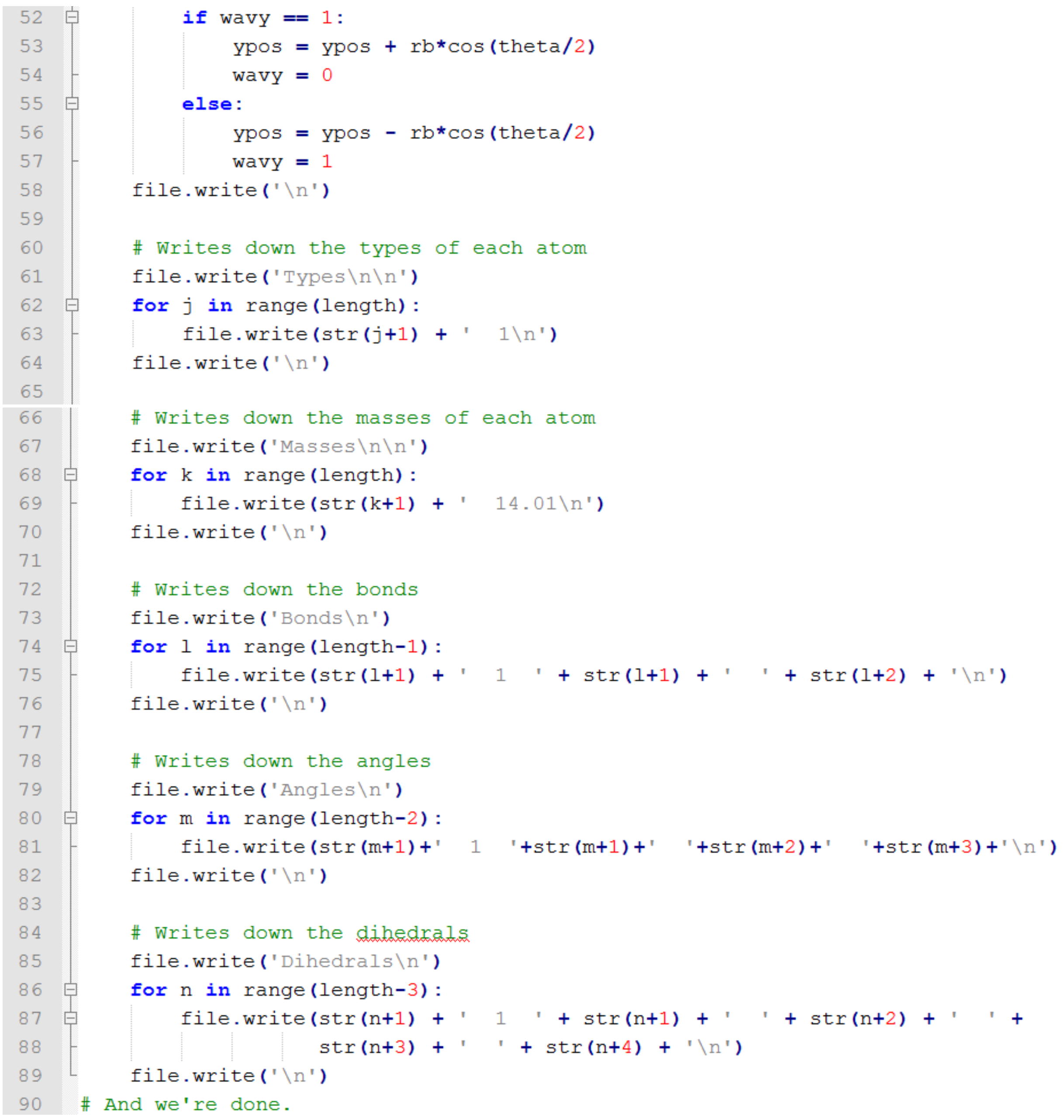


```python
while r < rmax:
    # Lennard-Jones domain
    U[i] = 4*epsilon*((sigma/r)**12 - (sigma/r)**6)
    F[i] = 24 * epsilon/r * (2*(sigma/r)**12 - (sigma/r)**6)
    i = i + 1
    r = r + step

# Getting the force derivatives at the end of each region (for more accurate
# splines)
fplow = -2 * alpha**2 * D * exp(-alpha*(rmin-rb)) * (2*exp(-alpha*(rmin-rb))-1)
fphi = 96 * epsilon / r**2 * (5*(sigma/r)**12 - (sigma/r)**6)

# Getting the current time, and saving it as a string
timestamp = datetime.strftime(datetime.now(), "%c")

# Writing the table
with open('morse.table','w') as file:
    # First lines
    file.write('# Bond potential with cutoff ' + str(cutoff) + '\n')
    file.write('# File generated on ' + timestamp + '\n\n')
    # Table header
    file.write(name + '\n')
    file.write('N ' + str(N) + ' FP ' + '{:9f}'.format(fplow) + ' ' +
               '{:9f}'.format(fphi) + ' EQ ' + str(rb) + ' \n\n')
    for i in range(N):
        # Index, bond-length, energy, force
        file.write(str(i+1) + ' ' + '{:9f}'.format(rmin + i*step) + ' ' +
                   '{:9f}'.format(U[i]) + ' ' + '{:9f}'.format(F[i]) + ' \n')
    # Adds a zero table
    file.write('\n' + name2 + '\n')
    file.write('N ' + str(N) + ' FP 0.0 0.0 EQ ' + str(rb) + ' \n\n')
    for i in range(N):
        file.write(str(i+1) + ' ' + '{:9f}'.format(rmin + i*step)+ ' 0.0 0.0\n')
# End of script
```

4.4 Alkane builder (Python)

This scripts builds a file that can be called by LAMMPS' `molecule` command. While the output file looks okay, I have not actually tried it in practice.

```python
# Python 3.3.2
# Written by Kristian Keilen
# Molecule builder

# Builds a molecule file that LAMMPS can access. Handy if you want to add
# molecules to a simulation.
# This molecule will be filled with CH2 pseudoatoms, with atomtype 1,
# bond type 1, angle type 1 and dihedral type 1.
# If you have more than one atom type in the system, beware!
from math import sin, cos
from datetime import datetime

length = int(input('Length of molecule 4-30: '))
if length < 4:
    # Too gassy
    length = 4
    print('length less than 4, defaults to 4')
elif length > 30:
    # Too bitumeny
    length = 30
    print('length larger than 30, defaults to 30')

# Geometry constants, respectively the bond length and bond angle.
rb = 1.53
theta = 1.911
# Accessing namearray[n] will give the name of the alkane of that length
namearray = ["","","","","butane","pentane","hexane","heptane","octane",
             "nonane","decane","undecane","dodecane","tridecane",
             "tetradecane","pentadecane","hexadecane","heptadecane",
             "octadecane","nonadecane","icosane","henisocane","docosane",
             "tricosane","tetracosane","pentacosane","hexacosane",
             "heptacosane","octacosane","nonacosane","triacontane"]

timestamp = datetime.strftime(datetime.now(), "%c")
with open(str(namearray[length]) + '.txt','w') as file:
    file.write(' # File generated by Python on ' + timestamp + '\n\n')
    # Writes down the number of atoms and such
    file.write(str(length)    + ' atoms\n' +
               str(length - 1)+ ' bonds\n' +
               str(length - 2)+ ' angles\n' +
               str(length - 3)+ ' dihedrals\n\n')
    # Writes down the coordinates of each atom
    file.write('Coords\n\n')
    wavy = 1
    xpos = 0
    ypos = 0
    for i in range(length):
        file.write(str(i+1) + '  ' + '{:9f}'.format(xpos)
                   + '  ' + '{:9f}'.format(ypos) +
                   '  ' + '{:9f}'.format(0) + '\n')
        xpos = xpos + rb*sin(theta/2)
```

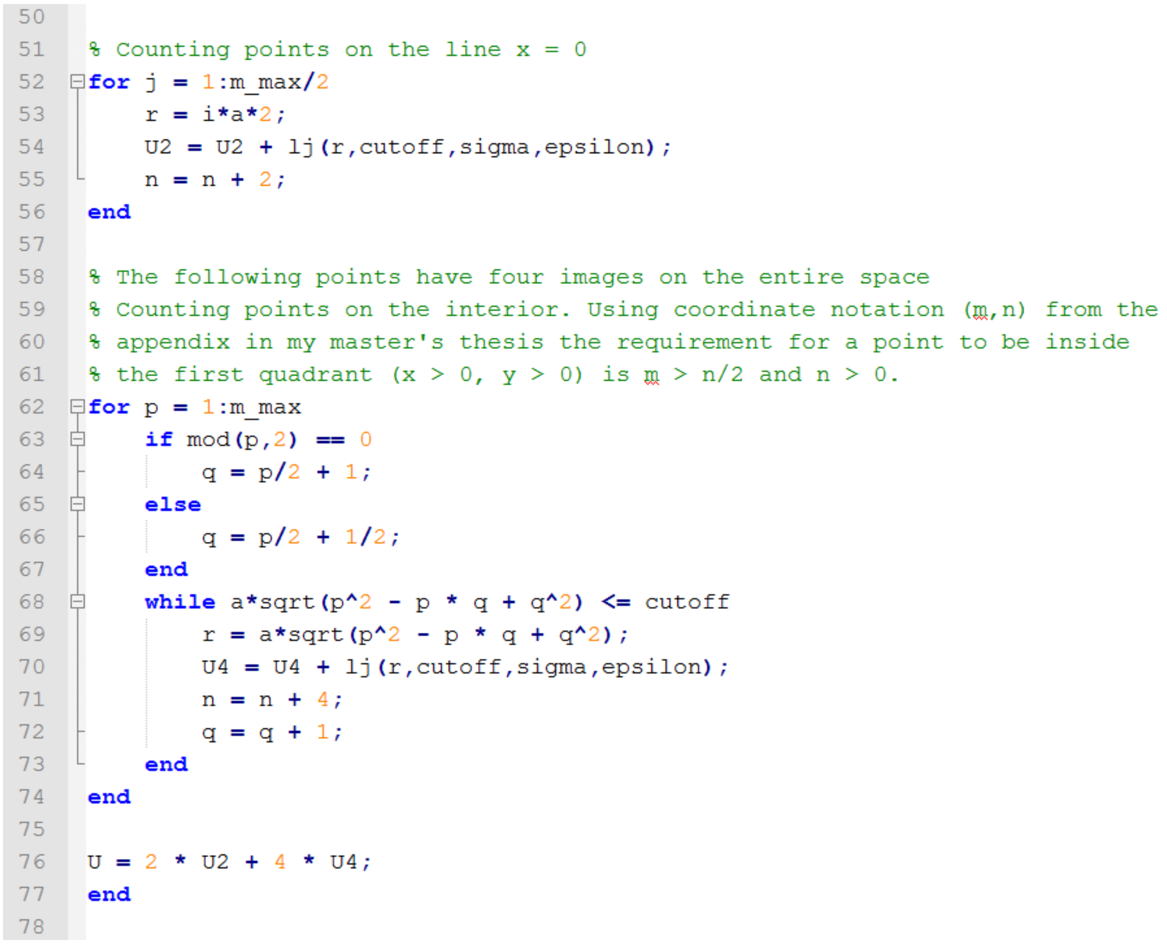


```python
        if wavy == 1:
            ypos = ypos + rb*cos(theta/2)
            wavy = 0
        else:
            ypos = ypos - rb*cos(theta/2)
            wavy = 1
    file.write('\n')

    # Writes down the types of each atom
    file.write('Types\n\n')
    for j in range(length):
        file.write(str(j+1) + '  1\n')
    file.write('\n')

    # Writes down the masses of each atom
    file.write('Masses\n\n')
    for k in range(length):
        file.write(str(k+1) + '  14.01\n')
    file.write('\n')

    # Writes down the bonds
    file.write('Bonds\n')
    for l in range(length-1):
        file.write(str(l+1) + '  1  ' + str(l+1) + '  ' + str(l+2) + '\n')
    file.write('\n')

    # Writes down the angles
    file.write('Angles\n')
    for m in range(length-2):
        file.write(str(m+1)+'  1  '+str(m+1)+'  '+str(m+2)+'  '+str(m+3)+'\n')
    file.write('\n')

    # Writes down the dihedrals
    file.write('Dihedrals\n')
    for n in range(length-3):
        file.write(str(n+1) + '  1  ' + str(n+1) + '  ' + str(n+2) + '  ' +
                   str(n+3) + '  ' + str(n+4) + '\n')
    file.write('\n')
# And we're done.
```

4.5 Cut-off distance script for hexagonal lattice (MATLAB)

There exists a variant of this script for the cubic lattice as well, and it is not hard to work out how that one looks like. See Appendix 3 for a motivation. The subroutine `lj(r,cutoff,sigma,epsilon)` calculates the Lennard-Jones 12-6 potential for two atoms with a distance r.

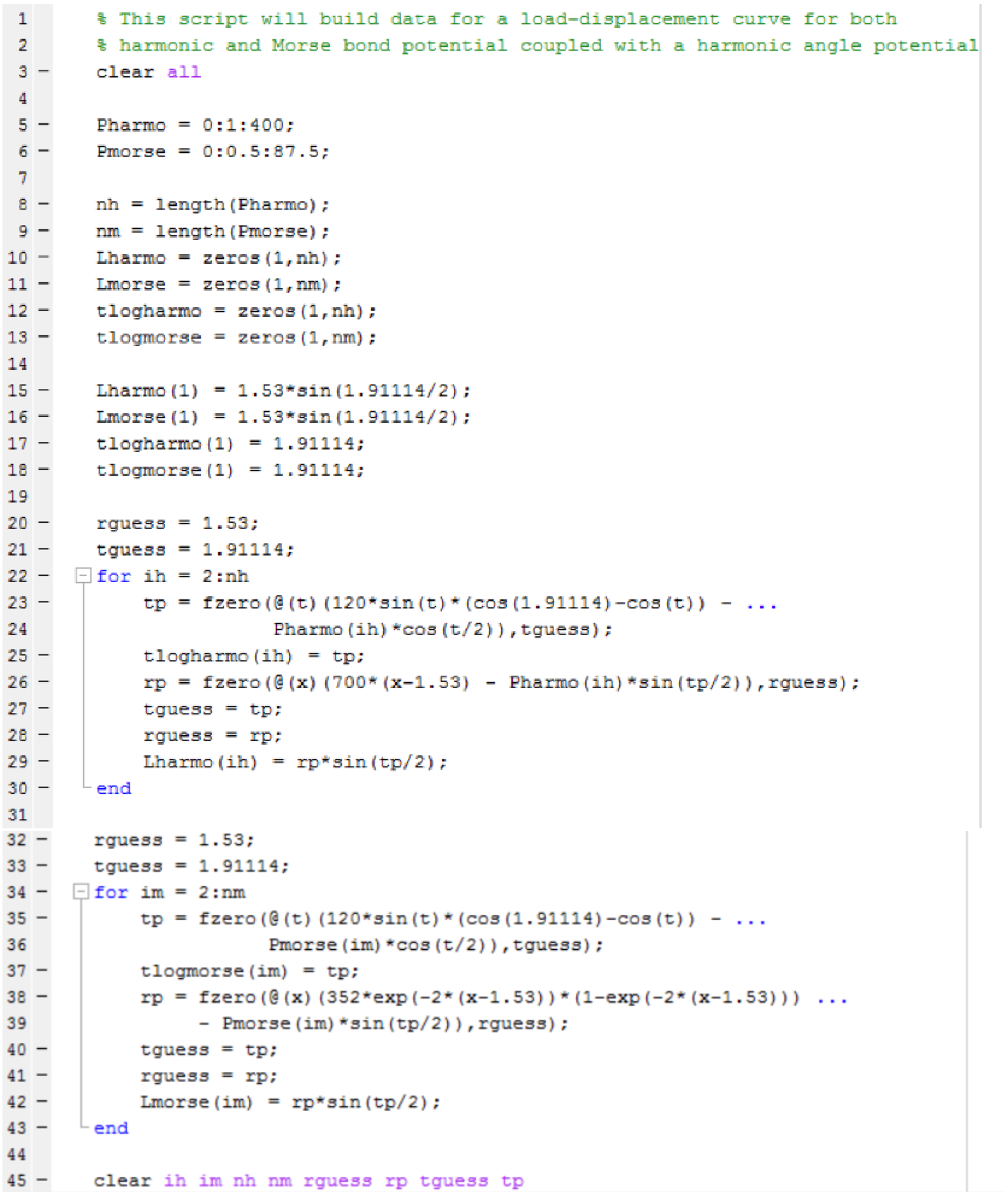
```
function [ U, n ] = hexcut( cutoff,a,sigma,epsilon )
%HEXCUT Calculates the pot. energy for an atom in a 2D hexagonal lattice
%   Returns two values, the potential energy U and the number of neighbors
%   n. Returns only U if only one value is asked for. You need to specify
%   the cutoff radius to run this, the other have default values set. The
%   cutoff radius must be scalar or matrix component for now, that means
%   that you can't enter a vector of cutoff radii and get a vector of U back

% This function counts points on the lattice bounded by the bounds
% x = 0, y = 0 and x^2 + y^2 = cutoff^2 on the first quadrant. The rest
% of the points follow by symmetry.

% Note that n is the number of neighbours for a atom, not the number of
% lattice points inside a circle like http://oeis.org/A000328 .
% The difference is 1, as the central atom is not counted.

% Sets default values
if nargin == 1
    a = 4.5;
    sigma = 4.01;
    epsilon = 0.112;
elseif nargin == 2
    sigma = 4.01;
    epsilon = 0.112;
elseif nargin == 3
    epsilon = 0.112;
end

% Treating a troublesome special case
if cutoff < a
    U = 0;
    n = 0;
    return
end

% Calculating the maximal lattice coordinate
m_max = cutoff/a;

U2 = 0;
U4 = 0;
n = 0;

% The following points have two images on the entire space
% Counting points on the line y = 0
for i = 1:m_max
    r = i*a;
    U2 = U2 + lj(r,cutoff,sigma,epsilon);
    n = n + 2;
end
```



```
% Counting points on the line x = 0
for j = 1:m_max/2
    r = i*a*2;
    U2 = U2 + lj(r,cutoff,sigma,epsilon);
    n = n + 2;
end

% The following points have four images on the entire space
% Counting points on the interior. Using coordinate notation (m,n) from the
% appendix in my master's thesis the requirement for a point to be inside
% the first quadrant (x > 0, y > 0) is m > n/2 and n > 0.
for p = 1:m_max
    if mod(p,2) == 0
        q = p/2 + 1;
    else
        q = p/2 + 1/2;
    end
    while a*sqrt(p^2 - p * q + q^2) <= cutoff
        r = a*sqrt(p^2 - p * q + q^2);
        U4 = U4 + lj(r,cutoff,sigma,epsilon);
        n = n + 4;
        q = q + 1;
    end
end

U = 2 * U2 + 4 * U4;
end
```

4.6 Theoretical Load-displacement curves (MATLAB)

This script was used to produce Figure 41 and Figure 42. Here `Pharmo` and `Pmorse` are the forces that are put upon the system, `Lharmo` and `Lmorse` are the response lengths l of a polyethylene subunit with the different potentials, and `tlogharmo` and `tlogmorse` are the angles from the angle potential. The bond length r can be calculated from equation (108).



```
% This script will build data for a load-displacement curve for both
% harmonic and Morse bond potential coupled with a harmonic angle potential
clear all

Pharmo = 0:1:400;
Pmorse = 0:0.5:87.5;

nh = length(Pharmo);
nm = length(Pmorse);
Lharmo = zeros(1,nh);
Lmorse = zeros(1,nm);
tlogharmo = zeros(1,nh);
tlogmorse = zeros(1,nm);

Lharmo(1) = 1.53*sin(1.91114/2);
Lmorse(1) = 1.53*sin(1.91114/2);
tlogharmo(1) = 1.91114;
tlogmorse(1) = 1.91114;

rguess = 1.53;
tguess = 1.91114;
for ih = 2:nh
    tp = fzero(@(t) (120*sin(t)*(cos(1.91114)-cos(t)) - ...
                Pharmo(ih)*cos(t/2)),tguess);
    tlogharmo(ih) = tp;
    rp = fzero(@(x) (700*(x-1.53) - Pharmo(ih)*sin(tp/2)),rguess);
    tguess = tp;
    rguess = rp;
    Lharmo(ih) = rp*sin(tp/2);
end

rguess = 1.53;
tguess = 1.91114;
for im = 2:nm
    tp = fzero(@(t) (120*sin(t)*(cos(1.91114)-cos(t)) - ...
                Pmorse(im)*cos(t/2)),tguess);
    tlogmorse(im) = tp;
    rp = fzero(@(x) (352*exp(-2*(x-1.53))*(1-exp(-2*(x-1.53))) ...
         - Pmorse(im)*sin(tp/2)),rguess);
    tguess = tp;
    rguess = rp;
    Lmorse(im) = rp*sin(tp/2);
end

clear ih im nh nm rguess rp tguess tp
```

4.7 Input testing (LAMMPS)

This simple program goes over the data file and checks if everything is valid. It outputs a dump-file and a picture of the system and quits.

```
variable filename string 20120w20h
################################################################################
# "Hello World"-style program to test data files. It takes a data-file containing
# atom positions and dumps an image. Replace variable in line 1 with
# any filename you want. Logs to log.lammps
#
# Kristian Keilen
#
################################################################################

# dimensions and boundary conditions

dimension   3
boundary    p s s   # Choose the type of boundary conditions to use in x-, y- and
                    # z-direction
units       real    # Distance: Å, Time: fs, Mass: g/mol Energy kCal/mol
atom_style  molecular # Sets read_data to read atoms, bonds, angles and dihedrals

# add simulation box and atoms, reads force field
pair_style  lj/cut 10.0
bond_style  table spline 1000
angle_style cosine/squared
dihedral_style multi/harmonic

read_data   data.${filename}
bond_coeff  1 morse.table BREAK
bond_coeff  2 morse.table ZERO

#neighbor   2.0 nsq  # nsq is a good neighbor style for non-periodic boxes with
                     # a small number of atoms

dump        atoms all atom 10 dumps\dump.test${filename}
dump        photo all image 50000 images\test${filename}_*.jpg element element &
            bond atom 0.2 size 1024 1024 view 35.3 -45 center &
            d 0.5 0.5 0.5 axes yes 0.1 0.02 zoom 1.4
dump_modify photo element C backcolor white

run         0
```

4.8 Energy minimization (LAMMPS)

This script takes a data file and runs two sessions of energy minimization (Part C, section 5) on it and then outputs a new data file, a dump file and three pictures.

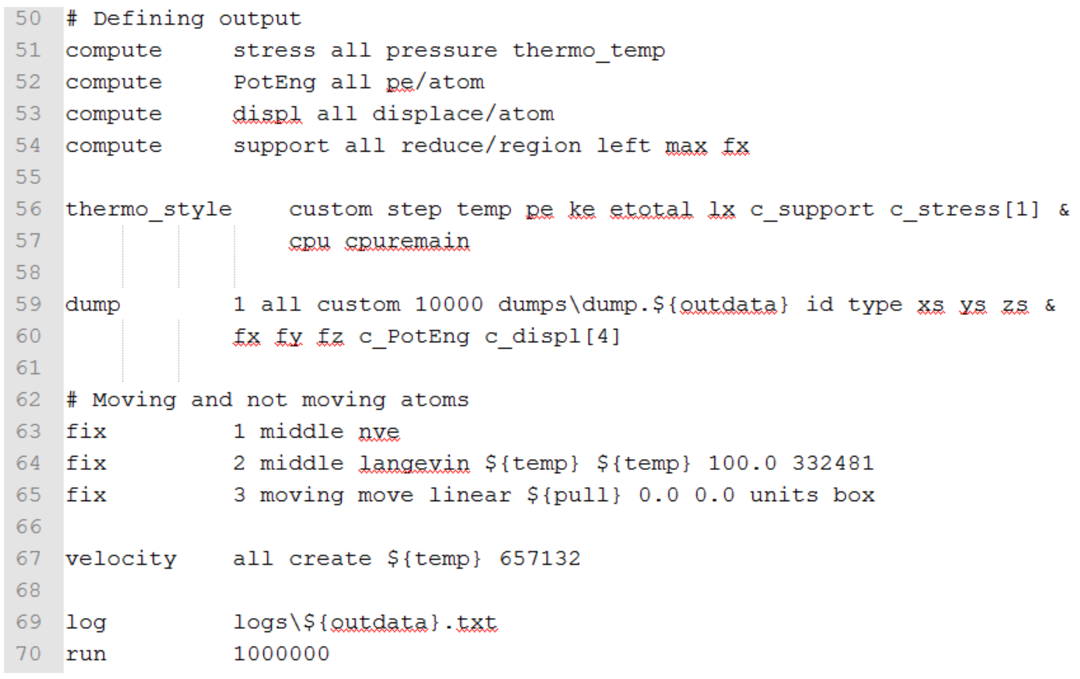
```
# in.justmini
# Kristian Keilen - Feb 2014
# LAMMPS script testing a force field by I. H. Sahputra
# This script will take any polymer structure supported by the force
# and perform two energy minimizations on it. For extra eye candy it will
# produce three images, one initial and one after each minimization.
# I recommend changing the boundary conditions to match those of the work
# script.

variable    indata string bsaw421
variable    outdata string opti_bsaw421

# Box changing factors (irrelevant for s-directions)
variable    bx equal 1.0
variable    by equal 1.0
variable    bz equal 1.0

dimension   3
boundary    s s s
units       real
atom_style  molecular

package     omp * force/neigh
suffix      omp

# Forcefield. The coefficients are given in the data and table files.
pair_style  lj/cut 10.0
bond_style  table spline 1000
angle_style cosine/squared
dihedral_style opls

read_data   data.${indata}
#neighbor   2.0 nsq # Uncomment for small data files.
bond_coeff  1 morse.table BREAK
bond_coeff  2 morse.table ZERO

dump        before all image 40000 images\${outdata}.*.jpg  &
            element element bond atom 0.2 size 4096 4096 &
            view 35.3 -45 center d 0.5 0.5 0.5 axes yes 0.1 0.02 &
            zoom 2.0
dump_modify before element C backcolor white

change_box  all x scale ${bx} y scale ${by} z scale ${by} units box
run         0


# Energy minimization
# Runs two energy minimization sessions
minimize    1.0e-6 1.0e-8 10000 100000
min_style   cg          # Polak-Ribiere conjugate gradient algorithm

minimize    1.0e-10 1.0e-12 10000 100000
min_style   hftn        # Hessian-free truncated Newton algorithm

# Saves a data file containing the optimized system. Use this one for the next run!
write_data      dumps/data.${outdata}

# Adds a dump file
dump        atoms all atom 1 dumps\dump.${outdata}
run         1
```

4.9 ppp simulation (LAMMPS)

```
# in.tensilesss
# Kristian Keilen - Apr 2014
# LAMMPS script for tensile testing an infinite-sized crystal of alkanes
# This script will deform the alkane by moving the outer atoms.
# Boundary conditions are ppp, and deformation is done with *expanding the box*
# It is assumed that an energy minimization or similar has taken place.

variable    indata string p5018w8h # The name of our indatafile (data.string)
variable    outdata string p5018w8h-300-tppp # What to name our outdata
variable    temp equal 300    # The temperature
variable    esr equal 5e-7 # Engineering strain rate

dimension   3
boundary    p p p
units       real    # Distance: Å, Time: fs, Mass: g/mol,
                    # Energy: kcal/mol, Force: kcal/mol.Å
                    # Pressure: atm, Charge: e
                    # 1 kcal/mol.Å = 7e-11 N approx.
atom_style  molecular
timestep    1
thermo      1000

# Parallelizes all the tasks underneath
package     omp * force/neigh
suffix      omp

# Forcefield (coefficients are given in the data file)
pair_style  lj/cut 10.0
#neighbor   2.0 nsq # Uncomment if LAMMPS gives you trouble about bin sizes
bond_style  table spline 1000
angle_style cosine/squared
dihedral_style opls
read_data   data.${indata}
bond_coeff  1 morse.table BREAK
bond_coeff  2 morse.table ZERO

# Builds a group of reference molecules that we will use to compute the stress
# of the system. A per-atom stress will be computed for all atoms in this group
# and averaged over the group to get a computable source.
variable    wlength equal xlo+1.5
region      window block INF ${wlength} INF INF INF INF units box
group       stressref region window

# Defining output
compute     stress all pressure thermo_temp
compute     PotEng all pe/atom
compute     displ all displace/atom
compute     stress2 stressref stress/atom NULL
compute     refstress all reduce/region window ave c_stress2[1]

# c_stress[1] is the x-component of the stress tensor for the entire system
# c_refstress is the average of the x-component of the stress tensor for atoms
#             in our reference region.

thermo_style    custom step temp pe ke etotal lx c_stress[1] &
                c_refstress cpu cpuremain
dump        1 all custom 5000 dumps\dump.${outdata} id type xs ys zs &
            fx fy fz c_PotEng c_displ[4]

fix         1 all nve
fix         2 all langevin ${temp} ${temp} 100.0 567110
fix         3 all deform 1 x erate ${esr}

velocity    all create ${temp} 3278410

log         logs\${outdata}.txt
run         1000000
```

4.10 sss simulation (LAMMPS)

```
# in.tensilesss
# Kristian Keilen - Apr 2014
# LAMMPS script for tensile testing a finite-sized crystal of alkanes
# This script will deform the alkane by moving the outer atoms.
# Boundary conditions are sss, and deformation is done with *moving atoms*
# It is assumed that an energy minimization or similar has taken place.

variable    indata string 10011w1h # The name of our indatafile (data.string)
variable    outdata string 10011w1h-300-90-tsss # What to name our outdata
variable    temp equal 300   # The temperature
variable    pull equal 5e-5 # Rate of deformation

dimension   3
boundary    s s s
units       real     # Distance: Å, Time: fs, Mass: g/mol,
                     # Energy: kcal/mol, Force: kcal/mol.Å
                     # Pressure: atm, Charge: e
                     # 1 kcal/mol.Å = 7e-11 N approx.
atom_style  molecular
timestep    1
thermo      1000

# Parallelizes all the tasks underneath
package     omp * force/neigh
suffix      omp

# Forcefield (coefficients are given in the data and table files)
pair_style  lj/cut 10.0
neighbor    2.0 nsq # Uncomment if LAMMPS gives you trouble about bin sizes
bond_style  table spline 1000
angle_style cosine/squared
dihedral_style opls

read_data   data.${indata}
bond_coeff  1 morse.table BREAK
bond_coeff  2 morse.table ZERO

# Partitioning the box into three parts, the fixed atoms, the moving atoms
# and the free atoms. The reason the region right is larger than the region
# left is that, there is initially some headroom on that side of the simulation
# box with no atoms.
variable    lowall equal xlo+0.8
variable    hiwall equal xhi-1.5
region      left block INF ${lowall} INF INF INF INF units box
region      right block ${hiwall} INF INF INF INF INF units box
group       fixed region left
group       moving region right
group       middle subtract all fixed moving

```



```
# Defining output
compute     stress all pressure thermo_temp
compute     PotEng all pe/atom
compute     displ all displace/atom
compute     support all reduce/region left max fx

thermo_style    custom step temp pe ke etotal lx c_support c_stress[1] &
                cpu cpuremain

dump        1 all custom 10000 dumps\dump.${outdata} id type xs ys zs &
            fx fy fz c_PotEng c_displ[4]

# Moving and not moving atoms
fix         1 middle nve
fix         2 middle langevin ${temp} ${temp} 100.0 332481
fix         3 moving move linear ${pull} 0.0 0.0 units box

velocity    all create ${temp} 657132

log         logs\${outdata}.txt
run         1000000
```

4.11 Thermodynamical run

Our thermodynamic run holds a system at a fixed temperature and outputs thermodynamic properties. The code given underneath is not exactly the same as the one used to output the thermodynamic runs in Part D, section 3, since those were produced with a harmonic bond length potential. These changes are to make the thermodynamical runs compatible with the data producing scripts in Appendix 4.1 and 4.2.

```
# in.thermo
# Kristian Keilen - Feb 2014
# LAMMPS script testing a force field by I. H. Sahputra
# The goal is to output thermodynamical variables for a given temperature,
# then perform a time average giving results like P(T) and U(T).
# From this, results like latent heat of fusion and/or vaporization can be
# computed.
# Make sure that the simulation has run through in.thermoboil first!

variable    indata string 50l10w10h # The name of our indatafile (data.string)
variable    outdata string 50l10w10h100 # What to name our outdata
variable    temp equal 100   # The temperature, saved as a float

dimension   3
boundary    p p p
units       real    # Distance: Å, Time: fs, Mass: g/mol,
                    # Energy: kcal/mol, Force: kcal/mol.Å
                    # Pressure: atm, Charge: e
                    # 1 kcal/mol.Å = 7e-11 N approx.
atom_style  molecular
timestep    1
thermo      1000

# Parallelizes all the tasks underneath
package     omp * force/neigh
suffix      omp

# Forcefield (coefficients are given in the data file)
pair_style  lj/cut 10.0
#neighbor   2.0 nsq # Uncomment if LAMMPS gives you trouble about bin sizes
bond_style  table spline 1000
angle_style cosine/squared
dihedral style opls
read_data   data.${indata}
bond_coeff  1 morse.table BREAK
bond_coeff  2 morse.table ZERO

# Doing some groundwork for our outputting.
fix         1 all nvt temp ${temp} ${temp} 100.0
velocity    all create ${temp} 324500121 rot yes
compute     poteng all pe/atom

# Outputting relevant thermodynamical data.
thermo_style    custom step temp etotal pe ke press vol enthalpy cpu cpuremain
dump        1 all custom 20000 dumps\dump.${outdata} type xs ys zs vx vy vz &
            fx fy fz c_poteng
            # This might become a huge dump file.
log         logs\${outdata}.txt
# Executing the task
run         100000

# Constructs a data file, in case one wants to restart this session
write_data  dumps/data.${outdata}
```