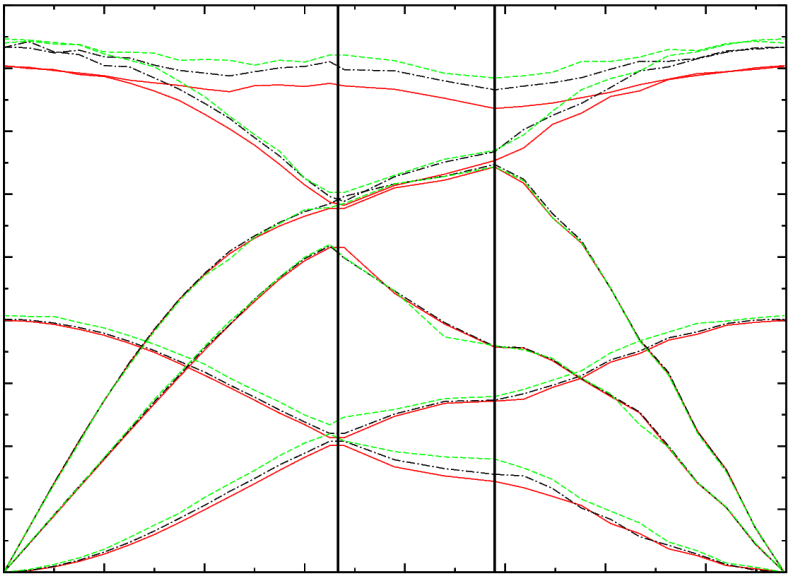
Bachelor Thesis

Luuk Coopmans

July 15, 2016

Temperature dependence of the phonons of graphene



Supervisors:
Prof. dr. Annalisa Fasolino
Dr. Jan Los

Second reader:
Dr. Gilles de Wijs

Theory of Condensed Matter
Institute of Molecules and Materials
Radboud University Nijmegen

Acknowledgements

First of all, I would like to thank my supervisor Prof. dr. Annalisa Fasolino for providing excellent support during my bachelor project. Thanks to her many intellectual advices my project has developed in the right direction. Besides her professional help, I experienced her as a warm and open person who cared about my personal well being.

The next person I want to thank is dr. Jan Los, a very good programmer who helped me solve some problems when I was writing the computer codes for this project. Moreover, I want to thank him for computing some of the parameters which I needed for my research.

Thirdly I want to thank the people from the TCM group, especially my roommates Edo van Veen, Hylke Donker and Lennert van Tilburg, for the enjoyable time I had while working in the group and for sometimes helping me when I had some issues. In particular the day out of the research group I liked really much.

To end, I would also like to thank my parents and some of my friends for the support they gave me when writing my thesis. Especially I want to thank Krijn Reijnders, a mathematics student, who listened to and gave his insights in the use of discrete Fourier transformations for my project.

Contents

Preliminaries

Physical constants:

Name of Constant	Symbol	Value
Speed of Light	c	$2.997\,924\,58 \times 10^{8}\,\mathrm{m\,s^{-1}}$
Boltzmann Constant	k_B	$1.380\,648\,52 \times 10^{-23}\,\mathrm{m^2\,kg\,s^{-2}\,K^{-1}}$
Mass Carbon	M_C	12.011 u
Angstrom	1 Å	10×10^{-10} m
Planck Constant	h	$4.135\,667\,662 \times 10^{-15}$ eV s
Lattice parameter graphene ($T = 0$K)	a	2.4595 Å
2D Mass density graphene ($T = 0$K)	ρ_{2D}	$7.6144 \times 10^{-7}\,\mathrm{kg\,m^{-2}}$

Unit conversion:

Name of quantity	Unit	Value in other units
Atomic mass unit	u	$1.660\,539\,040 \times 10^{-27}$ kg
Energy	1 eV	$1.602\,176\,57 \times 10^{-19}$ J
Frequency	1 $\mathrm{cm^{-1}}$	$2.997\,924\,56 \times 10^{10}\,\mathrm{s^{-1}}$
Frequency	1 meV	$8.065\,544\,005\,\mathrm{cm^{-1}}$

Abbreviations:

Abbreviation	Meaning
MD	Molecular Dynamics
FT	Fourier Transform
DFT	Discrete Fourier Transform
kVACS	k-space velocity autocorrelation sequence
2D/3D	two/three dimensional
ZA	Acoustic out-of-plane
ZO	Optical out-of-plane
TA	Acoustic transverse
TO	Optical transverse
LA	Acoustic longitudinal
LO	Optical longitudinal
LCBOP	Long range carbon bond order potential
PSD	Power spectral density

Introduction

The study of two dimensional (2D) systems has been important in many different research fields, varying from mechanics and statistical physics to chemistry and biology. Contrary to 3D bulk materials, the structural and elastic properties of 2D crystals were predicted to be dominated by thermal fluctuations, leading to unusual scaling behaviour with size [1]. Graphene, the first truly 2D crystal ever observed, is probably by now the best known example of such a 2D system [2]. Its simple hexagonal structure, made of only carbon atoms allows detailed and predictive theoretical and computational studies so that, thanks to graphene, scientists are able to explore the many extraordinary properties of 2D systems further.

The linear band dispersion [3], chiral tunneling [4] and Quantum Hall effect at room temperature [5] measured in graphene are examples of such peculiar properties that have initially attracted much attention. Besides these exceptional electronic properties, also special elastic properties have been predicted and/or observed in graphene [1]. One of those is the rippling of graphene sheets at any finite temperature [6], which is of great importance for the stability of graphene. These ripples stabilize graphene by means of the anharmonic coupling of in-plane and out-of-plane modes as predicted within the theory of membranes [1, 7]. This theory predicts scaling behaviour of several elastic moduli which has been verified by Monte Carlo simulations for graphene [8]. The temperature dependence of the bending rigidity found in this type of calculations [9] is however not fully explained by the theory of membranes. Since the bending rigidity also determines the dispersion of out-of-plane acoustic modes, it is interesting to compare the results obtained from the theory of membranes to those obtained by direct calculations of the phonon modes.

In this thesis we will focus on the temperature dependence of the phonons of graphene, from which the values of the bending rigidity and some other interesting elastic properties like the sound velocity can be extracted [1]. In general, the phonon spectrum does not depend much on temperature and can be calculated by diagonalization of the dynamical matrix [10]. The calculated frequencies can then be used within the quasiharmonic approximation to evaluate thermal effects [11]. However, it has been shown [12] that the quasiharmonic approximation is not applicable to graphene due to its negative thermal expansion. For the same reason also a generally more accurate method, the self consistent phonon approximation(SCAILD) does not lead to convergence [13].

So, determining the phonons of graphene at different temperatures is thus not trivial at all. However, E.N. Koukaras et al. recently proposed a new method based on Molecular Dynamics (MD) simulations and Fourier Analysis that already has succeeded in correctly obtaining the phonon frequencies of graphene at different temperatures [14]. The focus of that work was on the optical modes, whereas we are particularly interested in the low frequency acoustic modes that are directly related to the elastic moduli of the material. We therefore decided to use this new method and we wrote a new computer program to analyze the results of MD simulations based on the LCBOPII potential for carbon [15].

In this thesis we first explain our method for obtaining the temperature dependence of the phonon modes and bending rigidity and then we present our results and conclusions. In order to guide the reader to all the different steps that we take in our study, this thesis is structured in the following way:

Chapter 1: introduction to graphene, phonons and anharmonic effects.
Chapter 2: computational methods.

Chapter 3: Fourier theory and some test results.
Chapter 4: computational approach for graphene.
Chapter 5: numerical results: phonon spectrum and bending rigidity.
Chapter 6: summary, conclusion and an outlook for future research.

Chapter 1

Graphene

In 2004 Andre Geim and Konstantin Novoselov discovered graphene, a stable truly two dimensional (2D) crystal made of only carbon atoms. They were able to achieve this by using sticky tape to exfoliate single layers of graphene from the bulk material graphite [2]. This discovery led to a change in condensed matter research because for a long time it was considered impossible that a strictly 2D crystal could exist [16].

Besides the two dimensionality, graphene also attracts interest because it is considered one of the strongest materials [17], one of the few transparent conductors [18] and has the highest melting temperature of all materials in nature [19]. These properties make graphene a perfect material to be applied in the industry, for example in wafers or other electronic devices. However, it is still difficult to obtain large single layers of graphene and therefore more research is needed before it eventually can be applied on large scale.

It may be clear that graphene is a special material. For this reason this first chapter introduces the topic graphene and its specific physical properties that we are interested in. It includes information about the lattice structure, phonons and temperature dependence of graphene.

1.1 Lattice structure

We start with describing the structure of graphene at the atomic level. Figure 1.1 shows the honeycomb lattice structure of graphene as well as the distance between two nearest neighbour carbon atoms $a = 1.42$ Å. Later we will see that the lattice parameter is strongly temperature dependent, but for now we consider it at $T = 0$K. The unit cell of graphene is defined by the two lattice vectors:

$$\mathbf{a_1} = a \begin{bmatrix} \sqrt{3} \\ 0 \end{bmatrix} \tag{1.1}$$

$$\mathbf{a_2} = \frac{a}{2} \begin{bmatrix} \sqrt{3} \\ 3 \end{bmatrix} \tag{1.2}$$

shown in figure 1.1. The unit cell has a two atom basis described by

$$\vec{r}_i = \vec{r} + \delta_i \quad \text{for } i = 1,2 \tag{1.3}$$

with $\delta_1 = (0,0)$ and $\delta_2 = (a, a\sqrt{3})$. This hexagonal lattice structure can be divided into two triangular sub lattices A and B, denoted in figure 1.1 by red and blue circles, corresponding to the two atoms in the unit cell.

Since we are interested in the collective wavelike movement of the atoms, i.e. the phonons, it is convenient to describe it by means of the wave vectors in reciprocal space. The reciprocal lattice vectors can be found by using the relation

$$\vec{b}_i \cdot \vec{a}_j = 2\pi\delta_{ij} \tag{1.4}$$

with Kronecker delta δ_{ij}, which leads to

$$\mathbf{b_1} = \frac{2\pi}{a\sqrt{3}} \begin{bmatrix} 1 \\ -\frac{1}{\sqrt{3}} \end{bmatrix} \tag{1.5}$$

$$\mathbf{b_2} = \frac{4\pi}{3a} \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{1.6}$$

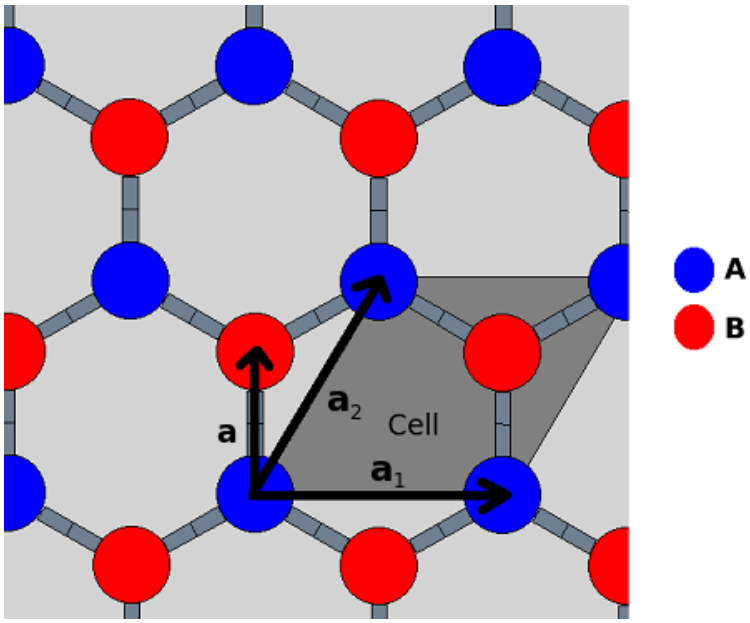


FIGURE 1.1: The hexagonal lattice structure of graphene, A en B denote the two sub lattices. $\boldsymbol{a_1}$ and $\boldsymbol{a_2}$ are the lattice vectors and $\boldsymbol{a}$ is the distance between two nearest neighbour carbon atoms.

By using the reciprocal lattice vectors (1.5) and (1.6) the first Brillouin zone of graphene in the reciprocal plane (k_x, k_y) can be sketched as shown in figure 1.2. It is enough to just look at the first Brillouin zone due to the periodicity of the crystal.

The points $\boldsymbol{\Gamma}$,$\boldsymbol{K}$ and $\boldsymbol{M}$ at the boundary of the Brillouin zone have the highest symmetries. These points and the paths between them have higher symmetry than other points in the Brillouin zone. The coordinates of these special points are given by

$$\boldsymbol{\Gamma} = \frac{2\pi}{a\sqrt{3}}\begin{bmatrix}0\\0\end{bmatrix} \qquad \boldsymbol{K} = \frac{2\pi}{a\sqrt{3}}\begin{bmatrix}\frac{2}{3}\\0\end{bmatrix} \qquad \boldsymbol{M} = \frac{2\pi}{a\sqrt{3}}\begin{bmatrix}\frac{1}{2}\\\frac{1}{2\sqrt{3}}\end{bmatrix} \tag{1.7}$$

and the coordinates for the paths between them are shown in table 1.1.

TABLE 1.1: Coordinates for the high symmetry paths.

Path	x coordinates	y coordinates
$\boldsymbol{\Gamma K}$	$0 \leq x \leq \frac{4\pi}{3a}$	$y = 0$
$\boldsymbol{KM}$	$\frac{\pi}{a} \leq x \leq \frac{4\pi}{3a}$	$0 \leq y \leq \frac{\pi}{a\sqrt{3}}$
$\boldsymbol{M\Gamma}$	$0 \leq x \leq \frac{\pi}{a}$	$0 \leq y \leq \frac{\pi}{a\sqrt{3}}$

1.2 Phonons

As said in the introduction we are interested in the phonons of graphene. In this section we will describe what phonons are and give some of their basic properties. After that we will discuss the currently known phonon dispersion of graphene and some of its special characteristics.

1.2.1 What are phonons

In section 1.1 we described the lattice structure of graphene as if it is rigid which is only true at $T = 0$. At finite temperatures the atoms vibrate around these equilibrium positions due to

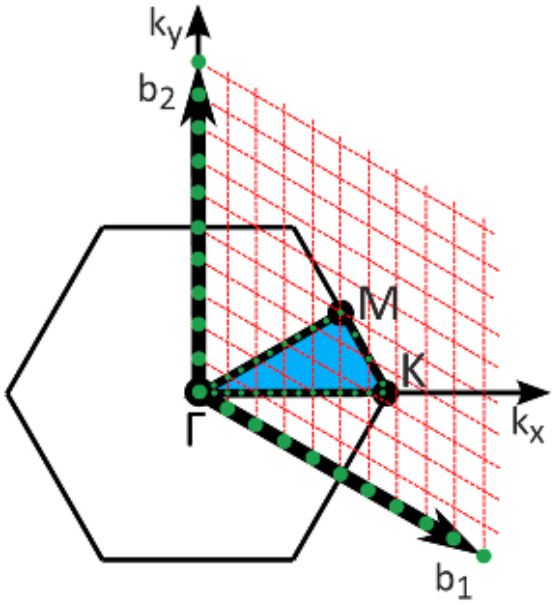


FIGURE 1.2: The Brillouin zone of graphene with reciprocal lattice vectors $\boldsymbol{b_1}$ and $\boldsymbol{b_2}$ and high symmetry points $\boldsymbol{\Gamma}$, $\boldsymbol{K}$ and $\boldsymbol{M}$. The green bullets denote the partitioning of the wave vectors which will be described in section 4.1. This picture is taken from [14].

the thermal energy they have as described by the equipartition theorem from classical statistical mechanics [10].

However, the atoms are still part of the periodic lattice and therefore cannot move freely. The movement of a single atom in the lattice is affected by the movements of the other atoms. As we know, the interaction between particles can be described by a potential. From a potential then the equations of motion of the atoms can be derived [10].

With help of Bloch's Theorem [10] we can find the solution of the equations of motions. It turns out that the solutions are collective waves which obey the periodicity of the lattice. In other words the frequencies of the wavelike solutions depend on the structure of the lattice. These frequencies are the normal modes of the lattice and the relation between the frequency and the reciprocal lattice vectors is given by the dispersion relation $\omega(k)$ [10].

So all the atoms in the lattice vibrate collectively according to a periodic wave solution that matches the periodicity of the lattice. The whole lattice is vibrating with superpositions of these normal modes. Our goal is to determine these normal modes, vibrations of the whole lattice, which are called phonons.

A more formal derivation of the definition of phonons as quasi-particles based on quantum mechanics and Bose-Einstein statistics is given in [10]. However, the just given (classical) description of phonons is sufficient for us because we study only the lattice vibrations at temperatures around room temperature. Intuitively, we could compare phonons to photons: phonons describe the propagation, dispersion, of sound through a crystal whereas photons describe the propagation of light through space [20].

Phonons can be divided into two branches: the acoustic branch and the optical branch [21]. The acoustic branch corresponds to the propagation of sound waves through the system and goes to zero for small wave vectors. The optical branch generally has higher frequencies and has a finite value for $\vec{k} = 0$. It is important to notice that for lattices with a one atom basis only the acoustic branch occurs at Γ, whereas for a lattice with a two atom basis, such as graphene, also the optical branch becomes visible.

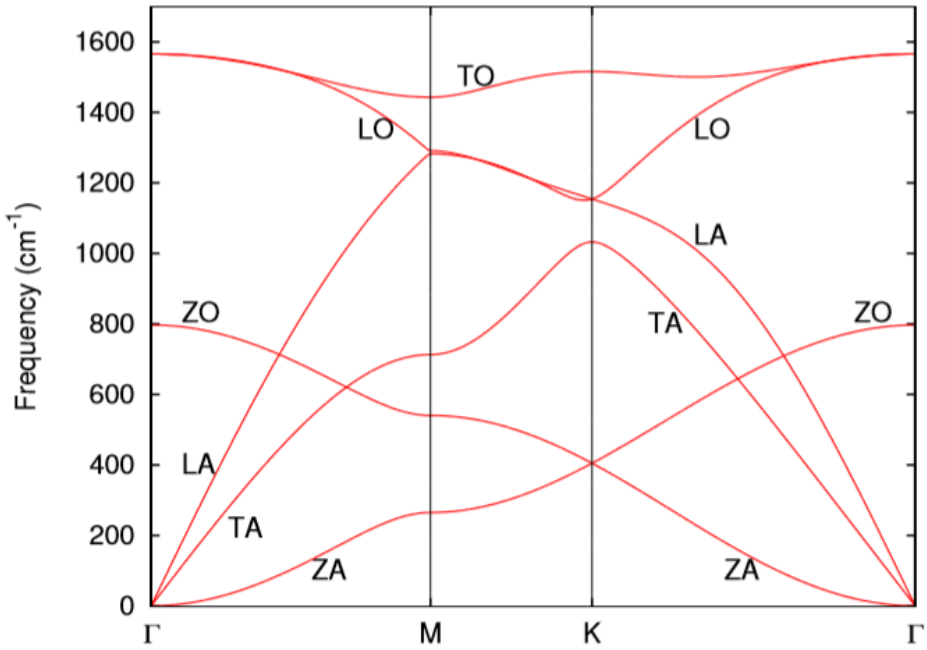


FIGURE 1.3: The phonon dispersion of graphene calculated by Leendertjan Karssemeijer [12] at $T = 0$ K based on the LCBOPII potential.

An extremely important point is that in the acoustic branch the atoms move in unison, while in the optical branch the atoms move out of phase [10]. For this reason, the vibrations of the two sub lattices A and B, as described in section 1.1, amplify each other for the acoustic branch, but cancel each other out for the optical branch. Therefore only one sub lattice needs to be taken into account if we also want to find normal modes for optical branches [14].

The fact that phonons can describe sound velocities in materials [22] and also their affect on electrical and heat transport in solids make them an interesting research topic [21].

1.2.2 Phonons of graphene

Figure 1.3 gives the phonon dispersion of graphene as computed by Leendertjan Karrsemeijer with the same potential LCBOPII that we are using [12]. At first sight, the dispersion looks not very special compared to phonon dispersions of other materials. We can recognize acoustic and optical branches as expected for a material with a two atom basis. The optical branches indeed have a finite frequency for $\vec{k} \to 0$ while the frequencies of the acoustic branches go to zero.

Secondly, we can recognize the high-symmetry points $\boldsymbol{\Gamma}$, $\boldsymbol{K}$ and $\boldsymbol{M}$ points in the picture. This are the zone boundaries of the Brillouin zone as described in section 1.1. At for instance the $\boldsymbol{K}$-point, the acoustic out-of-plane (ZA) mode is degenerate with the optical out-of-plane (ZO) mode, this is expected because we have a two atom basis with only carbon atoms ($M_1 = M_2$) [10].

When looking at the acoustic phonons for small wave vectors near the high-symmetry point Γ we see that the phonon dispersion $\omega(k)$ for the transverse acoustic (TA) mode and the longitudinal acoustic (LA) mode is linear. The ZA mode however, $\omega(k)$ has not a linear, but a quadratic dispersion which is a special characteristic of 2D layered materials [1]. We can clearly see this quadratic behaviour in figure 1.3. This mode is of particular interest because it is a bending mode which is linked to the bending of the surface [22]. We will explain this relation in the next section.

Finally, the behaviour of the ZO mode of the phonon dispersion of graphene is also interesting. We already discussed the ZA-mode shortly, but when looking closely at figure 1.3 we see that the

frequency of the ZO mode is much lower than the frequency of the transverse optical (TO) and the longitudinal optical (LO) modes. The explanation for this is that the ZO mode describes the movement in the out-of-plane direction, while the TO and LO modes describe the in-plane movements. For graphene movement in the out-of-plane direction is much easier than the in-plane movements since the atoms can move freely in the out-of-plane direction but not freely in the in-plane direction [22].

1.3 Bending Rigidity

As explained in the former section, the ZA phonon is linked to the bending of graphene. This ZA mode namely describes the in phase movement of the atoms in the out-of-plane direction. For long wavelengths λ, this movement of the atoms bends the surface resulting in ripples [22]. It is important to notice that this quadratic behaviour is only true for long wavelengths, so for small k-vectors since $\lambda \propto \frac{1}{k}$.

Lifschitz [23] derived that the quadratic out-of-plane dispersion for graphite depends on the bending rigidity κ, which is a measure for the amount of energy required to bend a crystal. In absence of strain, this relation is given by

$$\omega_{ZA}(\vec{k}) = \sqrt{\frac{\kappa}{\rho_{2D}}}\left|\vec{k}\right|^2 \tag{1.8}$$

where κ is the bending rigidity and ρ_{2D} a two dimensional mass density.

On the other hand, the bending rigidity κ of graphene is usually calculated from the bending energy

$$\epsilon = \frac{\kappa}{2}H^2 \tag{1.9}$$

with H the curvature. This bending energy can be derived from calculations of the total energy of carbon nanotubes of increasing radius R where $H = \frac{1}{R}$ in equation 1.9 [24]. Leendertjan Karssemeijer [22] determined in this way the bending rigidity $\kappa = 0.69$eV at zero temperature.

Here we will use the Lifschitz relation to the ZA mode as given in equation 1.8 to determine the bending rigidity κ. As described in section 1.4 we will see that this method is suitable to study the temperature dependence of κ due to anharmonic effects. In fact, the quasiharmonic approximation [11], which is usually sufficient to calculate the temperature dependence of phonon frequencies fails for graphene [12, 13] due to the fact that the phonon frequencies calculated for a lattice parameter smaller than the one at $T = 0$K (as said graphene contracts with temperature) become imaginary end cannot be used as for this approximation.

1.4 Anharmonic effects and temperature dependence

We end this chapter with a short discussion about why the harmonic approximation is not valid for graphene. The first indication that we need to take anharmonic effects into account is that the out-of-plane fluctuations of graphene are of the same order of magnitude as the interatomic distance since the atoms can move freely in this direction. The harmonic approximation only holds for small fluctuations compared to the interatomic distance so is not appropriate here [1].

Indeed, Peierls and Landau [16] proposed more than 70 years ago, that in the standard harmonic approximation, the long range order of the crystal structure would be destroyed at any finite temperature because of thermal fluctuations [16]. Moreover, Zakharchenko found that the lattice parameter of graphene depends on temperature [9], while the harmonic approximation predicts a constant lattice parameter. For all these reasons, we need look beyond the harmonic approximation, so considering anharmonic effects.

Anharmonic effects can be taken into account by including anharmonic terms in the potential for the calculation of the phonon dispersion. The potential that we used will be described in the next chapter. Anharmonic effects will directly contribute to the temperature dependence of the phonon modes without need of approximations.

Chapter 2

Numerical Methods

As discussed in chapter 1, lattice vibrations depend on the vibrations of single atoms in the lattice. So the first step we need to take for finding the lattice vibrations is to correctly calculate the positions and the velocities of the atoms. Studying the lattice dynamics can be done both quantum mechanically and empirically. Quantum methods, also called ab initio calculations, solve the Schrödinger equation using only fundamental constants [25]. Empirical methods instead, need some additional input parameters derived from experimental observations and thus are actually approximations.

Since we are interested in systems with many atoms it is better to use an empirical method because generally it computes faster than a quantum method [26]. Moreover, it has been shown that often the elastic properties could be well defined by empirical methods [1]. For these reasons we decided to use Molecular Dynamics (MD) simulations based on an empirical interatomic potential to simulate the positions and velocities of the atoms during a given time. Such a simulation helps us with finding the periodic behaviour of the atoms.

In this chapter we will describe the main concepts of this numerical method based on an empirical potential. Furthermore, in the end we will introduce a numerical technique to investigate whether the results of our simulations contain patterns, periodicity or are just random noise.

2.1 Empirical long-range bond-order potentials

We start with introducing empirical interatomic potentials. Such potentials describe the interactions between atoms, they give the energy of a system of atoms as a function of their coordinates. Many different forms of interatomic potentials have been developed describing all sorts of interactions. In MD simulations the so called bond order potentials are a very much used class of empirical potentials. In these potentials the strength of the bonds depends on the environment: number of bonds, bond length and bond angle [27]. One characteristic that is very important is that these potentials are reactive, namely they allow breaking and formation of bonds thereby allowing phase transitions [1].

The 'Long-range Carbon Bond-Order Potential' or LCBOP, especially developed for carbon systems, is such a potential [28]. This potential takes long-range interactions into account which are of great importance for carbon structures to correctly describe the weak bonding between graphitic planes [1].

A new version of LCBOP, the LCBOPII, has been developed to allow an accurate description of carbon in the liquid state (at $T = 6000$K). This LCBOPII included newly developed mid-range interactions besides the already existing short- and long-range interactions in LCBOP [15]. Figure 2.1 gives an schematic overview of all these interaction domains of LCBOPII. All these interactions contribute to the total binding energy

$$E_b = \frac{1}{2}\sum_{i>j}^{N}\left(S_{ij}^{sr}V_{ij}^{sr} + S_{ij}^{sr}V_{ij}^{lr} + \frac{1}{Z_i^{mr}}S_{ij}^{mr}V_{ij}^{mr}\right). \tag{2.1}$$

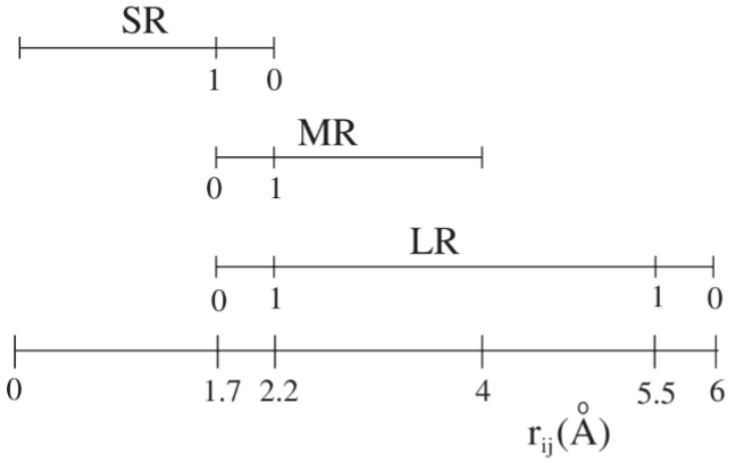


FIGURE 2.1: Schematic overview of the short-range (SR), middle-range (MR) and long-range (LR) interaction domains of LCBOPII. The bottom line gives the total interaction range r_{ij} from $r_{ij} = 0$ Å to $r_{ij} = 6$ Å. The cutoff functions S_{ij} are active in the domains between the numbers 0 and 1. This picture is taken from the thesis of Leendertjan Karssemeijer [12].

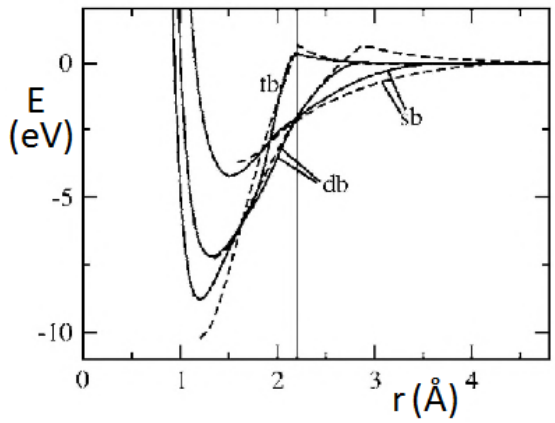


FIGURE 2.2: Bonding energies for single bond (sb), double bond (db) and triple bond (tb) carbon atoms calculated by density functional (dashed curves) and LCBOPII (solid curves). Picture copied from [15].

in which is summed over the distances $r_{ij} = |\mathbf{r_i} - \mathbf{r_j}|$ between the N atoms [15]. V_{ij}^{sr} accounts for the covalent short-range interactions which depend on the bond-order. The V_{ij}^{mr} function includes the middle-range effects, just after the short-range, like the presence of "dangling bonds". $\frac{1}{Z_i^{mr}}$ is the middle-range coordination number, which is nothing more than a correction factor. The long-range potential V_{ij}^{lr} represents non-bonded interactions like the van der Waals forces. Finally the S_{ij} terms are smooth cut off functions that make smooth connections between the three ranges [15].

Figure 2.2 gives a fit of this binding energy, it is interesting to notice that the resulting potential does not look harmonic. Actually it looks more like a Morse potential which includes anharmonic effects [29]. As discussed in section 1.4 the presence of such anharmonic effects in the potential is essential if we want to correctly describe graphene.

2.2 Molecular Dynamics Simulations

The aim of our study is to calculate the phonon dispersion of graphene at finite temperatures, as discussed in section 1.2 we therefore need to calculate the dynamics of the atoms. MD simulations is a numerical technique to compute the dynamics at atomic level [30]. This technique is based on the assumption that heavy atoms (such as carbon) obey the laws of classical mechanics. In systems of heavy atoms the nuclei have namely a much greater influence than the surrounding electrons. In this section we explain the details of this technique and the Verlet algorithm on which it is based. For most concepts in this section we used [31] and chapter 4 "Molecular Dynamics Simulations" from a book about Molecular simulation written by Daan Frenkel and Berend Smit [30].

First of all, a MD calculation can be divided into a few different steps:

1. read in the parameters that specify the initial positions and velocities
2. compute the forces on all the particles
3. integrate Newton's equations of motion
4. compute and print the averages of the measured quantities

In the first step, parameters like the number of particles, the size of the sample, the length of the time step, the duration of the simulation and the desired temperature are read into the program. After that, the program reads in the starting positions and the starting velocities of the atoms are derived by a Maxwellian distribution. These velocities are scaled so that the mean kinetic energy of the atoms is equal to the mean thermal energy as defined by the equipartition theorem [30]. This results in

$$\left\langle v_\alpha^2 \right\rangle = \frac{k_B T}{m} \tag{2.2}$$

for the mean velocities of an atom in a particular spatial direction α. We see in equation 2.2 that the mean velocity depends on the desired temperature, so changing the desired temperature will result in different velocities of the atoms.

Next, the actual simulation can start and the forces on the atoms are calculated using

$$\vec{F}_i = \nabla_{\vec{r}_i} V_{tot} \tag{2.3}$$

with V_{tot} the total potential that describes the interactions between all the atoms, which in our case is LCBOPII as described in section 2.1.

Going from forces to new positions is the most important step in MD simulations and is done by integrating Newton's equations of motion for the desired length of time using the given time step. This can be done in several ways using different algorithms with advantages and disadvantages [30]. The MD simulations program that we use is based on the Verlet algorithm which we will describe in detail in the next subsection.

Finally, the wanted quantities, in our case the new positions and new velocities, are computed and printed.

2.2.1 The Verlet algorithm

As stated in the previous section, the most important step in MD simulations is integrating Newton's equations of motion. To do this numerically, several algorithms have been developed and used through history [30]. The algorithm that we use is the Verlet algorithm which was rediscovered by Loup Verlet in the 1960s [31].

Since we want the positions and the velocities of the atoms at the same instant, we use a special form of the Verlet algorithm which is called velocity Verlet algorithm [31]. According to this algorithm, the new positions $\vec{r}(t + \Delta t)$ can be computed from the current positions $\vec{r}(t)$,

velocities $\vec{v}(t)$ and accelerations $\vec{a}(t)$ [30] by using

$$\vec{r}(t + \Delta t) = \vec{r}(t) + \vec{v}(t)\Delta t + \frac{1}{2}\vec{a}(t)\Delta t^2. \tag{2.4}$$

With the new positions a formula for the new accelerations $\vec{a}(t+\Delta t)$ can be derived from Newton's second law

$$\vec{F} = m\vec{a} \tag{2.5}$$

combined with equation 2.3 for the forces on the atoms resulting in

$$\vec{a}(t + \Delta t) = -\frac{1}{m}\nabla V(\vec{r}(t + \Delta t)). \tag{2.6}$$

Eventually, with help of these new accelerations, the new velocities can be calculated:

$$\vec{v}(t + \Delta t) = \vec{v}(t) + \frac{1}{2}(\vec{a}(t) + \vec{a}(t + \Delta t))\Delta t. \tag{2.7}$$

The velocity Verlet algorithm as described above is one of the best suitable algorithms in many situations. It is for instance very fast, time-reversal and area preserving. These last two are very important because if these two properties are not present, then the principle of energy conservation is not obeyed [30]. Also this algorithm requires as little memory as possible and approaches true Hamiltonian dynamics when taking the limit to infinitely small time steps. However, the main disadvantage of this algorithm is that it is not very accurate on long time steps [30], therefore we need to compute the forces on the particles many times using a very small time step Δt as we will discuss further in section 4.2.

2.2.2 Statistical uncertainties in simulations

It is important to notice that MD simulation is a statistical technique and its results are statistical averages [32]. If we want to compare MD simulation results with real experimental data we therefore need to find a way to define the precision of our calculations. Just like in real experiments, in computer experiments two kinds of errors can occur: systematic and statistical [32].

To begin with statistical errors, one way to see if there is a statistical error is by looking at the standard deviation of the statistical results. A low standard deviation indicates that the results at some instant are close to the mean value of the whole sequence. It turns out that this standard deviation depends on the duration of the simulation [32], so simulating longer will decrease the statistical uncertainties.

However, systematic errors on the other hand are not so easy to recognize and do not necessarily decrease with longer simulation times. In the book of J.M. Haille [32] an exact method to find such errors is given, but we will not discuss this here. For us, it is interesting to notice that systematic errors can decrease if we increase the number of atoms in the system due to lesser influence of boundary conditions [32]. Furthermore, to check if there is a systematic error we could compare our results to other studies with different simulation methods, for example Monte Carlo simulation [9], to see if there are large differences.

The most important thing to keep in mind is that we need to check everything carefully because this method is sensitive to mistakes.

2.3 Velocity Autocorrelation functions

MD simulation provides us with a large number of positions and velocities at different time steps, but it does not provide information about relations and periodicity in these quantities. Since we

are interested in the periodic behaviour of these quantities, we need to analyze the MD simulation results somehow. In this section we introduce a method of doing this and give an example.

First of all we need to find out if the MD simulation results that we obtained are really periodic or if they are just random noise. To investigate this we can use time correlation functions which measure how the value of some dynamic quantity $A(t)$ may be related to the value of some other quantity $B(t)$ [32]. Time correlation functions are only suitable for time dependent data and are defined by:

$$C(t) = \lim_{\tau\to\infty} \frac{1}{\tau} \int_0^\tau A(t_0)B(t_0 + t)dt_0 = \langle A(t_0)B(t_0 + t)\rangle \tag{2.8}$$

where is integrated over many different time origins t_0 [32]. It is important that each time origin t_0 is taken from a system at equilibrium. The bracket notation denotes time averages.

With equation 2.8 we are able to check whether two time dependent quantities $A(t)$ and $B(t)$ are correlated or not. If they are not, $C(t)$ simply gives $\langle A\rangle \langle B\rangle$. If they are correlated however, the result for $C(t)$ is much different. In our case we want to calculate the correlation of a velocity signal to itself to search for periodicity and therefore the quantities A and B are taken to be the same. In this special case 2.8 is called an autocorrelation function [32].

The main advantage of this technique is that it removes noise from the signals because $\langle R_{noise}\rangle = 0$ for the reason that noise is not correlated to anything else in the signal. Secondly, an autocorrelation function is invariant under translations of the time origin t_0, therefore it does not matter from which starting point in time the MD simulation results are obtained [32].

MD simulation uses finite time steps, therefore its results are discrete and we need to find a way to implement this in the autocorrelation method. Several algorithms have been designed to apply this technique to discrete signals, but we will only explain the details of the algorithm that we use in our study [14]. The solution is to define an autocorrelation sequence $\hat{r}$ instead of a continuous function.

This solution can be explained in the following way: consider a simulation of in total N data points measured at time steps with width Δt. We now want to derive a correct sequence from 2.8. First of all, the term $\lim_{\tau\to\infty}$ in 2.8 could not go to infinity anymore because the timescale is limited by the total duration of the simulation. Secondly, we define the time step in discrete form by m, which is a fixed number of data points. To make sure we stay in the set of N data points τ now becomes $N - m$.

In discrete form the integral in 2.8 will change into a sum over different starting data points which we will denote by n. Also these starting points need to be taken from the data signal and therefore n will at its maximum, as shown in figure 2.3, be equal to $N - m - 1$. Eventually all this leads to the following discrete autocorrelation sequence:

$$\hat{r}(m) = \frac{1}{N-m} \sum_{n=0}^{N-m-1} \langle A(n)A(n+m)\rangle\,. \tag{2.9}$$

Figure 2.3 gives a schematic illustration of the autocorrelation procedure for $m = 3$. It can be seen that each data point n is compared to data point $n + m = n + 3$ to check if there is any correlation between these two data points. An important feature of autocorrelation sequences is that an autocorrelation sequence of a periodic signal again forms a periodic signal [32]. This can be understood better by looking at the example shown in figure 2.4. We can see in the picture that the autocorrelation function removes the noise from the signal and gives a clear signal with a clear period.

It can be shown that this period is the same as the period from the original uncorrelated signal. In fact, the function has a maximum value when the two data points are completely correlated, or in other words have the same value. For a harmonic function this is the case when we look with exactly the period of the signal, $t = P$, because then shifting the time origin has no influence on

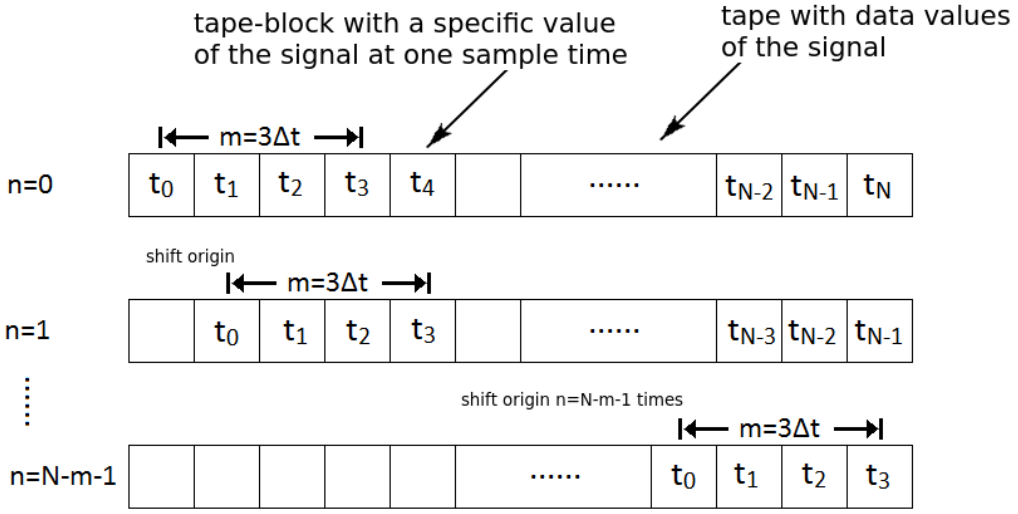


FIGURE 2.3: Schematic illustration of a tape of dynamic stored data at N times t_i for $i = 0,\ N-1$. Shifts in the time origins are shown for all possible values of n denoted by the boundary arrows around $m = 3\Delta t$. The data values at the tape-blocks do not shift and stay stored at the same place. It is immediately clear that n could not reach further than the $N - m - 1$ data point. Remake of a picture in [32].

the correlation between the data points since we always end up with two data points with the same value.

Furthermore, the autocorrelation function has a minimum when the two data points are totally uncorrelated, for a periodic signal this happens when comparing a minimum value to a maximum value. We know from harmonic functions that minima and maxima occur at half of the period of the signal $t = P/2$. So the autocorrelation function oscillates between maxima at $t = iP$ for $i = 0, 1, \ldots.$ and minima at $t = iP/2$ for $i = 1, 2, \ldots.$, which is a cosine with the same frequency as the original signal as illustrated in figure 2.4.

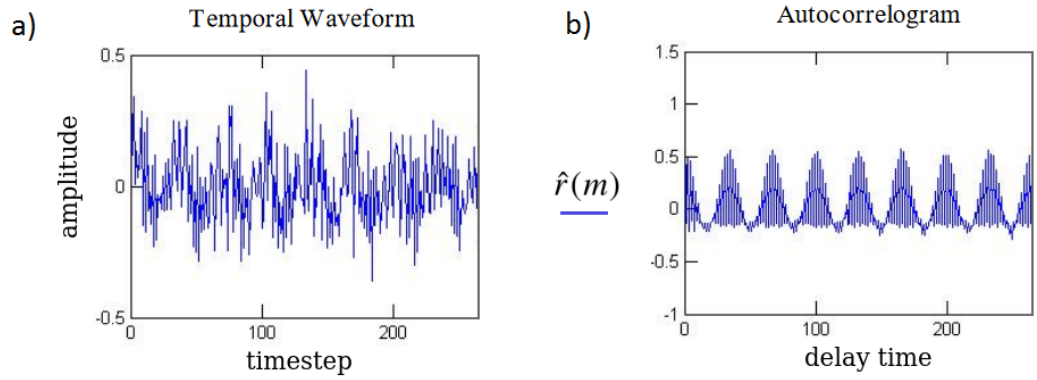


FIGURE 2.4: Plot a) is an example of a noisy signal measured at different timesteps, plot b) gives the autocorrelation sequence of this signal as a function of the delay time m as given in equation 2.9. The resulting signal is a cosine with the same frequency. These example signals were taken from [33].

Chapter 3

Fourier Analysis

As stated in the introduction, we use a recently proposed method based on MD simulations and Fourier Analysis to calculate the phonon frequencies of graphene at different temperatures [14]. Since we have only discussed MD simulations in detail, we will explain in this chapter the main concepts of Fourier Analysis which is the other important element in our method. We will start with an introduction to Fourier Theory, then we will describe how to apply this technique to discrete results from MD simulations and in the end we will present the Fourier test program that we have written.

3.1 Introduction to Fourier Analysis

Before the time of Joseph Fourier, mathematicians and physicists used Taylor Series to expand continuous functions in infinite polynomials. However, this technique failed for periodic functions and functions with discontinuities in them [34]. This was quite an issue because many physical quantities behave periodically. Fortunately Fourier proposed a solution which was originally intended to solve the heat equation [35]. He came with the idea that every function can be expressed by a sum of harmonic functions like sines and cosines. This expansion is called a Fourier Series and is in the complex representation defined as

$$f(t) = \sum_{-\infty}^{\infty} C_n e^{int} \tag{3.1}$$

with

$$C_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(t) e^{-int} dt. \tag{3.2}$$

Clearly this new representation of functions solved the problems with the Taylor Series and a new theory was born [35]. From then on the theory was further developed and became useful in many applications like signal processing and partial differential equations [36].

From the series representation 3.1 we can derive the so called Fourier transformation which is more useful in our study. This derivation leads to

$$g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \tag{3.3}$$

for the Fourier transformation and

$$f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(\omega) e^{i\omega t} d\omega \tag{3.4}$$

for the inverse Fourier transformation. The derivation of equations 3.4 and 3.3 and some properties of Fourier transformations are given in [34].

In expressions 3.3 and 3.4 we used the continuous variables angular frequency ω and time t. Such a transformation is called a Temporal Fourier transform and transforms time signals to

corresponding frequency distribution functions. For this reason they can help to determine the eigenfrequencies in a signal composed of a superposition of different frequencies. When ω_1 is for example an eigenfrequency of the signal $h(t)$, than the Fourier transform $h(\omega)$ will give a sharp peak at $h(\omega_1)$.

Note that we choose the related variables ω and t, while we could also have chosen any other pair of related variables. The condition for this relation is that the first variable needs to have the inverse units of the second variable. In the chapter 4 we will see, for example, that we need to transform from real space to reciprocal space. This can be done easily with a Fourier transform because $\vec{x}$ has the unit m and $\vec{k}$ has the related inverse unit m^{-1}. Such a transformation is called a spatial Fourier transformation.

3.2 Discrete Fourier Transformation

As discussed in chapter 2 a MD simulations program uses finite time steps resulting in signals of physical quantities measured in discrete time intervals. The continuous Fourier transformation as described in the previous section is not suitable for such signals. We therefore in this section shortly derive a discrete version of equation 3.4.

First consider we have a discrete sequence of N steps Δt which contains at each step the value of a quantity f(t). We can visualize this sequence as

$$f(n\Delta t) = f(0), f(\Delta t), f(2\Delta t), ..., f((N-1)\Delta t) \tag{3.5}$$

with $n = 0, ..., N-1$. Obviously this sequence is not continuous and therefore the Fourier transform has to be treated as such.

Changing the continuous function $f(t)$ in equation 3.4 with the discrete sequence $f(n\Delta t)$ in 3.5 results in a discrete frequency distribution $g(k\Delta\omega)$ after the transformation. The value of the discrete frequency steps $\Delta\omega$ is determined by the length of the complete sequence $t = N\Delta t$. This results in

$$\Delta\omega = \frac{2\pi}{N\Delta t} \tag{3.6}$$

for the quanta of discrete frequencies [34].

Next, we notice that in equation 3.4 is integrated over the time between $t = -\infty$ and $t = \infty$. Our sequence however, is not infinitely long: it is limited by the first measured value at $n = 0$ and the last measured value at $n = N-1$. To make the integral discrete, we have to replace it with a sum over all the measured values $f(n\Delta t)$ between $n = 0$ and $n = N-1$. Together with 3.6 this eventually gives

$$g(k\Delta\omega) = \sum_{n=0}^{N-1} f(n\Delta t) e^{-ik\Delta\omega n\Delta t} = \sum_{n=0}^{N-1} f(n\Delta t) e^{\frac{-i2\pi kn}{N}} \tag{3.7}$$

as expression for the discrete Fourier transform (DFT).

This was just one way of deriving the DFT. A more elegant way is by looking at the complex respresentation of the Fourier series 3.1 which already is in discrete form. The details of this different derivation from 3.1 to 3.7 can be found in [34].

3.2.1 Fast Fourier Transformation

Although DFT is a suitable method for analyzing discrete periodic sequences, it unfortunately also is a very slow method. There are $O(N^2)$ operations necessary for analyzing one sequence. Cooley and Tukey [37] recognized that a lot of these operations are unnecessary because the same calculations are done multiple times. For this reason they developed an improved method called

fast Fourier transformation (FFT) which reduces the number of computations to $O(N\log(N))$. In this subsection we will briefly describe this method.

First of all, we can recognize double calculations in the DFT by looking carefully at equation 3.7. In the exponent we recognize the two indices k and n, this exponent is calculate many times when summed over n. However, some of these calculations are done twice or more when $kn = nk$. So, for example, the calculation of the exponent for $n = 1$ and $k = 7$ has the same result as the calculation for $n = 7$ and $k = 1$. This is thus not an efficient method at all.

Cooley and Tukey [37] were the first who recognized this and developed an algorithm to get rid of this. Their basic idea was to split up the original DFT 3.7 into two sums: one over the even indices n and one over the odd indices n. After a short derivation it can be shown that those two sums are also DFT's, but both only sum over $\frac{N}{2}$ indices [34]. This reduces the number of computations from $N^2 \mapsto 2(\frac{N}{2})^2$.

This is actually not the end, it turns out that also these two new DFT's can be devided into new sums over $\frac{N}{4}$ indices and that also these sums are again DFT's. We can continue with this splitting process until there is only one point left to sum over, that is when $\frac{N}{2^l} = 1$, so $N = 2^l$. The resulting speed is $O(N\log(N))$ which is much faster than the original DFT.

So if we want to develop a fast program that recognizes frequencies, we need to use FFT. Unfortunately, this technique requires that N needs to be a power of 2 which we can not guarantee to be true for our signals. For this reason since we chose to use DFT instead of FFT.

3.2.2 Wiener-Khintchine Theorem

Before continuing to the test DFT program which we wrote, we will first shortly explain a theorem that connects Fourier analysis with autocorrelation. This theorem, the Wiener-Khintchine theorem, states that the Fourier transform 3.4 of an autocorrelation function 2.8, indicated as $F(C_{\text{autc}}(t))$ with $C_{\text{autc}}(t)$ an autocorrelation function, is the power spectral density $|P(\omega)|^2$ [34]. In mathematical form this is given by

$$F(C_{\text{autc}}(t)) = |P(\omega)|^2 . \tag{3.8}$$

Such a power spectral density (PSD) describes how power is distributed over the frequencies. In this way we can derive which frequency ω has the most power and therefore can be seen as an eigenfrequency.

3.3 A Fourier test program

Before applying Fourier analysis to our real velocity signals, we first wrote a test program to test if we could correctly apply the just described DFT algorithm on a sequence of data points with a known frequency. In this section we will first shortly describe this program, than give the results and finally comment on the results. The FORTRAN90 code for this program can be found in appendix B.

3.3.1 The program

We started with making a sequence of $l = 1000$ measurements of a known function with a known frequency. The test function that we first used was a cosine with frequency $\omega = 10$. Note that we not work with units for this test program and we used $\Delta t = 1$. The value $f(i\Delta t)$ of this function was calculated for $i = 0, \ldots 999$ and written to a file.

In the second step, we read in the values $f(i\Delta t)$ of the first step and than Fourier transform this discrete function by using equation 3.7. The resulting sequence of frequency components $g(j\Delta\omega)$

was written to another file. Where $j = 0, 1, 2, ..., 999$ iterates the different frequencies as shown in Appendix B.

Finally, to test if the Fourier transform had worked and was not producing big errors, we took the inverse Fourier transform and compared the result to the original function. The value of the original function, the back transformed function and the difference between them was afterwards written to a file so that we could analyze the data.

3.3.2 Results

Running the program resulted in figures 3.1 and 3.2.

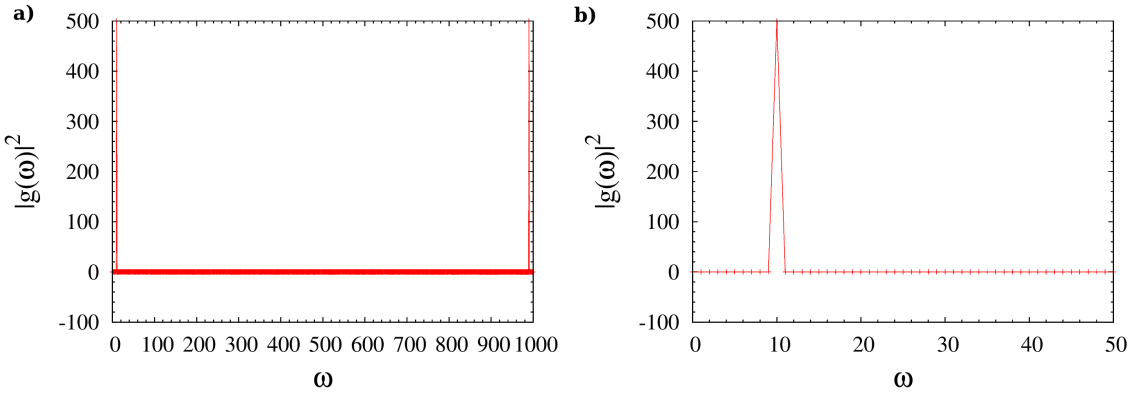


FIGURE 3.1: a) Resulting Fourier spectrum of a cos(10t) function. b) Zoomed version of the Fourier spectrum illustrated in a). We recognize a peak at $\omega = 10$.

From the Fourier spectrum in figure 3.1 can be read that $\omega = 10$ is the frequency, which is in accordance with our given frequency. The peak is very sharp between $\omega = [9.5, 10.5]$ and we cannot see much noise. The peak at $\omega = 90$ is very interesting, because this is quite strange while this frequency is not the frequency that we had given the function. However, it turns out that $f(\omega) = f(-\omega)$ is a feature of DTF's as described in [34]. For us it is important that this has a consequence for the values of ω which we could evaluate, namely from $\omega = 0$ to $\omega = \frac{N}{2}\Delta\omega$, this highest frequency is also called the Nyquist frequency. So we must take many data points N if we want to study very high frequencies.

In figure 3.2 the original and the back Fourier transformed function are illustrated. The back transformed function looks like an exact copy of the original function which is an indication that the Fourier transformation has worked correctly. Further analyzing the data file for the back transform reveals an error of $O(10^{-13})$ which is very small.

3.3.3 Comments and further analysis of our Fourier test program

So from our first results we could say that the program seems to work. To be sure we tested the program for some other functions, including one with a combination of powers of cosines and sines, with different frequencies. For all these test functions we were able to find the correct frequencies and the difference between the original and the back transformed functions were negligible.

It is however important to notice that we need to simulate for a very long time with small time steps for two different reasons. Firstly, the maximum frequency that we can analyze is the Nyquist frequency which is limited by the number of measurements N. Secondly, we can evaluate frequencies in steps of $\Delta\omega$ as given in equation 3.6, increasing N will thus lead to a higher accuracy since we can evaluate more distinct frequencies.

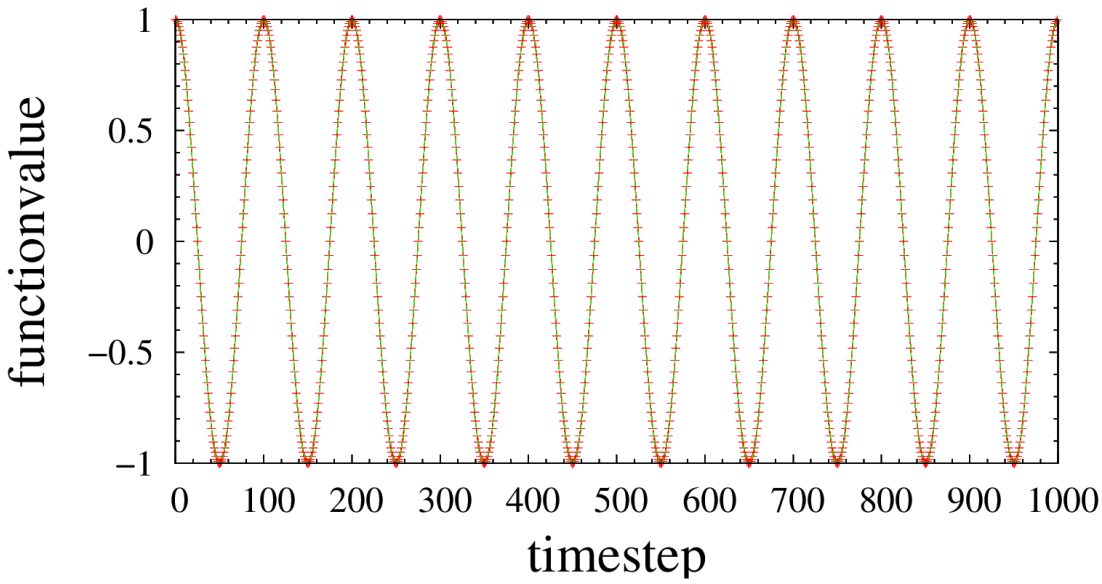


FIGURE 3.2: The original function given by a green line and the forth and back Fourier transformed function given by the red crosses.

Chapter 4

Method

In this fourth chapter we will shortly describe all the different steps that we used to calculate the phonons of graphene at different temperatures. We will also show how we derived the bending rigidity κ from the phonons. We will start this chapter with giving an overview of the method and after that we will shortly describe each step while sometimes referring back to the information in previous chapters.

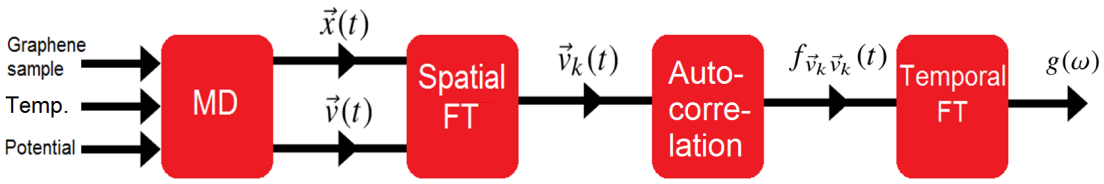


FIGURE 4.1: Schematic overview of all the steps of the used method including the computational methods MD simulations, autocorrelation and Fourier Transformation (FT) illustrated by the red boxes. Arrows indicate the input parameters and the output variables needed for and resulting from these computational methods.

A schematic overview of the method we have used is shown in figure 4.1. For the computational methods we made use of existing FORTRAN90 codes for MD simulations, but we wrote new FORTRAN90 codes for FT and autocorrelation. Note that the functions in figure 4.1 are displayed as if they are continuous, but we will see in the next paragraphs that we need to use discrete functions instead. In the next sections we will describe each step from the process in figure 4.1 in more detail.

4.1 Creating the input

We need to start with defining the input parameters for which we want to run our calculations. This includes three main things. First of all we need a sample of graphene for which we want to know the phonon dispersion. It is important to choose this carefully because as seen in section 1.1 the wave vectors are derived from the crystal structure of graphene. If we want accurate data its best to choose a sample that is as big as possible, because this provides more discrete values of the wave vectors. Another important thing to take into account is that the size of the sample also affects the speed of the calculations which grows with N, the number of atoms.

In our case we chose to make a sample of $N = 800$ atoms which is as big as the sample used in [14]. With this sample relatively fast computation speeds can be reached. After we have defined the number of atoms in the sample, we need to make sure that we make a roughly squared computational cell to cover equally the whole Brillouin zone. This resulted in the 20×20 computational cell of 800 atoms shown in figure 4.2.

From the computational cell we can derive the wave vectors with help of equations (1.5) and (1.6) in section 1.1. In this case we do not have just one unit cell, but we have $n = 20$ unit cells.

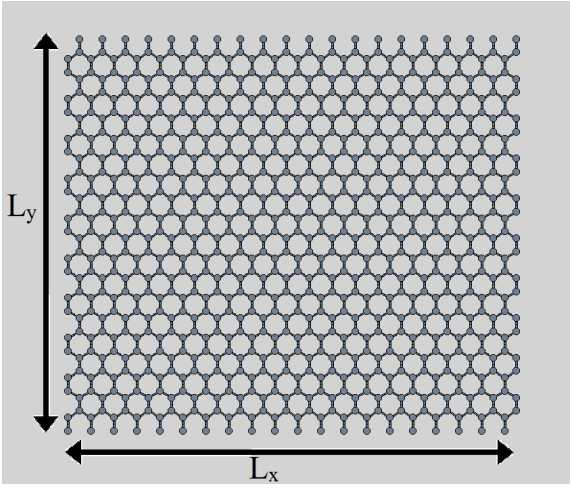


FIGURE 4.2: The graphene sample of 20×20 unit cells that we used for our calculations. L_x denotes the length and L_y denotes the width of the sample, these values vary for different temperatures as shown in 4.2.

For a sample of n unit cells the general form of wave vectors (1.5) and (1.6) is partitioned into n segments as shown in figure 1.2 in section 1.1. Adding these two wave vectors for the different directions together gives

$$\boldsymbol{k}_{ij} = \frac{2\pi}{a\sqrt{3}n} \begin{bmatrix} i \\ -i\frac{1}{\sqrt{3}} + j\frac{2}{\sqrt{3}} \end{bmatrix}. \tag{4.1}$$

Indexes i and j in equation 4.1 enumerate the partition sections along the reciprocal lattice vectors $\mathbf{b}_1$ and $\mathbf{b}_2$. The values of these indexes are derived from the coordinates of the high symmetry paths given in table 1.1 which result in the values reported in table 4.1.

TABLE 4.1: Index values and the spacing between the k_x and k_y values for the different symmetry paths for our computational cell of $n = 20$ unit cells. For the calculation of the spacing we used the lattice vector at $T = 300$K.

Path	**i values**	**spacing $\mathbf{k_x}$ [Å^{-1}]**	**j values**	**spacing $\mathbf{k_y}$ [Å^{-1}]**
$\boldsymbol{\Gamma K}$	$0 \leq i \leq \frac{2n}{3}$	0.127732	$j = \frac{i}{2}$	0.0
$\boldsymbol{KM}$	$\frac{\sqrt{3}n}{2} \leq i \leq \frac{2n}{3}$	0.127732	$j = n - i$	0.221239
$\boldsymbol{M\Gamma}$	$0 \leq i \leq \frac{\sqrt{3}n}{2}$	0.127732	$j = i$	0.073747

For creating the sample it is important to know that the lattice parameter is temperature dependent. In fact, the lattice parameter decreases with temperature contrary to 3D materials like diamond. This fact is predicted theoretically [11] and measured up to $T = 400$K [38], but only for $T > 500$K the lattice parameter of graphene starts to grow. Dr. Jan Los calculated for our study the lattice parameter at different temperatures using Monte Carlo simulations in the NPT ensemble. In this way, setting $P = 0$, he could find the equilibrium area of the sample at a given temperature. This resulted in different samples sizes at different temperatures as in shown table 4.2.

The temperature is the second input parameter. We already know the dispersion at $T = 0$K and moreover, performing MD simulations is impossible at this temperature. For this reason it is for us not interesting to look at $T = 0$K. On the other hand we are interested in temperature dependence and since we have just seen that the lattice parameter varies strongly between $T = 100$K and $T = 2000$K we therefore chose to perform our calculations at $T_1 = 100$K, $T_2 = 300$K, $T_3 = 1000$K and $T_4 = 2000$K. Furthermore, simulating at these temperatures makes it easy to compare our results

to other studies like the study of Zakharchencko [9]. In that study the temperature dependence of the bending rigidity was calculated from the power law behaviour of normal-normal correlation functions, according to the theory of membranes.

TABLE 4.2: The values of the lattice parameter and the sizes of the samples at different temperatures.

Temperature[K]	**Lattice Parameter[Å]**	**length L_x[Å]**	**width L_y[Å]**
0	2.4595		
100	2.4578	49.16	42.57
300	2.4566	49.13	42.55
1000	2.4571	49.14	42.56
2000	2.4633	49.27	42.67

The last thing we need as input for our calculations is a potential that describes the interactions between the carbon atoms. For this we used LCBOPII as described in section 2.1.

4.2 Performing MD simulations

The next step is to calculate the trajectories

$$\vec{r}_i(k\Delta t) \tag{4.2}$$

and the velocities

$$\vec{v}_i(k\Delta t) \tag{4.3}$$

of all the single atoms $i = 1, 2, ..., N$ using MD simulations based on the velocity Verlet algorithm as described in section 2.2. Note that the output is in discrete sequences because Verlet integration is performed using finite time steps Δt. This time step and the total duration of the simulations $M\Delta t$ can be chosen. For our calculations we chose time steps between $\Delta t = 0.05$ fs and $\Delta t = 0.2$ fs which yields energy conservation as it should for an isolated system. Note that in equation 4.2 and 4.3 we save the results every 10 time steps $k = 0, 10, 20, ..., M$ because this reduces the amount of memory space we use. Moreover, we will see when we discuss the results in the next chapter that saving the data more than every 10 time steps is not necessary.

As discussed in section 2.2.2 simulating longer will result in smaller statistical errors. But we do not have infinitely long time for our simulations, therefore we chose a total of $M = 500000$ time steps resulting in a simulation of in total 25 picoseconds (ps). Another reason for this duration is that we wanted to measure the lowest phonon frequencies of graphene close to $\omega = 0$. From equation 3.6 we can derive $\Delta\omega \approx 0.13$ cm^{-1}, so we can indeed measure very low phonon frequencies. A test simulation revealed that it took a standard budget computer approximately one hour to run the total MD simulation which is a reasonable time.

4.3 K-space velocity distribution

To go from the velocities and positions of all atoms in the lattice to the dynamics of the whole lattice we compute, the k-space velocity distribution

$$\vec{v}_{\vec{k}}(k\Delta t) = \sum_{j=0}^{N} \vec{v}_j(k\Delta t) e^{-i\vec{k}\cdot\vec{R}_j}. \tag{4.4}$$

In this k-space velocity distribution given by equation 4.4 is summed over all the atoms $j = 1, ..., N$ which results in an average over all the velocity contributions of the single atoms to the lattice.

This is actually a transformation of the lattice from real space to reciprocal space, in other words a discrete Spatial Fourier transformation as discussed in section 3.2. This transformation to reciprocal space is necessary since we are interested in the wave-like behaviour of the whole lattice. Notice that for each wave vector $\vec{k}$ defined in equation 4.1 we need to perform this transformation separately.

4.4 Obtaining the eigenfrequencies

The following step is to remove noise from the k-space velocity distribution 4.4 and extract a sequence with the main frequencies. To this purpose we use the autocorrelation sequence from section 2.3. Rewriting equation 2.9 with our k-space velocity distribution 4.4 results in the k-space velocity autocorrelation sequence (kVACS) given by

$$\hat{r}_{\vec{v}_k \vec{v}_k}(m) = \frac{1}{M-m} \sum_{n=0}^{M-m-1} \langle \vec{v_k}(n+m)\vec{v_k}(n) \rangle \tag{4.5}$$

with $m = 0, 1, ..., N-1$ the different time delays and M the total number of time steps. Note that also this sequence has to be calculated for each different k-vector in equation 4.1. We finally normalize the autocorrelation sequence:

$$\frac{\hat{r}_{\vec{v}_k \vec{v}_k}(m)}{\hat{r}_{\vec{v}_k \vec{v}_k}(0)}. \tag{4.6}$$

According to the Wiener-Khintchine theorem from section 3.2.2 the discrete temporal FT 3.7 of this autocorrelation sequence gives us, for each k-vector, the power spectral density (PSD). Reading out the peaks of these spectra eventually provides us with the phonon frequencies of our graphene sample $\omega(k)$.

For these last three steps after MD simulations we wrote a new program in FORTRAN90 that reads out the MD simulations results and prints the power spectral density for each k-vector to a file. This written program can be found in Appendix C. When writing this program we tested each step thoroughly and looked carefully at the results as shown in the next chapter.

4.5 Determining the dispersion and obtaining the bending rigidity

From these power spectral densities we determined for each k-vector the phonon frequencies in the acoustic and optical branch resulting in the phonon dispersion of graphene. For obtaining the bending rigidity as described in section 1.3 we fitted the parabola

$$f(x) = cx^2 \tag{4.7}$$

against the frequencies of the lowest k-vectors for the acoustic out of plane mode as shown in figure 4.3. By using equation 1.8 we can then extract $\kappa = c^2 \rho_{2D}$ as shown in appendix A.

To determine for which values of the k-vector this quadratic fit to the ZO-mode is appropriate, and thus the relation for the bending rigidity as described in section 1.3 holds, we first plot the frequencies ω against $|k|^2$ as shown in figure 4.4. We then look for which k-vector this fit is not linear anymore and this gives us the highest k-vector we could use for the fit of equation 4.7.

To obtain κ for different temperatures we redo the whole process illustrated in figure 4.1 at the temperatures given in section 4.1. In the end we plot the resulting values for κ against the different temperatures.

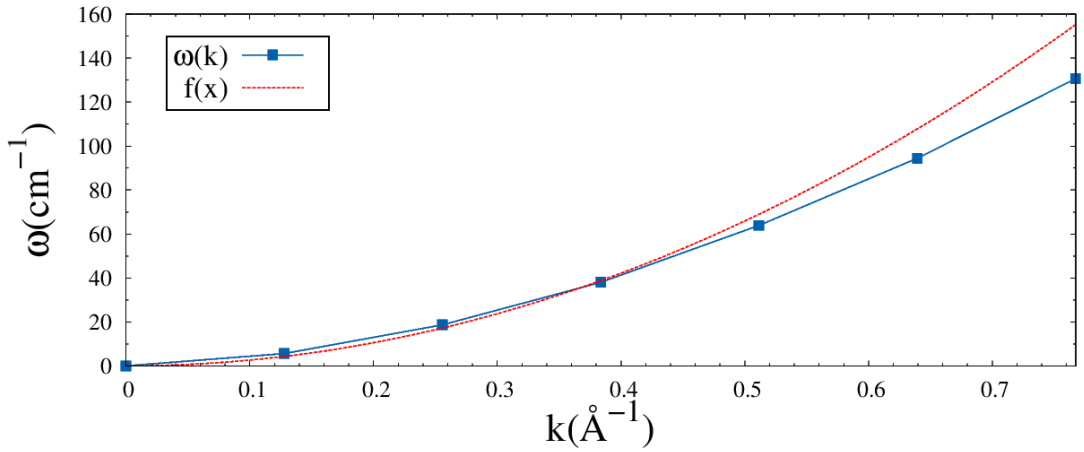


FIGURE 4.3: Fitting the parabola f(x) from equation 4.7 to the measured ZA phonon mode of graphene at $T = 300K$. This fit was based on the first 4 measured data points.

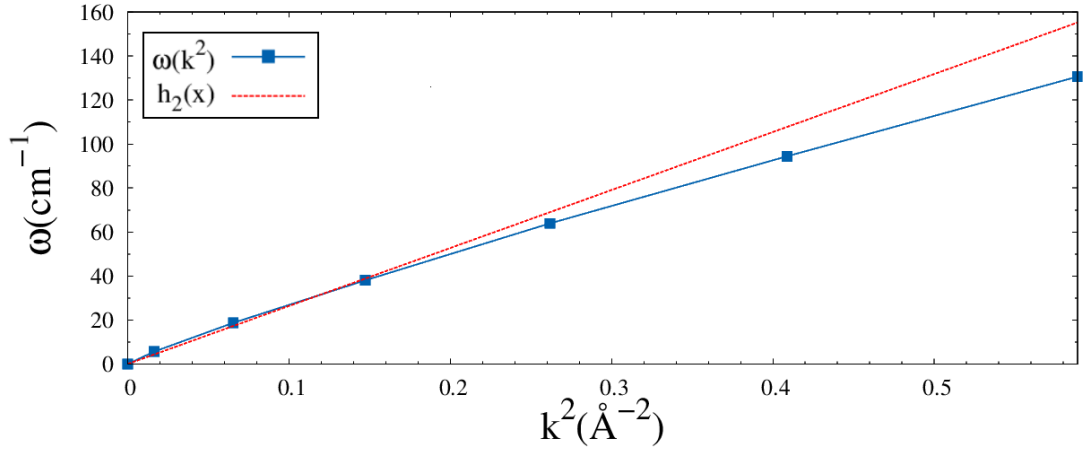


FIGURE 4.4: Fitting a linear line to the measured ZA phonon mode of graphene at $T = 300K$ for k-vector squared to determine for which value of $\vec{k}$ the ZA phonon is still quadratic. We notice that the first 4 measured data points, including (0,0) are almost linear.

Chapter 5

Results

In this chapter we present the results that we obtained from our simulations and calculations as described in chapter 4. We show the results step by step as in figure 4.1 from the former chapter. At every step we explain shortly some interpretations from the results and we also give the necessary conditions.

5.1 MD simulations

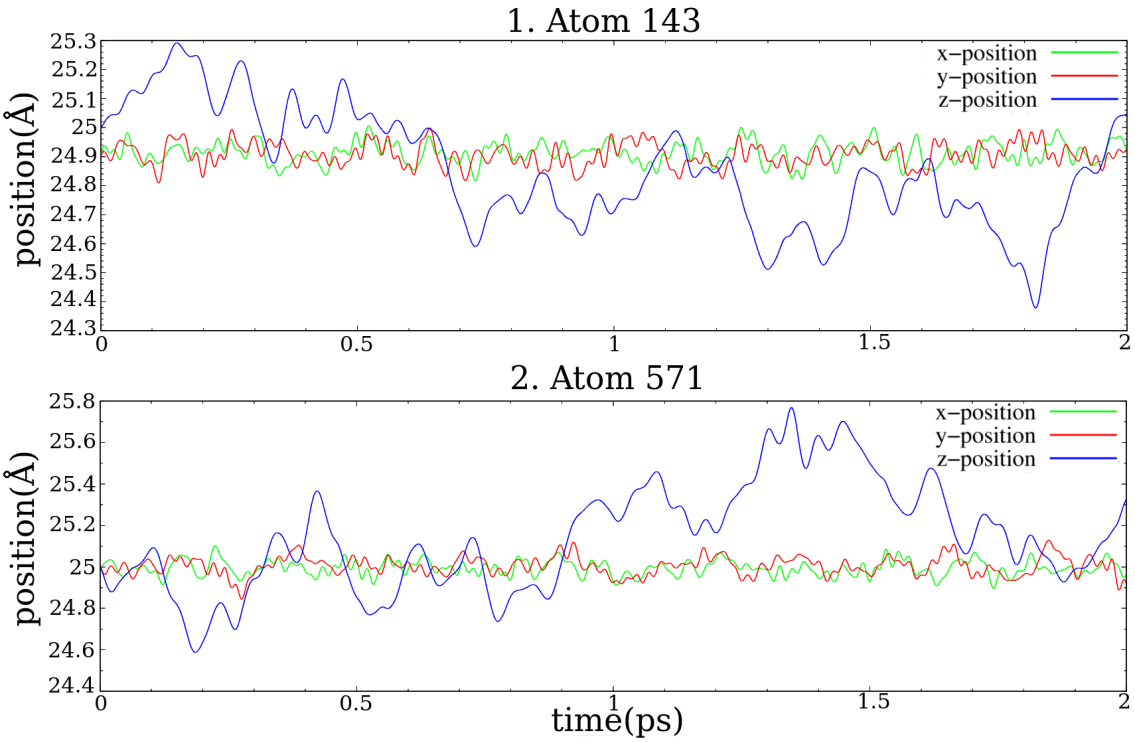


FIGURE 5.1: Trajectories in the three spatial directions of atom 143 and atom 571 in our 800 atom graphene sample. The trajectories are derived from a $t = 2$ ps MD simulation with time step $\Delta t = 0.2$ fs. The trajectories are shifted to the same position region so that we could easily compare them to each other by means of the fluctuations.

In figure 5.1 we show the results for the trajectories of two atoms during the time derived from the MD simulations at $T = 300$ K with time step $\Delta t = 0.2$ fs. We shifted the three curves for the x-, y- and z-direction by constant values so that the curves are plotted in the same position region and we could easily compare the behaviours for the different directions. The first thing which is interesting is that the fluctuations of the out-of-plane, z-direction, position (about 0.9 – 1.0 Å) of both atoms are much higher than the in-plane, x- and y-direction, fluctuations of the positions (about 0.1 – 0.2 Å). The z-fluctuations are actually on the same order of magnitude as

the interatomic distances ($d_i \approx 1.42$ Å). Referring back to section 1.4, we see that this as expected since for this reason the harmonic approximation is not valid and we need to consider anharmonic effects.

Moreover, we see in both graphs that the positions vary periodically in time, so the carbon atoms of graphene are in periodic motion as expected. We can also see that the frequency of the x- and y-position is much higher than that of the z-position, this is also as expected. So everything from figure 5.1 seems to be in accordance with the theory about the dynamics of the atoms of graphene, therefore we believe that the positions were correctly computed in the MD simulation process.

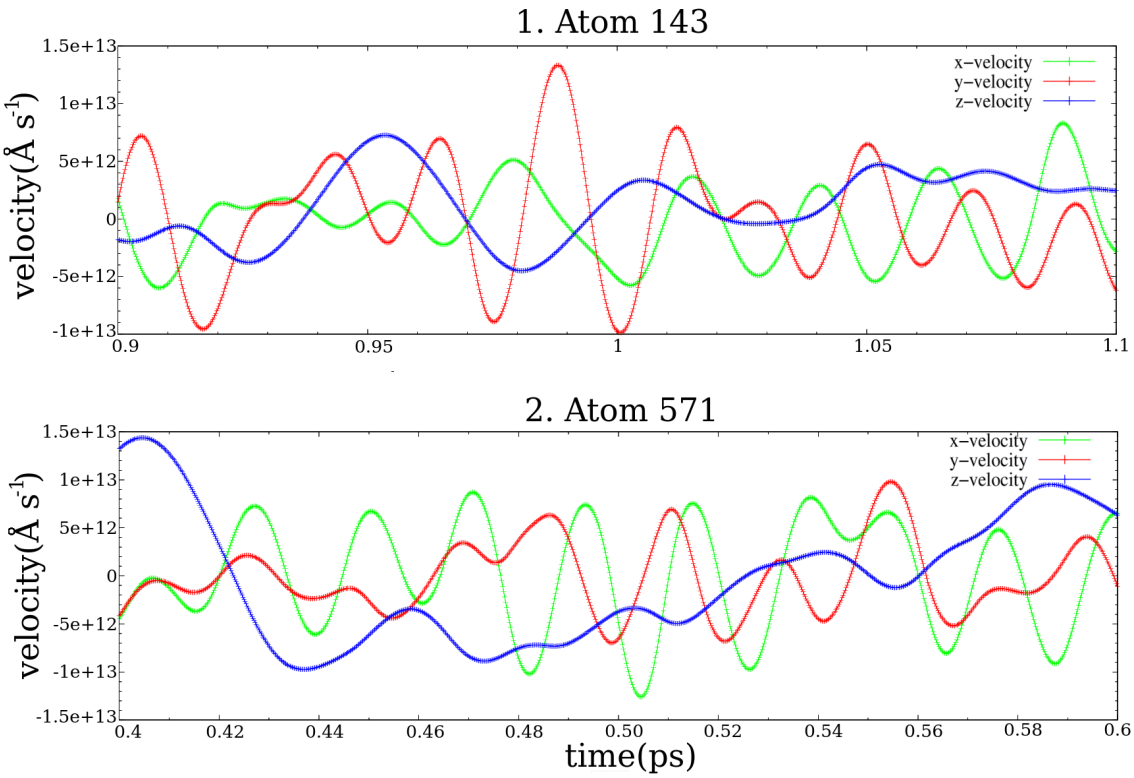


FIGURE 5.2: Velocities in the three spatial directions of atom 143 and atom 571 in our 800 atom graphene sample. The velocities are derived from a $t = 2$ ps MD simulation with time step $\Delta t = 0.2$ fs.

In figure 5.2 we plotted the results for the velocities, derived from the same simulation. Now we see that also the velocities vary periodically in time, however the magnitude of the z-velocity is not much higher than the x- and y-velocities. The magnitudes are approximately 0.5×10^{13} Å s^{-1}, which is in accordance with the equipartition theorem shown in equation 2.2 which predicts a velocity of approximately 0.46×10^{13} Å s^{-1} for carbon atoms at room temperature.

For this reason we think that also the velocities are correctly computed. To check this even better, we plotted the x-velocity and x-position of one atom in one graph as shown in figure 5.3. Carefully looking at figure 5.3 reveals that the period of the position is approximately the same as the period of the velocity, which in accordance with the theory.

One last check is to look at the exact frequency, from figure 5.3 we can easily determine the period, which is approximately $P = 0.32 - 0.35$ ps. Calculating the frequency from this period gives $\omega \approx 100$ cm^{-1}. Which is on the right order of magnitude when comparing to the phonon dispersion of graphene given in figure 1.3.

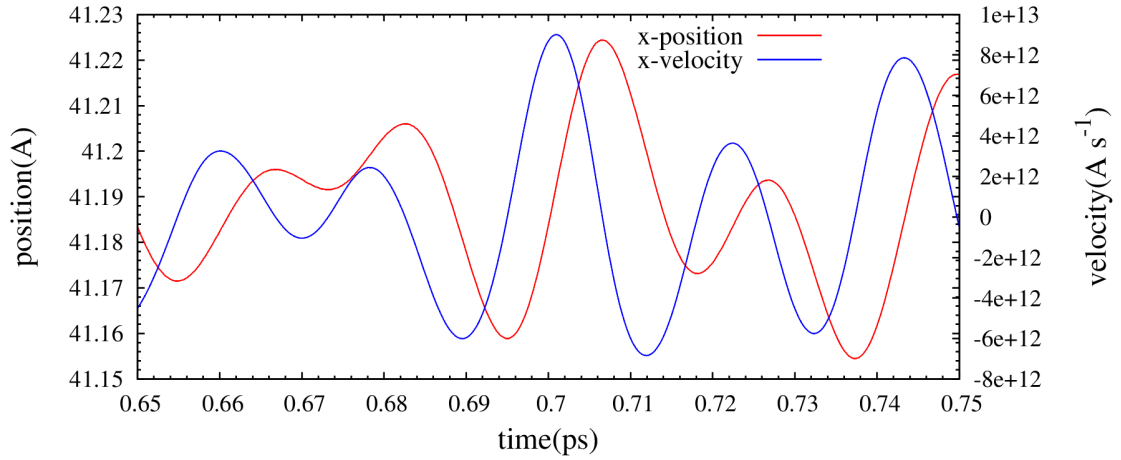


FIGURE 5.3: x-component of the position and velocity of atom 571 from our 800 atom graphene sample. The velocity and position are derived from a $t = 2$ ps MD simulation with time step $\Delta t = 0.2$ fs.

5.2 k-space velocity distributions

5.2.1 Defining the k-vectors

To calculate the k-space velocity distributions, we first need to calculate the values of the k-vectors according to equation 4.1. Therefore, in the first step of the program, we read the lattice parameter and the simulation cell into the program and then derive the corresponding k-vectors. As said before, this lattice parameter is different for different temperatures, for this reason we get slightly different k-vectors at the different temperatures. Figure 5.4 shows the results of this calculation for the k-vectors at $T = 300$ K. It can be seen that the Brillouin zone of graphene is correctly reproduced when we compare it to figure 1.2.

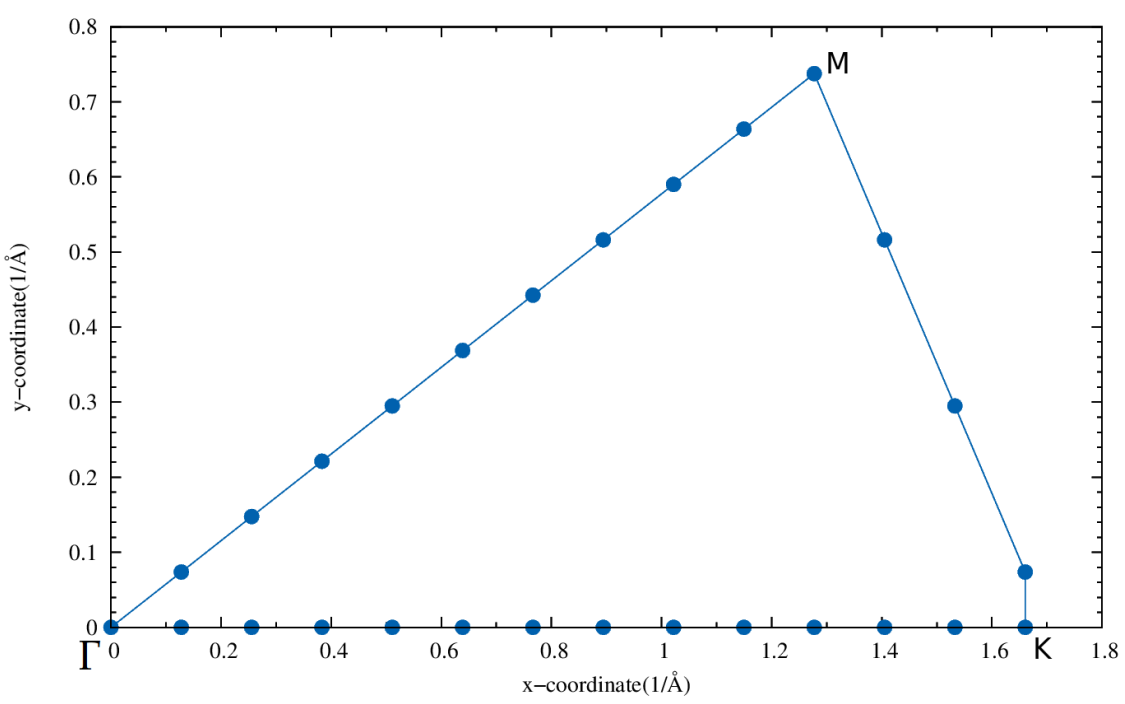


FIGURE 5.4: Plot of the used k-vectors for the spatial discrete Fourier transformation. This result is derived from reading in the simulation cell at $T = 300$ K.

For this sample the boundaries of the Brillouin zone could almost exactly be reached. The $\boldsymbol{K}$-point has coordinates $(1.660522, 0)$ which is a good approximation of $\boldsymbol{K}$ in equation 1.7. The coordinates of the $\boldsymbol{M}$-point are even better, it exactly lies on the Brillouin zone boundary.

5.2.2 k-space velocity distributions

Since we now have the correct wave vectors, we can compute the k-space velocity distributions as given in equation 4.4. In figure 5.5 the result of this computation is given for k_6 in all three spatial directions. Looking more closely at this graph reveals that the magnitude of the k-space velocity distribution in the z-direction is much higher than in the in-plane directions. This interesting behaviour was also found when looking at other k-vectors. Possibly this is due to the fact that the Z modes are more easily excited due to their low frequency.

Furthermore, from the k-space velocity distributions we can determine the frequencies of the oscillations. This results in $\omega \approx 119\ \mathrm{cm}^{-1}$ for the z-frequency for k_6. This is on the right order of magnitude for k_6 when we compare it to figure 1.3. Moreover, in figure 5.5 can be seen that the frequencies for the in-plane velocity distributions are, as expected, higher than the out-of-plane frequency.

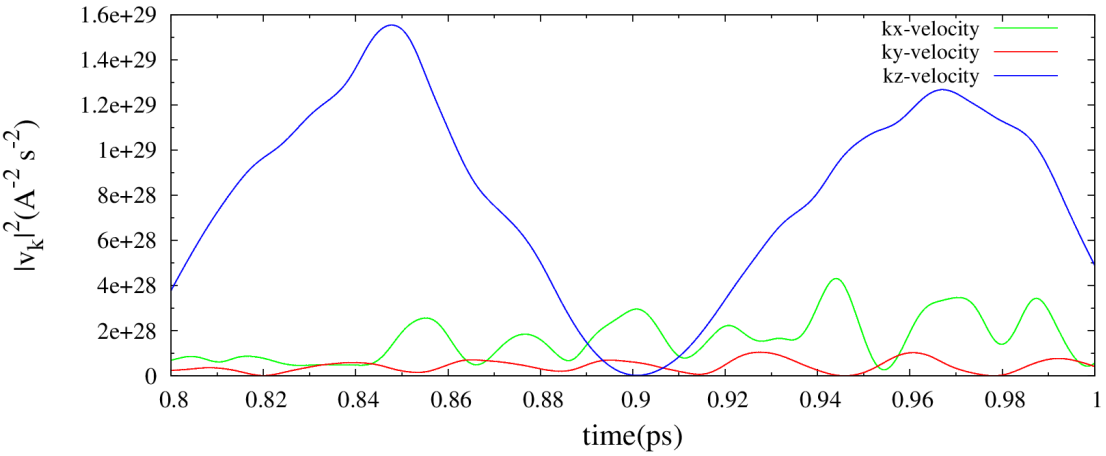


FIGURE 5.5: K-space velocity distribution for k_6 derived with equation 4.4 using the results of a $t = 25$ ps MD simulation with time step $\Delta t = 0.05$ fs at $T = 300$ K.

To examine the behaviour for different wave vectors, we plotted in figure 5.6 the k-space velocity distributions in the z-direction for all the different k-vectors that we have. This graph looks a bit messy, but we can clearly recognize patterns. We could for example see that the resulting k-space velocity distribution for every k-vector looks periodic, but all with a different frequency. This is as expected because otherwise we would have a constant dispersion relation. It is also interesting to notice that the amplitude differs for the different k-vectors. We can interpret this in the following way: the different k-vectors have all a different contribution to the vibrations of the lattice, so the lattice will vibrate as a superposition of these different k-vector contributions.

5.3 k-space velocity autocorrelation sequence

In the next step we wanted to investigate if the k-space velocity distributions that we have obtained in the former section are random or contain some main frequencies. The graph in the top panel of figure 5.7 shows that it is diffecult to see whether there are some main frequencies in the k-space velocity distribution or not. Carefully looking reveals for example that this k-space velocity distribution is not smooth, but it has some weirdly distributed sharp peaks. Also we can see in the top panel of figure 5.7 that the amplitude varies enormously in time.

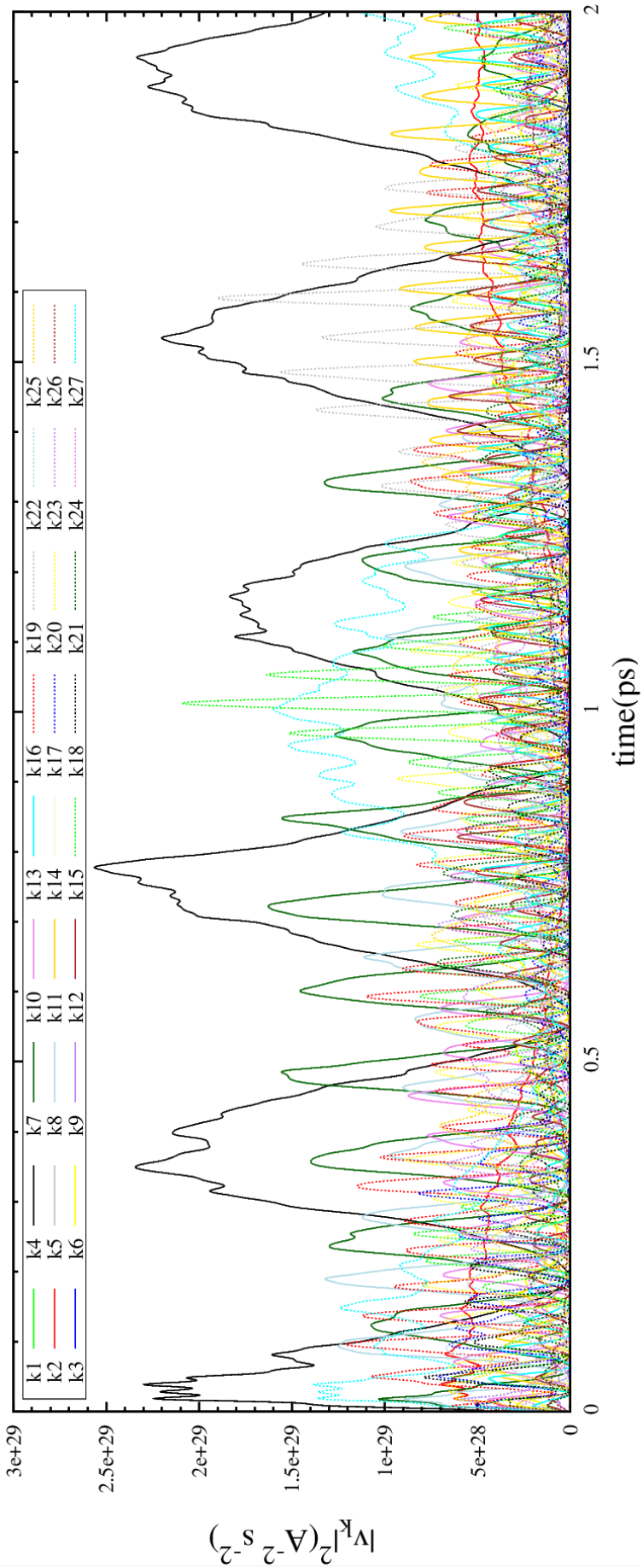


FIGURE 5.6: K-space velocity distributions in the z-direction for all k-vectors. These results were derived with equation 4.4 using the results of a t = 25 ps MD simulation with time step Δt = 0.05 fs at T = 300 K.

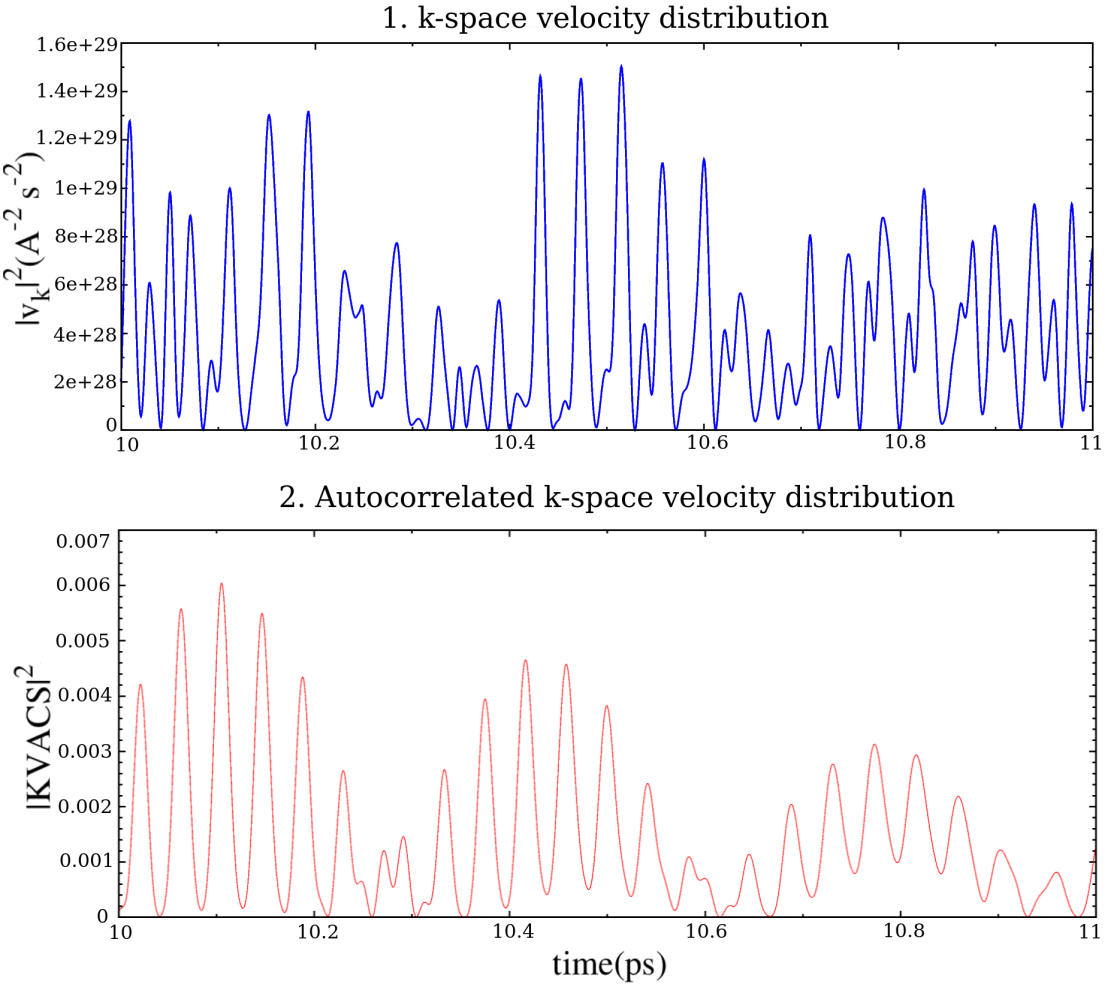


FIGURE 5.7: Top panel: 1. K-space velocity distribution for k_3 at $T = 2000$K derived with equation 4.4 using the results of a $t = 25$ ps MD simulation with time step $\Delta t = 0.05$ fs. Bottom panel: 2. the autocorrelated k-space velocity distribution k_3 at $T = 2000$K derived with equation 4.5 using the k-space velocity distrubution for k_3 at $T = 2000$K from the top panel as input.

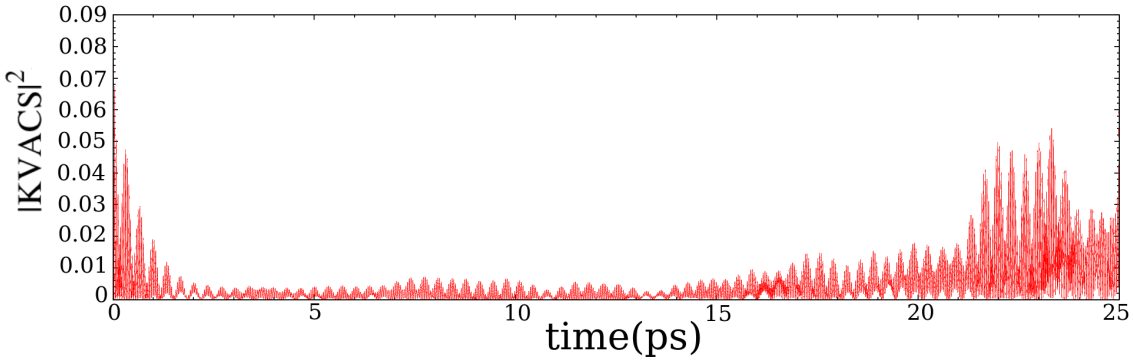


FIGURE 5.8: Whole kVACS for k_3 at $T = 2000$ K derived with equation 4.5 using tthe k-space velocity distrubution for k_3 at $T = 2000$K from the top panel in figure 5.7 as input. We can recognize the wavepackets. Notice the cosine behaviour at the beginning of the sequence.

For these reasons, as proposed in section 4.4, we calculate the k-space velocity autocorrelation sequence (kVACS) as given in equation 4.5 to obtain a signal with the main phonon frequencies. Figure 5.7 shows in the bottom panel the kVACS for k_3. We can clearly see that the resulting kVACS signal shows much clearer periodicity than the original signal in the top panel. It looks like as if the kVACS is a superposition of two main frequencies since we recognize large wave packets synthesized of a faster oscillating harmonic function. These two distinct frequencies correspond to the acoustic mode, the wave packet, and the optical mode, the fast oscillating harmonic function. Deriving the values of these frequencies from the graph in figure 5.7 gives $\omega_1 \approx 42\text{cm}^{-1}$ for the acoustic mode and $\omega_2 \approx 833\text{cm}^{-1}$ for the optical mode. Comparing these frequencies to [14] and figure 1.3 reveals that these results are in the right order of magnitude.

Finally to check if we computed this step correctly, in figure 5.8 we show the graph of the whole kVACS for k_3. This clearly looks like a cosine signal because of the maximum at $t = 0$. We know from autocorrelation functions that they always look like a cosine because at $t = 0$ every data point is correlated to itself due to the zero time delay. On the other hand, we see that the amplitude increases after approximately $t = 20$ ps. The fact that the delay times in that region are really large compared to the total time of the signal could explain this increasing behaviour since correlation is difficult for such large delay times. To accurately determine the phonon frequencies, it is therefore the best to analyze the frequency between $t = 3$ ps and $t = 15$ ps.

5.4 phonon dispersion

5.4.1 Fourier transformation

After the autocorrelation we can eventually determine the actual phonon frequencies. We perform a temporal DFT of the autocorrelation sequence from equation 4.5 for each different k-vector in every spatial direction as given by equation 3.8. This results in an enormous amount of power spectral densities (PSD's) from which we can identify the peaks which correspond to the phonon frequecies. Figure 5.9 shows the resulting PSD's for the three smallest wave vectors of the ZA phonon at $T = 300$ K. We can clearly recognize a sharp peak for each wave vector. The width $\Delta\omega \approx 2.5\ \text{cm}^{-1}$ of the peaks is about the same for all three wave vectors. For these low acoustic modes the peak is rather broad, in the conclusion we will discuss if we can somehow make the peaks sharper resulting in a higher accuracy.

By comparing the three graphs belonging to the three increasing k-vectors shown in figure 5.9, one can see that the peak shifts to higher frequencies when increasing the k-vector. This is exactly as expected because on the $\Gamma\mathbf{K}$-path the frequencies have to increase as shown in figure 1.3. This comparison also shows a somewhat quadratic behaviour since the amount by which the frequency increases becomes bigger when we look at increasing k-vectors. For example the frequency ω increases by approximately 15cm^{-1} when $k_1 \mapsto k_2$ and approximately 20cm^{-1} when $k_2 \mapsto k_3$.

Note that we show in figure 5.9 only the PSD's of the lowest values of ω, but when we look at the whole spectrum we can recognize more distinct peaks as shown in figure 5.10. Each different peak corresponds to one phonon mode, however it is not always trivial to determine which peak belongs to which phonon mode as shown in the both spectra in figure 5.10. In particular the in-plane phonon modes as shown in the top panel in figure 5.10 are difficult to recognize due to the fact that the in-plane k-vectors are not completely well defined since we have a hexagonal lattice. For this reason we find 4 peaks, two for each in-plane phonon branch. Each peak corresponds to a frequency of either a longitudinal mode or a transverse mode.

Another difficulty occurs when identifying the frequencies near the high symmetry points, for example the **K**-point as shown in the bottom panel in figure 5.10 for the longitudinal modes. Looking closely reveals that there are two peaks very near to each other at about 1200cm^{-1}, but it is difficult to determine which peak corresponds to the acoustic longitudinal mode and which peak corresponds to the optical longitudinal mode. Since we know that before and after the the high

symmetry point **K** the optical frequency is higher, we have decided that the peak with the highest frequency corresponds to the optical mode.

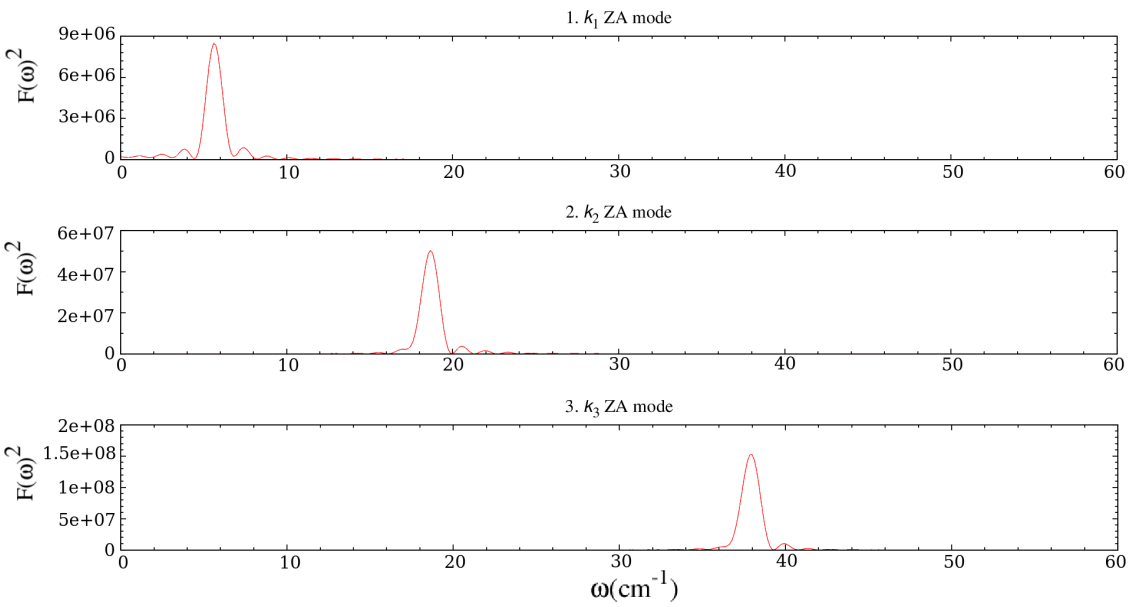


FIGURE 5.9: PSD's for the ZA mode at $T = 300$K with wave vectors: 1. k_1, 2. k_2 and 3. k_3. You can see that the peak from 1. shifts from $\omega_{k_1} \approx 5\text{cm}^{-1}$ to $\omega_{k_2} \approx 18\text{cm}^{-1}$ for 2. and to $\omega_{k_3} \approx 38\text{cm}^{-1}$ for 3. The used time step for the $t = 25$ ps MD simulations is still $\Delta t = 0.05$ fs

5.4.2 First test results

Before giving the results for the analyzed Fourier spectra from the former step, we first show how important it is to choose the right time step and the right sample by showing some of the first test results. In figure 5.11 we show the result that we first got by using 10000 time steps of $\Delta t = 0.2$ fs, which are not many time steps. We can observe two main failures in this graph. Firstly the dispersion the lowest k-vectors looks linear while it must be quadratic. This can be explained by the width of the time step which is quite big. Therefore the values of ω that could be analyzed were in integer values of $\Delta\omega = 16.68$ cm^{-1}, which is huge and for this reason small difference in the frequency for the lowest k-vectors are impossible to measure. Another explaination for this incorrect linear dispersion is that within the MD simulations we used a lattice parameter based on the interatomic distance $a = 1.42$Å which is not exactly the same at $T = 300$ K because, as seen, the lattice parameter varies for different temperatures.

The second mistake that can be observed in figure 5.11 is the peak at k_{18} which corresponds to the highly symmetric **M**-point. For the same reason as discussed in previous section, it can sometimes be difficult to distinguish two peaks around high symmetry points. Due to the rough values of $\Delta\omega$ that can be measured with this time step, it could happen that we cannot distinghuish or recognize two peaks at all meaning that we only observe one of the two peaks which is thus not accurate enough to observe all the branches.

From this we can learn that the time step and duration of the MD simulations are indeed very important, and moreover we need to make sure that we use the right lattice parameter at the corresponding temperature.

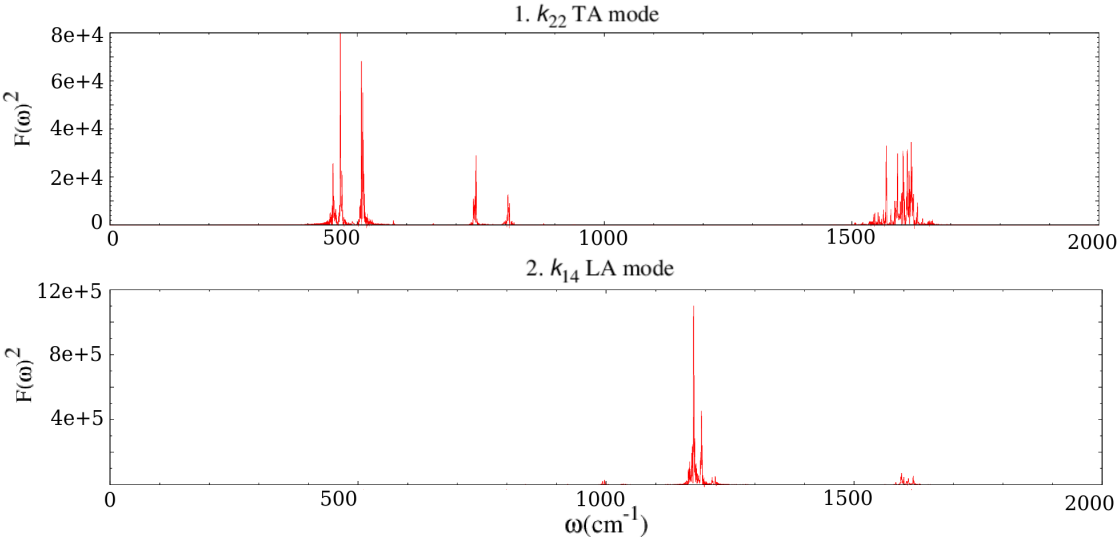


FIGURE 5.10: PSD's for 1. k_{22} at $T = 300$K in the y-direction and 2. for k_{14} at $T = 300K$ in the x-direction. In 1. we can recognize in total 4 or 5 peaks. In 2. we see just one peak with a small peak besides it.

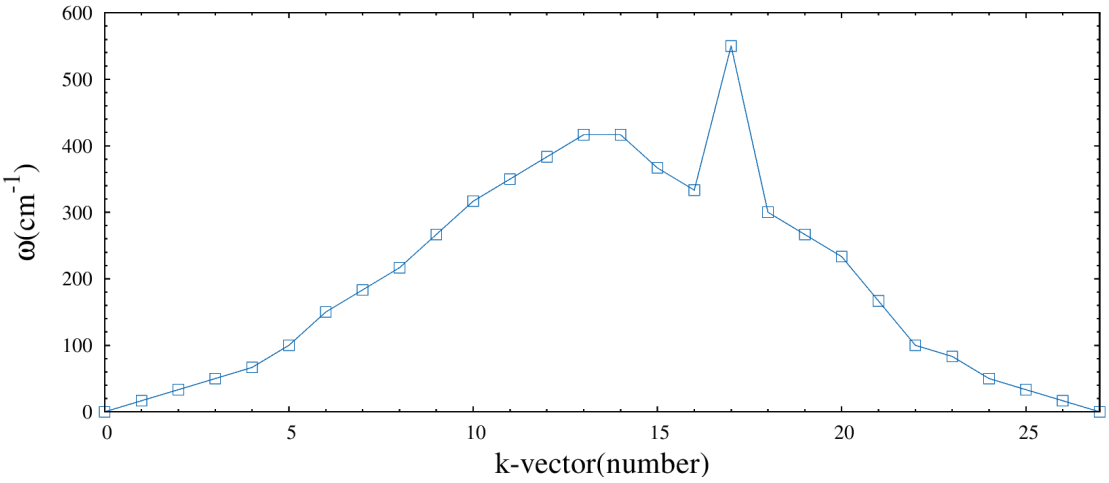


FIGURE 5.11: ZA phonon determined from a simulation runned at 10000 time steps of $\Delta t = 0.2$ fs at $T = 300$K. Contrary to the other MD simulations as explained in 4.2, in this simulation the results were saved for each time step seperately.

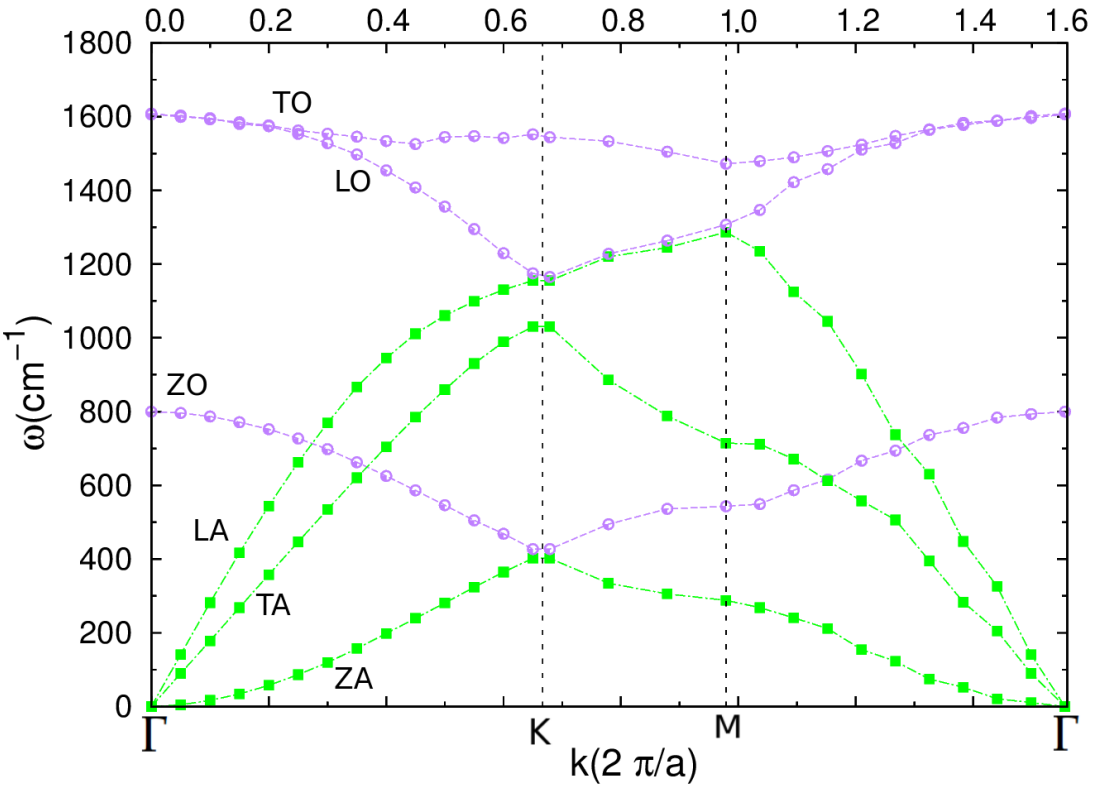


FIGURE 5.12: Phonons of our 800 atom graphene sheet for all branches at $T = 100$K. Based on MD simulations with time step $\Delta t = 0.05$fs.

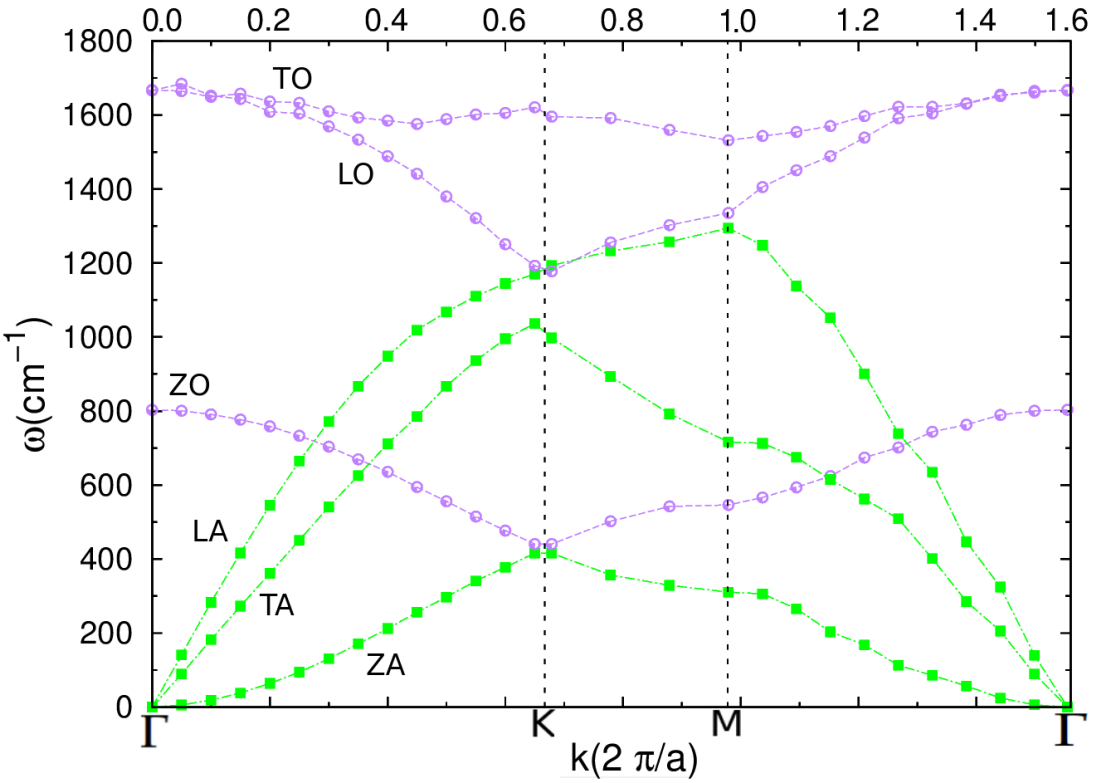


FIGURE 5.13: Phonons of our 800 atom graphene sheet for all branches at $T = 300$K. Based on MD simulations with time step $\Delta t = 0.05$fs.

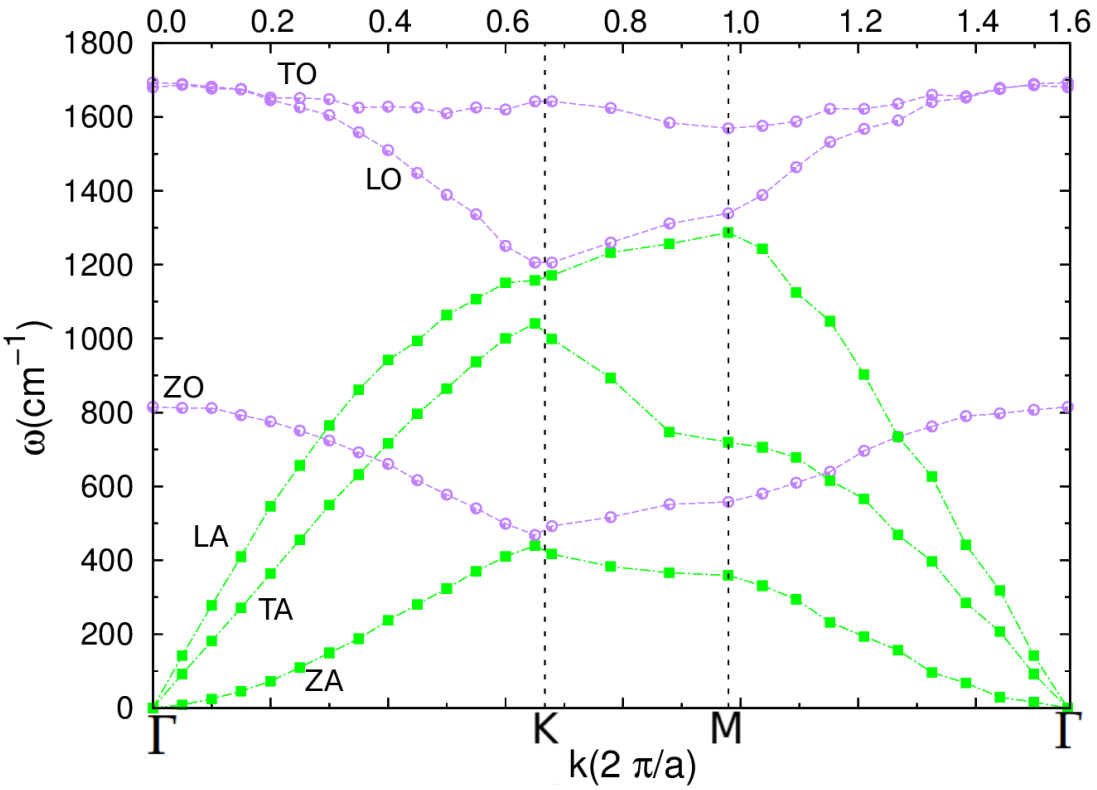


FIGURE 5.14: Phonons of our 800 atom graphene sheet for all branches at $T = 1000$K. Based on MD simulations with time step $\Delta t = 0.05$fs.

5.4.3 Phonon dispersion

By evaluating the PSD's for all the 27 different k-vectors in all three spatial directions we can derive all the frequencies necessary to draw the phonon dispersion of graphene. In figures 5.12, 5.13 and 5.14 we show these resulting phonon dispersions for our 800 atom graphene sample at the three different temperatures as defined in section 4.1. At first sight, our results seem to look exactly the same as the phonon dispersion given in figure 1.3. The results on the $\Gamma\mathbf{K}$-path really look fine without much deviations from the trends. However on the $\mathbf{M}\Gamma$-path we observe some wiggles in the lines, especially at high temperatures. The reason for this could be that the points on the $\mathbf{M}\Gamma$-path are really close to the Brillouin zone boundary and therefore boundary effects need to be taken into account. For our research this is not a huge problem since we want to derive the bending rigidity κ from the $\Gamma\mathbf{K}$-path. Some general results of these dispersion graphs for the different temperatures can be found in table 5.1.

TABLE 5.1: Results for dispersionproperties of graphene at the four different temperatures

Temperature[K]	$\omega_{\Gamma_{E2g}}$	ω_K **ZA Mode[cm^{-1}]**	ω_M **ZA Mode[cm^{-1}]**
100	1607	402	288
300	1667	416	310
1000	1687	419	359
2000	1611	450	394

Results at $T = 2000$K including a try for the error margins

As mentioned in section 4.1 we are also interested in the phonon properties at $T = 2000$ K. We did not calculate the whole phonon dispersion again at this temperature, but we just looked at the most important phonon frequencies which are interesting for our study. Some general phonon frequencies calculated at this temperature at the high symmetry points are reported in table 5.1. Moreover, we took the errors into account at this temperature to see if they have much influence on our results. We estimated the errors by the total width of the peaks in the PSD's. This results in the graph shown in figure 5.15. We can recognize that the errors are very small compared to the values of the frequencies, however a more argued estimate of the errors as explained in section 2.2.2 is necessary if we want to be sure that our found values are accurate.

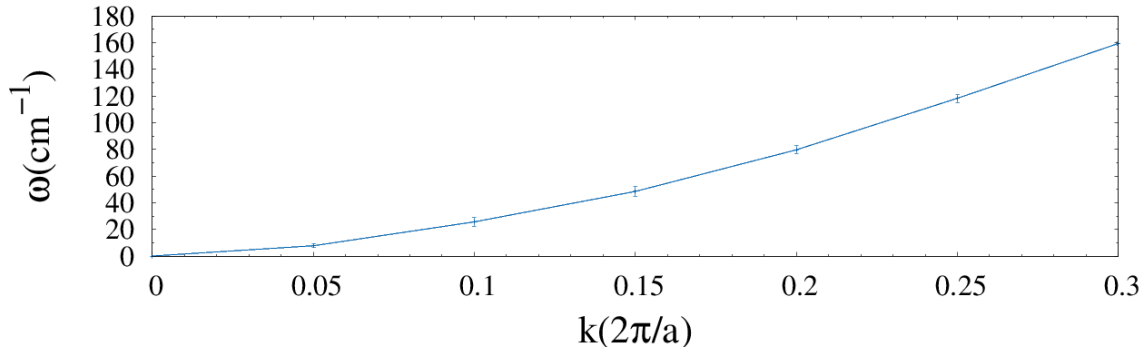


FIGURE 5.15: Results of the phonon dispersion of our 800 atom graphene sample with error bars for the first few k-vectors of ZA mode at $T = 2000K$.

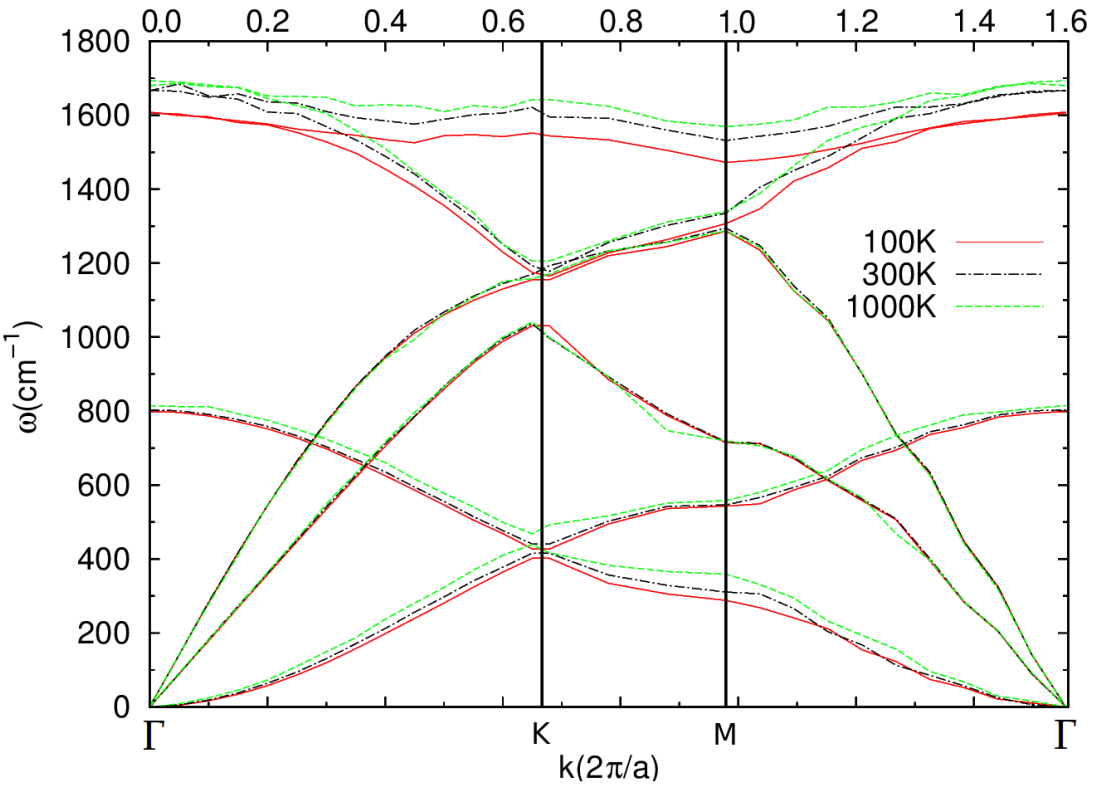


FIGURE 5.16: Combined graph of figures 5.12, 5.13 and 5.14 which shows the differences between the phonons of graphene at $T = 100$ K, $T = 300$ K and $T = 1000$ K.

5.5 Comparing the different temperatures

In our study we are interested in the temperature dependence, for this reason we evaluate in this section if there are any differences between the calculated phonon dispersions for the different temperatures. We begin with a general discussion and then derive the temperature dependence of the bending rigidity and sound velocity from the calculated phonon dispersions of graphene.

It is difficult to compare the phonon dispersions at different temperatures directly from figures 5.12, 5.13 and 5.14, therefore we made one combined graph of all the dispersion results for all the different temperatures as shown in figure 5.16. Now we are able to see some differences for the optical modes. It is clearly visible that the optical modes for the different temperatures deviate more from each other than the acoustic modes. For example the TO shifts by about $\Delta\omega \approx 100\ \mathrm{cm}^{-1}$, which is an enormous difference. This can also be seen in table 5.1, especially the phonon frequency at $\omega_{\Gamma_{E2g}}$, which is the phonon frequency for the double degenerate TO and LO mode at Γ, is in our interest. We can compare these data namely to the study of E.N. Koukeras et Al. [14] which more focused on the behaviour of these optical modes. In figure 5.17 we show the calculated temperature dependence of this double degenerate mode. When we compare it to the study of [14], given in figure 5.18, we see that our results are in agreement. The only difference is that the behaviour of $\omega_{\Gamma_{E2g}}$ which we found softens a bit harder for temperatures higher then $T = 900K$. However, this last fact can be explained by the fact that we used an other potential and also because we only compared 4 different temperatures whereas [14] evaluated 7 different temperatures. These results predict that the optical phonons of graphene are temperature dependent.

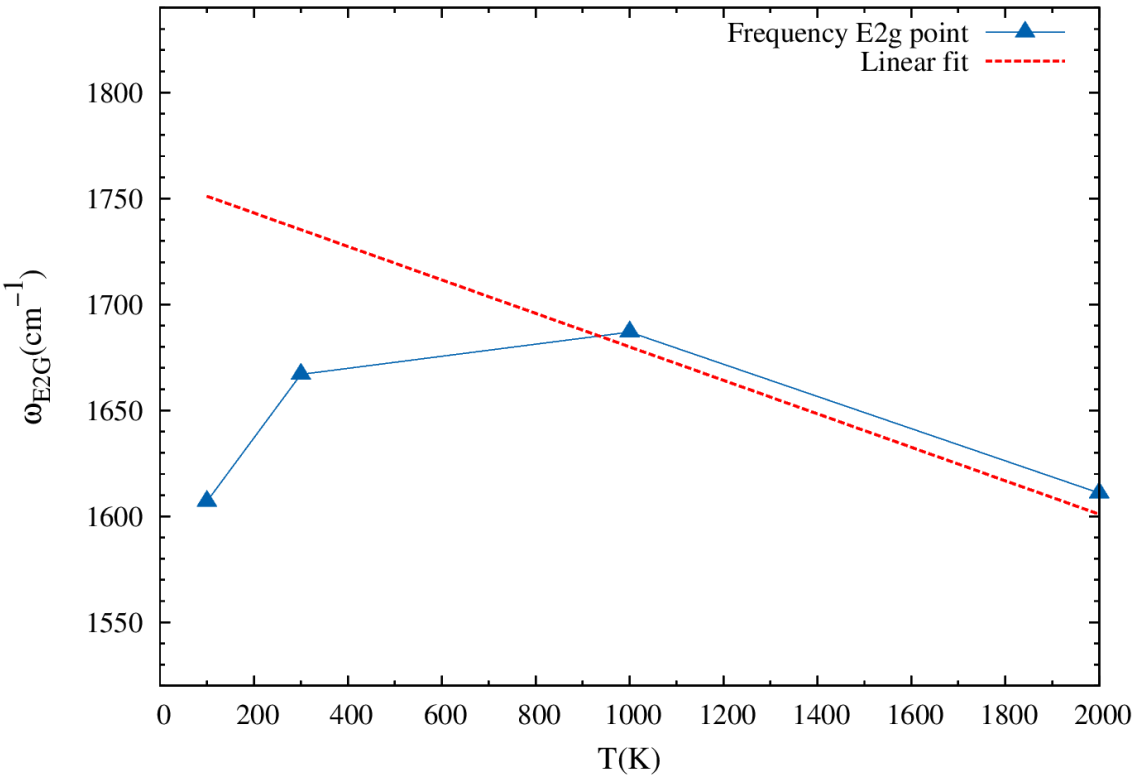


FIGURE 5.17: A fit of the temperature dependence of the double degenerate (TO and LO) phonon mode of graphene. The red dashed line denotes a linear to the softening of this temperature dependence after $T = 900$K.

For the acoustic modes it is much more difficult to see any differences between the different temperatures in figure 5.16. Therefore we made a zoom of the lowest k-vectors of the acoustic

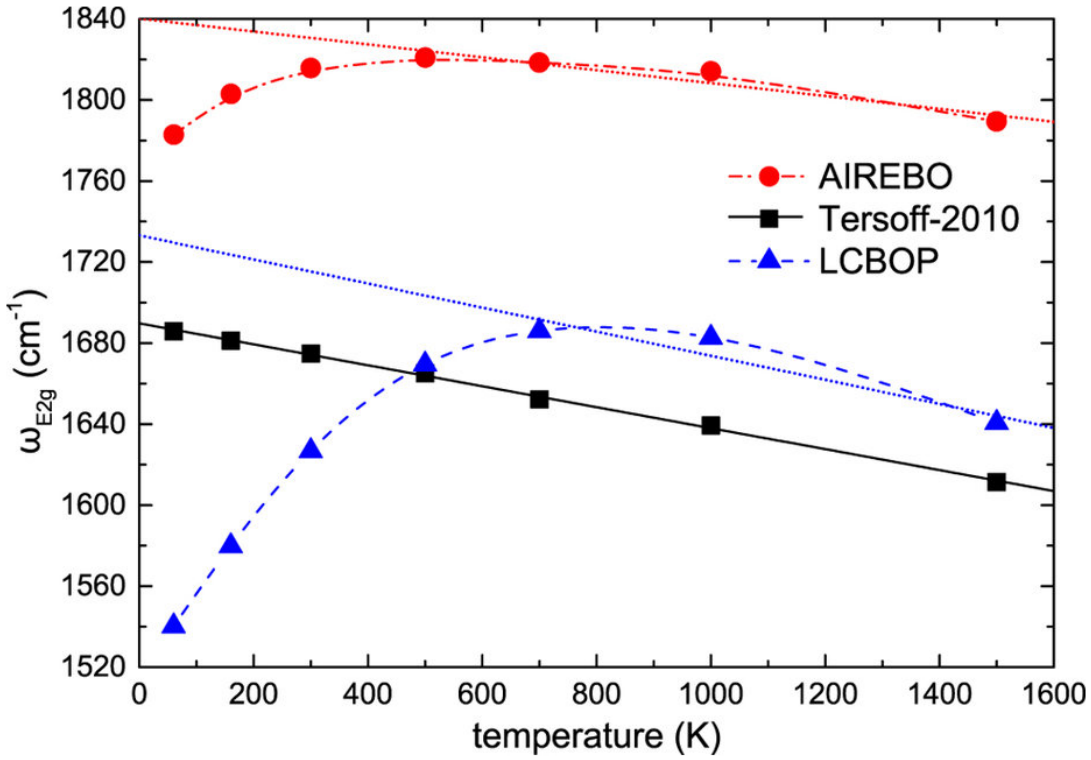


FIGURE 5.18: Temperature dependence of the double degenerate (TO and LO) phonon frequency in graphene for the AIREBO (red circles), Terseff-2010 (black squares), and LCBOP (blue triangles) potentials. Picture taken from [14].

modes as shown in figure 5.19. In this figure we can see much more detail. Firstly, the in-plane acoustic modes, TA and LA, seem to be independent of temperature because the lines lie almost exactly on top of each other. Secondly, it is interesting to look at the ZA-mode in figure 5.19. It is clearly visible that the lines corresponding to the different temperatures deviate from each other, meaning that there are differences for the different temperatures. How higher the temperature how faster the frequencies grow with the k-vector. So from this we could derive that the ZA-mode is also, like the optical modes, temperature dependent.

5.5.1 Temperature dependence of the bending rigidity and the sound velocity

As said and showed in figure 5.19, the in-plane acoustic modes, TA and LA, seem to be independent of temperature. To check this we can derive from these phonon modes the sound velocity in graphene by making a linear fit as described in Appendix A. Doing this calculation results in the values reported in table 5.2. In this table can be seen that the sound velocity does not vary very

TABLE 5.2: Results for the sound velocities at the different temperatures.

Temperature[K]	**TA mode v_{sound}[km s^{-1}]**	**LA mode v_{sound}[km s^{-1}]**
100	13.2	20.6
300	13.4	20.6
1000	13.3	20.3
2000	13.1	20.0

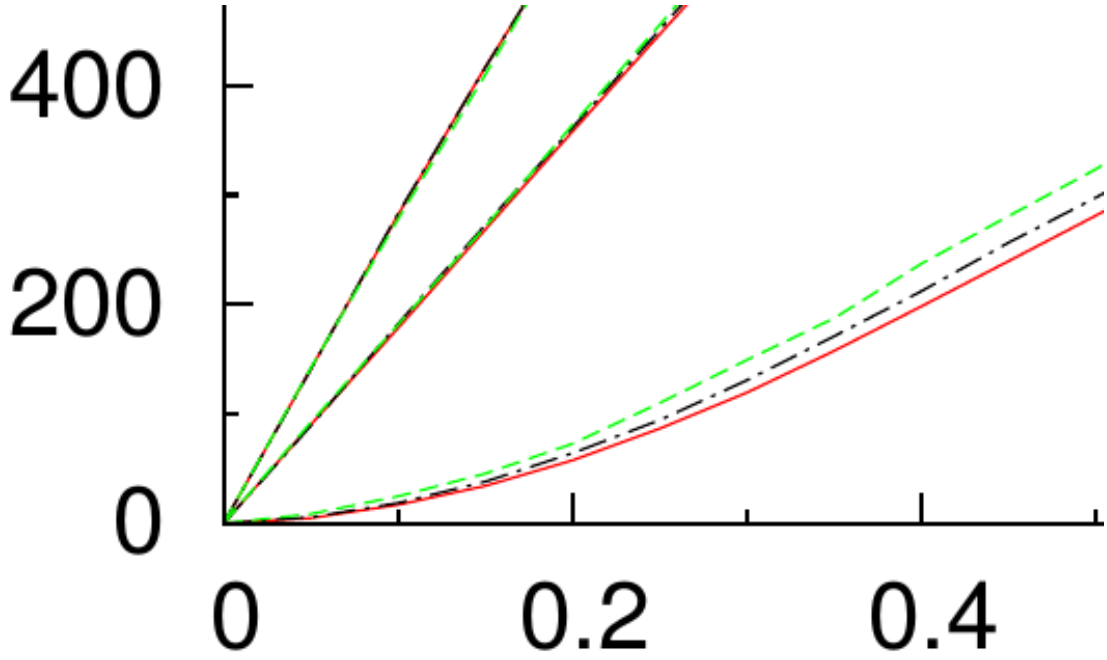


FIGURE 5.19: Zoom of the temperature difference graph shown in figure 5.16 for the lowest acoustic modes at the lowest k-vectors

much for different temperatures, this indicates that indeed the in-plane acoustic phonons are temperature independent. Comparing these results to the sound velocities calculated by Leendertjan [12] reveals that we have almost the same results.

Finally we determine the bending rigidity κ by fitting a parabola to our dispersion graphs for graphene as discussed in section 1.3. This results in the values reported in table 5.3, as one can see, the bending rigidity κ clearly increases for higher temperatures. We finally made the graph shown in figure 5.20 to determine the behaviour of this bending rigidity for different temperatures and to easily compare them to other studies. This behaviour looks a bit like a square root.

TABLE 5.3: Results for bending rigidity

Temperature[K]	**Lattice Parameter[Å]**	**Particle density [Å^{-2}]**	**Bending rigidity[eV]**
100	2.4578	0.3827	0.95
300	2.4566	0.3823	1.17
1000	2.4571	0.3825	1.76
2000	2.4633	0.3806	2.05

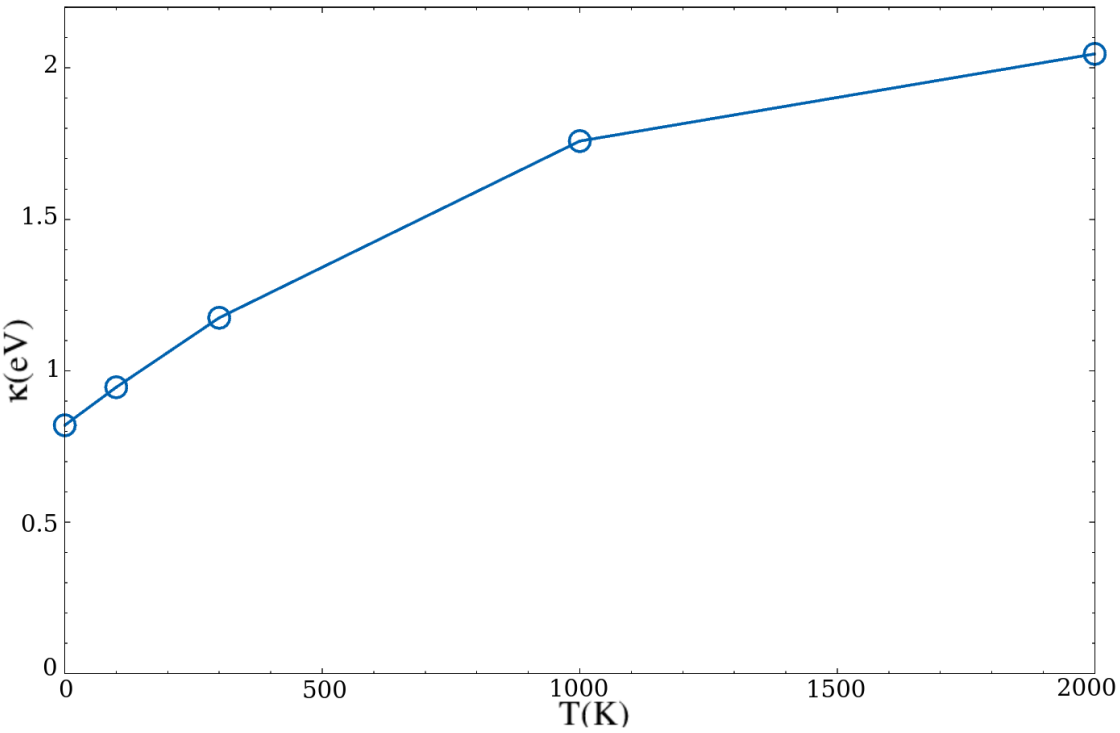


FIGURE 5.20: The calculated results for the bending rigidity $\kappa(T)$ at 5 different temperatures between $T = 0$ K and $T = 2000$ K.

Chapter 6

Conclusion and Discussion

In this chapter we interpret and summarize our main results. We start with comparing our results with some other studies to similar effects. Than we give the final overall findings of this research and give an outlook for further research.

6.1 Comparison to other studies

In this section we compare our results to four different other studies about the phonons of graphene. First of all we compared our results with the the Master thesis of Leendertjan Karssemeijer [12] and of Inka Locht [13] because they also investigated the temperature dependence of the phonons of graphene with the same potential as we did. To begin with Leendertjan, he computed the phonon dispersion of graphene at zero Kelvin using the force constant matrix. The results are reported in figure 1.3. Comparing that figure with our own phonon dispersion reveals that they look almost exactly the same, as expected because we are using the samen potential. Only on the **M**Γ-path, Leendertjan got smoother curves. Looking more closely reveals that our curves are a bit higher in frequency at for example the **K**-point and the M-point for the ZA-phonon. This can be explained because we were looking at a higher temperature. In his thesis Leendertjan compared his phonon dispersion with the experimental values and he argued that the experimental values were higher. In that sense, we could say that we have a good results for the phonon dispersions.

Besides the phonons, Leendertjan also looked at the bending rigidity κ according to his phonon dispersion at $T = 0$ K. He obtained $\kappa = 0.69$eV which compatible with our result as extrapolation to $T = 0$ K. He also looked at the thermal expansion of carbon nanotubes which revealed that the bending rigidity is not temperature dependent. This contradicts our results, but it could be that this has to do with the form of graphene in either a flat sheet or in a nanotube.

On the other hand, Inka Locht investigated the temperature dependence of the phonons and the bending rigidity of graphene using the SCAILD method, a selfconsistent extension of the quasiharmonic approximation. She was able to argue that κ depends on temperature, but she was not able to compute this temperature dependence for lack of convergence. In that sense, our results confirms here finding of temperature dependence, and moreover our results even give a good prediction for the temperature dependence.

The second study that we compare our results with is the paper that we have used for our new method [14]. It is good to compare our data with this study because we can see if we apply the method correctly to a different potential. Looking at our results reveals that we indeed have used the method correctly and it also confirms the data in [14]. This paper namely also found a strong temperature dependence for the optical modes, but dit not look at the temperature dependence of the out-of-plane phonons in which we were interested.

Finally we compare our results with the results from Zakharchenko [9], who obtained a temperature dependence for the bending rigidity by using Monte Carlo simulations and studying normal-normal correlation functions. It is interesting to notice that we used a completely different method but also obtained almost the same temperature dependence of the bending rigidity

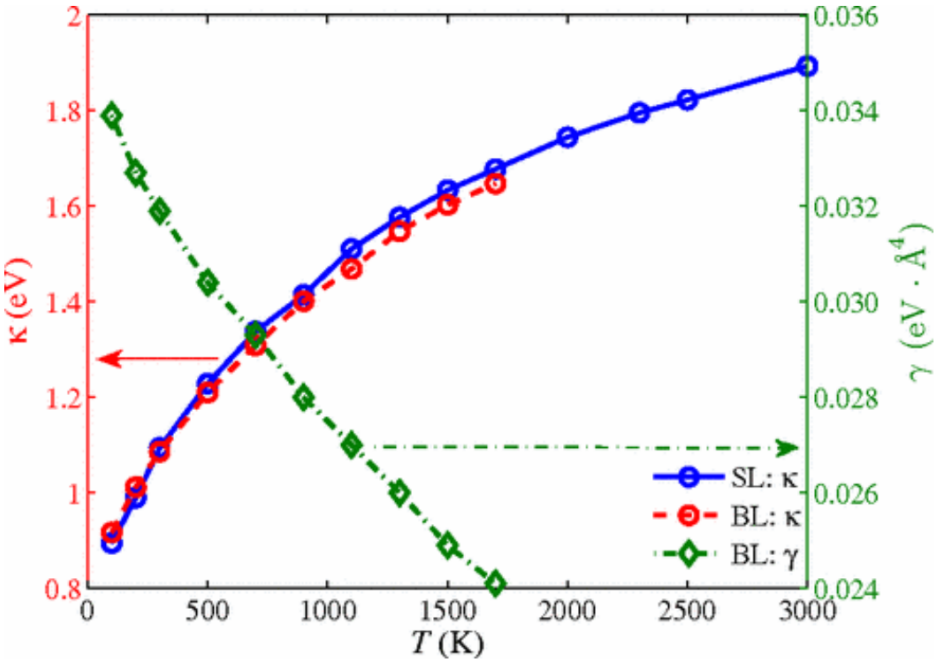


FIGURE 6.1: Temperature dependence of the bending rigidity of single-layer (sl), bi-layer (bl) graphene calculated by Zakharchencko [9] Also the lattice parameter of bi-layer graphene is plotted.

as shown in figure 5.20 and figure 6.1. The only differences are that our relation grows a bit faster and to 10% higher values for κ compared to the results of Zakharchencko.

To summarize, our results are in accordance with other studies except from the fact that Leendertjan did not find a temperature dependence of the bending rigidity for carbon nanotubes. For this reason we believe that our results with a whole different method are reasonable and might be predict the correct temperature dependence.

6.2 Summary, conclusions and outlook

In summary we were interested in the elastic properties of single layer graphene sheets and have therefore derived the phonons of graphene from MD simulations using afterwords Fourier transformations and autcorrelation techniques. It turns out that this new recently proposed technique is a good method to determine the phonon dispersion of graphene. A nice thing to notice is that this technique could also easily be applied to other materials with other lattice structures as long as we can define the wave vectors from the Brillouin zone.

Looking at our results, we found that the optical phonon modes of graphene are strongly temperature dependent by factors of about $\Delta\omega = 100\,\mathrm{cm}^{-1}$ when shifting the temperature from $T =$ 100K to $T = 1000$K. We focused on the acoustic phonon branch and we found that the traversal acoustic and the longitudinal acoustic phonon modes are as expected not temperature dependent and the sound velocity is in the right order of magnitude. These modes turned out to be linear for small wave vectors, as expected according to the theory. What was more interesting is that we found that the out-of-plane ZA mode clearly depends on temperature for small wavevectors. For small wave vectors the curvature of this mode more than doubled for higher temperatures which is in fact a quite astonishing result.

The most appealing result we got is the that the bending rigidity, derived from this ZA phonon, increases for higher temperatures. This contradicts the findings for carbon nanotubes but confirms the data found in other studies using Monte Carlo simulations. This result makes sense because,

as said before, single layer graphene sheets are rippled at finite temperatures and we know that rippled sheets are more difficult to bend than flat sheets hence the increasing bending rigidity.

However, to draw definite conclusions on the debate about the temperature dependence of the bending rigidity of graphene we need to improve our results a bit by making use of a bigger sample with more carbon atoms, simulating longer or do both. We also need to find a way to better estimate the error margins in our findings so that we can look how accurate our data is. Making these improvements is relatively easy with this new method and therefore new results could be obtained relatively quickly.

Finally we want to end this thesis with an outlook for future research and an explanation of the importance of our findings for the bending rigidity. As said, we still need to improve our results a little, but we have strong indications that the bending rigidity is temperature dependent. It would be useful to find a method to quantify the role of anharmonicity and coupling between the phonon modes. Why the bending rigidity for nanotubes is not temperature dependent is in that sense rather strange and unexplained. More research in comparing nanotubes to flat graphene sheets might therefore be interesting, especially for elastic properties. The consequences of this temperature dependent bending rigidity are important when considering elastic repsonse of the materials but also to understand the role of anharmonic coupling for membranes.

Appendix A

Unit conversion, bending rigidity and sound velocity

A.1 Converting the frequency units Herz(Hz) to inverse length(cm^{-1})

From the Fourier transformation given by equation 3.4 we get the frequency ω in Herz[s^{-1}]. Together with equation

$$\lambda = \frac{c}{f} \tag{A.1}$$

for the wavelength with c the speed of light we can determine frequency in units of inverse length:

$$\omega(\mathrm{cm}^{-1}) = \frac{\omega(\mathrm{Hz})}{100c}. \tag{A.2}$$

In equation A.2 the factor 100 accounts for the conversion from meters to centimeters.

Finally we can evaluate the dimensions of equation A.2 to see if it is a correct conversion:

$$[\omega] = \frac{[f]}{[c]} = \frac{1}{\cancel{s}} \times \frac{\cancel{s}}{\mathrm{cm}} = \frac{1}{\mathrm{cm}} \tag{A.3}$$

A.2 Deriving the value of κ in electronVolts

From equation 1.8 we can express the bending rigidity κ in terms of the fitting parameter c:

$$\kappa = c^2 \rho_{2D} \tag{A.4}$$

with $\rho 2D$ a 2D mass density with unit kgm^{-2}. We can derive the unit of fitting parameter c from equation 4.7 together with the units of the data we fitted the function to. This results in

$$[c] = \frac{[\omega]}{[k^2]} = \mathrm{cm}^{-1}\mathrm{m}^2. \tag{A.5}$$

The unit of κ in equation A.4 can then be determined resulting in:

$$[\kappa] = [c^2][\rho_{2D}] = \frac{\mathrm{m}^{\cancel{4}}}{\mathrm{cm}^2} \times \frac{\mathrm{kg}}{\mathrm{m}^{\cancel{2}}} = \frac{\mathrm{kgm}^2}{\mathrm{s}^2} = J \tag{A.6}$$

where we have used equation A.2 to convert cm^{-1} to s^{-1}. In equation A.6 we can see that the bending rigidity is given in Joule which can be easily converted electronVolt as given in the constant and unit conversion section in the preliminaries.

A.3 Deriving the value of the speed of sound in kms^{-1}

For graphene we have for small wave vectors linear dispersion in the in-plane direction given by dispersion relation

$$\omega = vk. \tag{A.7}$$

So fitting a linear line to the data of the in-plane acoustic gives directly the sound velocity v as fitting parameter. The unit of v can be derived:

$$[v] = \frac{[\omega]}{[k]} = \frac{1}{\mathrm{cm}} \times \frac{1}{\frac{1}{\mathrm{m}}} = \frac{\mathrm{km}}{\mathrm{s}} \tag{A.8}$$

where we have again used equation A.2 to convert cm^{-1} to s^{-1} and afterwards devided by 1000 to go from meter to kilometer.

Appendix B

FORTRAN90 code of the Fourier test program

```
! This program writes a function and then determines the frequency of the function
! with a discrete fouriertransform
! OUTDAT/function.dat gives the values of the function and OUTDAT/fourier.dat
! gives the values of the fouriertransform
     program testfourier
     implicit double precision(a-h,o-z)
     parameter(m=1000,m2=1000)                          !give in the number of steps
     integer it, iomega, im
     Real*8 ar
     real*8 f(0:m-1),ftr(0:m2-1)
     real*8 fti(0:m2-1),fbr(0:m-1),fbi(0:m-1)
     twopi=2.0d0*dacos(-1.d0)                           !define twopi
     tr=twopi/m

     open(12,file='OUTDAT/function.dat')                !open file to write the function values
     do it=0,(m-1)                                      !calculate the function values
       ar=10*tr*dfloat(it)
       f(it)=dcos(ar)
       write(12,'(i5,1e14.6)')it, f(it)
     enddo

     call FTfreq(f,m,m2,twopi,ftr,fti,tr)               !apply the FT in subroutine
     open(13,file='OUTDAT/fourier.dat')                 !open file to write the FT values
     do iomega=0,(m2-1)                                 !write the FT values
       write(13,'(i7,3e14.6)')iomega,ftr(iomega),
       &fti(iomega),(ftr(iomega)**2+fti(iomega)**2)
     enddo

     call IFT(fbr,fbi,m,twopi,ftr,fti,tr)               !apply the inverse FT in subroutine
     open(14,file='OUTDAT/fourierback.dat')             !open file to write the inverse FT values
     write(14,*)'#timestep, original function,
                &fourierbackreal,fourierbackimaginair, difference'
     do im=0,m-1
       write(14,'(i5,4e14.6)')im,f(im),fbr(im),fbi(im),(f(im)-fbr(im))
     enddo

     end program testfourier
!=============================================================================================!
      subroutine FTfreq(f,m,m2,twopi,ftr,fti,tr)        !subroutine for the FT
      implicit double precision(a-h,o-z)
      real*8 f(0:m2-1),ftr(0:m2-1),fti(0:m2-1)
      integer iomega, im, ik


      ftr(0:m2-1)=0.d0
      fti(0:m2-1)=0.d0

      do iomega=0,(m2-1)                                !calculate the FT for each time step
            do im=0,m-1
               ar=im*iomega*tr
               ftr(iomega)=ftr(iomega)+(f(im)*dcos(ar))
               fti(iomega)=fti(iomega)-(f(im)*dsin(ar))
            enddo
```

```
      enddo
       end subroutine FTfreq
!=============================================================================================!
      subroutine IFT(fbr,fbi,m,twopi,ftr,fti,tr)         !subroutine for the inverse FT
      implicit double precision(a-h,o-z)
      real*8 fbi(0:m-1),fbr(0:m-1),ftr(0:m-1),fti(0:m-1)
      integer iomega, im


      fbr(0:m-1)=0.d0
      fbi(0:m-1)=0.d0

       do im=0,m-1                                       !calculate the IFT for each time step
            do iomega=0,m-1
               ar=im*iomega*tr
               fbr(im)=fbr(im)+(ftr(iomega)*dcos(ar)-fti(iomega)*dsin(ar))
               fbi(im)=fbi(im)+(ftr(iomega)*dsin(ar)+fti(iomega)*dcos(ar))
            enddo
            fbr(im)=fbr(im)/m
            fbi(im)=fbi(im)/m
       enddo
      end subroutine IFT
```

Appendix C

FORTRAN90 code of our program to determine the PSD's

```
! This program reads configurations from the files of Molecular Dynamics Simulations
! and creates a file with the Fourier plots after defining the BZ and velocity autocorrelation.
      program getvelcfxyz
      implicit double precision(a-h,o-z) !working with double precision variables
      parameter(npm=1008,nspcm=1)  !defining the input variables
      parameter(nbst=2000,ncm=50001,nrsm=100,nrpsm=100,m=50000)
      character*2 atom,atspc(nspcm)
      character*3 ext,ext1,ext2,lstrs(0:nrpsm,nrsm),yes
      character*10 config,chaine,path
      integer ipbc(3),isp(npm),nei(npm)
      dimension nrps(nrsm),nconflst(nrpsm,nrsm),nmclst(nrpsm,nrsm)
      dimension hh(3),h(3),atpos(3,npm),atvel(3,npm),rnnbspsq(nspcm,nspcm)
      dimension templst(2,nrpsm,nrsm),preslst(2,nrpsm,nrsm),atpos0(3,npm)
!     dimension qlst(2,60)
      integer:: i1v1, i2v1,ncells,nc,nci
      Real:: i1v2, i2v2
      real*8 qvec(2),vkr(3),vki(3),kint(3,60) !making the necessary arrays
      real*8 kvelt(2,3,60,ncm),rrseq(27,3,0:m),riseq(27,3,0:m),psdr(27,3,0:m),psdi(27,3,0:m)
      real*8 afreqr(27,3,0:m),afreqi(27,3,0:m),fwsq(27,3,0:m),qlst(2,60)

      config='config.xxx'
      chaine='0123456789'
      open(10,file='OUTDAT/confall.xyz') !open the file with positions and velocities of the atoms

      ipbc(1)=1
      ipbc(2)=1
      ipbc(3)=1
      atspc(1)='C '
!      atspc(2)='H '
      rnnbspsq(1,1)=1.85d0**2
!      rnnbspsq(2,2)=3.0d0**2
!      rnnbspsq(3,3)=1.6d0**2
!      do i=1,nspcm
!      do j=1,nspcm
!        rnnbspsq(2,1)=dsqrt(rnnbspsq(i,i)*rnnbspsq(j,j))
!      enddo
!      enddo
      zmax=-1000.d0

      open(10,file='OUTDAT/sample.000',form='unformatted')!open file sample characteristics
      read(10)np                    !read the number of particles
      read(10)(h(i),i=1,3)          !read the lenght/width/height of the sample
      do ip=1,np
        read(10)atom,(atpos0(i,ip),i=1,3),(atvel(i,ip),i=1,3) !read starting pos./vel.
      enddo
      close(10)
```

```
      ncells=20            !define the number of unit cells
      alat=h(1)/dfloat(ncells) !calculate the lattice parameter from the size of the sample
      print*,alat,h(1),h(2),h(3) !print them on screen to check if correct
      pause
!     write(6,*)'Are you in directory SimLog ? (yes/no)'
!     read(5,*)yes
      yes='no '
      path(1:10)=' '
      if(yes(1:1).eq.'n')then
        path(1:7)='OUTDAT/'
      else
        write(6,*)'From which subdirectory do you want data ?'
        read(5,*)path
        ind=index(path,' ')
        path(ind:ind)='/'
      endif
      write(6,*)'path=',path

      open(16,file=path(1:index(path,' ')-1)//'inp.log')
      read(16,*)nrun,nspc,(atspc(i),i=1,nspc)
      if(nspc.gt.nspcm)then
        write(6,*)'ERROR: nspc > nspcm'
        write(6,*)'set parameter nspc >= nspcm and retry'
      endif
      do irun=1,nrun
        read(16,*,iostat=istatus)iextinp,iextout,nmcext,invwcf,temp1,temp2
        if(istatus>=0)then
          call mkrunseq(nrsm,nrpsm,chaine,nrs,nrps,lstrs,irun,iextinp,iextout,&
          nmcext,invwcf,temp1,temp2,pres1,pres2,nconflst,nmclst,templst,preslst)
        else
          exit
        endif
      enddo

      write(6,'(a28,i5)')'   Number of run sequences :',nrs !show all the data sequences
      do irs=1,nrs
        write(6,*)
        write(6,'(a30,i6)')'   run sequence              :',irs
        write(6,'(a30,i6)')'   no runs for this sequence :',nrps(irs)
        write(6,*)'  irs  extinp  extout  nmcext  nconf  temp1      temp2    pres1    pres2'
        write(6,*)'  ---  ------  ------  ------  -----  -----      -----    -----    -----'
        do irps=1,nrps(irs)
        write(6,'(i5,4x,a3,2x,a3,x,a3,x,a3,x,2i7,1x,f8.2,a4,f8.2,f7.2,a4,f7.2)&
          ') irps,lstrs(irps-1,irs),'-->',lstrs(irps,irs),nmclst(irps,irs),&
          nconflst(irps,irs),templst(1,irps,irs),' -->',templst(2,irps,irs),&
```

```
        preslst(1,irps,irs),' -->',preslst(2,irps,irs)
      enddo
    enddo

    write(6,*)!determine which sequence of configurations is used
    if(nrs.gt.1)then
      write(6,*)'From which run sequence do you want configurations.'
      read(5,*)irs
    else
      irs=1
    endif
    write(6,*)'Do you want configurations for the complete run&
            sequence ?(yes/no)'
    read(5,*)yes
    if(yes(1:1).eq.'y')then
      irps1=1
      irps2=nrps(irs)
    else
      write(6,*)'Give the lower and upper sequence number'
      read(5,*)irps1,irps2
    endif

    write(6,*)'Give the number of configurations to be skipped'
    read(5,*)iskip

    write(6,*)'Give skip modulo'
    read(5,*)nskip

!     nqx=h(1)/2.4595d0+1
!     nqy=h(2)/4.26d0+1
!     nq=max0(nqx,nqy)+10
!     nqx=nq
!     nqy=nq
!     write(6,*)'nqm,nq :',nqm,nq
!     write(6,*)'nqx,nqy:',nqx,nqy
!     if(nq.gt.nqm)then
!       write(6,*)'ERROR: nqm too small'
!       write(6,*)'make sure that nqm >',nq
!       stop
!     endif
    twopi=2.d0*dacos(-1.d0) !define twopi

!! Make qlst with allowed qvectors that obey the Brillouin Zone
    iv1max=2*ncells/3
    iv1min=ncells/2
```

```
    do i1v1=1,iv1max !path GammaK
      i1v2=Real(i1v1*0.5)
       iq=i1v1
       qx=twopi/(alat*ncells)*i1v1
       qy=twopi/(alat*ncells)*((-i1v1+2.0*i1v2)/SQRT(3.0))
          qlst(1,iq)=qx
       qlst(2,iq)=qy
    enddo
    do i2v1=iv1min, iv1max   !path KM
       i2v2=-i2v1+ncells
       iq2=iq+iv1max - i2v1+1 !put in array from high to low
       qx=twopi/(alat*ncells)*i2v1
       qy=twopi/(alat*ncells)*(-i2v1/SQRT(3.0) +2.0*i2v2/SQRT(3.0))
          qlst(1,iq2)=qx
       qlst(2,iq2)=qy
    enddo
    do i3v1=1, (iv1min-1)   !path MGamma
       i3v2=i3v1
       iq3=iq2+iv1max-i3v1
       qx=twopi/(alat*ncells)*i3v1
       qy=twopi/(alat*ncells)*(-i3v1/SQRT(3.0) +2.0*i3v2/SQRT(3.0))
          qlst(1,iq3)=qx
       qlst(2,iq3)=qy
    enddo
    nq=iq3+iv1min-1
!    open(11,file='OUTDAT/qlist.dat')   !creates a outputfile with the qvectors
!    write(11,*),'#The coordinates of the',nq,'q-vectors of this sample'
!    write(11,*),'#number x-coordinate y coordinate'
!       do iq=1, nq
!         write(11,'(i3,2e14.6)')iq,(qlst(1:2,iq))
!       enddo
 print*,'qlist is created!' !print on screen to see in which step the program currently is
pause

!          dqx=twopi/(h(1))
!          dqy=twopi/(h(2))
!     open(12,file='OUTDAT/positionvelocityxyzatom143_2000K.dat') !make some output files
!     write(12,*),' #positions and velocities atom 143 in 800 sample'
!     write(12,*),' #t     x  y  z   vx  vy  vz'
!     open(13,file='OUTDAT/positionvelocityxyzatom571_2000K.dat')
!     write(13,*),' #positions and velocities atom 571 in 800 sample'
!     write(13,*),' #t     x  y  z   vx  vy  vz'
!     open(14,file='OUTDAT/positionvelocityxyzatom71_2000K.dat')
!     write(14,*),' #positions and velocities atom 71 in 800 sample'
!     write(14,*),' #t     x  y  z   vx  vy  vz'
!     open(14,file='OUTDAT/kspacevx.dat')
```

```
!     write(14,*),' #kspace x-velocity for',nq,'kvectors in BZ'
!     write(14,*),' #t     vk1x    vk2x    ....''      vk',nq,'x'
!     open(15,file='OUTDAT/kspacevy.dat')
!     write(15,*),' #kspace y-velocity for',nq,'kvectors in BZ'
!     write(15,*),' #t     vk1y     vk2y   ....''      vk',nq,'y'
      open(16,file='OUTDAT/kspacevz2000K.dat')
      write(16,*),' #kspace z-velocity for ',nq,'kvectors in BZ'
      write(16,*),' #t     vk1z    vk2z    .....''      vk',nq,'z'
      nctot=0
      do irps=irps1,irps2
        nci=0
        ext=lstrs(irps,irs)
        if(ext.eq.'999')cycle
        config(8:10)=ext(1:3)
        open(8,file=path(1:index(path,' ')-1)//config,status='old',form='unformatted')
         do nc=0,ncm !read in all the positions and velocities for all the timesteps in the sequence
          read(8,iostat=istatus)np,(h(i),i=1,3),&
          ((atpos(k,i),k=1,3),i=1,np),(isp(i),i=1,np),((atvel(k,i),k=1,3),i=1,np) !read in for all atoms
            write(12,'(i5,6e14.6)')nc,atpos(1:3,143),atvel(1:3,143) !create output files for pos./vel.
            write(13,'(i5,6e14.6)')nc,atpos(1:3,571),atvel(1:3,571)
            write(14,'(i5,6e14.6)')nc,atpos(1:3,71),atvel(1:3,71)

           hh(:)=0.5d0*h(:)
           if(istatus>=0)then
             if(iskip.gt.0)then !skip some data if given
               iskip=iskip-1
               cycle
             endif
             if(mod(nc,nskip).ne.0)cycle
             nci=nci+1
             do iq=1,nq !take the fouriertransform for different q values
               qvec(1:2)=qlst(1:2,iq)                  !read in qvector
               call FTvel(np,atpos,atvel,qvec,vkr,vki)   !determine velocity distribution for qvector
               kvelt(1,1:3,iq,nci)=vkr(1:3)
               kvelt(2,1:3,iq,nci)=vki(1:3)
               kint(1:3,iq)=kvelt(1,1:3,iq,nci)**2+kvelt(2,1:3,iq,nci)**2
             enddo
!            write(14,'(i5,27e14.6)')nc,kint(1,1:nq) !create output file with intensity kvelocities
!            write(15,'(i5,27e14.6)')nc,kint(2,1:nq)
            write(16,'(i5,27e14.6)')nc,kint(3,1:nq)
!             print*,'# of atoms to be shown in xyz file=',np
           else
             nctot=nctot+nci
             print*,nc,' snapshots are read from file ',config
             exit
           endif
```

```
        enddo
       enddo
!      print*,'datafiles atom 143 and atom 571 are created!'
      print*,'datafiles kspacevelocities are created!' !print on screen to show how far the program is
      pause
      call kVACS(nctot,m,kvelt,rrseq,riseq,psdr,psdi,nq,ncm) !do the velocity autocorrelation

!      open(12,file='OUTDAT/psdreel.dat') !make output file for velocity autocorrelation functions
!      write(12,*),' #normalised psdz  per kvector '

!      do im=0,m  !write the velocity autcorrelation functions to an output file
!        write(12,'(i5,27e14.6)')im,(psdr(1:nq,3,im)**2+psdi(1:nq,3,im)**2)
!      enddo
!pause
      print*,'kVACS for all qvectors have been created!' !print again how far the program is
     ! write(6,*)'PSD Real, Imaginair:',psdi
      call FTfreq(psdr,afreqr,afreqi,qlst,nq,m,psdi) !do the final Fourier transform
      print*,'fouriertransform to frequency has been done' !print again how far the program is
      open(18,file='OUTDAT/omegaintensitiesx.dat') !write the results to some files
      write(18,*),' omegax intensity for every k-vector'
      write(18,*),' omega in steps of 0.1333 wki1   wki2 ....  wki(nq)'
      open(19,file='OUTDAT/omegaintensitiesy.dat')
      write(19,*),' omegay intensity for every k-vector'
      write(19,*),' omega in steps of 0.1333 wki1   wki2 ....  wki(nq)'
      open(20,file='OUTDAT/omegaintensitiesz.dat')
      write(20,*),' omegaz intensity for every k-vector'
      write(20,*),' omega in steps of 0.1333 wki1   wki2 ....  wki(nq)'
        do iomega=0,20000
          do iq=1,nq
             fwsq(iq,1:3,iomega)=afreqr(iq,1:3,iomega)**2+afreqi(iq,1:3,iomega)**2
          enddo
          write(18,'(i6,27e14.6)')iomega,fwsq(1:27,1,iomega)
          write(19,'(i6,27e14.6)')iomega,fwsq(1:27,2,iomega)
          write(20,'(i6,27e14.6)')iomega,fwsq(1:27,3,iomega)
        enddo
      print*,'finished!'    !print that the program has just finished
      print*,'number of config_ files =',irps2-irps1+1,', all are read'
      print*,'total # of snapshots=',nctot

      end program getvelcfxyz

!=============================================================!
      subroutine FTvel(np,atpos,atvel,qvec,vkr,vki) !subroutine for the first Fourier Transform
      implicit double precision(a-h,o-z)
      real*8 atpos(3,np), atvel(3,np),qvec(2)    !input
      real*8 vkr(3),vki(3)                       !output
```

```
      vkr(1:3)=0.d0
      vki(1:3)=0.d0

      do ip=1,np,2   !!steps of 2 because of using just 1 sublattice
        qr=qvec(1)*atpos(1,ip)+qvec(2)*atpos(2,ip)  !inproduct qvector with position atoms
          vkr(1:3)=vkr(1:3)+atvel(1:3,ip)*dcos(qr) !real components FT
          vki(1:3)=vki(1:3)+atvel(1:3,ip)*-dsin(qr) !imaginary components FT
      enddo

      end subroutine FTvel

!=====================================================================!
      subroutine kVACS(nctot,m,kvelt,rrseq,riseq,psdr,psdi,nq,ncm) !subroutine autocorrelation
      implicit double precision(a-h,o-z)
      real*8 kvelt(2,3,60,ncm),rrseq(27,3,0:m),riseq(27,3,0:m),psdr(27,3,0:m),psdi(27,3,0:m)
      integer im,nctot,n,step,nq,i

      rrseq(1:27,1:3,0:m)=0.d0 !make sure the arrays are empty
      riseq(1:27,1:3,0:m)=0.d0
      psdr(1:27,1:3,0:m)=0.d0
      psdi(1:27,1:3,0:m)=0.d0

      do iq=1,nq
        do im=0,m
          do n=1,nctot-im
            step=n+im
            do i=1,3 !calculate the real and imaginary component of the kVACS
              rrseq(iq,i,im)=rrseq(iq,i,im)+&
                        kvelt(1,i,iq,step)*kvelt(1,i,iq,n)+&
                        kvelt(2,i,iq,step)*kvelt(2,i,iq,n)
              riseq(iq,i,im)=riseq(iq,i,im)-&
                        kvelt(1,i,iq,step)*kvelt(2,i,iq,n)+&
                        kvelt(2,i,iq,step)*kvelt(1,i,iq,n)
            enddo

          enddo
          pf=1.d0/(dfloat(nctot-im))
          rrseq(iq,1:3,im)=pf*rrseq(iq,1:3,im)
          riseq(iq,1:3,im)=pf*riseq(iq,1:3,im)
          psdr(iq,1:3,im)=rrseq(iq,1:3,im)/(rrseq(iq,1,0)+rrseq(iq,2,0)+& !normalise
                        rrseq(iq,3,0))
          psdi(iq,1:3,im)=riseq(iq,1:3,im)/(rrseq(iq,1,0)+rrseq(iq,2,0)+&
                        rrseq(iq,3,0))
!          print*,'', psdr(iq,1:3,im)
```

```
      enddo
    enddo

    end subroutine kVACS

!=================================================================!
    subroutine FTfreq(psdr,afreqr,afreqi,qlst,nq,m,psdi) !subroutine FT, equal to FT test program
    implicit double precision(a-h,o-z)
    real*8 psdr(27,3,0:m),afreqr(27,3,0:m-1),afreqi(27,3,0:m-1),psdi(27,3,0:m),omega
    dimension qlst(2,60)
    integer nq, iomega, im,iq

    twopi=2.d0*dacos(-1.d0)
    tr=twopi/dfloat(m)
    afreqr(1:27,1:3,0:m-1)=0.d0
    afreqi(1:27,1:3,0:m-1)=0.d0

     do iq=1,nq
      do iomega=0,20000  !frequencies higher than Nyquist(iomega=25000) not evaluated
                         !frequencies are not expected to be higher than iomega=20000
        do im=0,m-1
          ar=dfloat(im)*(dfloat(iomega)/10)*tr  !devide by 10 because we saved the data every
                                                !10 time steps
          afreqr(iq,1:3,iomega)=afreqr(iq,1:3,iomega)+& !calculate FT from kVACS
                        (psdr(iq,1:3,im)*dcos(ar)+&
                        psdi(iq,1:3,im)*dsin(ar))
          afreqi(iq,1:3,iomega)=afreqi(iq,1:3,iomega)+& !again also imaginary components
                        (-psdr(iq,1:3,im)*dsin(ar)+&
                        psdi(iq,1:3,im)*dcos(ar))
!         write(12,*)",ar,afreqr(iq,1:3,iomega),psdr(iq,1:3,im)
!          pause
        enddo
      enddo
!     write(6,*)",afreqr(iq,1:3,0:20)
!     pause
    enddo
    end subroutine FTfreq
```

Bibliography

[1] M. I. Katsnelson and A. Fasolino. Graphene as a prototype crystalline membrane. *Accounts of Chemical Research*, 46(1):97–105, **2013**.

[2] K. S. Novoselov, A. K. Geim, S. V. Morozov, D. Jiang, Y. Zhang, S. V. Dubonos, I. V. Grigorieva, and A. A. Firsov. Electric Field Effect in Atomically Thin Carbon Films. *Science*, 306(5696):666–669, **2004**.

[3] K. S. Novoselov, A. K. Geim, S. V. Morozov, D. Jiang, M. I. Katsnelson I. V. Grigorieva, S. V. Dubonos, and A. A. Firsov. Two-dimensional gas of massless dirac fermions in graphene. *Nature*, 438(7065):197–200, **2005**.

[4] M. I. Katsnelson, K. S. Novoselov, and A. K. Geim. Chiral tunnelling and the klein paradox in graphene. *Nature physics*, 2(9):620–625, **2006**.

[5] K. S. Novoselov, Z. Jiang, Y. Zhang, S. V. Morozov, H. L. Stormer, U. Zeitler, J. C. Maan, G. S. Boebinger, P. Kim, and A. K. Geim. Room-temperature quantum hall effect in graphene. *Science*, 315(5817):1379–1379, **2007**.

[6] A. Fasolino, J. H. Los, and M. I. Katsnelson. Intrinsic ripples in graphene. *Nat Mater*, 6(11):858–861, **2007**.

[7] D. R. Nelson and L. Peliti. Fluctuations in membranes with crystalline and hexatic order. *Journal de physique*, 48(7):1085–1092, **1987**.

[8] J. H. Los, M. I. Katsnelson, O. V. Yazyev, K. V. Zakharchenko, and A. Fasolino. Scaling properties of flexible membranes from atomistic simulations: application to graphene. *Physical Review B*, 80(12):121405, **2009**.

[9] K. V. Zakharchenko, J. H. Los, M. I. Katsnelson, and A. Fasolino. Atomistic simulations of structural and thermodynamic properties of bilayer graphene. *Phys. Rev. B*, 81:235439, **2010**.

[10] C. Kittel. *Introduction to solid state*. John Wiley & Sons, **1966**.

[11] N. Mounet and N. Marzari. First-principles determination of the structural, vibrational and thermodynamic properties of diamond, graphite, and derivatives. *Phys. Rev. B*, 71:205214, **2005**.

[12] L. J. Karssemeijer. Thermal expansion of carbon structures. Master's thesis, Radboud University Nijmegen, Heyendaalseweg 135, the Netherlands, **2010**.

[13] I. Locht. The effect of temperature on the phonon dispersion relation in graphene. Master's thesis, Radboud University Nijmegen, Heyendaalseweg 135, the Netherlands, **2012**.

[14] E. N. Koukaras, G. Kalosakas, C. Galiotis, and K. Papagelis. Phonon properties of graphene derived from molecular dynamics simulations. *Scientific Reports*, 5, **2015**.

[15] J. H. Los, L. M. Ghiringhelli, E. J. Meijer, and A. Fasolino. Improved long-range reactive bond-order potential for carbon. i. construction. *Physical Review B*, 72(214102):1–14, **2005**.

[16] A. K. Geim and K. S. Novoselov. The rise of graphene. *Nat Mater*, 6(3):183–191, **2007**. 10.1038/nmat1849.

[17] C. Lee, X. Wei, J. W. Kysar, and J. Hone. Measurement of the Elastic Properties and Intrinsic Strength of Monolayer Graphene. *Science*, 321(5887):385–388, **2008**.

[18] Y. Song, W. Fang, R. Brenes, and J. Kong. Challenges and opportunities for graphene as transparent conductors in optoelectronics. *Nano Today*, 10(6):681–700, **2015**.

[19] J. H. Los, K. V. Zakharchenko, M. I. Katsnelson, and Annalisa Fasolino. Melting temperature of graphene. *Phys. Rev. B*, 91:045415, **2015**.

[20] Quanta: A handbook of concepts. second edition (atkins, p.w.). *Journal of Chemical Education*, 69(5):A167, **1992**.

[21] P. Hofmann. *An introduction to Solid State Physics*. Wiley-VCH Verlag GmbH & Co. KGaA., 2 edition, **2015**.

[22] L. J. Karssemeijer and A. Fasolino. Phonons of graphene and graphitic materials derived from the empirical potential LCBOPII. *Surface Science*, 605:1611–1615, **2011**.

[23] L. D Landau and E. M. Lifschitz. *Theory of Elasticity*. Pergamon Press, Oxford, **1970**.

[24] J. Tersoff. Energies of fullerenes. *Phys. Rev. B*, 46:15546–15549, **1992**.

[25] M. C. Payne, M. P. Teter, D. C. Allan, T. A. Arias, and J. D. Joannopoulos. Iterative minimization techniques for total-energy calculations: molecular dynamics and conjugate gradients. *Rev. Mod. Phys.*, 64:1045–1097, **1992**.

[26] M. E. Tuckerman, P. J. Ungar, T. von Rosenvinge, and M.L. Klein. Ab initio molecular dynamics simulations. *The Journal of Physical Chemistry*, 100(31):12878–12887, **1996**.

[27] D. W. Brenner, O. A. Shenderova, J. A. Harrison, S. J. Stuart, B. Ni, and S. B. Sinnott. A second-generation reactive empirical bond order (rebo) potential energy expression for hydrocarbons. *Journal of Physics: Condensed Matter*, 14(4):783, **2002**.

[28] J. H. Los and A. Fasolino. Intrinsic long-range bond-order potential for carbon: Performance in monte carlo simulations of graphitization. *Physical Review B*, 68(024107):1–14, **2003**.

[29] E. V. Anslyn and D. A. Dougherty. Modern physical organic chemistry. *Journal of Chemical Education*, 83(3):387, **2006**.

[30] B. Frenkel, D.and Smit. *Chapter 4 - Molecular Dynamics Simulations*. Academic Press, San Diego, second edition edition, **2002**.

[31] L. Verlet. Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules. *Phys. Rev.*, 159:98–103, **1967**.

[32] J. M. Haile. *Molecular Dynamics Simulation: Elementary Methods*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, **1992**.

[33] K. Satyanarayana. Characteristics of noise received by software de ned radio. From Department of Electrical Engineering Indian Institute of Technology Madras publicated on http://www.gaussianwaves.com, **2012**.

[34] P. L. De Vries and J. E. Hasbun. *A First Course in Computational Physics*. Jones & Bartlett Learning, LLC, **2011**.

[35] T. N. Narasimhan. Fourier's heat conduction equation: History, influence, and connections. *Proceedings of the Indian Academy of Sciences - Earth and Planetary Sciences*, 108(3):117–148, **1999**.

[36] G. B. Folland. *Fourier analysis and its applications*, volume 4. American Mathematical Soc., **1992**.

[37] C. Van Loan. *Computational frameworks for the fast Fourier transform*, volume 10. Siam, **1992**.

[38] Bao Wenzhong, Miao Feng, Chen Zhen, Zhang Hang, Jang Wanyoung, Dames Chris, and Lau Chun Ning. Controlled ripple texturing of suspended graphene and ultrathin graphite membranes. *Nat Nano*, 4(9):562–566, **2009**.