Theoretical Materials Science:
Electronic Structure Theory at the Computer

Prepared by Hagen-Henrik Kowalski,
based on an exercise by Christian Carbogno
Berlin, June 2017

Some rules on expected documentation from this exercise.

The computational exercises are intended as "hands-on" experience with actual, numerical electronic structure theory. Our main goal is to fill some of the basic concepts with life for real systems.

If you are fast, you may finish the exercise below in the actual exercise class. If not, we ask you to try finish as much as you can, and then come back at another point, follow the script, and finish the exercise.

You can do the actual computations and hand in your solutions to the exercise in pairs of two. What is expected is a record of the basic data that we ask for (e.g., in table form), rough answers to the questions asked (answers can be short, but should be there, and should indicate that you understood the meaning of your data), and plots, where required. You can just qualitatively draw the plots by hand in your homework or print them on paper or screenshot them and then hand-in your homework electronically via email as PDF.

Please hand in all your solutions as usual at the beginning of the exercise in the following week.

Contents

1 Important Note

If any problems occur please do not hesitate to contact me.
My email address is kowalski@fhi-berlin.mpg.de

2 Background

In this exercise, we will use the structure predictions of exercises 14 and 15 as a basis to investigate the vibrational properties of diamond silicon.

In the present exercise, we will use the forces calculated in Kohn-Sham density-functional theory (DFT-GGA) to approximate the potential-energy surface (PES) in a second order Taylor expansion around the equilibrium volume,

$$V^{\mathrm{BO}}(\{\mathbf{R}\}) \approx V^{\mathrm{BO}}(\mathbf{R}_0) + \frac{1}{2} \sum_{\alpha\beta,IJ} \underbrace{\frac{\partial^2 V^{\mathrm{BO}}(\mathbf{R}_0)}{\partial R_I^\alpha \partial R_J^\beta}}_{\Phi_{IJ}^{\alpha\beta}} s_{\alpha,I} s_{\beta,J}. \tag{1}$$

Here the $\Phi_{IJ}^{\alpha\beta}$ are the force constants (or Hessian), $\mathbf{s}$ the displacement vectors, and V^{BO} the Born-Oppenheimer surface.
We already know from the lecture that, strictly, this "approximation" will be limited in reach as it does not account for anharmonic effects.

In exercise 23 you have learned how the thermal expansion can be calculated by explicitly accounting for the Volume dependence of the Hessian matrix $\Phi_{IJ}^{\alpha\beta}$. However this approximation can still fail for highly anharmonic systems in which case the full potential-energy surface needs to be considered.
This can be accomplished by molecular dynamics techniques in which the classical motion of the nuclei is evaluated by stepwise integrating the Newtonian equations of motion:

$$\mathbf{F}_I = M_I \mathbf{a}_I \tag{2}$$

3 Important: Necessary scripts and modules

For the exercise 24, you need to download the molecular dynamics simulation package LAMMPS as well as a few additional scripts. How to obtain and install LAMMPS is described in Sec. 5. The remaining **scripts and modules** for exercise 24 can be downloaded from the link:
→ https://thcloud.rz-berlin.mpg.de/s/Od9CDHzyMUb6XjK
Important: Please use the "Thermal_expansion.py", and "classical_phonopy" versions included in the downloaded archive (link above) as they contain some updates.
Alternatively, you can just re-download and re-install the updated **virtual machine** instance that already contains these packages in form of tar files from
→ https://thcloud.rz-berlin.mpg.de/s/kElVkeik4aSfMEQ .

3.1 Classical calculations using phonopy

IMPORTANT: please reinstall "classical_phonopy" as it contains a bugfix.
To complete exercise 24, you need a modified version of the phonopy package. Be careful when using this package. To avoid any conflicts between the modified and usual phonopy installed on your machine, it might be useful to name the folder containing the modified phonopy version, e. g. **classical_phonopy**. Accordingly, when you want to use the new installation, just change the path and the python path to:

```
export PATH=/home/theory/classical_phonopy/bin:$PATH
export PYTHONPATH=/home/theory/classical_phonopy/lib/python
```

The installation works in exactly the same way as described in Sec. 4. **IMPORTANT**: when you use the normal phonopy, do not forget to change the paths back to the original ones described in Sec. 4.

3.2 Necessary scripts for exercise 24 and a short recap how to set the paths for ASE and Thermal_expansion.py

The other scripts included in the download are required to perform quasi-harmonic part of exercise 24. These are python modules or scripts which require no installation. Their exact purpose will become clear in the following sections. To make use of the scripts, safe the new version of "Thermal_expansion.py" in a new folder, for instance "modules" and change your python path accordingly to (without blanks or line break)

```
export PYTHONPATH=/home/theory/phonopy/lib/python:
/home/theory/modules:/home/theory/modules/ase-3.12.0
```

Unless you are simply replacing the old version in which case you can leave the path as it is.

4 Reminder: Phonopy in the Virtual Machine

To calculate phonons in the harmonic approximation, we will use the phonopy package but before you can get started some other packages have to be installed. Please install pip, h5py, and pyyaml by typing:

```
sudo apt install python-pip
pip install h5py
pip install pyyaml
```

Once these are installed, download **phonopy** from the link
→ https://thcloud.rz-berlin.mpg.de/s/6FEiHHM3ftK6F8B.
After downloading, unpack the file and name it simply `phonopy`. Before installing phonopy, you have to add a line to your python path in your .bashrc file (this file lies in the home folder). Open the file by typing:

```
gedit ~/.bashrc
```

or use any other editor of your choice. At the end of the .bashrc file, please add the lines:

```
export PATH=/home/theory/phonopy/bin:$PATH
export PYTHONPATH=/home/theory/phonopy/lib/python
```

After saving and exiting the .bashrc file do not forget to execute it in the current shell by typing:

```
source .bashrc
```

Next, change into the folder `phonopy` and install the package by typing:

```
python setup.py install --home=.
```

Now you can run the program simply by typing `phonopy-FHI-aims` into the terminal.
Later in the exercise you will also need some **pre-calculated simulations**. They can be downloaded from the link
→ https://thcloud.rz-berlin.mpg.de/s/3PgxVnmhRMT2LqI.
These folders contain force evaluations with a number of atoms that would not be possible on a local machine let alone on a virtual one.

5 LAMMPS in the Virtual Machine

Given the computational demands of an *ab-initio* molecular dynamics simulation we cannot use FHI-aims for exercise 24. To give you an idea of molecular dynamics nonetheless we will use the molecular dynamics simulator LAMMPS. Unlike FHI-aims this code uses predefined "potential-energy surfaces" such as the Lennard-Jones potential and is thus computational far less demanding. Before any calculations can be performed the program must first be downloaded and installed which is relatively straight forward.

5.1 Installing LAMMPS

First download the code from this link:

```
http://lammps.sandia.gov/download.html
```

Go to the point "Download a tarball" and choose the last stable release from 31. Mar. 2017. Unpack the file by typing

```
tar -xzvf file.tar.gz
```

into the terminal. To install the program itself go to the directory "lammps-31Mar17/src/" and type

```
make serial
if a serial installation is requested or
make mpi
if a parallel installation is requested
```

However to ensure compatibility with the provided scripts a serial installation is recommend. Once the installation is finished type

```
make yes-manybody
```

This will generate the executable "lmp_serial" or "lmp_mpi" in the directory lammps-31Mar17/src/.

5.2 Using LAMMPS

Using LAMMPS is generally more challenging then using FHI-aims which is why we will provide you with the necessary input files and scripts. Nonetheless we shall explain the basics needed to perform this exercise here. To run LAMMPS an input file similar to FHI-aims first needs to be written. An example related to this exercise is given below.

```
# Sample LAMMPS input script
units metal # defines units used internally
variable T equal 70.0 # sets the temperature of the system for example fs
variable V equal vol
variable dt equal 0.001 # sets how long each time step is in fs here 1 ps
variable p equal 20000 # determins how many steps dt are to be evauated
variable s equal 1 # sample interval
variable d equal p*s # dump interval
variable lattconst equal 5.376 # lattice constant
variable seed equal 12782318 # sets the seed has to be different for each trajectory

# convert from LAMMPS real units to SI
variable kB equal 1.3806504e-23 # [J/K] Boltzmann
variable eV2J equal 1.6e-19 # ev to Joule
variable A2m equal 1.0e-10
variable ps2s equal 1.0e-12
variable convert equal $eV2J*$eV2J/$ps2s/$A2m
variable run_thermalization equal 20000 # sets time of how long to run thermalization here 20000ps
variable dump_thermalization equal 10000 # 1 ps
variable run_NPT equal 30000 # sets time of how long to run actural calculation here 30000ps
variable dump_NPT equal 10 # sets at which dimes the results are written into the output
variable size equal 2 # sets size of the supercell


# setup problem dimension 3
boundary p p p # defines the problem to be periodic in all 3 dimensions lattice diamond ${lattconst} orient x 1 0 0 orient y 0 1 0 orient z 0 0 1 # sets up first cubis supercell
region box block 0 $size 0 $size 0 $size # specifies which supercell is used
create_box 1 box
create_atoms 1 box
mass 1 28.085 # defines atoms mass
#pair_style lj/cut 24.0 # describes which potential to use for simulation in this case Lennard-Jones
#pair_coeff * * 0.01034398278 3.405 # Fixes Parameters used for Lennar-Jones potential in eV and Å
pair_style tersoff # describes which potential to use for simulation in this case the Tersoff potential pair_coeff * * PATH_TO_PARAMETERS Si(D) # defines parameters for the Tersoff potential
(unlike Lennard-Jones the Tersoff potential requires multiple parameters)
timestep ${dt} #sets time step

# equilibration and thermalization
velocity all create ${T} ${seed} mom yes rot yes dist gaussian #Create gaussian distribution of velocities at specified temperature with random seed, while zeroing angular and linear momentum.
fix 1 all npt temp ${T} ${T} 1.0 iso 0.0 0.0 1.0 # fixes which ensemble to use in this case NPT with an isotropic pressure of 0 Bar
thermo_style custom step vol temp
```

```
thermo_modify format float %32.12f
thermo ${dump_thermalization}
run ${run_thermalization} # starts thermalization

#Production:
reset_timestep 0
velocity all zero linear


# Output (total system)
thermo_style custom step temp etotal vol
thermo_modify format float %32.12f
thermo ${dump_NPT}
# Run
run ${run_NPT}
```

In principle this input file can be arbitrarily named, however in this exercise you should use the name convention "lammps.in". Using the file "lammps.in" a calculation can be started by typing

```
~/lammps-31Mar17/src/lmp_serial < lammps.in > lammps.out
or
~/lammps-31Mar17/src/lmp_mpi < lammps.in > lammps.out
```

into the terminal depending on your installation.

6 Exercise 24: Using Molecular Dynamics to Describe Highly Anharmonic Systems

In the previous exercises you have learned how to determine phonon spectra from the harmonic approximation and even how to describe the thermal expansion by means of the quasi-harmonic approximation. However the assumptions made are only valid for materials with a moderate degree of anharmonicity. To overcome this limitation, the full anharmonicity, i.e. the complete parts of the potential-energy surface that are accessible at certain thermodynamic conditions, have to be taken into account in the calculation. This can be achieved via molecular dynamics simulations, in which the classical motion of the nuclei is computed by evaluating the Newtonian equations of motion

$$\mathbf{F}_I = M_I \mathbf{a}_I. \tag{3}$$

For the exercises you will be provided with scripts that will help you to carry out some of the more technical parts of the exercise.

6.1 Exercise 24 b) Converging the Molecular Dynamics Simulation

For this part you will need the scripts "replace.py" and "ensemble_average.py" as well as the provided input-file "lammps.in". Make sure to use the input-file that is intended for silicon. This is of importance because for silicon a different empiric potential has to be chosen than for argon namely the Tersoff potential for which a more detailed description can be found under the link
→ http://lammps.sandia.gov/doc/pair_tersoff.html .
The script "replace.py" will carry out the molecular dynamics simulation specified in the input-file "lammps.in" for a set of temperatures and a number of different trajectories. First open the script for example by typing in the terminal:

```
gedit replace.py
```

You will find the lines

```
Temp = []
Input_name = "lammps.in"
num_traj = 10
LAMMPS = ~/lammps-31Mar17/src/lmp_serial
```

Temp is a list in which you have to type the temperatures for which you want to run a molecular dynamics simulation. For example if you wish to run a simulation at 200, 300, and 500 K just change the list to

```
Temp = [200, 300, 500]
```

The rest of the lines define the name of the input-file, the number of trajectories, and the path to the LAMMPS executable. Now take a closer look at the file "lammps.in". Here you need to pay special attention to the lines

```
variable run_thermalization equal 20000
variable dump_thermalization equal 10000
variable run_NPT equal 50000
variable dump_NPT equal 10
```

The first line defines how long to "thermalize" the system in this case 20000 ps. During this time the actual expansion takes place. Accordingly it is important to choose a length that allows the system to reach its equilibrium volume. The second line defines in which intervals the volume is written to the output file. Accordingly this value should be decreased for part b) to accurately assess how long it takes the system to equilibrate.
The third line defines how long the system runs in equilibrium. This is important to obtain reliable results as the volume fluctuates over time and we thus have to average over a sufficiently large number of volumes. Finally the fourth line defines the lengths of the intervals after which the results of "run_NPT" are written to the output.
If you now execute the script "replace.py" a number of folders will be generated named according to the temperature of the molecular dynamics simulation. In each folder you will find 10 sub-folders named consecutively from 0 to 9. These contain different trajectories. However you do not need to evaluate these by yourself, for this purpose you are provided with the script "ensemble_average.py". Once all molecular dynamic runs are finished you should execute the script "ensemble_average.py". It prints out the standard deviation for each trajectory and writes the ensemble average over the ten trajectories for the volume and the statistical errors into the files "Volume.dat" and "total_standard_deviation.dat". Additionally it also writes the volume during the thermalization and the actual run at the chosen simulation step into each temperature folder. To assess if the results are converged with respect to the thermalization time you should make sure that the system is indeed in equilibrium.
In case of the actual runtime you have to closely inspect the standard deviation per trajectory which is printed out by the script "ensemble_average.py". If they are similar for each trajectory you most likely reached convergence. You also have to check how much the resulting lattice constant changes – a few mÅ are normal.
Important: Do not set your settings too high otherwise the calculations may take very long. You may also choose one temperature, for example 800 K, to find suitable times.

6.2 Exercise 24 c), d), and f) Comparing the Quasi-Harmonic Approximation to Molecular Dynamics Results

Before starting this part make sure you have set the correct "PATH" and "PYTHONPATH". It is important to not mix the classical and normal phonopy installations. One possible source of error is that you may have adjusted the "PYTHONPATH" to the phonopy version you wish to use but forgot to do the same for the "PATH" variable. This is of importance because the scripts "Calculate.py" and "runfree.py" rely one the "PATH" to run phonopy-FHI-aims.

6.2.1 Silicon in the Quasi-Harmonic Approximation

Start by calculating the thermal expansion the quasi-harmonic approximation using the provided scripts.
You may use the folder included in the tar file you have downloaded earlier. However first check if you use the updated version of the module "Thermal_expansion.py". If so proceed by executing the script "Calculate.py" this will generate a folder "Si" in which you find folders which contain prepared phonopy displacements at different volumes similar to exercise 23. Now copy the scripts "Expansion.py", "LammpsEq.py", "Lammpsrun.py", "run_lammps.bash", "run_free.py", "calculate_all.sh", and "Si_single_point_calculation.py" to the folder Si.
First execute the script "run_lammps.bash" by typing

```
bash run_lammps.bash
```

This script executes the python script "LammpsEq.py" and exports the path to the LAMMPS executable in the line

```
export LAMMPS_COMMAND="/home/theory/lammps-31Mar17/src/lmp_serial"
```

This will generate a folder Murn_data which contains the static energies saved in the file "Etot.dat". Next execute the script "Lammpsrun.py" by typing

```
python Lammpsrun.py
```

This will carry out the force evaluations in the folder containing the phonopy supercells and write them into a format compatible with phonopy-FHI-aims by using the scripts "Si_single_point_calculation.py" and "calculate_all.bash". Similar to "run_lammps.bash" "calculate_all.bash" exports the path to the LAMMPS executable via an identical line.
Afterwards either run phonopy-FHI-aims manually in each directory named "Phononxxx" or use the script runfree.py to accomplish this automatically. Finally run the script "Expansion.py", this will generate the files "VolumePhonon.dat" and "Thermal_expansionPhonon.dat".
For part d) repeat this calculation **but** switch to the classical phonopy version.

6.2.2 Thermal Expansion of Silicon using Molecular Dynamics

The proceedings in this part are very similar to those described in Sec. 6.1. The main difference is that here you are supposed to use the thermalization and run time you found in part b) and use them to find the equilibrium volume. For this purpose open the script "replace.py" and add the corresponding temperatures to the list "Temp" as shown below

```
Temp = [200,300,400,500,600,700,800]
```

Reminder: Do not forget to use the correct "lammps.in" file provided in the folder "Molecular_Dynamics/Silicon". As in Sec. 6.1 first execute the script "replace.py" to perform the molecular dynamics simulations and use the script "ensemble_average.py" to evaluate the results. To make the results comparable to those calculated using the quasi-harmonic approximation, both volumes and the total standard deviation are divided by a factor of eight. This is a necessary step as in the quasi-harmonic calculation the first cubic supercell was chosen as the unit cell which then was transformed into the second cubic supercell for the force evaluation.

6.2.3 Thermal Expansion of Argon

This part essentially is identical to parts b) and c) only that you repeat the calculations for argon. Remember however that argon melts above a temperature of 80 K thus do not use larger temperatures in your convergence test. Choose the classical phonopy installation from the beginning to obtain the thermal expansion in the quasi-harmonic approximation. You do **not** have to repeat this calculation using the normal phonopy installation. Given that argon is bound by van der Waals forces we choose the Lennard-Jones potential to describe energies and forces.
Accordingly the "lammps.in" for argon differs in some points from the one for silicon. The scripts used to evaluate the quasi-harmonic approximation are adapted for argon as well while the naming convention stays the same. Only the script "Si_single_point_calculation.py" was renamed "Ar_single_point_calculation.py". However the general proceedings are identical.