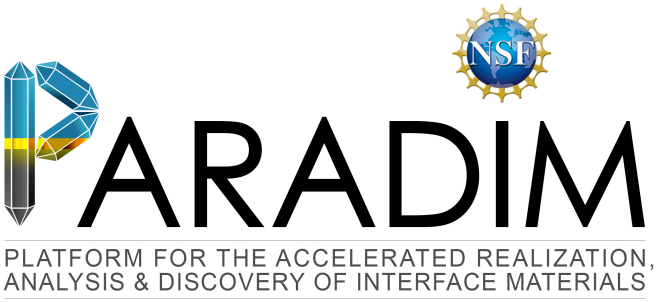


An Introduction to Density Functional Theory for Experimentalists

Feliciano Giustino

University of Oxford & Cornell University

Ithaca, 8-14 July 2018

Lecture 1.1

Ab initio materials modelling

Organization of the DFT module

Betül Pamuk

Guru Khalsa

Duminda Samarakoon

FG

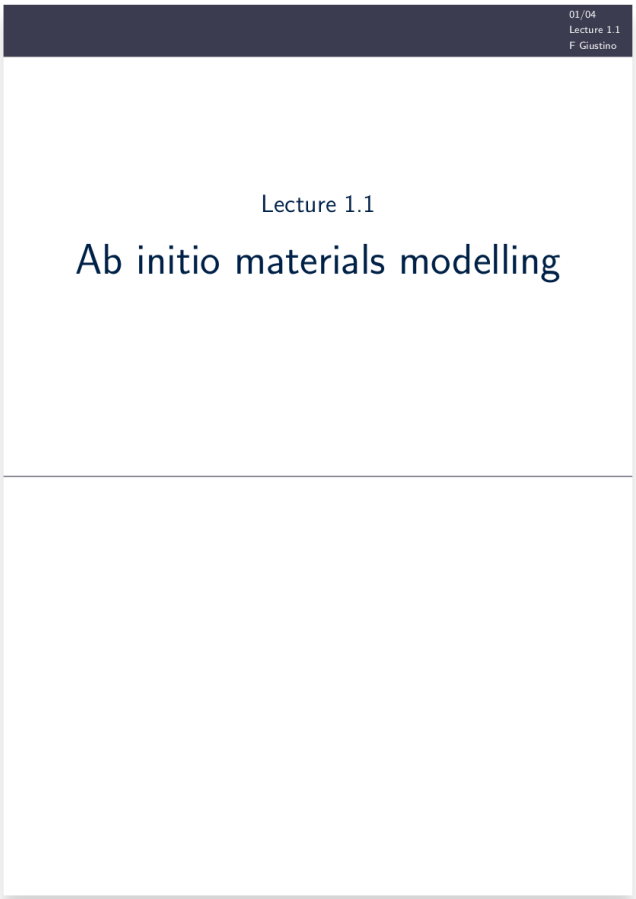
Organization of the DFT module

Monday–Tuesday and Thursday–Saturday

9:30–10:15	Theory Lecture 1	45 min	FG
10:15–10:45	Coffee Break	30 min	
10:45–11:30	Theory Lecture 2	45 min	FG
11:30–13:30	Lunch Break	2 h	
13:30–14:15	Technical Lecture	45 min	FG
14:15–14:30	Short Break	15 min	
14:30–16:30	Hands-on Session	2 h	Betül, Guru, Duminda, FG

Handouts and Tutorial Sheets

Handouts



Tutorial Sheets

PARADIM

An introduction to density functional theory for experimentalists

Tutorial 1.1

Login shell and compilation

We will perform calculations on the Blue Crab Linux cluster of MARCC. Blue Crab hosts 676 Intel Haswell dual socket 12-core processors, and for this tutorial we will be using between 4 and 24 cores at a time. In order to work on Blue Crab we need to establish a secure connection. We first open a terminal (on Ubuntu/Mac machines; from Windows we launch Putty), then we type:

```
$ ssh -X gateway2.marcc.jhu.edu -l fgiusti1.temp@jhu.edu
```

where fgiusti1.temp@jhu.edu must be replaced by the username that you have been assigned. After entering your password you will see something like:

```
[fgiusti1.temp@jhu.edu@login-node01 ~]$
```

We can customize the Unix 'shell' environment by shortening the prompt, creating a couple of "aliases", and adding modules that we will need later on. We copy/paste the following into the terminal (it is important to copy/paste exactly as it is, since the bash shell is very strict with spaces):

```
cat >> .bashrc << EOF
PS1="$ "
alias c="clear"
alias l="ls -lh"
module load openmpi/intel/2.0.1
module load fftw3/intel/3.3.7
module load xcrysden/1.5.53
module load gnuplot/5.0.0
EOF
source ~/.bashrc
```

From now on the prompt will be simply '$' and the command 'c' and 'l' will clear the screen and list the content of a directory, respectively.
We can now create our working directory for this school:

```
$ mkdir scratch/summerschool ; cd ~/scratch/summerschool
```

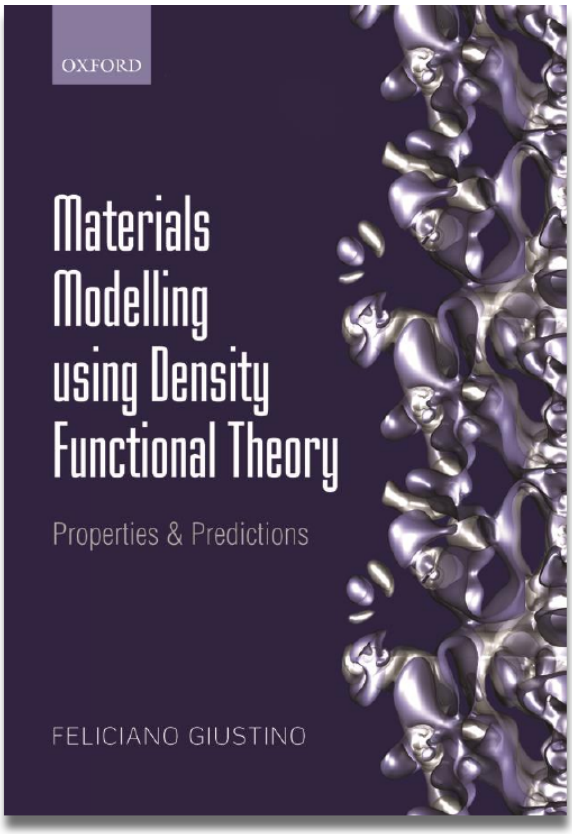
In this school we will be using the Quantum ESPRESSO (QE) software package. QE is an open-source suite of *ab initio* electronic structure codes based on pseudopotentials and planewaves.

The project website can be found at www.quantum-espresso.org

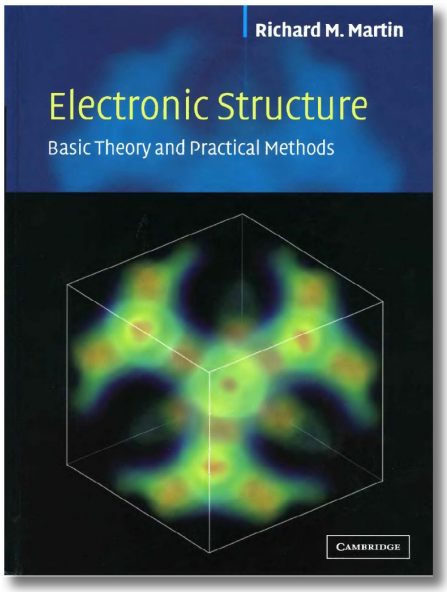
Paradim@Cornell, July 8-14, 2018 F Giustino Tutorial 1.1 | 1 of 8

Textbooks

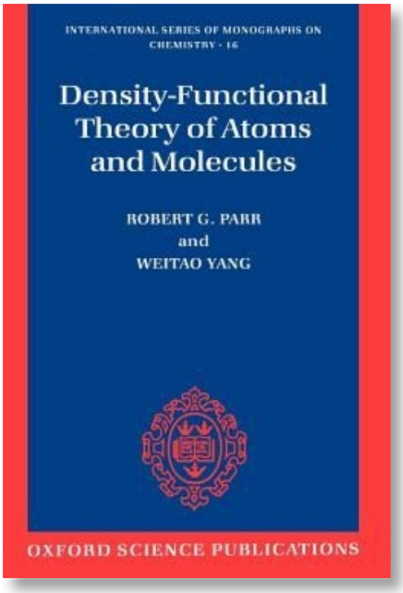
MSc and 1st year PhD



Advanced PhD level



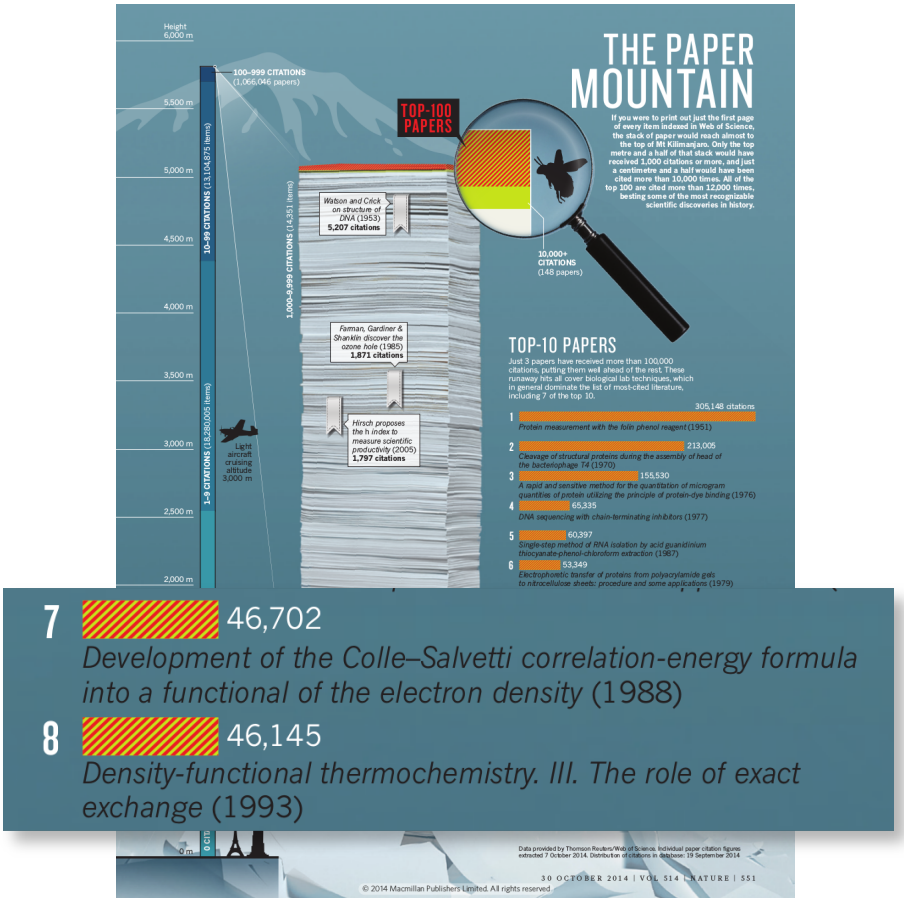
Theoretical foundations



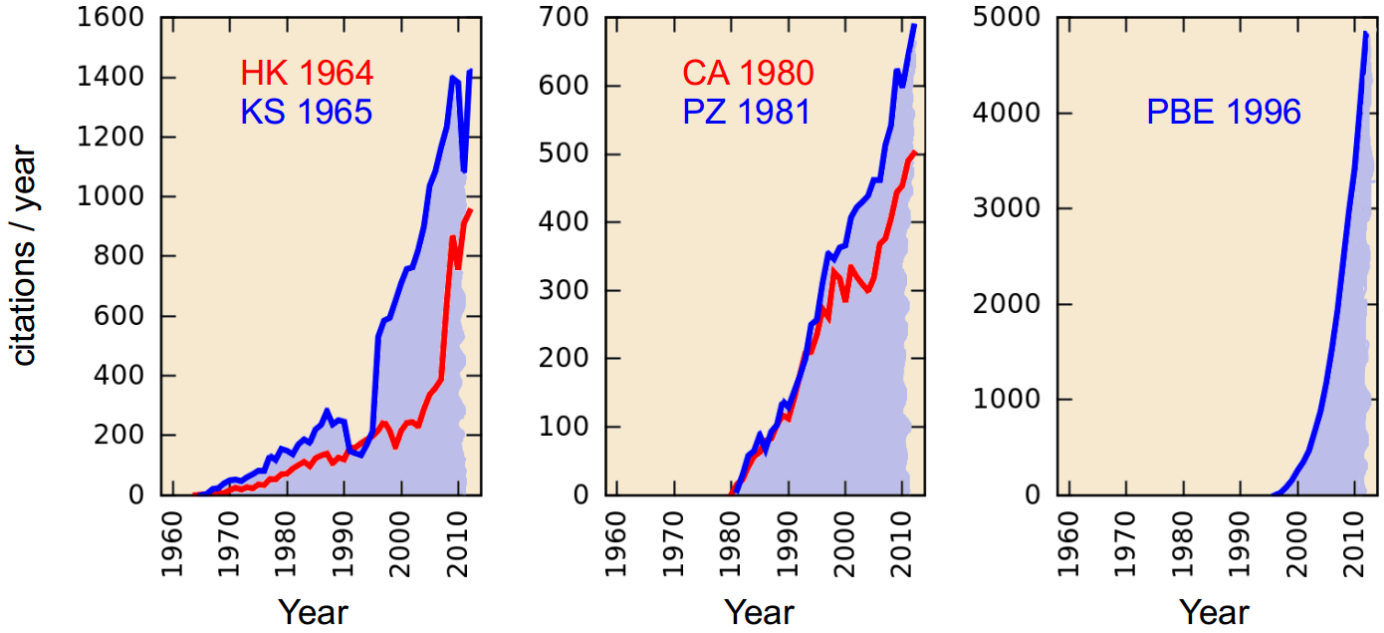
Outline

THE TOP 100 PAPERS

Interview by R. Van Norden
Nature 514, 550 (2014)



Impact of DFT



HK 1964 Hohenberg, Kohn, Phys. Rev. 136, B864 (1964)
KS 1965 Kohn, Sham, Phys. Rev. 140, A1133 (1965)
CA 1980 Ceperley, Alder, Phys. Rev. Lett. 45, 566 (1980)
PZ 1981 Perdew, Zunger, Phys. Rev. B 23, 5048 (1981)
PBE 1996 Perdew, Burke, Ernzerhof, Phys. Rev. Lett. 77, 3865 (1996)

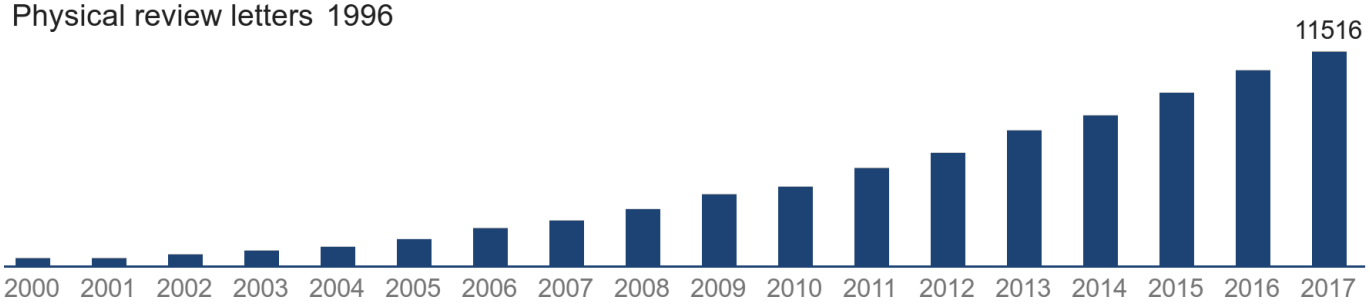
Impact of DFT

Generalized gradient approximation made simple

John P Perdew, Kieron Burke, Matthias Ernzerhof

Physical review letters 1996

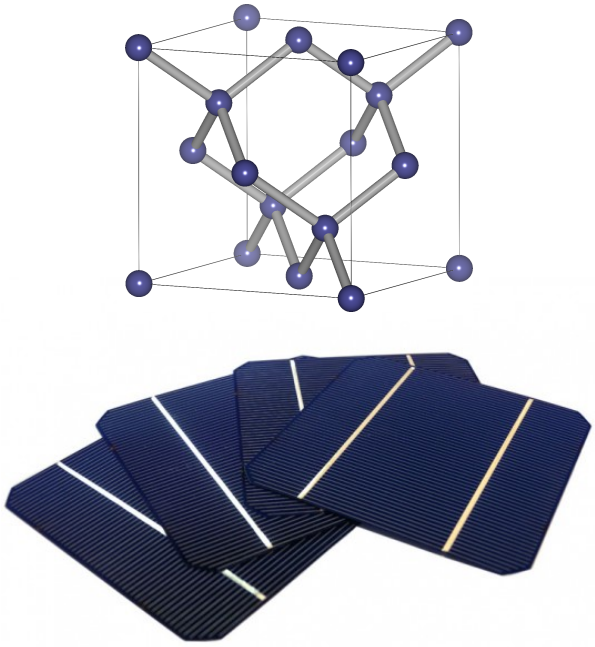
Total citations
83631

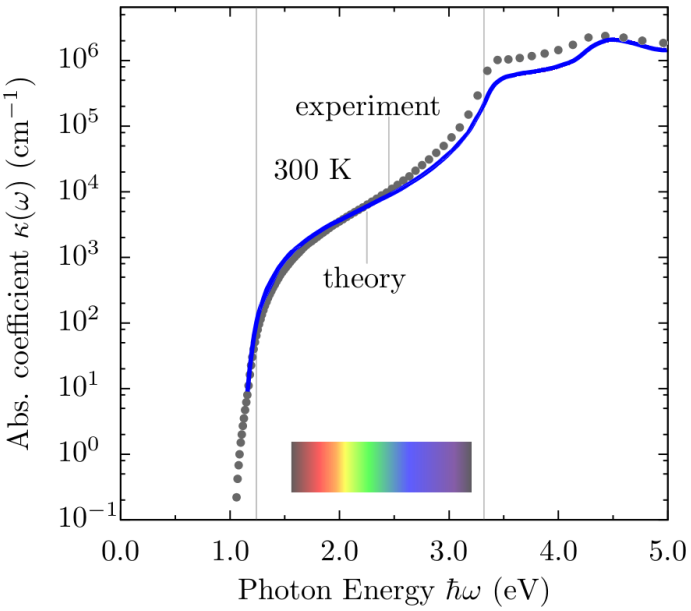


The B3LYP papers ranked #7 and #8 in 2014 are now at ∼75k cites

Examples of calculations based on DFT

Predictive calculations of optical properties





Zacharias, Patrick, and FG, Phys. Rev. Lett. 115, 177401 (2015)

Examples of calculations based on DFT

Predictive calculations of transport properties

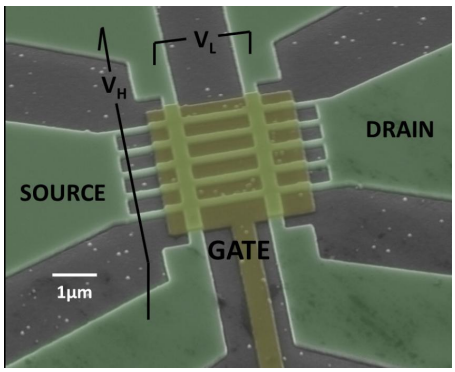
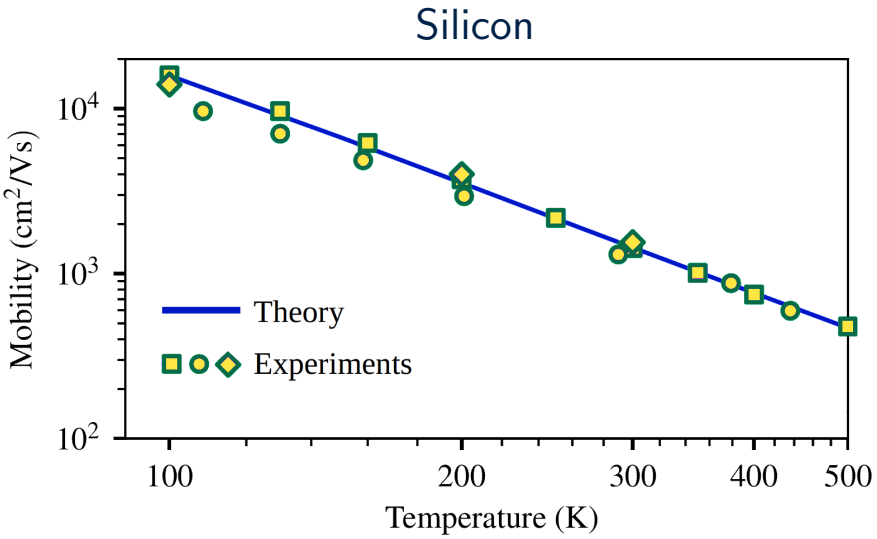


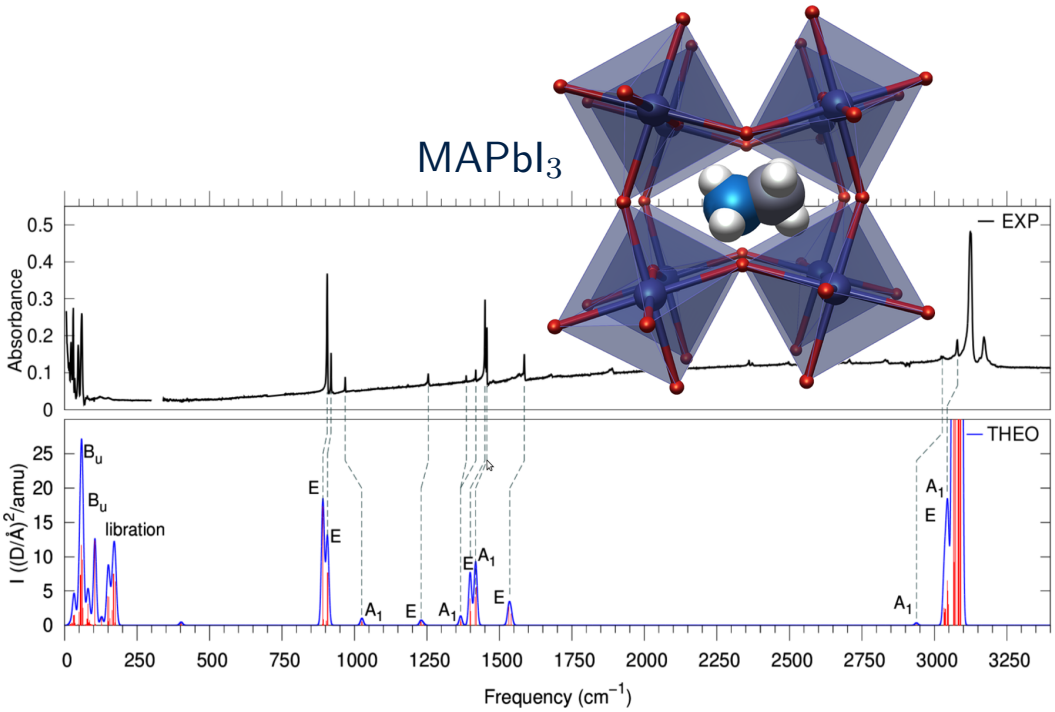
Image from Thathachary et al, Nano Lett. 14, 626 (2014)



Poncé, Margine, and FG, Phys. Rev. B(R) 97, 121201 (2018)

Examples of calculations based on DFT

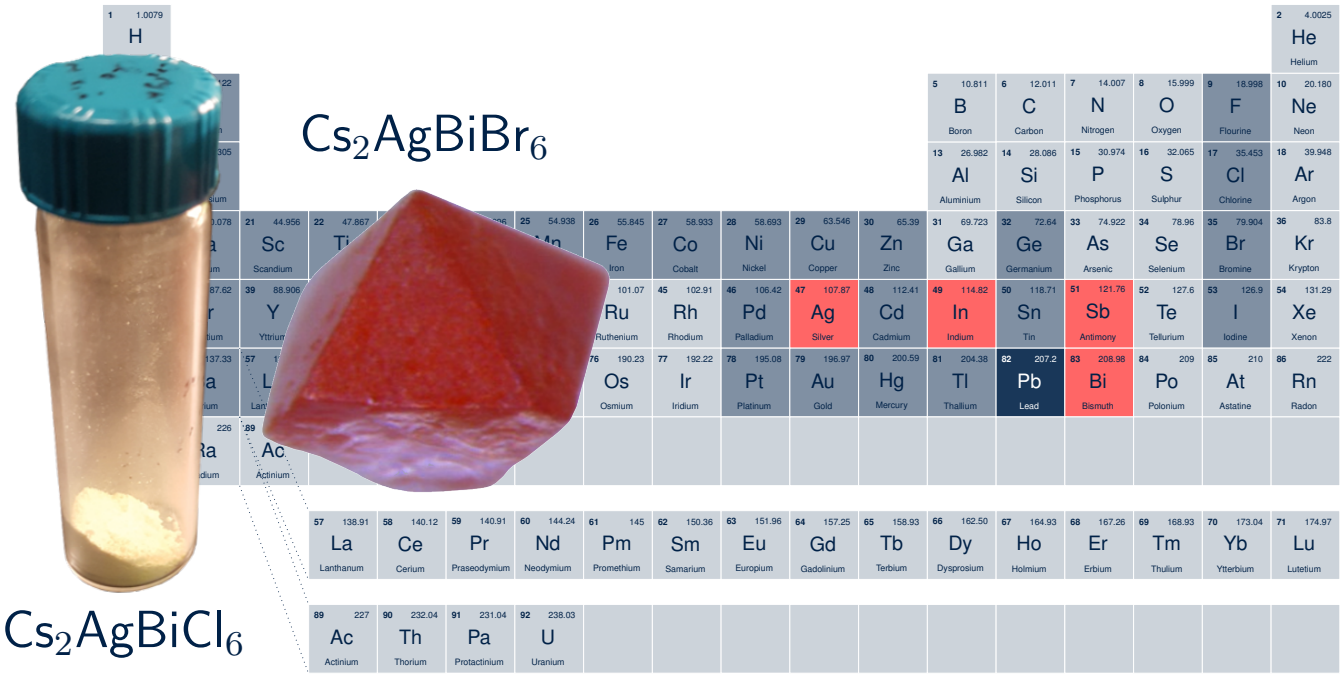
Materials characterization via vibrational spectroscopy



Perez-Osorio, Milot, Filip, Patel, Herz, Johnston, and FG, J. Phys. Chem. C 119, 25703 (2015)

Examples of calculations based on DFT

Computational materials discovery

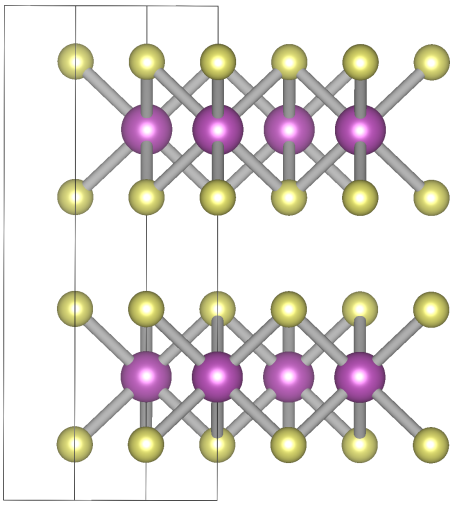


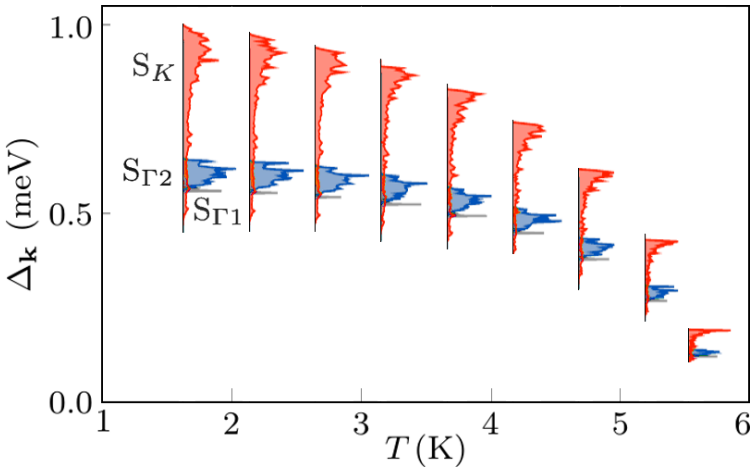
Volonakis, Filip, Haghighirad, Sakai, Wenger, Snaith, and FG, J. Phys. Chem. Lett. 7, 1254 (2016)

Examples of calculations based on DFT

Predictive calculations of the superconducting critical temperature

NbS_2



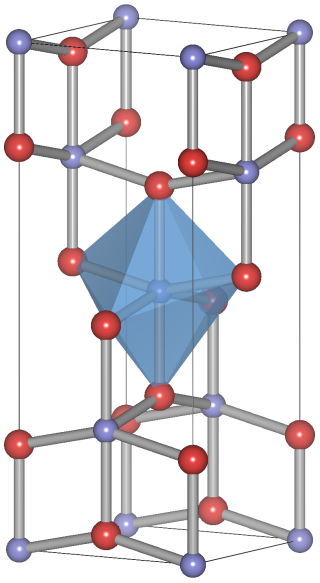


Heil, Poncé, Lambert, Schlipf, Margine, and FG, Phys. Rev. Lett., 119, 087003 (2017)

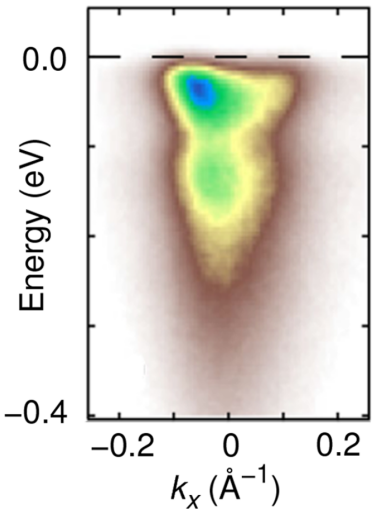
Examples of calculations based on DFT

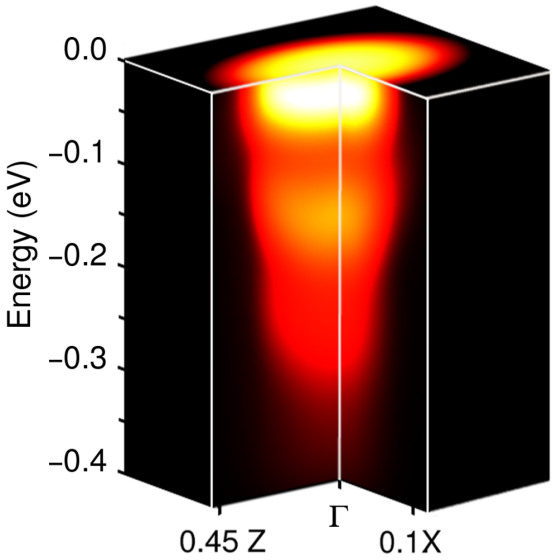
Many-body effects in ARPES

TiO_2 anatase



Moser et al,
PRL 110, 196403 (2013)





Verdi, Caruso, and FG, Nat. Commun. 8, 15769 (2017)

Timeline of DFT methods

1964 Hohenberg–Kohn theorem and Kohn–Sham formulation

1972 Relativistic extension of density functional theory

1980 Local density approximation for exchange and correlation

1984 Time-dependent density functional theory

1985 First-principles molecular dynamics

1986 Quasiparticle corrections for insulators

1987 Density functional perturbation theory

1988 Towards quantum chemistry accuracy

1991 Hubbard-corrected density functional theory

1996 The generalized gradient approximation

Exponential rate of progress in the past two decades

Why is DFT so popular?

- **Transferability**
  We can use the same codes/methods for very different materials
- **Simplicity**
  The Kohn-Sham equations are conceptually very similar to the Schrödinger equation for a single electron in an external potential
- **Reliability**
  Often we can predict materials properties with high accuracy, sometimes even before experiments
- **Software sharing**
  The development of DFT has become a global enterprise, e.g. open source and collaborative software development
- **Robust platform**
  Often the shortcomings of DFT can be cured by using more sophisticated approaches, which still use DFT as their starting point

How many papers using DFT will be published worldwide during this school?

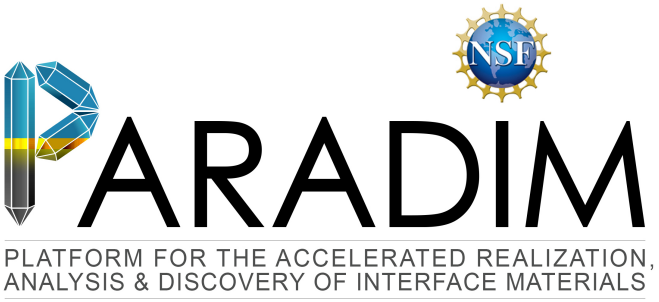
A Ten

B At least two hundred

C Ten thousand

D More than a million

E I have no idea



An Introduction to Density Functional Theory for Experimentalists

Feliciano Giustino

University of Oxford & Cornell University

Ithaca, 8-14 July 2018

Lecture 1.2

Many-body problem

Many-body Schrödinger equation

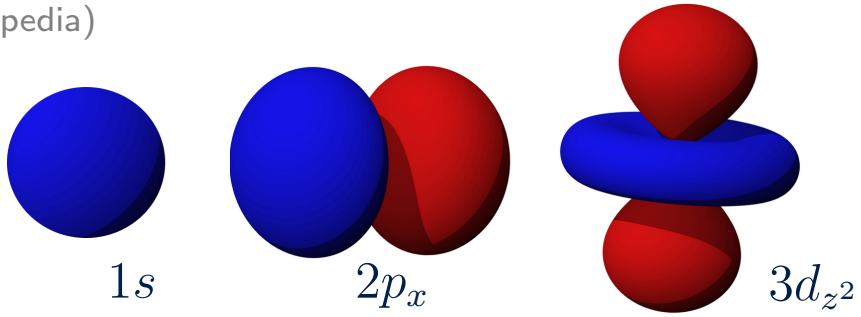
$$\text{Materials} = \text{Electrons} + \text{Nuclei}$$

- Schrödinger equation for the H atom (nucleus at $\mathbf{r} = 0$)

$$-\frac{\hbar^2}{2m_e}\nabla^2\psi(\mathbf{r}) - \frac{e^2}{4\pi\epsilon_0}\frac{1}{|\mathbf{r}|}\psi(\mathbf{r}) = E_{\text{tot}}\psi(\mathbf{r})$$

- wavefunctions of H

(from Wikipedia)



Many-body Schrödinger equation

- Many-body wavefunction (keep it simple: only 3 electrons)

$$\psi(\mathbf{r}) \rightarrow \Psi = \Psi(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$$

- Probability of finding electron $\#1$ at the point $\mathbf{r}$

$$\text{prob}(\mathbf{r}_1 = \mathbf{r}) = \int |\Psi(\mathbf{r}, \mathbf{r}_2, \mathbf{r}_3)|^2 \, d\mathbf{r}_2 d\mathbf{r}_3$$

- Electron density at the point $\mathbf{r}$

$$n(\mathbf{r}) = \text{prob}(\mathbf{r}_1 = \mathbf{r}) + \text{prob}(\mathbf{r}_2 = \mathbf{r}) + \text{prob}(\mathbf{r}_3 = \mathbf{r})$$

- Electrons are indistinguishable

$$n(\mathbf{r}) = 3 \int |\Psi(\mathbf{r}, \mathbf{r}_2, \mathbf{r}_3)|^2 \, d\mathbf{r}_2 d\mathbf{r}_3$$

Many-body Schrödinger equation

$$(\,\text{kinetic energy} + \text{potential energy}\,)\,\Psi = E_{\text{tot}}\Psi$$

- kinetic energy, electrons and nuclei

$$-\sum_{i=1}^{N}\frac{\hbar^2}{2m_e}\nabla_i^2 - \sum_{I=1}^{M}\frac{\hbar^2}{2M_I}\nabla_I^2$$

- potential energy, electron-electron repulsion

$$\frac{1}{2}\sum_{i\neq j}\frac{e^2}{4\pi\epsilon_0}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|}$$

- potential energy, nucleus-nucleus repulsion

$$\frac{1}{2}\sum_{I\neq J}\frac{e^2}{4\pi\epsilon_0}\frac{Z_I Z_J}{|\mathbf{R}_I-\mathbf{R}_J|}$$

- potential energy, electron-nucleus attraction

$$-\sum_{i,I}\frac{e^2}{4\pi\epsilon_0}\frac{Z_I}{|\mathbf{r}_i-\mathbf{R}_I|}$$

Many-body Schrödinger equation

$$\left[-\sum_i \frac{\hbar^2}{2m_e}\nabla_i^2 - \sum_I \frac{\hbar^2}{2M_I}\nabla_I^2 + \frac{1}{2}\sum_{i\neq j} \frac{e^2}{4\pi\epsilon_0}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|} + \frac{1}{2}\sum_{I\neq J} \frac{e^2}{4\pi\epsilon_0}\frac{Z_I Z_J}{|\mathbf{R}_I-\mathbf{R}_J|} - \sum_{i,I} \frac{e^2}{4\pi\epsilon_0}\frac{Z_I}{|\mathbf{r}_i-\mathbf{R}_I|} \right] \Psi = E_{\rm tot}\Psi$$

Hartree atomic units

- masses in units of $m_{\rm e}$ (electron mass)
- lengths in units of a_0 (Bohr radius)
- energies in units of $e^2/4\pi\epsilon_0 a_0$ (Hartree)

$$\left[-\sum_i \frac{1}{2}\nabla_i^2 - \sum_I \frac{1}{2M_I}\nabla_I^2 + \frac{1}{2}\sum_{i\neq j} \frac{1}{|\mathbf{r}_i-\mathbf{r}_j|} + \frac{1}{2}\sum_{I\neq J} \frac{Z_I Z_J}{|\mathbf{R}_I-\mathbf{R}_J|} - \sum_{i,I} \frac{Z_I}{|\mathbf{r}_i-\mathbf{R}_I|} \right] \Psi = E_{\rm tot}\Psi$$

Many-body Schrödinger equation

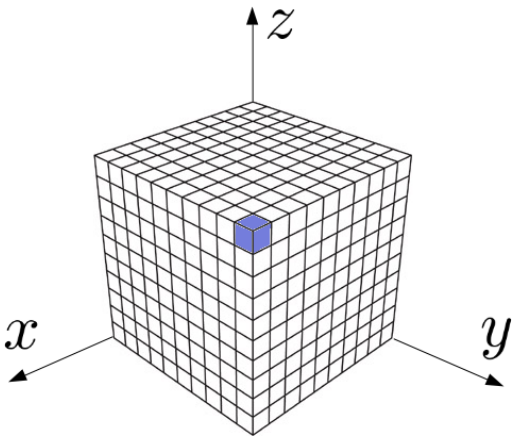
MBSE

MBSE in Hartree units

Exponential wall

Storage requirements for the many-body wavefunction of one unit cell of silicon in the diamond structure

- $\Delta x \sim 0.1$ Å
- $a = 5.43$ Å
- $N_{\rm p} = (a^3/4)/(\Delta x)^3 \sim 40,000$
- 8 valence electrons per unit cell
- $\Psi = 40,000^8$ complex numbers



10^{26} Terabytes

Clamped nuclei approximation

Set nuclear masses $M_I = \infty$:

$$\left[-\sum_i \frac{1}{2}\nabla_i^2 - \cancel{\sum_I \frac{1}{2M_I}\nabla_I^2} + \frac{1}{2}\sum_{i\neq j}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} + \cancel{\frac{1}{2}\sum_{I\neq J}\frac{Z_I Z_J}{|\mathbf{R}_I - \mathbf{R}_J|}} - \sum_{i,I}\frac{Z_I}{|\mathbf{r}_i - \mathbf{R}_I|} \right] \Psi = E_{\rm tot}\Psi$$

$$\left[-\sum_i \frac{\nabla_i^2}{2} + \sum_i V_{\rm n}(\mathbf{r}_i) + \frac{1}{2}\sum_{i\neq j}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} \right] \Psi = E\,\Psi$$

Electronic structure theory in a nutshell

Independent electrons approximation

- Independent particle Hamiltonian

$$\hat{H}_0(\mathbf{r}) = -\frac{1}{2}\nabla^2 + V_{\rm n}(\mathbf{r})$$

- Independent particles + Coulomb

$$\left[\sum_i \hat{H}_0(\mathbf{r}_i) + \frac{1}{2} \sum_{i \neq j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} \right] \Psi(\mathbf{r}_1, \dots \mathbf{r}_N) = E \ \Psi(\mathbf{r}_1, \dots \mathbf{r}_N)$$

- If we neglect this electron-electron Coulomb repulsion, the electrons will not 'feel' each other $\longrightarrow$ joint probability of independent events

$$\Psi(\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N) = \phi_1(\mathbf{r}_1) \cdots \phi_N(\mathbf{r}_N)$$

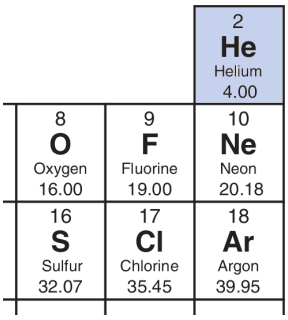
$$\hat{H}_0(\mathbf{r}) \phi_i(\mathbf{r}) = \varepsilon_i \phi_i(\mathbf{r})$$

$$E = \varepsilon_1 + \cdots + \varepsilon_N$$

Example: Ground state of He

- Independent particle equation ($Z = 2$)

$$-\frac{1}{2}\nabla^2\phi(\mathbf{r}) - \frac{2}{|\mathbf{r}|}\phi(\mathbf{r}) = \varepsilon\phi(\mathbf{r})$$



- Lowest-energy solution

$$\phi_{1s}(\mathbf{r}) = \frac{2^{3/2}}{\sqrt{\pi}}\exp\left(-2|\mathbf{r}|\right) \quad \text{with} \quad E_{1s} = -\frac{2^2}{2}$$

- Ground-state many-body wavefunction in the approximation of independent electrons

$$\Psi(\mathbf{r}_1, \mathbf{r}_2) = \phi_{1s}(\mathbf{r}_1)\phi_{1s}(\mathbf{r}_2) = \frac{8}{\pi}\exp\left[-2(|\mathbf{r}_1| + |\mathbf{r}_2|)\right]$$

- Ground-state energy in the approximation of independent electrons

$$E = 2E_{1s} = -4 \text{ Ha} = -108.8 \text{ eV}$$

Exclusion principle

- Let us calculate the electron density for 2 electrons in the independent-electron approximation

$$n(\mathbf{r}) = 2\int |\Psi(\mathbf{r},\mathbf{r}_2)|^2\, d\mathbf{r}_2 = 2\int |\phi_1(\mathbf{r})|^2 |\phi_2(\mathbf{r}_2)|^2\, d\mathbf{r}_2 = 2\,|\phi_1(\mathbf{r})|^2$$

- Admissible wavefunctions must be *antisymmetric* w.r.t. exchange of space and spin variables $\longrightarrow$ Slater determinant (spin-unpolarized)

$$\Psi(\mathbf{r}_1,\mathbf{r}_2) = \frac{1}{\sqrt{2}}\left[\phi_1(\mathbf{r}_1)\phi_2(\mathbf{r}_2) - \phi_1(\mathbf{r}_2)\phi_2(\mathbf{r}_1)\right]$$

- Let us try the density again

$$n(\mathbf{r}) = 2\int |\Psi(\mathbf{r},\mathbf{r}_2)|^2\, d\mathbf{r}_2 = \boxed{|\phi_1(\mathbf{r})|^2 + |\phi_2(\mathbf{r})|^2}$$

Mean-field approximation

- The electron density can be used to determine the electrostatic field generated by the electrons

$$\left[-\frac{1}{2}\nabla^2 + V_{\rm n}(\mathbf{r})\right]\phi_i(\mathbf{r}) = \varepsilon_i\phi_i(\mathbf{r})$$

$$n(\mathbf{r}) = \sum_i |\phi_i(\mathbf{r})|^2 \quad \longleftarrow$$

$$V_{\rm H}(\mathbf{r}) = \int d\mathbf{r}' \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}$$

$$\left[-\frac{1}{2}\nabla^2 + V_{\rm n}(\mathbf{r}) + V_{\rm H}(\mathbf{r})\right]\phi_i^{\rm new}(\mathbf{r}) = \varepsilon_i^{\rm new}\phi_i^{\rm new}(\mathbf{r})$$

- This is Hartree's self-consistent field approximation (1928)
  - ☺ No need for the many-body wavefunction
  - ☹ Requires iterative solution

Exchange energy

- The Hartree approximation does not incorporate the constraint on the antisymmetry of the many-body wavefunction, $\Psi(\mathbf{r}_2,\mathbf{r}_1) = -\Psi(\mathbf{r}_1,\mathbf{r}_2)$
- Incorporating this constraint in the mean-field equation leads to a new potential energy contribution, the **Fock exchange**

$$\left[-\frac{\nabla^2}{2} + V_{\rm n}(\mathbf{r}) + V_{\rm H}(\mathbf{r})\right]\phi_i(\mathbf{r}) + \int d\mathbf{r}'\, V_{\rm X}(\mathbf{r},\mathbf{r}')\,\phi_i(\mathbf{r}') = \varepsilon_i\,\phi_i(\mathbf{r})$$

$$V_{\rm X}(\mathbf{r},\mathbf{r}') = -\sum_{j\in{\rm occ}} \frac{\phi_j^*(\mathbf{r}')\phi_j(\mathbf{r})}{|\mathbf{r}-\mathbf{r}'|}$$

- The Fock potential enforces Pauli's principle by making sure that
  - same-spin electrons repel each other
  - opposite-spin electrons attract each other
- The Fock potential is **non-local**

Correlation energy

- So far we assumed that electrons are independent, that is uncorrelated

$$\text{prob}(\mathbf{r}_1, \mathbf{r}_2) = \text{prob}(\mathbf{r}_1) \times \text{prob}(\mathbf{r}_2)$$

- This is not true since electrons do repel each other, therefore the 'true' wavefunction cannot be expressed as a Slater determinant

$$\Psi_{\text{true}}(\mathbf{r}_1, \mathbf{r}_2) \neq \frac{1}{\sqrt{2}} \left[\phi_1(\mathbf{r}_1)\phi_2(\mathbf{r}_2) - \phi_1(\mathbf{r}_2)\phi_2(\mathbf{r}_1)\right]$$

- Since the Slater determinant is really useful for practical calculations, we keep it and we describe correlations by adding a fictitious potential

$$\left[-\frac{1}{2}\nabla^2 + V_{\text{n}} + V_{\text{H}} + V_{\text{X}} + V_{\text{c}}\right] \phi_i = \varepsilon_i \, \phi_i$$

V_{c}: correlation potential

Example: Ground state of He

		2 **He** Helium 4.00
8 **O** Oxygen 16.00	9 **F** Fluorine 19.00	10 **Ne** Neon 20.18
16 **S** Sulfur 32.07	17 **Cl** Chlorine 35.45	18 **Ar** Argon 39.95

Kinetic energy + electron-nucleus interaction	-3.89 Ha	-105.8 eV
Hartree energy	$+2.05$ Ha	$+55.8$ eV
Fock exchange energy	-1.02 Ha	-27.8 eV
Correlation energy	-0.04 Ha	-1.1 eV
Total energy	-2.90 Ha	-78.9 eV

Kohn-Sham equations

$$\left[-\frac{1}{2}\nabla^2 + V_{\rm n} + V_{\rm H} + V_{xc}\right]\phi_i = \varepsilon_i\,\phi_i$$

e**X**change and **C**orrelation potential

Quizz Time

We want to study the many-body wavefunction of a unit cell of Sr_2RuO_4.
We discretize the volume using 100,000 mesh points.

How many terabytes would we need to store this wavefunction?

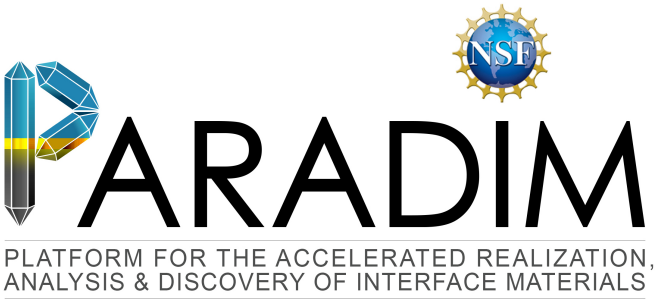
A Less than 1 TB

B 10 TB

C 10^{784} TB

D Infinity

E How much is a terabyte?



An Introduction to Density Functional Theory for Experimentalists

Feliciano Giustino

University of Oxford & Cornell University

Ithaca, 8-14 July 2018

Lecture 2.1

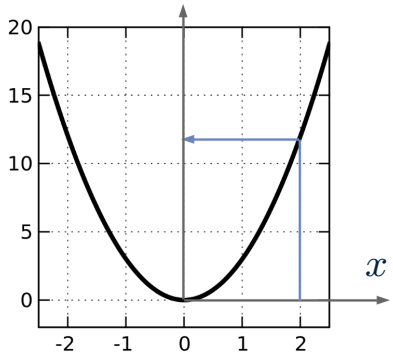
Density-functional theory

Density **F**unctional **T**heory = theory about the energy of electrons being a functional of their density

function

number → number

$g(x) = 3x^2$

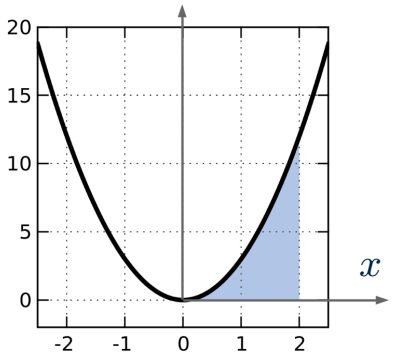


$g(2) = 12$

functional

function → number

$g(x) = 3x^2$



$F[g] = \int_0^2 dx\, g(x) = 8$

Functionals

The total energy is a functional of the wavefunction

$$\hat{H}\,\Psi = E\,\Psi \;\longrightarrow\; E = \int d\mathbf{r}_1 \ldots d\mathbf{r}_N \;\; \Psi^*\,\hat{H}\,\Psi$$

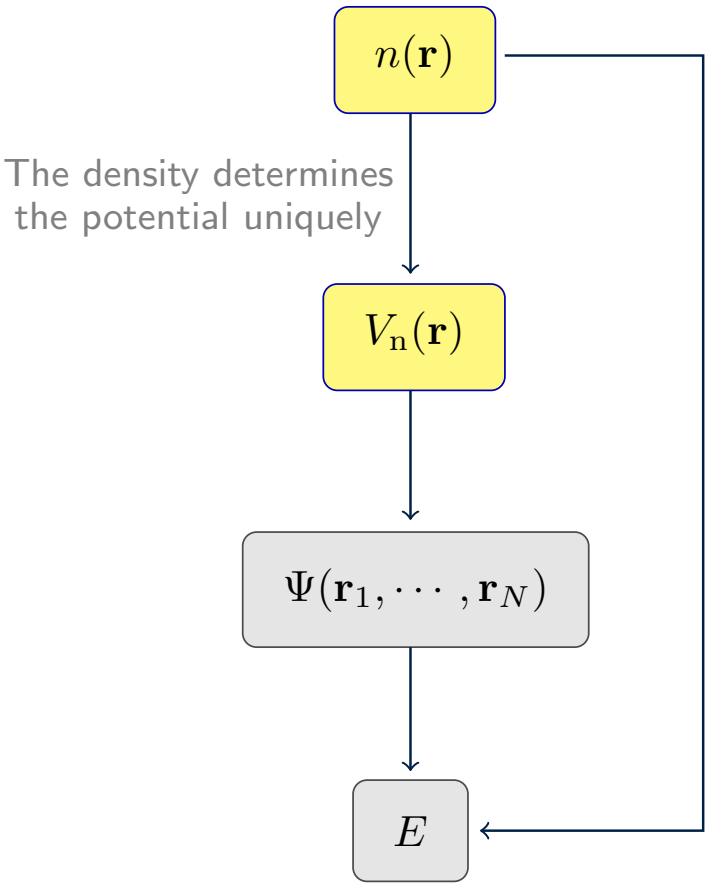
So for a generic quantum state we have

$$\Psi(\mathbf{r}_1,\ldots,\mathbf{r}_N) \longrightarrow E \qquad\qquad E = E[\Psi(\mathbf{r}_1,\ldots,\mathbf{r}_N)]$$

In 1964 Hohenberg and Kohn noted that, for the lowest-energy state, the total energy is also a functional of the electron density

$$n(\mathbf{r}) \longrightarrow E \qquad\qquad E = E[n(\mathbf{r})]$$

Hohenberg-Kohn theorem



HK theorem

In the ground-state the electron density $n_0(\mathbf{r})$ uniquely determines the total energy E_0

HK variational principle

Any $n(\mathbf{r}) \neq n_0(\mathbf{r})$ yields $E > E_0$.

Hohenberg-Kohn theorem

In Lecture 1.2 we had:

$$\left[-\sum_i \frac{1}{2}\nabla_i^2 + \sum_i V_{\rm n}(\mathbf{r}_i) + \frac{1}{2}\sum_{i\neq j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} \right] \Psi = E\,\Psi$$

In order to prove the HK theorem we rewrite the energy in compact notation

$$E = \int d\mathbf{r}\; n(\mathbf{r})\, V_{\rm n}(\mathbf{r}) + \langle \Psi | \hat{U} | \Psi \rangle, \qquad \hat{U} = -\sum_i \frac{1}{2}\nabla_i^2 + \frac{1}{2}\sum_{i\neq j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}$$

Now we want to show the following:

In the ground state, $n(\mathbf{r})$ uniquely determines $V_{\rm n}(\mathbf{r})$

Hohenberg-Kohn theorem

- Assume there are two potentials V_1 and V_2 for the same density (we temporarily suppress the subscript 'n' in $V_{\rm n}$)
- By solving the MBSE for each potential we find the lowest-energy states E_1, Ψ_1 and E_2, Ψ_2, respectively
- Since Ψ_1 **is** the ground state of V_1 we have

$$\int n\, V_1 + \langle \Psi_1 | \hat{U} | \Psi_1 \rangle = E_1$$

- Since Ψ_1 **is not** the ground state of V_2 we have

$$\int n\, V_2 + \langle \Psi_1 | \hat{U} | \Psi_1 \rangle > E_2$$

- The difference gives

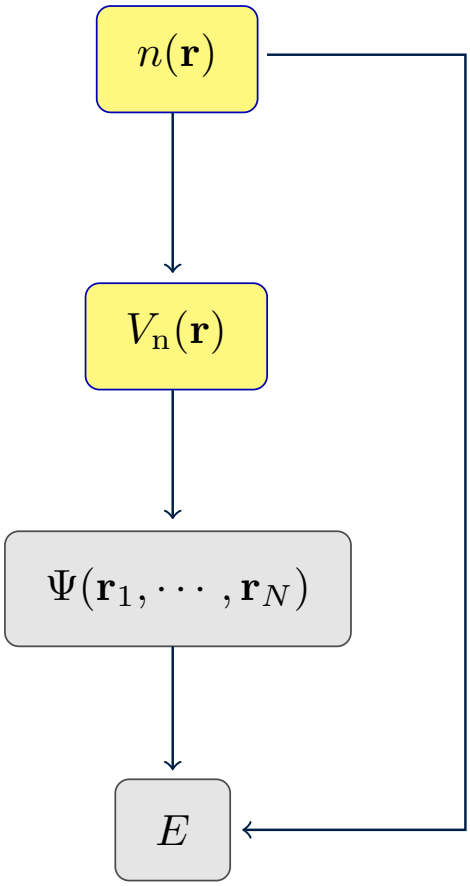
$$\int n\, (V_1 - V_2) > E_1 - E_2$$

- By repeating the same argument starting from Ψ_2 we have

$$\int n\, (V_2 - V_1) > E_2 - E_1$$

- The sum of the last two equations yields the **contradiction** $0 > 0$

Hohenberg-Kohn theorem



The energy functional

The HK theorem states that, in the ground state, the total energy of many electrons is a functional of their density, $E = E[n(\mathbf{r})]$.

What is this functional?

The energy functional is unknown

The scream by E. Munch (1910)

The energy functional

$$E[n] = \underbrace{\int d\mathbf{r}\, n(\mathbf{r}) V_{\rm n}(\mathbf{r})}_{\text{External potential}} + \underbrace{\frac{1}{2}\int d\mathbf{r}\, d\mathbf{r}'\, \frac{n(\mathbf{r})n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}}_{\text{Hartree energy}} + \text{Everything Else}$$

Kohn and Sham (1965) proposed to

(1) Express the electron density **as if** we had a system of independent electrons

$$n(\mathbf{r}) = \sum_{i\in \rm occ} |\phi_i(\mathbf{r})|^2$$

(2) Take out the kinetic energy of these electrons from the "everything else"

$$\text{Everything Else} = -\sum_i \int d\mathbf{r}\, \phi_i^*(\mathbf{r}) \frac{\nabla^2}{2} \phi_i(\mathbf{r}) + \text{Unknown Terms}$$

Kohn-Sham equations

Total energy

$$E[n] = \int d\mathbf{r}\, n(\mathbf{r}) V_{\rm n}(\mathbf{r}) + \frac{1}{2}\int d\mathbf{r} d\mathbf{r}' \frac{n(\mathbf{r})n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} - \sum_i \int d\mathbf{r}\, \phi_i^*(\mathbf{r}) \frac{\nabla^2}{2} \phi_i(\mathbf{r}) + E_{xc}[n]$$

We find the lowest energy state by looking for stationary points of $E[n]$

$$\begin{cases} \dfrac{\delta E}{\delta n} & = & 0 \\ \langle \phi_i | \phi_j \rangle & = & \delta_{ij} \end{cases}$$

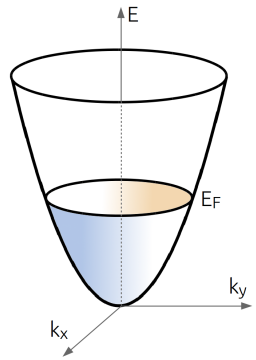
This leads to the Kohn-Sham equations

$$\left[-\frac{1}{2}\nabla^2 + V_{\rm n}(\mathbf{r}) + V_{\rm H}(\mathbf{r}) + V_{xc}(\mathbf{r}) \right] \phi_i(\mathbf{r}) = \varepsilon_i \phi_i(\mathbf{r})$$

$V_{xc} = \dfrac{\delta E_{xc}}{\delta n}$ Exchange and Correlation Potential

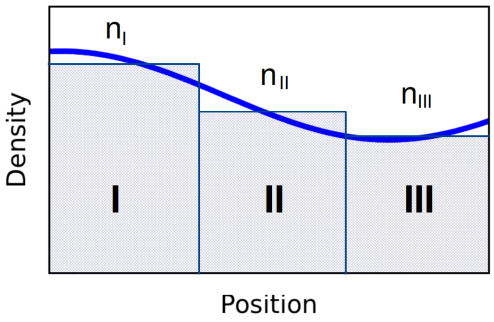
Local Density Approximation (LDA)

We consider the homogeneous electron gas (uniform gas of electrons in a positive compensating background)



$$n(\mathbf{r}) = \text{constant}$$

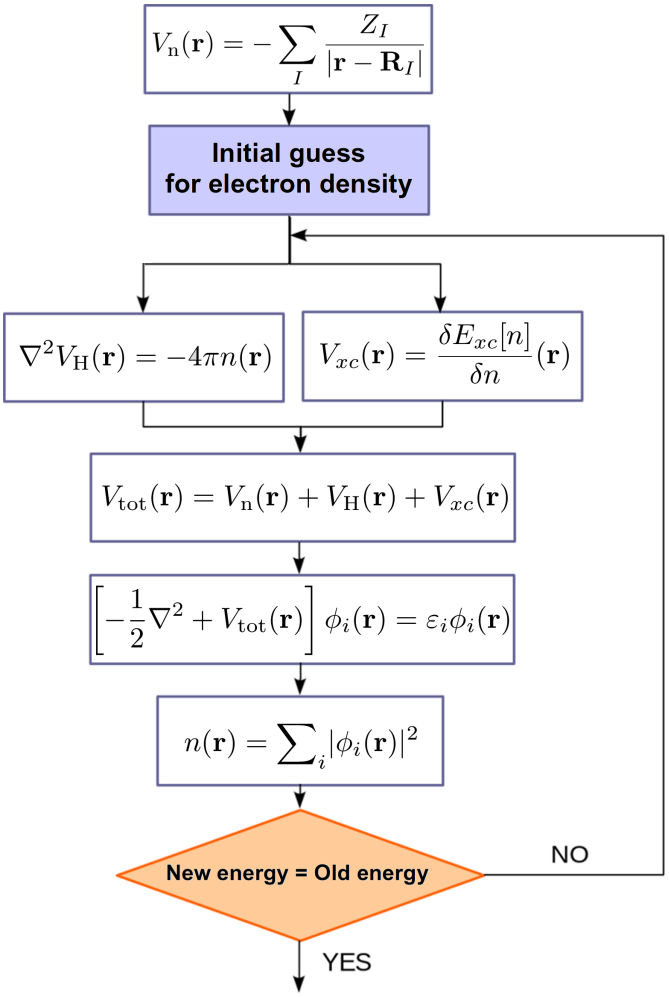
$$E_x^{\text{HEG}} = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{\frac{1}{3}} n^{\frac{4}{3}} V$$



$$E_x^{\text{LDA}} = \int_V \frac{E_x^{\text{HEG}}[n(\mathbf{r})]}{V}\, d\mathbf{r} = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{\frac{1}{3}} \int_V n^{\frac{4}{3}}(\mathbf{r}) d\mathbf{r}$$

$$V_x^{\text{LDA}} = \frac{\delta E_x^{\text{LDA}}}{\delta n} = -\left(\frac{3}{\pi}\right)^{\frac{1}{3}} n^{\frac{1}{3}}(\mathbf{r})$$

Self-consistent field calculations (SCF)



Successes and failures

	1 1A												13 3A	14 4A	15 5A	16 6A	17 7A	18 8A
1	1 **H** Hydrogen 1.01	2 2A																2 **He** Helium 4.00
2	3 **Li** Lithium 6.94	4 **Be** Beryllium 9.01											5 **B** Boron 10.81	6 **C** Carbon 12.01	7 **N** Nitrogen 14.01	8 **O** Oxygen 16.00	9 **F** Fluorine 19.00	10 **Ne** Neon 20.18
3	11 **Na** Sodium 22.99	12 **Mg** Magnesium 24.31	3 3B	4 4B	5 5B	6 6B	7 7B	8	9 8B	10	11 1B	12 2B	13 **Al** Aluminum 26.98	14 **Si** Silicon 28.09	15 **P** Phosphorus 30.97	16 **S** Sulfur 32.07	17 **Cl** Chlorine 35.45	18 **Ar** Argon 39.95
4	19 **K** Potassium 39.10	20 **Ca** Calcium 40.08	21 **Sc** Scandium 44.96	22 **Ti** Titanium 47.87	23 **V** Vanadium 50.94	24 **Cr** Chromium 52.00	25 **Mn** Manganese 54.94	26 **Fe** Iron 55.85	27 **Co** Cobalt 58.93	28 **Ni** Nickel 58.69	29 **Cu** Copper 63.55	30 **Zn** Zinc 65.39	31 **Ga** Gallium 69.72	32 **Ge** Germanium 72.61	33 **As** Arsenic 74.92	34 **Se** Selenium 78.96	35 **Br** Bromine 79.90	36 **Kr** Krypton 83.80
5	37 **Rb** Rubidium 85.47	38 **Sr** Strontium 87.62	39 **Y** Yttrium 88.91	40 **Zr** Zirconium 91.22	41 **Nb** Niobium 92.91	42 **Mo** Molybdenum 95.94	43 **Tc** Technetium (98)	44 **Ru** Ruthenium 101.07	45 **Rh** Rhodium 102.91	46 **Pd** Palladium 106.42	47 **Ag** Silver 107.87	48 **Cd** Cadmium 112.41	49 **In** Indium 114.82	50 **Sn** Tin 118.71	51 **Sb** Antimony 121.76	52 **Te** Tellurium 127.60	53 **I** Iodine 126.90	54 **Xe** Xenon 131.29
6	55 **Cs** Cesium 132.91	56 **Ba** Barium 137.33	57 **La** Lanthanum 138.91	72 **Hf** Hafnium 178.49	73 **Ta** Tantalum 180.95	74 **W** Tungsten 183.84	75 **Re** Rhenium 186.21	76 **Os** Osmium 190.23	77 **Ir** Iridium 192.22	78 **Pt** Platinum 195.08	79 **Au** Gold 196.97	80 **Hg** Mercury 200.59	81 **Tl** Thallium 204.38	82 **Pb** Lead 207.2	83 **Bi** Bismuth 208.98	84 **Po** Polonium (209)	85 **At** Astatine (210)	86 **Rn** Radon (222)
7	87 **Fr** Francium (223)	88 **Ra** Radium (226)	89 **Ac** Actinium (227)	104 **Rf** Rutherfordium (261)	105 **Db** Dubnium (262)	106 **Sg** Seaborgium (266)	107 **Bh** Bohrium (264)	108 **Hs** Hassium (269)	109 **Mt** Meitnerium (268)									

58 **Ce** Cerium 140.12	59 **Pr** Praseodymium 140.91	60 **Nd** Neodymium 144.24	61 **Pm** Promethium (145)	62 **Sm** Samarium 150.36	63 **Eu** Europium 151.96	64 **Gd** Gadolinium 157.25	65 **Tb** Terbium 158.93	66 **Dy** Dysprosium 162.50	67 **Ho** Holmium 164.93	68 **Er** Erbium 167.26	69 **Tm** Thulium 168.93	70 **Yb** Ytterbium 173.04	71 **Lu** Lutetium 174.97
90 **Th** Thorium 232.04	91 **Pa** Protactinium 231.04	92 **U** Uranium 238.03	93 **Np** Neptunium (237)	94 **Pu** Plutonium (244)	95 **Am** Americium (243)	96 **Cm** Curium (247)	97 **Bk** Berkelium (247)	98 **Cf** Californium (251)	99 **Es** Einsteinium (252)	100 **Fm** Fermium (257)	101 **Md** Mendelevium (258)	102 **No** Nobelium (259)	103 **Lr** Lawrencium (262)

How many terabytes would we need in DFT to study wavefunctions in a unit cell of Sr_2RuO_4 ?

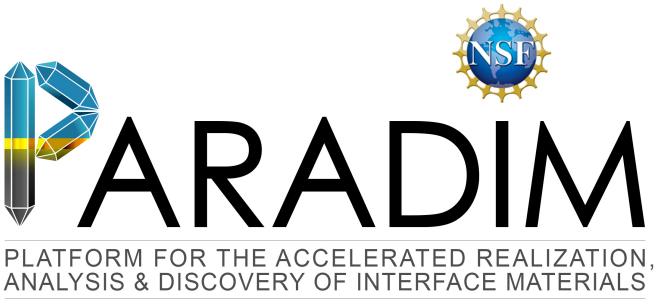
A 10 TB

B 10^{784} TB

C 1 MB

D 250 MB

E I need coffee



An Introduction to Density Functional Theory for Experimentalists

Feliciano Giustino

University of Oxford & Cornell University

Ithaca, 8-14 July 2018

Lecture 2.2

Planewaves and pseudopotentials

Born-von Kármán boundary conditions

DFT calculations require the numerical solution of the KS equations

$$-\frac{1}{2}\nabla^2\phi_i(\mathbf{r}) + V_{\rm tot}(\mathbf{r})\phi_i(\mathbf{r}) = \varepsilon_i\phi_i(\mathbf{r})$$

2nd order PDE $\longrightarrow$ for every y and z we need two boundary conditions on x

- **Localized system**
  atom, molecule, quantum dot, nanowire, slab
  $\phi_i(x,y,z) = 0 \quad \text{for} \quad x = -\infty, \qquad \phi_i(x,y,z) = 0 \quad \text{for} \quad x = +\infty$

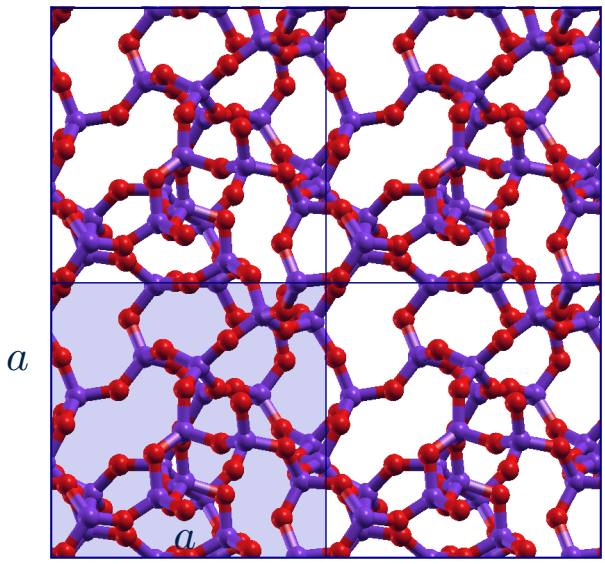
- **Extended system**
  solid, liquid
  $\phi_i(x+a,y,z) = \phi_i(x,y,z), \qquad \nabla\phi_i(x+a,y,z) = \nabla\phi_i(x,y,z)$

Periodic (BvK) boundary conditions

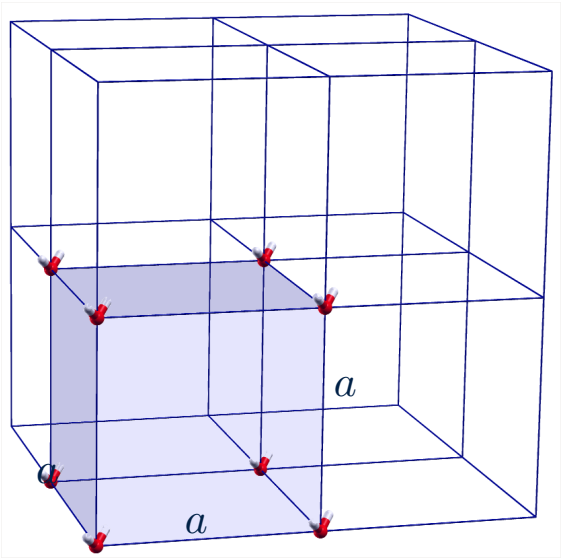
Born-von Kármán boundary conditions

DFT calculations for solids, liquids, interfaces, and nanostructures are performed using BvK boundary conditions

amorphous SiO_2

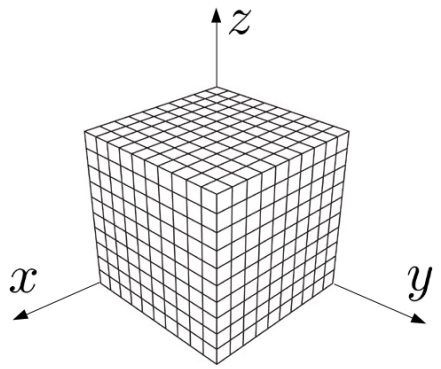


H_2O molecule

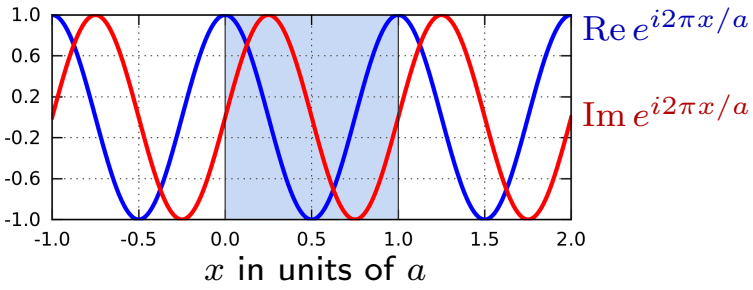


Planewaves

A convenient way of handling the KS wavefunctions is by expanding them in a basis of planewaves $\longrightarrow$ standard Fourier transform



1D case $\phi(x) = \sum_{n=-\infty}^{+\infty} c_n \, \mathrm{e}^{i2\pi nx/a}$



BvK conditions built in

$$\phi(x+a) = \phi(x)$$

$$\nabla\phi(x+a) = \nabla\phi(x)$$

Planewaves

In 2D and 3D we replace $2\pi/a$ by the primitive vectors of the reciprocal lattice

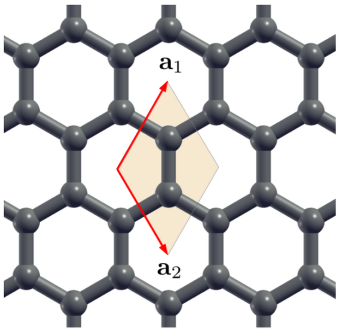
$$\mathbf{b}_1 = 2\pi \frac{\mathbf{a}_2 \times \mathbf{a}_3}{\mathbf{a}_1 \cdot \mathbf{a}_2 \times \mathbf{a}_3}$$

Reciprocal lattice vectors

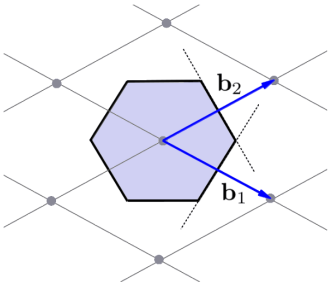
$$\mathbf{G} = m_1\mathbf{b}_1 + m_2\mathbf{b}_2 + m_3\mathbf{b}_3, \text{ with } m_1, m_2, m_3 \text{ integers}$$

Example: graphene

direct lattice



reciprocal lattice



Planewave in 2D or 3D $\exp(i\mathbf{G}\cdot\mathbf{r})$

Planewaves

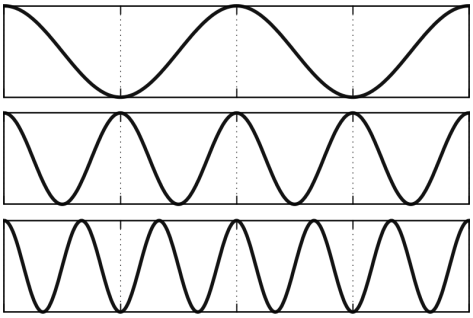
Kohn-Sham wavefunction in a basis of planewaves

$$\phi_i(\mathbf{r}) = \sum_{\mathbf{G}} c_i(\mathbf{G}) \exp(i\mathbf{G}\cdot\mathbf{r})$$

By replacing in the KS equations we obtain

$$\frac{|\mathbf{G}|^2}{2} c_i(\mathbf{G}) + \sum_{\mathbf{G}'} V_{\rm tot}(\mathbf{G}-\mathbf{G}') c_i(\mathbf{G}') = \varepsilon_i \, c_i(\mathbf{G})$$

How many planewave $\mathbf{G}$-vectors should we include in the expansion?



$|\mathbf{G}| = 2\pi/a$

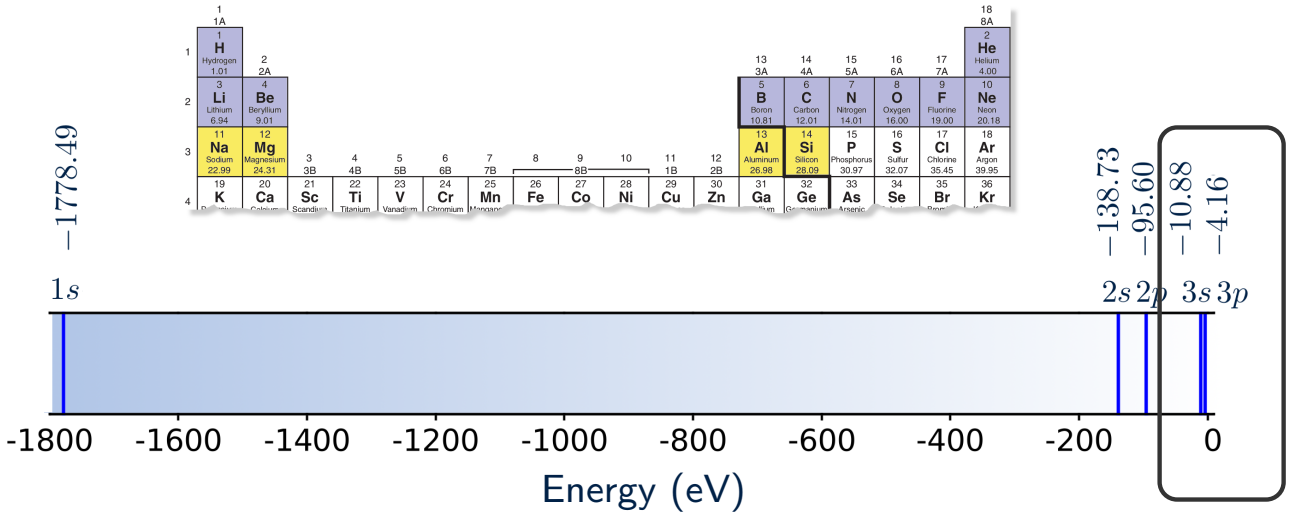
$|\mathbf{G}| = 2 \cdot 2\pi/a$

$|\mathbf{G}| = 3 \cdot 2\pi/a$

$$E_{\rm cut} = \frac{\hbar^2 |\mathbf{G}_{\rm max}|^2}{2m_{\rm e}}$$

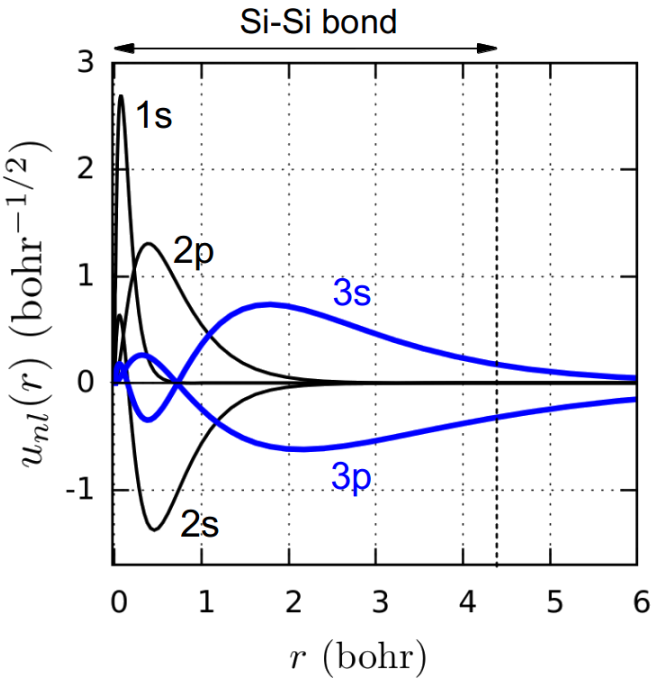
planewaves kinetic energy cutoff

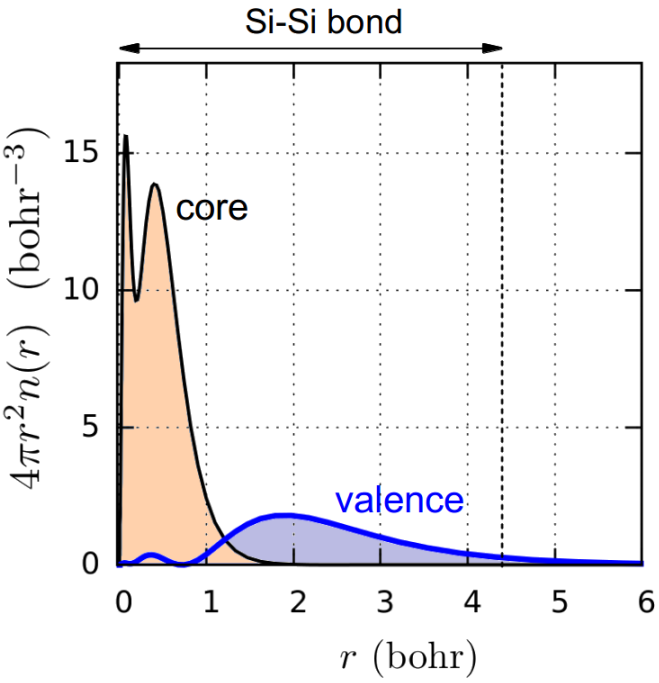
Atomic wavefunctions of silicon (DFT/LDA)



Only valence electrons are important for bonding

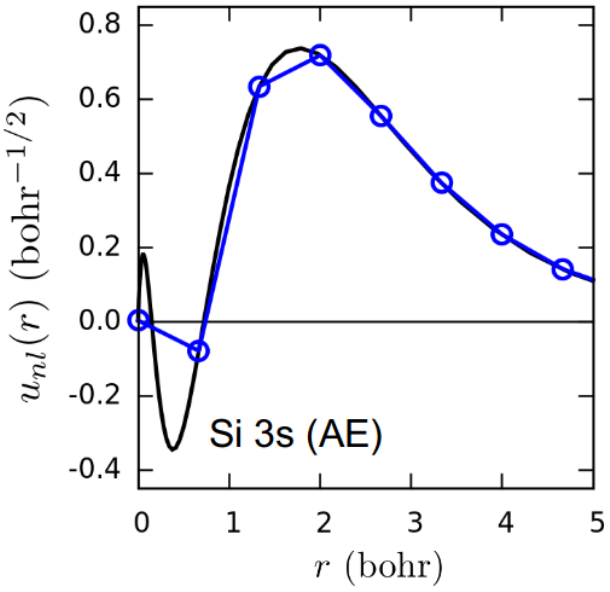
Atomic wavefunctions of silicon (DFT/LDA)

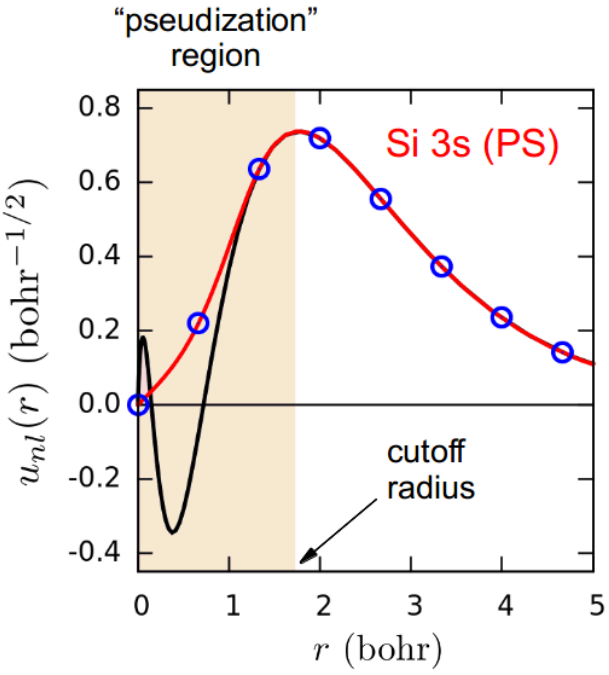




$$\psi_{nl}(\mathbf{r}) = \frac{u_{nl}(r)}{r} Y_{lm}\left(\frac{\mathbf{r}}{r}\right)$$

Pseudization: make wavefunctions smooth by removing the nodes

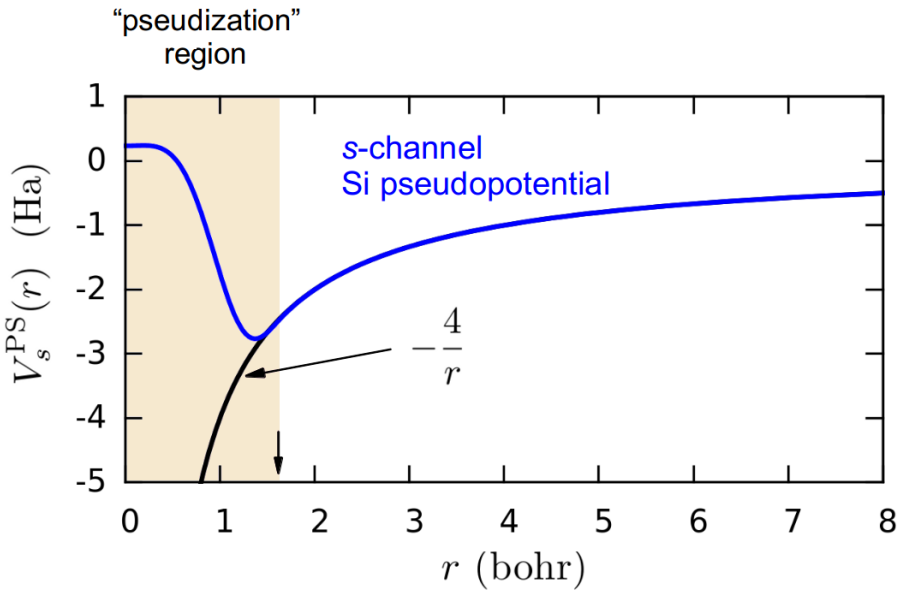




Pseudopotentials

Reverse-engineer the pseudo-potential potential which yields the pseudo-wavefunction as solution of the atomic Schrödinger equation

$$-\frac{1}{2}\frac{d^2}{dr^2}\,u_{3s}^{\rm PS}+V_{3s}^{\rm PS}\,u_{3s}^{\rm PS}=E_{3s}\,u_{3s}^{\rm PS}\quad\longrightarrow\quad V_{3s}^{\rm PS}=E_{3s}+\frac{1}{2u_{3s}^{\rm PS}}\frac{d^2u_{3s}^{\rm PS}}{dr^2}$$

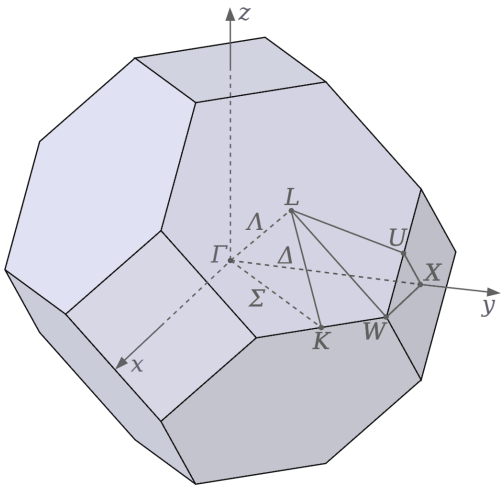


Brillouin zone sampling

In **crystalline** solids we label electronic states by their **Bloch wavevector** $\mathbf{k}$

Bloch theorem $\quad \phi_{i\mathbf{k}}(\mathbf{r}) = \mathrm{e}^{i\mathbf{k}\cdot\mathbf{r}} u_{i\mathbf{k}}(\mathbf{r}) \;$ with $\; u_{i\mathbf{k}}(\mathbf{r}+\mathbf{R}) = u_{i\mathbf{k}}(\mathbf{r})$

$$n(\mathbf{r}) = \sum_{i\in\mathrm{occ}} |\phi_i(\mathbf{r})|^2 \;\longrightarrow\; \sum_{i\in\mathrm{occ}} \int_{\mathrm{BZ}} \frac{d\mathbf{k}}{\Omega_{\mathrm{BZ}}} |u_{i\mathbf{k}}(\mathbf{r})|^2 \;\simeq\; \frac{1}{N_{\mathbf{k}}} \sum_{\mathbf{k}\in\mathrm{BZ}} \sum_{i\in\mathrm{occ}} |u_{i\mathbf{k}}(\mathbf{r})|^2$$



Brillouin zone of fcc crystal (e.g. Si, Cu)

- We discretize this volume using a uniform mesh
- We reduce the number of $\mathbf{k}$-points using the crystal symmetry operations

What is a pseudopotential ?

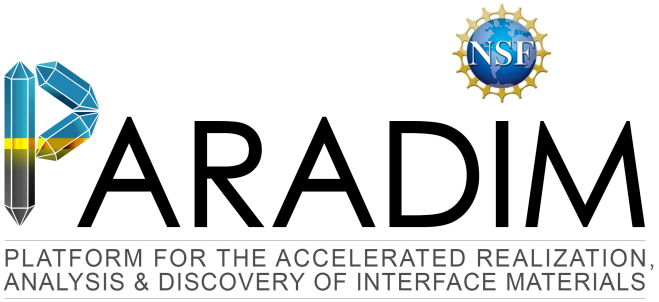
A An effective atomic potential describing nucleus & core electrons

B A potential describing the pseudo spin

C A false potential

D The potential in the Kohn-Sham equations of DFT

E This question is too easy



An Introduction to Density Functional Theory for Experimentalists

Feliciano Giustino

University of Oxford & Cornell University

Ithaca, 8-14 July 2018

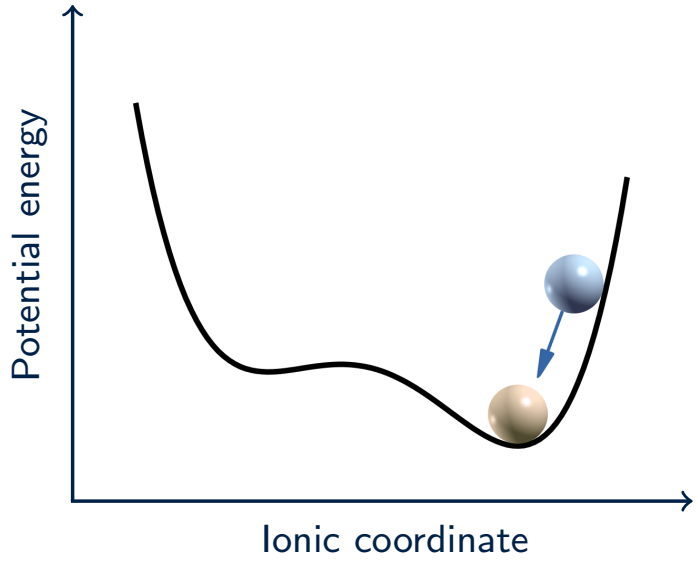
Lecture 3.1

Equilibrium structures

Equilibrium structures

In order to find the equilibrium structures of materials

1) We determine the **potential energy surface** of the ions
2) We look for the minima of this surface $\longrightarrow$ zero net forces on the ions



~~Clamped nuclei approximation~~

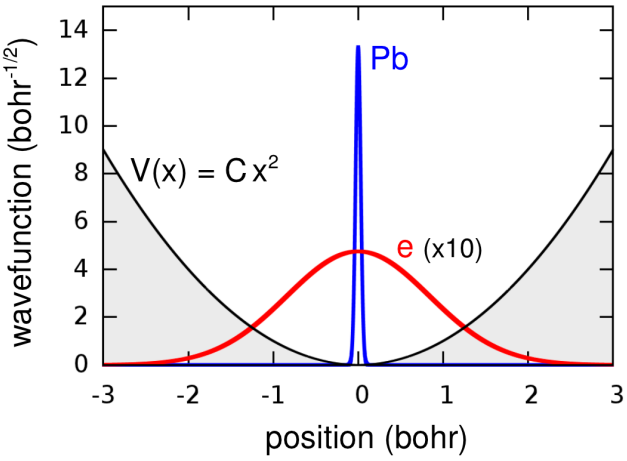
Back to the complete many-body Schrödinger equation for electrons & nuclei

$$\left[-\sum_i \frac{\nabla_i^2}{2} - \sum_I \frac{\nabla_I^2}{2M_I} - \sum_{i,I} \frac{Z_I}{|\mathbf{r}_i - \mathbf{R}_I|} + \frac{1}{2}\sum_{i\neq j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} + \frac{1}{2}\sum_{I\neq J} \frac{Z_I Z_J}{|\mathbf{R}_I - \mathbf{R}_J|}\right] \Psi = E_{\rm tot}\Psi$$

Here $\Psi = \Psi(\mathbf{r}_1, \ldots, \mathbf{r}_N, \mathbf{R}_1, \ldots, \mathbf{R}_M)$

Example
The wavefunction of an electron vs the wavefunction of the Pb nucleus

$$-\frac{1}{2M}\frac{d^2\psi(x)}{dx^2} + Cx^2\psi(x) = E\psi(x)$$



Born-Oppenheimer approximation

Born and Oppenheimer (1927) proposed the following approximation

- Factorize the electron-nuclear wavefunction

$$\Psi(\mathbf{r}_1,\ldots,\mathbf{r}_N,\mathbf{R}_1,\ldots,\mathbf{R}_M) \simeq \Psi_{\mathbf{R}}(\mathbf{r}_1,\ldots,\mathbf{r}_N)\,\chi(\mathbf{R}_1,\ldots,\mathbf{R}_M)$$

- Find the electronic part as the ground state of Schrödinger equation with the nuclei clamped at $\mathbf{R}_1,\ldots,\mathbf{R}_M$

$$\left[-\sum_i \frac{\nabla_i^2}{2} + \sum_i V_{\rm n}(\mathbf{r}_i;\mathbf{R}) + \frac{1}{2}\sum_{i\neq j}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|}\right]\Psi_{\mathbf{R}} = E(\mathbf{R}_1,\ldots,\mathbf{R}_M)\,\Psi_{\mathbf{R}}$$

- Replace the result in the complete MBSE of the previous slide

$$\left[-\sum_I \frac{\nabla_I^2}{2M_I} + \frac{1}{2}\sum_{I\neq J}\frac{Z_I Z_J}{|\mathbf{R}_I-\mathbf{R}_J|} + E(\mathbf{R}_1,\ldots,\mathbf{R}_M)\right]\chi = E_{\rm tot}\,\chi$$

Schrödinger equation for nuclei

Potential energy surface

$$\underbrace{-\sum_I \frac{\nabla_I^2}{2M_I}}_{\text{Kinetic Energy}}\chi + \underbrace{\left[\frac{1}{2}\sum_{I\neq J}\frac{Z_I Z_J}{|\mathbf{R}_I-\mathbf{R}_J|} + E(\mathbf{R}_1,\ldots,\mathbf{R}_M)\right]}_{\text{Potential Energy}}\chi = E_{\text{tot}}\,\chi$$

Potential energy surface

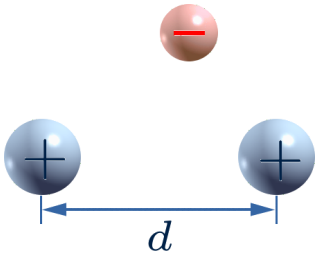
$$U(\mathbf{R}_1,\ldots,\mathbf{R}_M) = \frac{1}{2}\sum_{I\neq J}\frac{Z_I Z_J}{|\mathbf{R}_I-\mathbf{R}_J|} + E(\mathbf{R}_1,\ldots,\mathbf{R}_M)$$

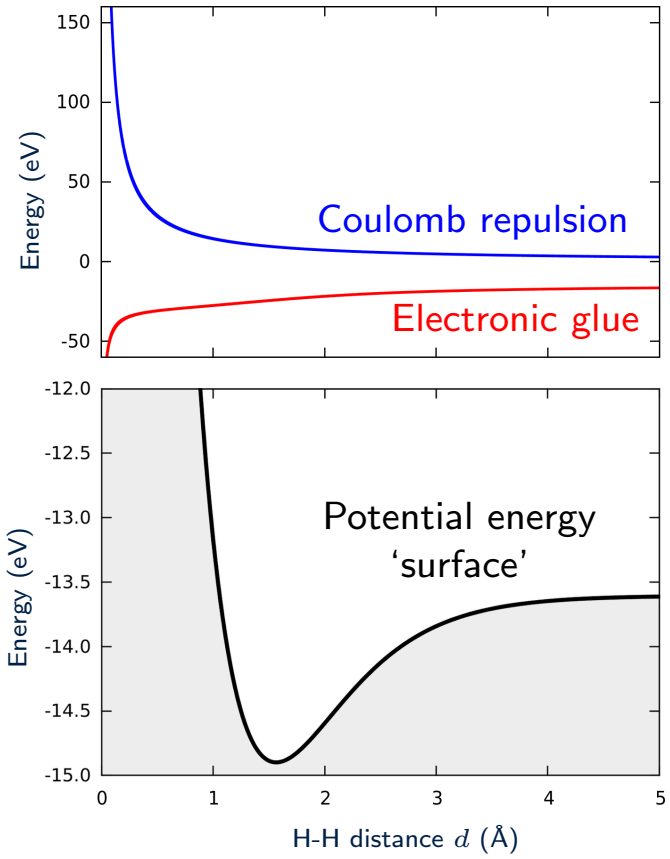
Coulomb repulsion between positive nuclei

Glue resulting from the negative charge of the electrons (from DFT)

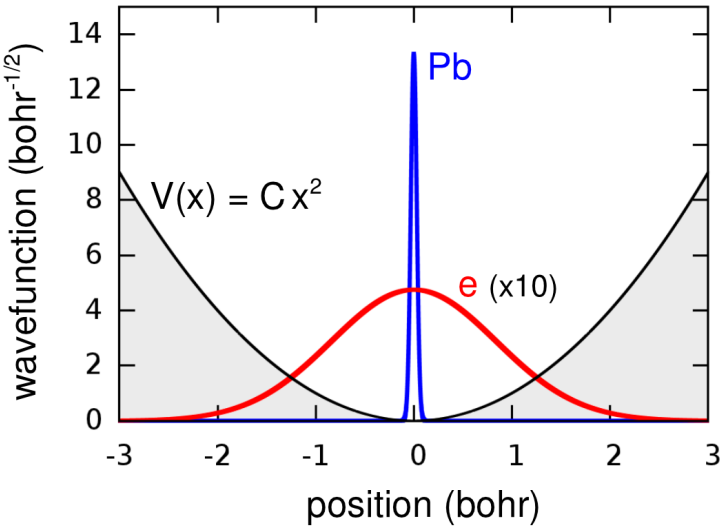
Potential energy surface

H_2^+ molecular ion





Classical nuclei



$$\hat{H} = -\sum_I \frac{\nabla_I^2}{2M_I} + U(\mathbf{R}_1, \ldots, \mathbf{R}_M)$$

$$\downarrow$$

$$H = -\sum_I \frac{\mathbf{P}_I^2}{2M_I} + U(\mathbf{R}_1, \ldots, \mathbf{R}_M)$$

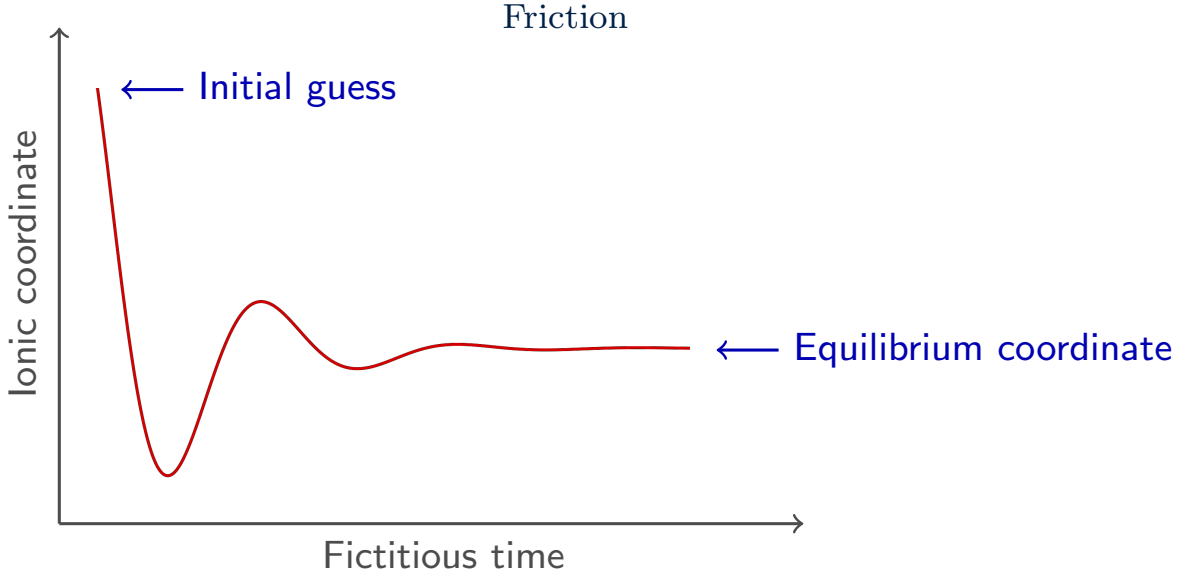
$$\downarrow$$

Newton's equation for nuclei

$$M_I \frac{d^2\mathbf{R}_I}{dt^2} = \mathbf{F}_I = -\frac{\partial U}{\partial \mathbf{R}_I}$$

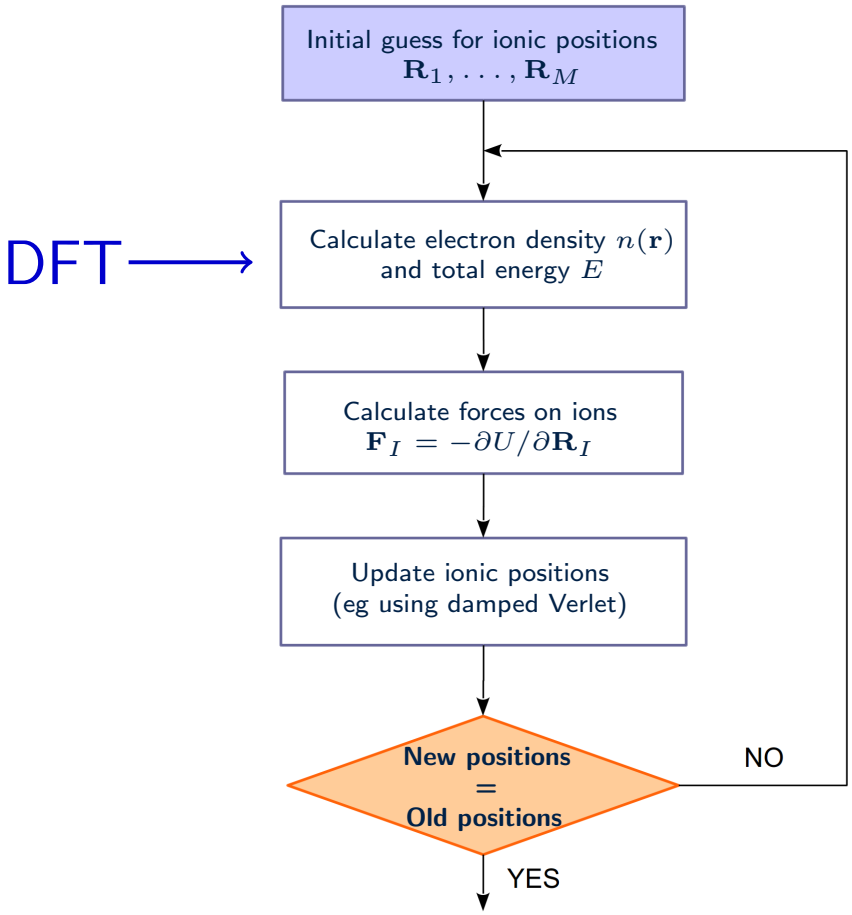
Example: Finding the minima of the PES using molecular dynamics

$$M\frac{d^2x}{dt^2} = -\frac{\partial U}{\partial x} \underbrace{-\frac{2M}{\tau}\frac{dx}{dt}}_{\text{Friction}}$$



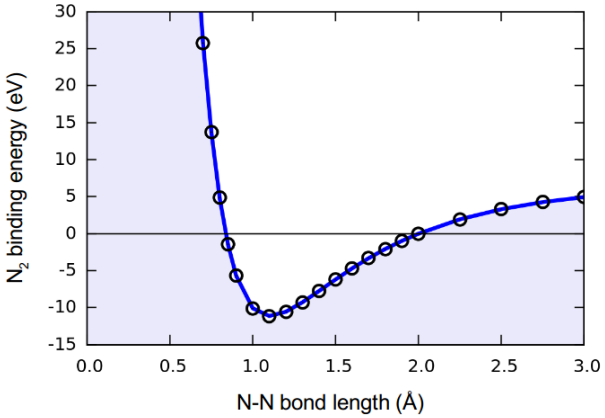
(Verlet) $M\frac{x_{i+1}-2x_i+x_{i-1}}{\Delta t^2} = -\frac{U(x_i)-U(x_{i-1})}{x_i-x_{i-1}} - \frac{2M}{\tau}\frac{x_{i+1}-x_{i-1}}{2\Delta t}$

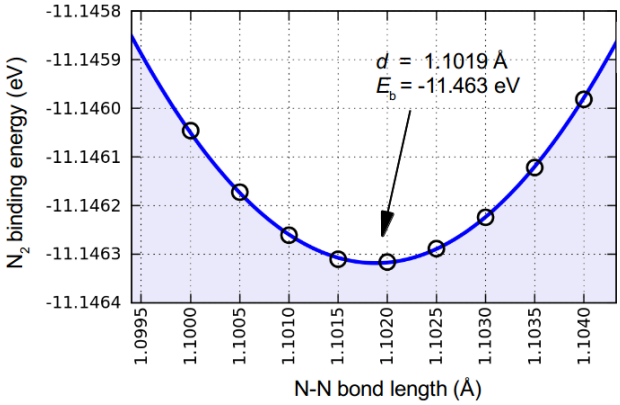
Structural optimization



Examples

Simplest case of structural optimization: N_2 diatomic molecule

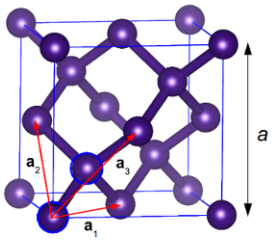


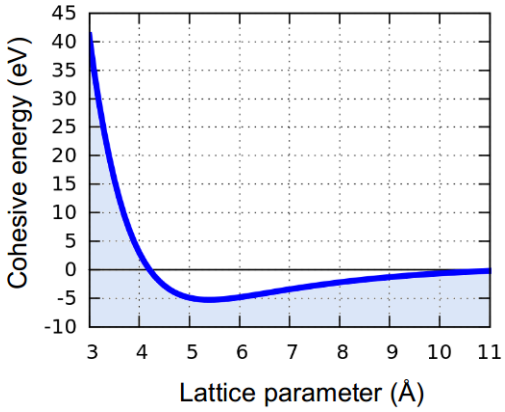


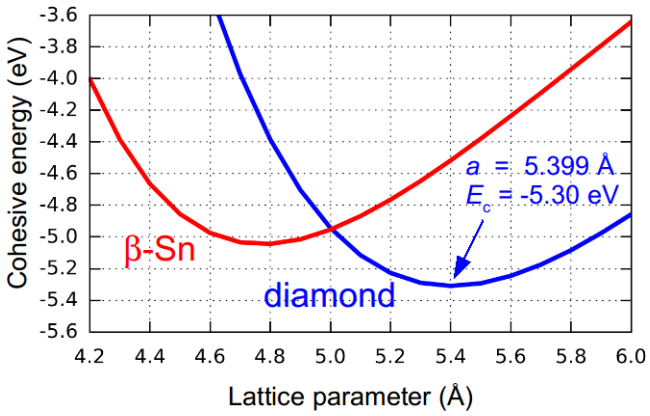
	DFT/LDA	Experiment	Rel. Error
bond length (Å)	1.102	1.098	0.4%
binding energy (eV)	11.46	9.76	17%

Note Nitrogen is $[1s^2]2s^2p^3$, therefore it has a spin $S = 3/2$ after the Hund's first rule. As the above calculations are spin-unpolarized, the energy at infinity is higher than twice the energy of one N atom.

Structural optimization of bulk crystals: Silicon

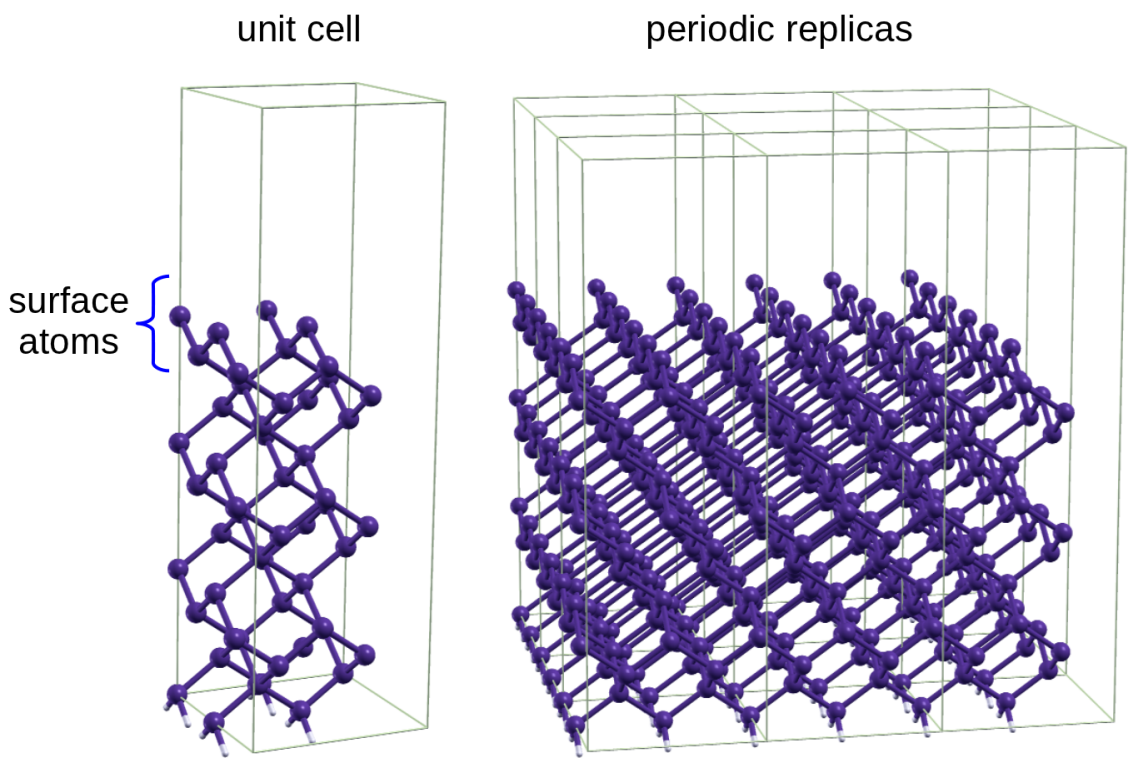




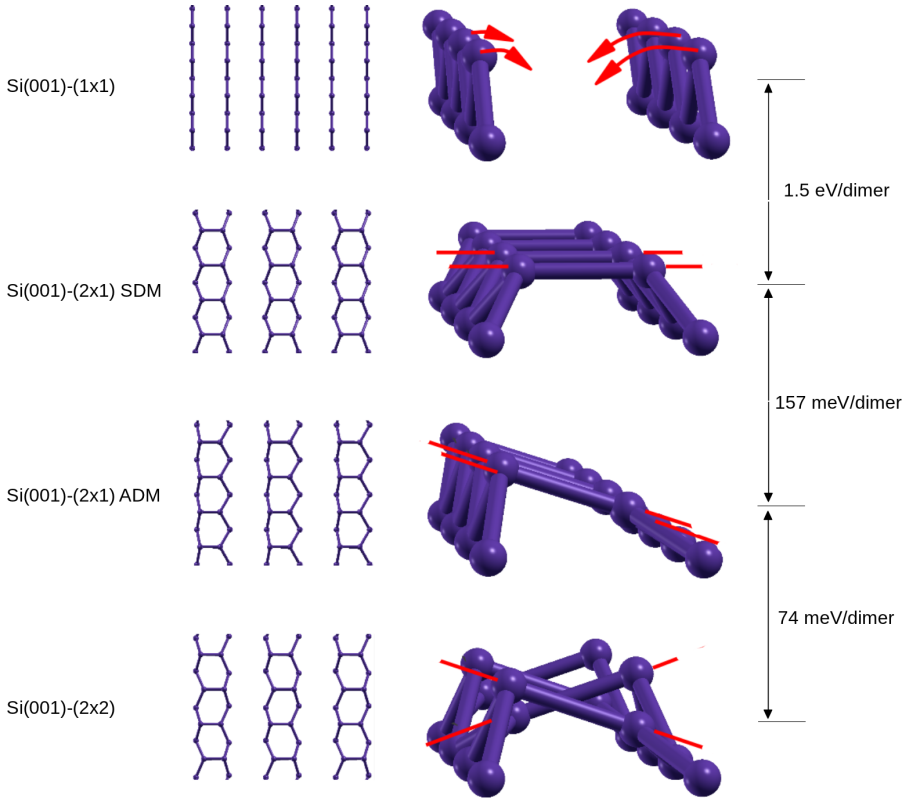


	DFT/LDA	Experiment	Rel. Error
lattice parameter (Å)	5.40	5.43	0.6%
cohesive energy (eV)	5.30	4.62	15%

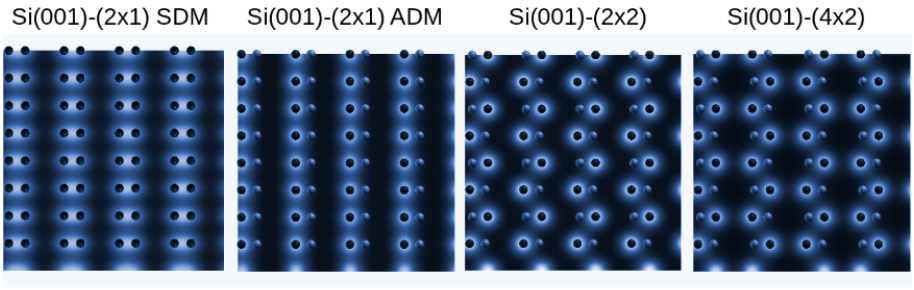
Structural optimization of surfaces: Clean Si(001) surface



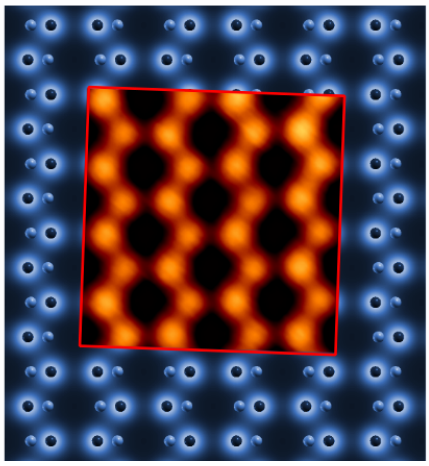
Structural optimization of surfaces: Clean Si(001) surface



Clean Si(001) surface: Calculated vs. measured STM images

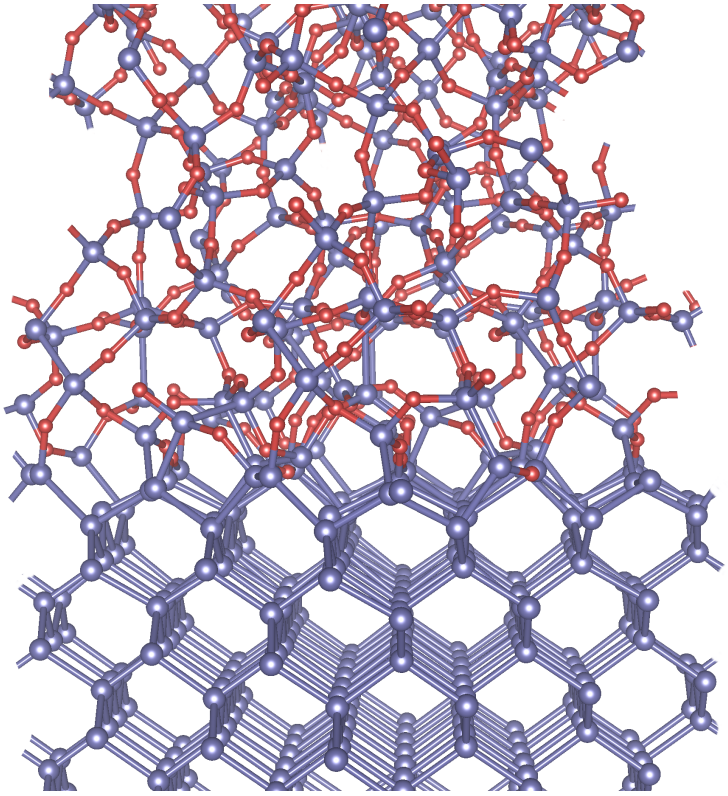


Si(001)-(4x2): DFT vs. experiment



Experimental STM image courtesy of T. Yokoyama
http://dx.doi.org/10.1103/PhysRevB.61.R5078

Si(001)/a-SiO_2 interface



from Giustino and Pasquarello
https://doi.org/10.1063/1.1923185

In which of the following systems the Born-Oppenheimer approximation breaks down?

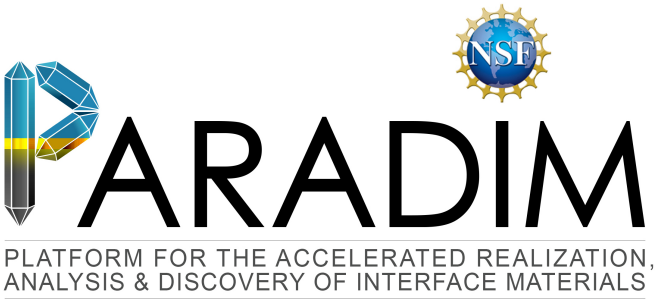
A Organic-inorganic lead halide perovskites

B Sulfur hydride high-temperature superconductors

C Diamond

D Graphene

E What is the Born-Oppenheimer approximation?



An Introduction to Density Functional Theory for Experimentalists

Feliciano Giustino

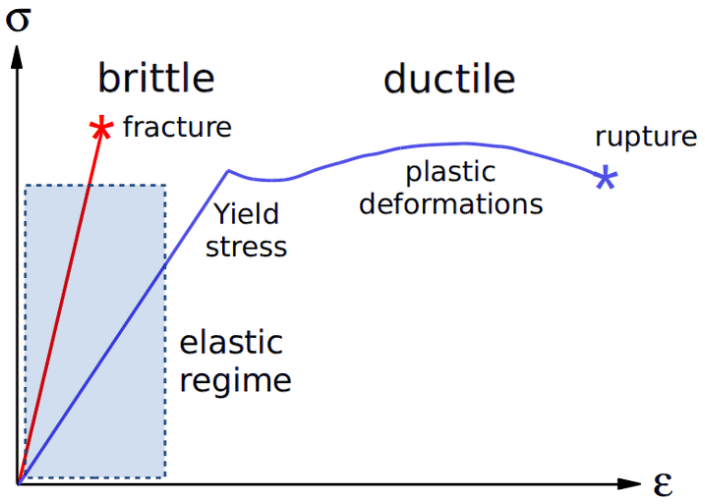
University of Oxford & Cornell University

Ithaca, 8-14 July 2018

Lecture 3.2

Elastic properties

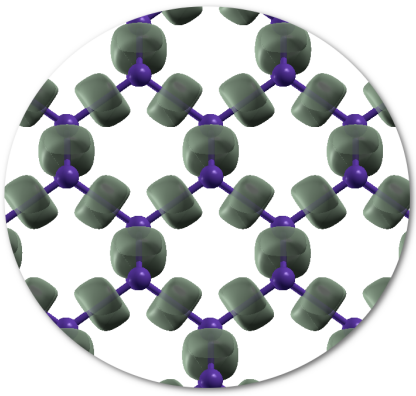
Elastic properties

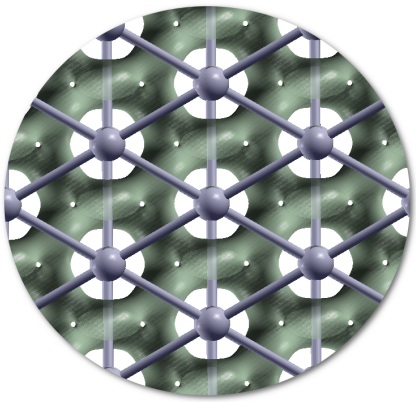


silicon

bcc tungsten

brittle



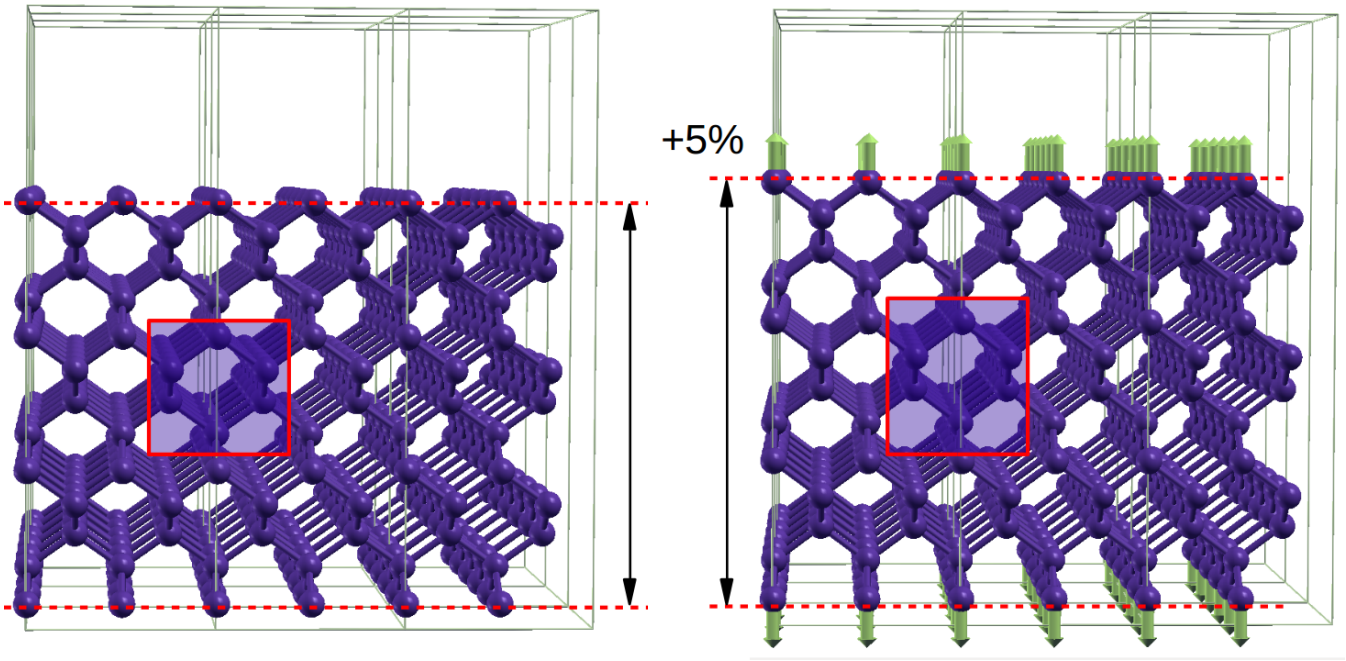


ductile

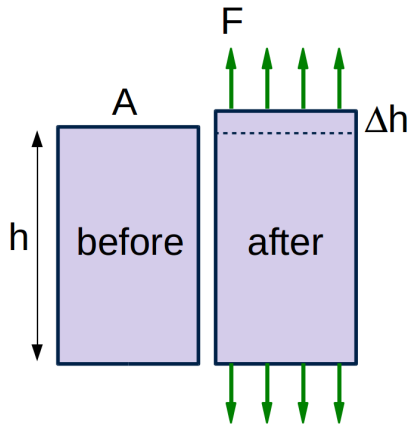
A simple computer experiment on silicon

Equilibrium structure

Stretched slab

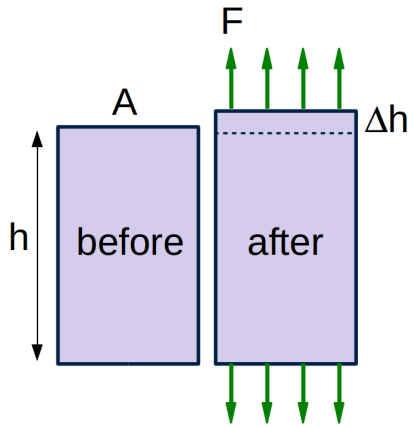


Heuristic approach to elasticity



$$\sigma = \frac{F_{\rm tot}}{A} \qquad \epsilon = \frac{\Delta h}{h} \qquad \sigma = Y\epsilon$$

transverse size	7.635 Å
no. of surface atoms	4
force per atom	0.682 eV/Å
σ	7.5 GPa
ϵ	5%
Y	150 GPa
experiment	166 GPa
deviation	11%



Work-energy theorem

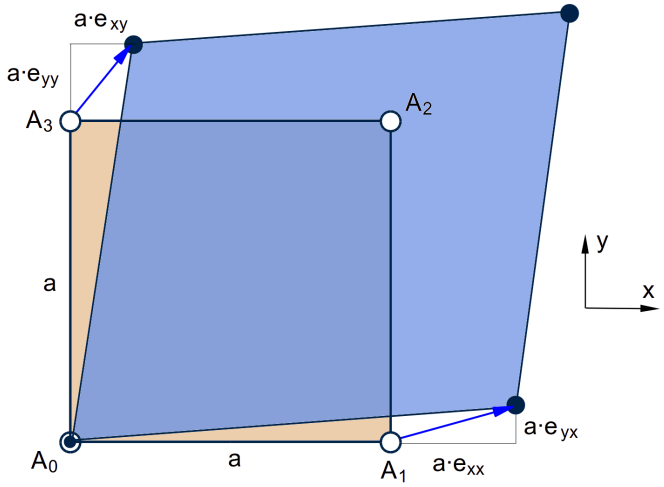
The work of external forces on a system equals the increase of its total potential and kinetic energies

$$\Delta U = \int_h^{h+\Delta h} F(z)\,dz = A\,h \int_h^{h+\Delta h} \frac{F(z)}{A}\,\frac{dz}{h} = \Omega \int_0^{\epsilon} \sigma\,d\epsilon = \Omega \int_0^{\epsilon} C\,\epsilon\,d\epsilon = \frac{1}{2}\Omega\,C\,\epsilon^2$$

$$\sigma = \frac{1}{\Omega}\frac{\partial U}{\partial \epsilon} \qquad C = \frac{1}{\Omega}\frac{\partial^2 U}{\partial \epsilon^2}$$

General transformation of the atomic coordinates: the strain tensor

$$R'_{I\alpha} = \sum_{\beta} (\delta_{\alpha\beta} + \mathrm{e}_{\alpha\beta}) R_{I\beta}$$



$$\sigma_{\alpha\beta} = \frac{1}{\Omega} \frac{\partial U}{\partial \epsilon_{\alpha\beta}} \qquad \frac{\Delta U}{\Omega} = \frac{1}{2} \epsilon_{\alpha\beta} \epsilon_{\gamma\delta}$$

Voigt notation

$$\epsilon_{\alpha\beta} = \begin{pmatrix} \epsilon_{xx} & \epsilon_{xy} & \epsilon_{xz} \\ \epsilon_{xy} & \epsilon_{yy} & \epsilon_{yz} \\ \epsilon_{xz} & \epsilon_{yz} & \epsilon_{zz} \end{pmatrix} \longrightarrow \epsilon_i = \begin{pmatrix} \epsilon_1 & \epsilon_6 & \epsilon_5 \\ & \epsilon_2 & \epsilon_4 \\ & & \epsilon_3 \end{pmatrix}$$

Engineering strain

$$\frac{\Delta U}{\Omega} = \frac{1}{2} C_{ij} u_i u_j \qquad \text{with} \qquad u_i = \begin{cases} \epsilon_i & \text{if } i = 1,2,3 \\ 2\epsilon_i & \text{if } i = 4,5,6 \end{cases}$$

$6^2 = 36$ constants

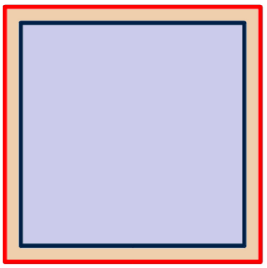
(21 independent)

Example: Cubic system

$$C_{ij} = \begin{pmatrix} C_{11} & C_{12} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{12} & 0 & 0 & 0 \\ C_{12} & C_{12} & C_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{44} \end{pmatrix}$$

Practical calculations

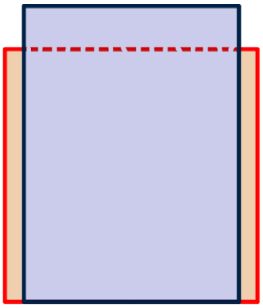
Isotropic deformation



$$u_1 = u_2 = u_3 = \eta$$
$$u_4 = u_5 = u_6 = 0$$

$$\frac{\Delta U}{\Omega} = \frac{3}{2}(C_{11} + 2\,C_{12})\,\eta^2$$

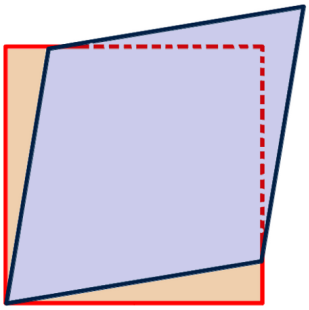
Tetragonal deformation



$$u_1 = u_2 = -\eta$$
$$u_3 = 2\eta$$
$$u_4 = u_5 = u_6 = 0$$

$$\frac{\Delta U}{\Omega} = 3(C_{11} - C_{12})\,\eta^2$$

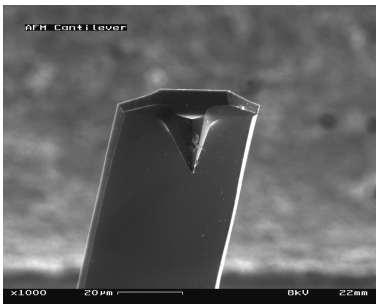
Trigonal deformation



$$u_1 = u_2 = u_3 = 0$$
$$u_4 = u_5 = 0$$
$$u_6 = \eta$$

$$\frac{\Delta U}{\Omega} = \frac{1}{2}C_{44}\,\eta^2$$

Example: Silicon and Tungsten



en.wikipedia.org/wiki/Cantilever

Elastic constants of silicon in GPa

	DFT/LDA	Experiment	Rel. Error
C_{11}	161	165.6	3%
C_{12}	62	63.9	3%
C_{44}	78	79.5	2%

en.wikipedia.org/wiki/Tungsten

Bulk modulus of tungsten in GPa

$$B = \Omega \frac{\partial^2 U}{\partial \Omega^2} = \frac{1}{3}(C_{11} + 2\,C_{12})$$

	DFT/LDA	Experiment	Rel. Error
B	328	314	4%

General remarks

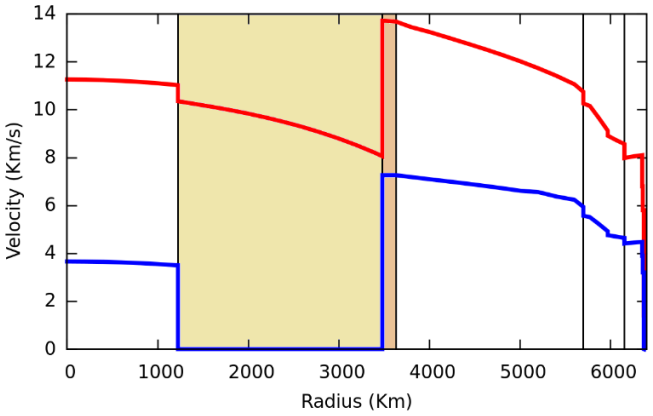
- The stress tensor can also be calculated in DFT w/o considering explicit distortions: stress theorem

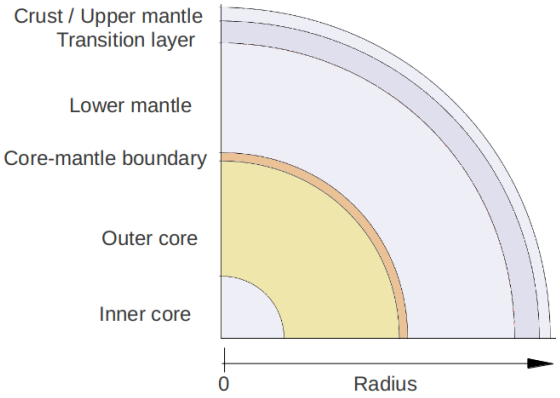
  Nielsen and Martin, Phys. Rev. Lett. 50, 697 (1983)

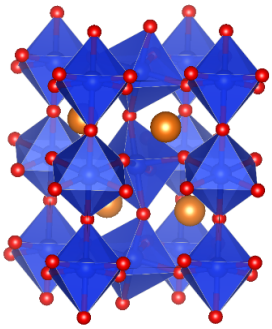
- It is possible to perform DFT calculations for an abritrary external load, e.g. to study pressure dependence

  Parrinello and Rahman, J. Appl. Phys. 52, 7182 (1981)

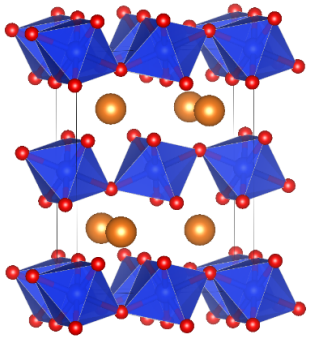
Example: Post-perovskite $MgSiO_3$



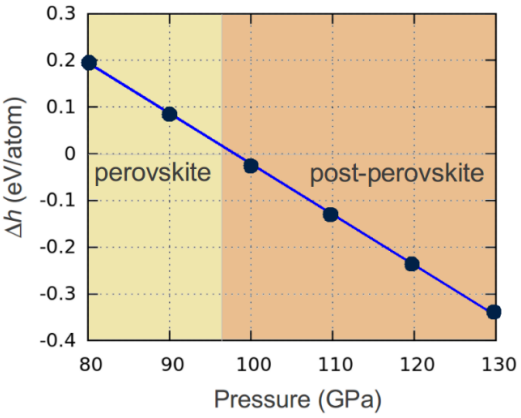




perovskite



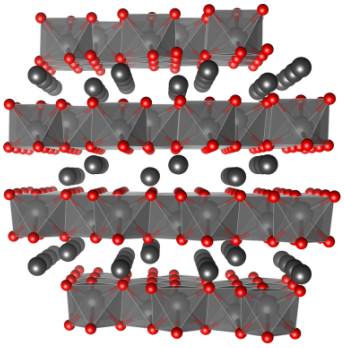
post-perovskite



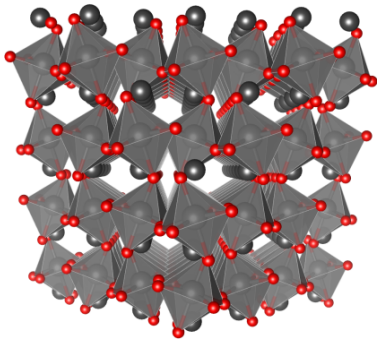
Iitaka et al, Nature 430, 442 (2004)

Example: Perovskite phase of Fe_2O_3

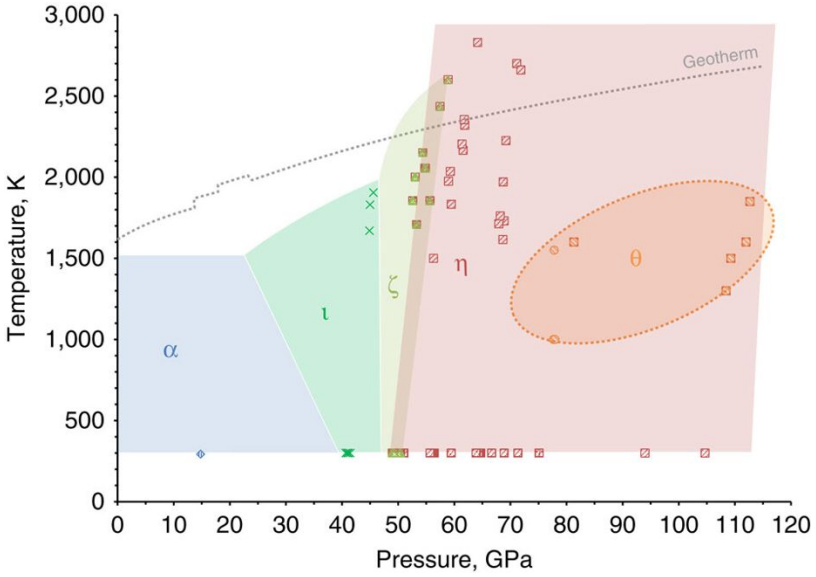
corundum



140 GPa (DFT)



perovskite



Experimental figure (right) from: Bykova et al, Nat Commun 8, 10661 (2016)
DFT calculation (left) from: Filip & FG, Proc Natl Acad Sci USA 115, 5397 (2018)

How many DFT total energy calculations are needed to calculate all the elastic constants of GaAs?

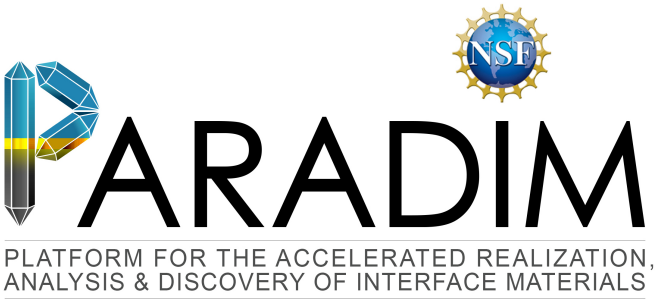
A 3

B 4

C 7

D 21

E 81



An Introduction to Density Functional Theory for Experimentalists

Feliciano Giustino

University of Oxford & Cornell University

Ithaca, 8-14 July 2018

Lecture 4.1

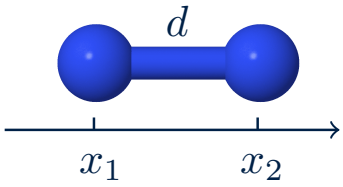
Phonons in DFT

Harmonic approximation

From Lecture 3.1: Newton's equation for classical ions

$$M_I \frac{d^2\mathbf{R}_I}{dt^2} = -\frac{\partial U}{\partial \mathbf{R}_I} \longleftarrow \text{Potential energy surface from DFT}$$

N_2 molecule



$$M_{\rm N} \frac{d^2 x_1}{dt^2} = -\frac{\partial U}{\partial x_1}$$

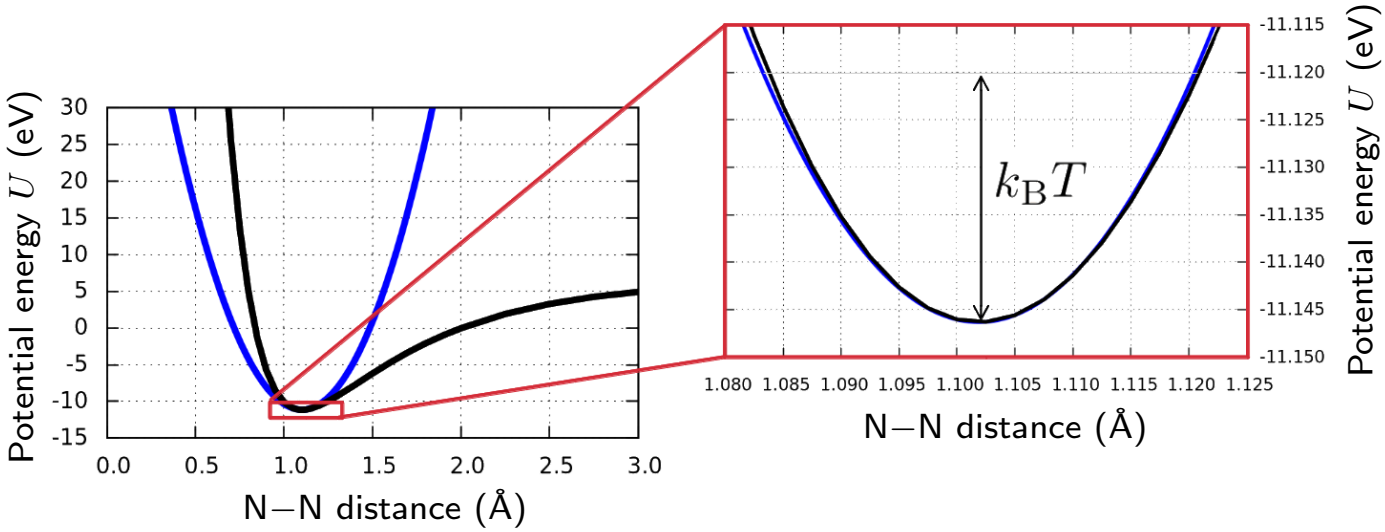
$$M_{\rm N} \frac{d^2 x_2}{dt^2} = -\frac{\partial U}{\partial x_2}$$

$$M_{\rm N}\, \ddot{d} = M_{\rm N} \frac{d^2}{dt^2}(x_2 - x_1) = -\left(\frac{\partial U}{\partial x_2} - \frac{\partial U}{\partial x_1}\right) = -2\frac{\partial U}{\partial d}$$

$$\frac{\partial U}{\partial x_2} = \frac{\partial U}{\partial \overline{x}} \frac{\partial \overline{x}}{\partial x_2} + \frac{\partial U}{\partial d} \frac{\partial d}{\partial x_2} = \frac{1}{2}\frac{\partial U}{\partial \overline{x}} + \frac{\partial U}{\partial d}$$

Harmonic approximation

$$\frac{M_{\rm N}}{2}\,\ddot{d} = -\frac{\partial U}{\partial d}$$



$$U(d) = U_0 + \frac{1}{2}K(d-d_0)^2 \longrightarrow \frac{\partial U}{\partial d} = K(d-d_0)$$

Harmonic approximation (Hooke's law)

Stretching frequency of N_2

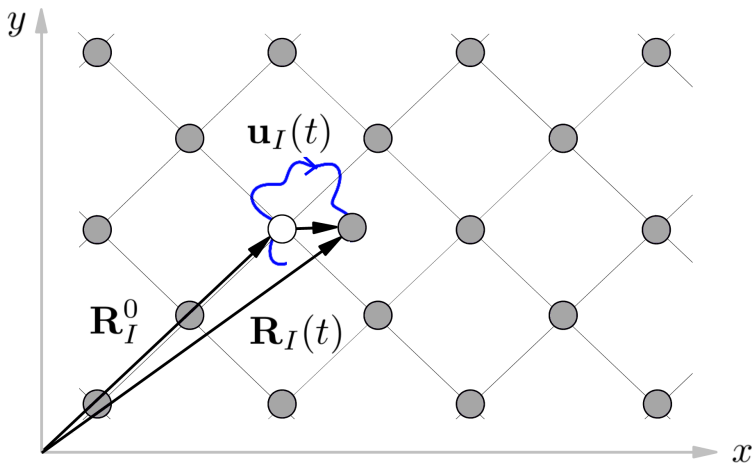
$$\ddot{d} = -\frac{2K}{M_{\rm N}}(d - d_0)$$

$$\omega_{\rm N_2} = \sqrt{\frac{2K}{M_{\rm N}}}$$

	DFT/LDA	experiment	deviation
$\hbar\omega_{\rm N_2}$ (meV)	288.9	300.4	4%

	harmonic	anharmonic	deviation
$\hbar\omega_{\rm N_2}$ (meV)	288.9	304.2	5%

Vibrational eigenmodes and eigenvalues



$$\mathbf{R}_I(t) = \mathbf{R}_I^0 + \mathbf{u}_I(t)$$

$$M_I \ddot{\mathbf{u}}_I = -\frac{\partial U}{\partial \mathbf{u}_I}$$

Taylor expansion of potential energy surface for small displacements

$$U = U_0 + \sum_{I\alpha} u_{I\alpha} \frac{\partial U}{\partial R_{I\alpha}} + \frac{1}{2} \sum_{I\alpha,J\beta} \frac{\partial^2 U}{\partial R_{I\alpha} \partial R_{J\beta}} u_{I\alpha} u_{J\beta} + \mathcal{O}(u^3)$$

$$K_{I\alpha,J\beta}$$

matrix of force constants

Vibrational eigenmodes and eigenvalues

Equation of motion in the harmonic approximation

$$M_I \, \ddot{u}_{I\alpha} = -\sum_{J\beta} K_{I\alpha,J\beta} \; u_{J\beta}$$

Try with $u_{I\alpha}(t) = u^0_{I\alpha} \exp(i\omega t)$ and rearrange

$$\sum_{J\beta} K_{I\alpha,J\beta} \; u^0_{J\beta} = M_I \, \omega^2 \, u^0_{I\alpha}$$

Get rid of masses

$$\sum_{J\beta} \underbrace{\frac{K_{I\alpha,J\beta}}{\sqrt{M_I M_J}}}_{D_{I\alpha,J\beta}} \underbrace{\sqrt{M_J} u^0_{J\beta}}_{e_{J\beta}} = \omega^2 \underbrace{\sqrt{M_I} \, u^0_{I\alpha}}_{e_{I\alpha}}$$

dynamical matrix — mass-scaled eigenmode

Vibrational eigenmodes and eigenvalues

Dynamical matrix

$$D_{I\alpha,J\beta} = \frac{1}{\sqrt{M_I M_J}} \frac{\partial^2 U}{\partial R_{I\alpha} \partial R_{J\beta}}\bigg|_{\text{Equil.}}$$

$$\mathbf{D}\,\mathbf{e} = \omega^2\,\mathbf{e}$$

$3M \times 3M$
(M atoms)

Standard matrix eigenvalue problem

Calculations of vibrational frequencies

1) Frozen-phonon method

$$\frac{\partial^2 U}{\partial R_{I\alpha}^2} \simeq \frac{U(R_{I\alpha}^0 + u) - 2U_0 + U(R_{I\alpha}^0 - u)}{u^2}$$

or alternatively using the forces

$$\frac{\partial^2 U}{\partial R_{I\alpha}^2} \simeq -\frac{F_{I\alpha}(R_{I\alpha}^0 + u) - F_{I\alpha}(R_{I\alpha}^0 - u)}{2u}$$

2) Density-functional perturbation theory (DFPT)

The method of choice for crystals

see Baroni et al, Rev. Mod. Phys. 73, 515 (2001)

Calculations of vibrational frequencies

Example: water molecule

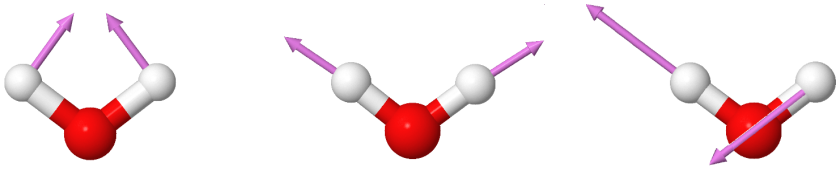
$$\hbar^2\mathbf{D} = \begin{pmatrix} 82 & 82 & -0 & -164 & -164 & -176 & -164 & -164 & 176 \\ & 82 & -0 & -164 & -164 & -176 & -164 & -164 & 176 \\ & & 103 & -136 & -136 & -206 & 136 & 136 & -206 \\ & & & 710 & 711 & 625 & -55 & -55 & -81 \\ & & & & 710 & 625 & -55 & -55 & -81 \\ & & & & & 791 & 81 & 81 & 31 \\ & & & & & & 710 & 711 & -625 \\ & & & & & & & 710 & -625 \\ & & & & & & & & 791 \end{pmatrix} \cdot 100\,\mathrm{meV}^2$$

9 eigenvalues

- 3 rigid-body translations
- 2 rigid-body librations
- 1 rigid-body spinning
- 3 internal vibrations

Calculations of vibrational frequencies

Example: water molecule

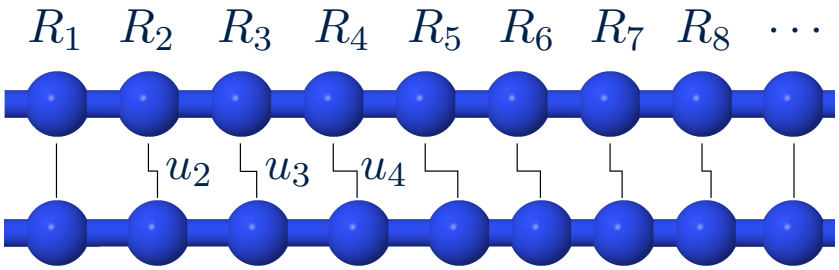


bending | symmetric stretching | asymmetric stretching

	bending	symmetric stretching	asymmetric stretching
DFT frequency (meV)	189	453	466
Expt. frequency (meV)	198	458	473
deviation	4.8%	1.1%	1.5%

Vibrations in crystals

Simplest model: 1D chain of atoms moving along 1 direction



$$M\ddot{u}_I = -\sum_J K_{IJ}\, u_J$$

Sound waves $\quad \dfrac{1}{v_s^2}\dfrac{\partial^2 p}{\partial t^2} = \dfrac{\partial^2 p}{\partial x^2} \longrightarrow p(x,t) = p_0\, e^{i(qx-\omega t)}$

By analogy we can try $\quad u_I(t) = u_0\, \mathrm{e}^{i(qR_I-\omega t)}$

Vibrations in crystals

$$M\,\omega^2\,\mathrm{e}^{iqR_I} = \sum_J K_{IJ}\,\mathrm{e}^{iqR_J}$$

$$\omega^2 = \frac{1}{M}\sum_J K_{IJ}\,\mathrm{e}^{iq(R_J-R_I)}$$

In crystals the force constants are translationally invariant

$$\omega^2(q) = \frac{1}{M}\sum_J K_{0J}\,\mathrm{e}^{iqR_J}$$

A Fourier transform of the force constants yields the eigenfrequencies

Dynamical matrix for crystals

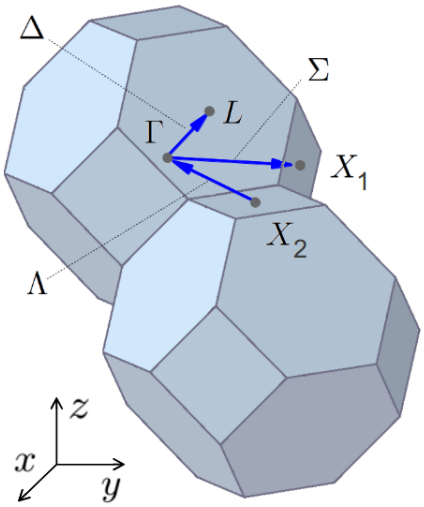
$$D_{I\alpha,J\beta}(\mathbf{q}) = \frac{1}{\sqrt{M_I M_J}}\sum_{\mathbf{R}} e^{i\mathbf{q}\cdot\mathbf{R}}\mathrm{e}^{i\mathbf{q}\cdot(\tau_J-\tau_I)}\,K_{0I\alpha,\mathbf{R}J\beta}$$

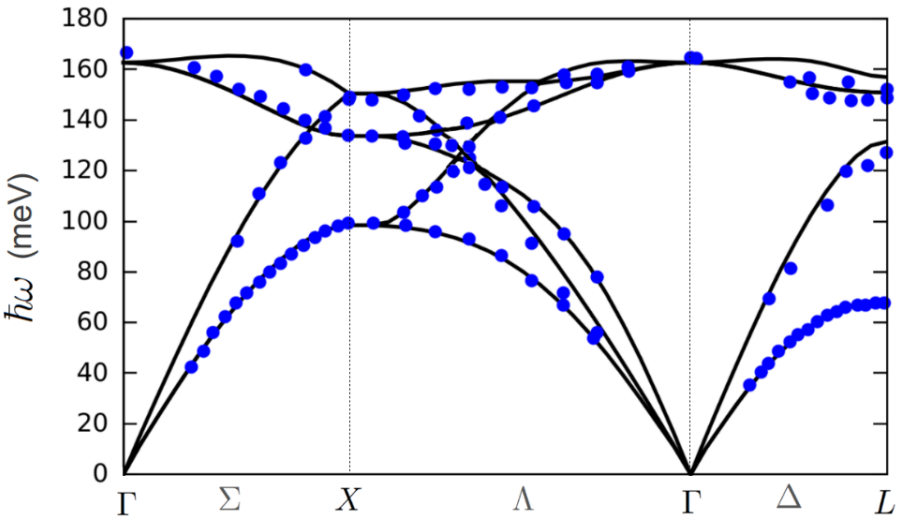
$\mathbf{R}$ reciprocal lattice vector
τ_I position of atom I in the primitive unit cell

Phonon dispersion relations

Example: diamond

en.wikipedia.org/wiki/Oppenheimer_Diamond



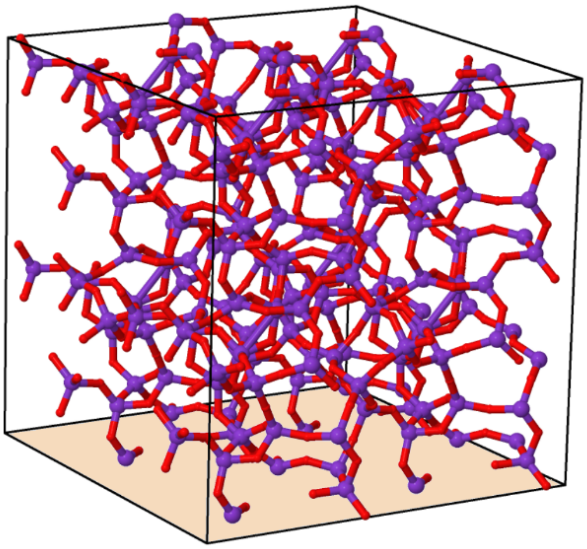


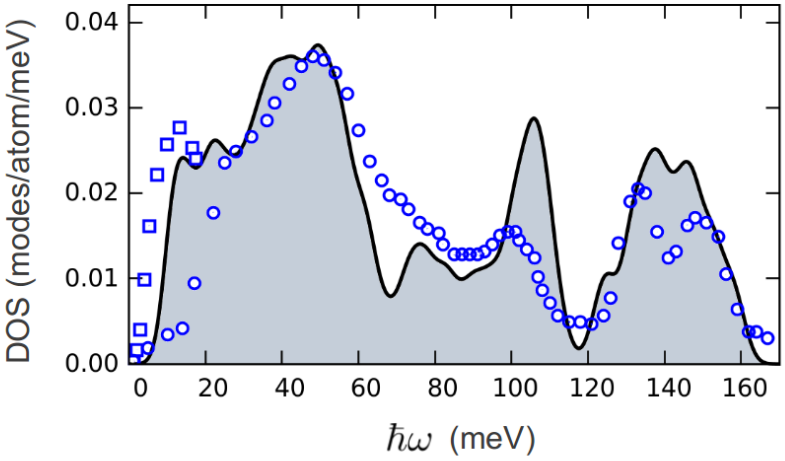
— DFT/LDA

● Inelastic neutron scattering

Phonon density of states

Example: amorphous SiO_2





○ DFT/LDA
— Inelastic neutron scattering

What is the size of the dynamical matrix of (monolayer) graphene?

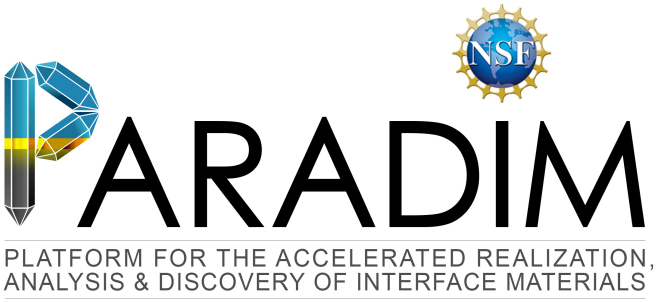
A 2×2

B 3×3

C 4×4

D 6×6

E I don't know what the dynamical matrix is



An Introduction to Density Functional Theory for Experimentalists

Feliciano Giustino

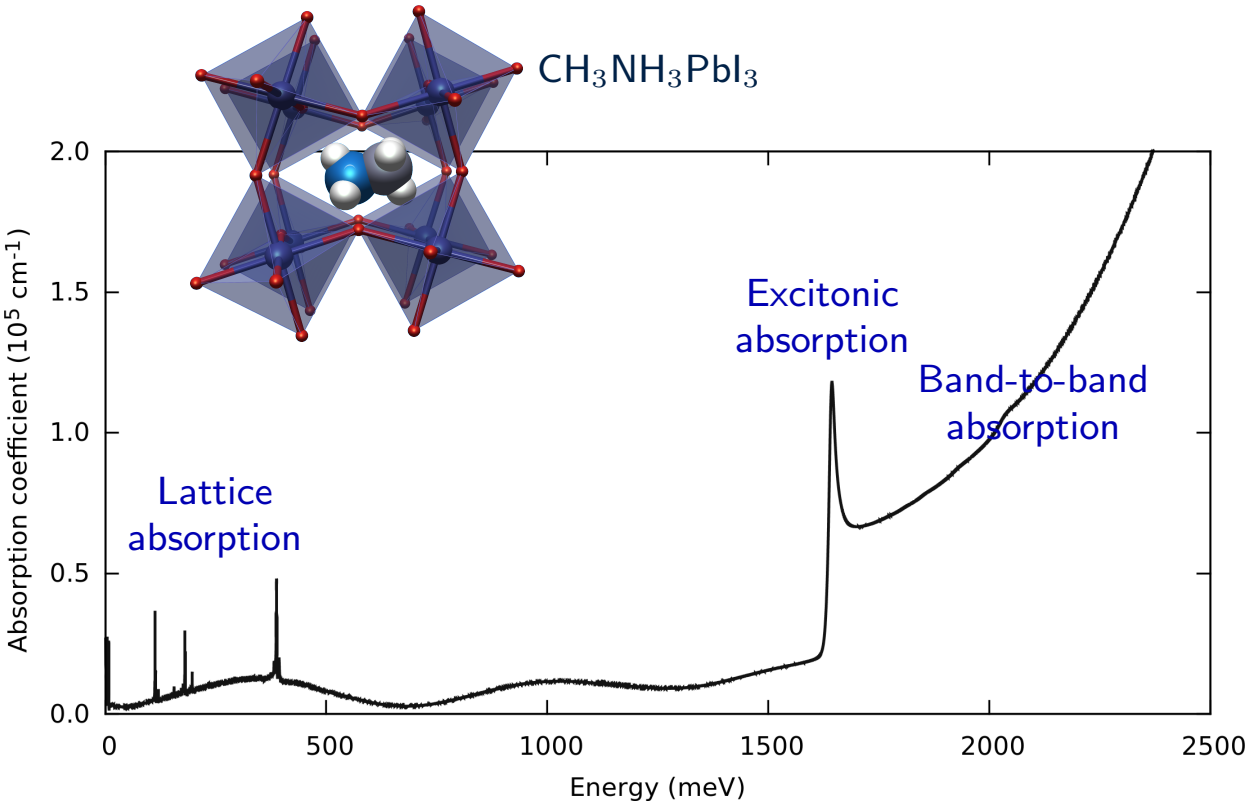
University of Oxford & Cornell University

Ithaca, 8-14 July 2018

Lecture 4.2

IR spectra & dielectric constants

Infrared absorption spectrum



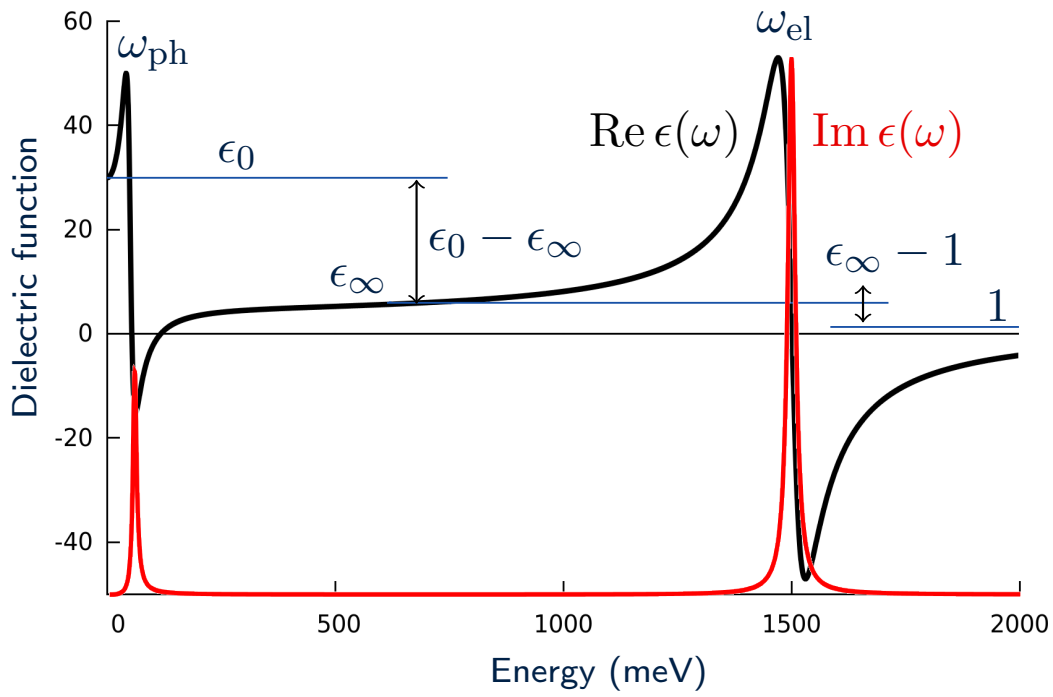
Experiment by R. Milot & M. Johnston, Oxford Physics

Perez et al, J. Phys. Chem. C 119, 25703 (2015)

Dielectric function

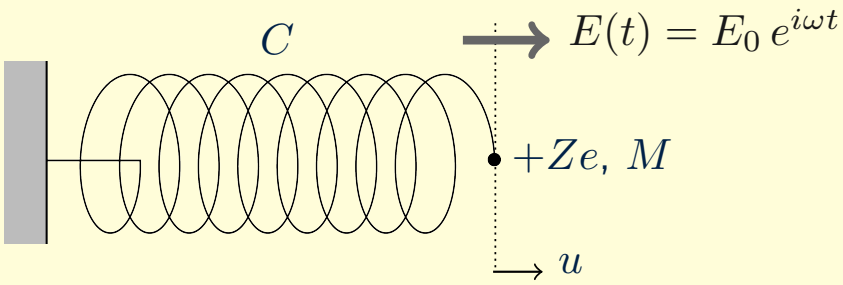
General structure of the frequency-dependent dielectric function

$$\epsilon(\omega) = 1 + \frac{\epsilon_\infty - 1}{1 - (\omega/\omega_{\rm el})^2} + \frac{\epsilon_0 - \epsilon_\infty}{1 - (\omega/\omega_{\rm ph})^2}$$



Simplified model

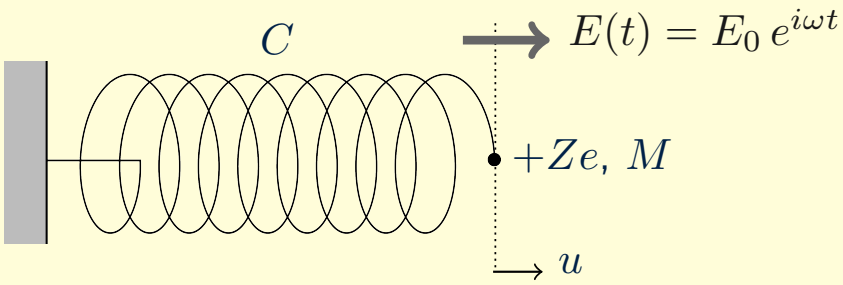
Classical point charge attached to spring



Equation of motion	$M\ddot{u}(t) = -C\,u(t) + Ze\,E(t)$
Set	$u(t) = u_0\, e^{i\omega t}$
We obtain	$-M\omega^2 u_0 = -C\,u_0 + Ze\,E_0$
w/o electric field	$-M\omega_0^2 u_0 = -C\,u_0$
Difference	$M(\omega_0^2 - \omega^2)u_0 = Ze\,E_0$
Induced displacement	$u_0 = \dfrac{e}{M}\dfrac{Z}{\omega_0^2 - \omega^2}E_0$

Simplified model

Classical point charge attached to spring



Induced displacement $$u_0 = \frac{e}{M}\frac{Z}{\omega_0^2 - \omega^2}E_0$$

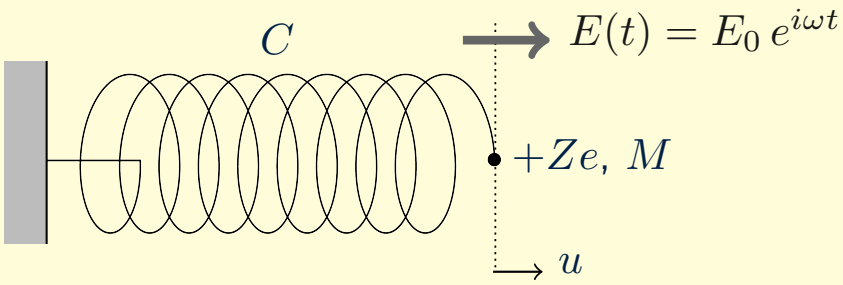
Ionic polarization $$P(t) = \frac{1}{\Omega}Ze\,u(t) = P_0\, e^{i\omega t} \rightarrow P_0 = \frac{1}{\Omega}Ze\,u_0$$

Total polarization $$P_0 + P_{\rm el} = \epsilon_0\epsilon_1(\omega)E_0, \qquad P_{\rm el} = \epsilon_0\epsilon^\infty E_0$$

Dielectric function $$\epsilon_1(\omega) = \epsilon^\infty + \frac{e^2}{4\pi\epsilon_0}\frac{4\pi}{\Omega}\frac{1}{M}\frac{Z^2}{\omega_0^2 - \omega^2}$$

Simplified model

Classical point charge attached to spring



Dielectric function

$$\epsilon_1(\omega) = \epsilon^\infty + \frac{e^2}{4\pi\epsilon_0}\frac{4\pi}{\Omega}\frac{1}{M}\frac{Z^2}{\omega_0^2 - \omega^2}$$

Static dielectric constant

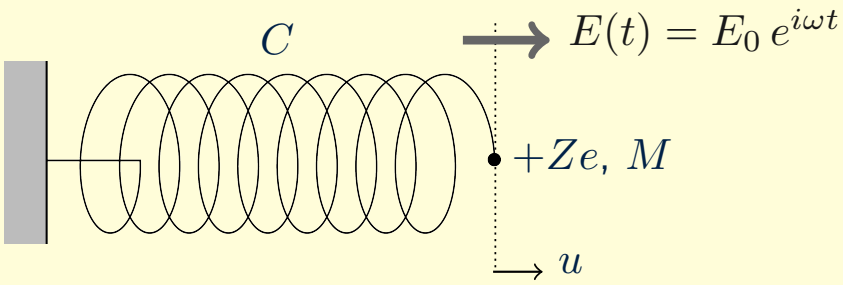
$$\epsilon_0 = \epsilon^\infty + \frac{e^2}{4\pi\epsilon_0}\frac{4\pi}{\Omega}\frac{1}{M}\frac{Z^2}{\omega_0^2}$$

IR absorption intensity
(Sec. 10.2.1 of Book)

$$I(\omega) \propto \omega\epsilon_2(\omega) \propto Z^2\delta(\omega - \omega_0)$$

Simplified model

Classical point charge attached to spring



		Born effective charge
Polarization	$P_0 = \frac{1}{\Omega} Ze\, u_0$	$Z = \frac{\Omega}{e} \frac{\partial P}{\partial u}$
Electric force	$F_0 = Ze\, E_0$	$Z = \frac{1}{e} \frac{\partial F}{\partial E}$

Low-frequency dielectric constant tensor ($\mathbf{q} \to 0$)

$$\epsilon_{1,\alpha\beta}(\omega) = \epsilon^{\infty}_{1,\alpha\beta} + \frac{e^2}{4\pi\varepsilon_0}\frac{4\pi}{\Omega}\frac{1}{M_0}\sum_n \frac{Z_{\alpha n} Z_{\beta n}}{\omega_n^2 - \omega^2}$$

High-frequency dielectric constant

Phonon frequencies at $\mathbf{q} = 0$

DFT calculations

Low-frequency dielectric constant tensor ($\mathbf{q} \to 0$)

$$\epsilon_{1,\alpha\beta}(\omega) = \epsilon^{\infty}_{1,\alpha\beta} + \frac{e^2}{4\pi\varepsilon_0}\frac{4\pi}{\Omega}\frac{1}{M_0}\sum_n \frac{Z_{\alpha n} Z_{\beta n}}{\omega_n^2 - \omega^2}$$

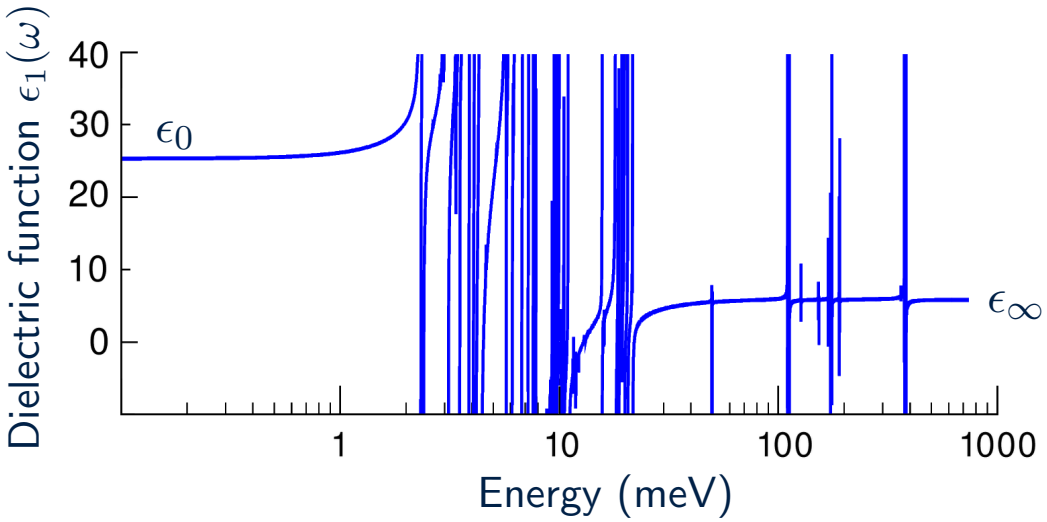
Mode effective charge vector $Z_{\alpha n} = \sum_{J\beta} Z^*_{J,\alpha\beta} \sqrt{\frac{M_0}{M_J}}\, e_{J\beta,n}$

Born effective charge tensor $Z^*_{J,\alpha\beta} = \frac{\Omega}{e}\frac{\partial P_\alpha}{\partial R_{J\beta}}$

Phonon eigenmodes at $\mathbf{q} = 0$

DFT calculations

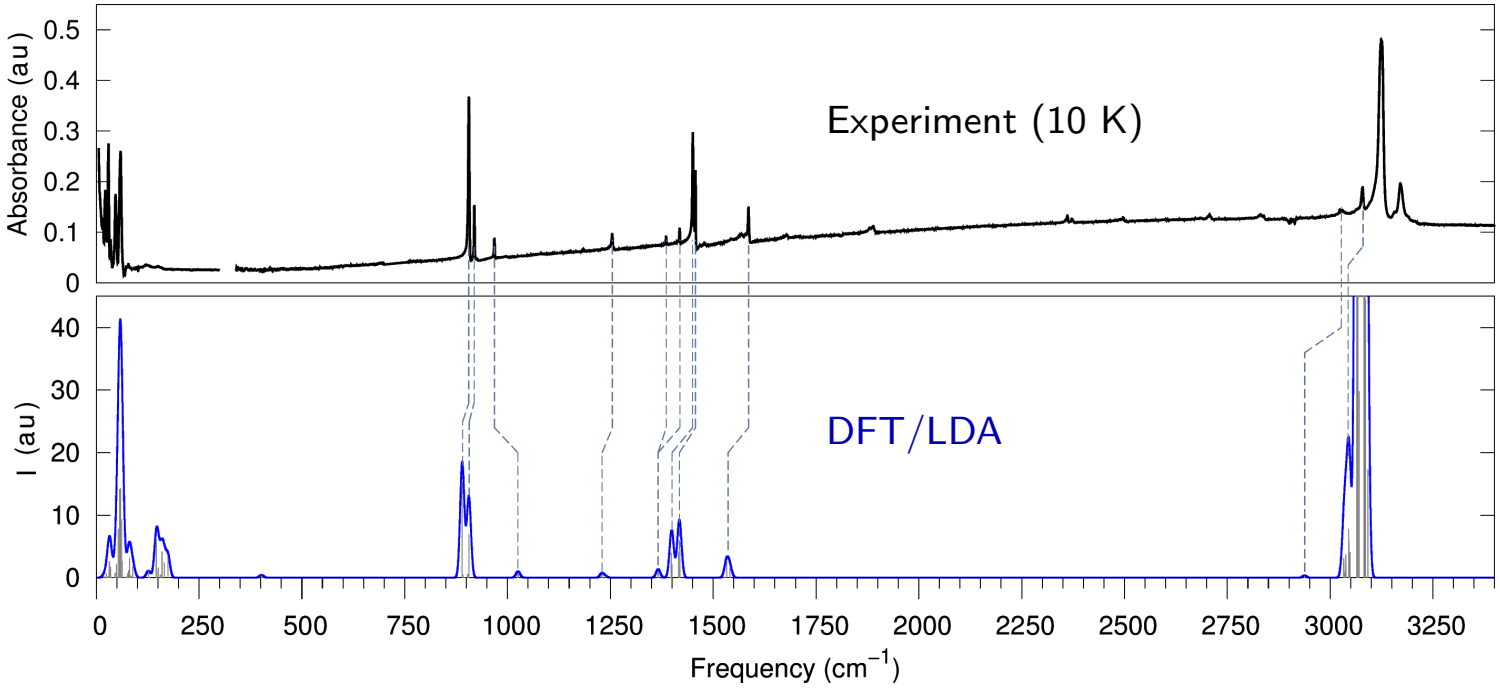
Example: Dielectric function of the hybrid perovskite $CH_3NH_3PbI_3$



	DFT/LDA	Experiment	deviation
ϵ_∞	5.9	6.5	10%
ϵ_0	25.3	30.5	20%

Perez et al, J. Phys. Chem. C 119, 25703 (2015)

Example: IR absorption of the hybrid perovskite $CH_3NH_3PbI_3$



Perez et al, J. Phys. Chem. C 119, 25703 (2015)

Born effective charges

$$Z^*_{I,\alpha\beta} = \begin{cases} \dfrac{\Omega}{e}\dfrac{\partial P_\alpha}{\partial R_{I\beta}} \\ \dfrac{1}{e}\dfrac{\partial F_{I\alpha}}{\partial E_\beta} \end{cases}$$

How the electronic polarization changes when we displace one atom

How the force on one atom changes when we apply an electric field

$CH_3NH_3PbI_3$

	Nominal charge	Born charge tensor			Isotropic Born charge
Pb	$+2$	4.52	-0.14	-0.77	$+4.42$
		-0.19	4.42	0.60	
		0.66	-0.27	4.31	
I	-1	-0.58	0.00	0.08	-1.88
		0.01	-4.15	0.01	
		0.16	0.01	-0.92	

Quiz Time

What is the Born effective charge of Pb in fcc lead?

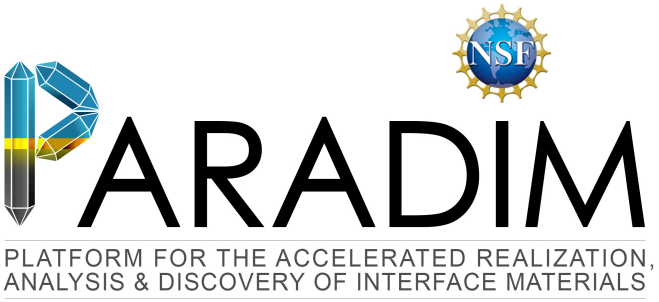
A $+4.42$

B $+4$

C $+1$

D 0

E Time for lunch



An Introduction to Density Functional Theory for Experimentalists

Feliciano Giustino

University of Oxford & Cornell University

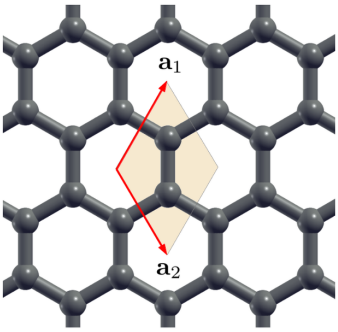
Ithaca, 8-14 July 2018

Lecture 5.1

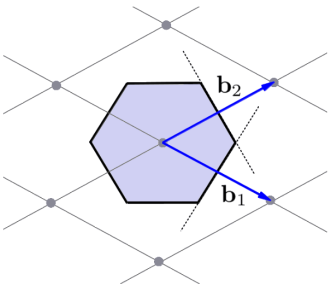
Band structures & optical spectra

Example: graphene

direct lattice



reciprocal lattice

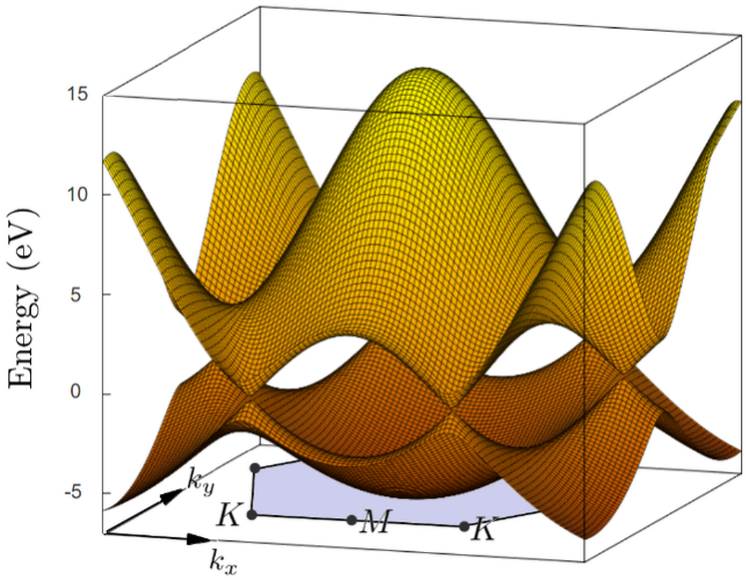


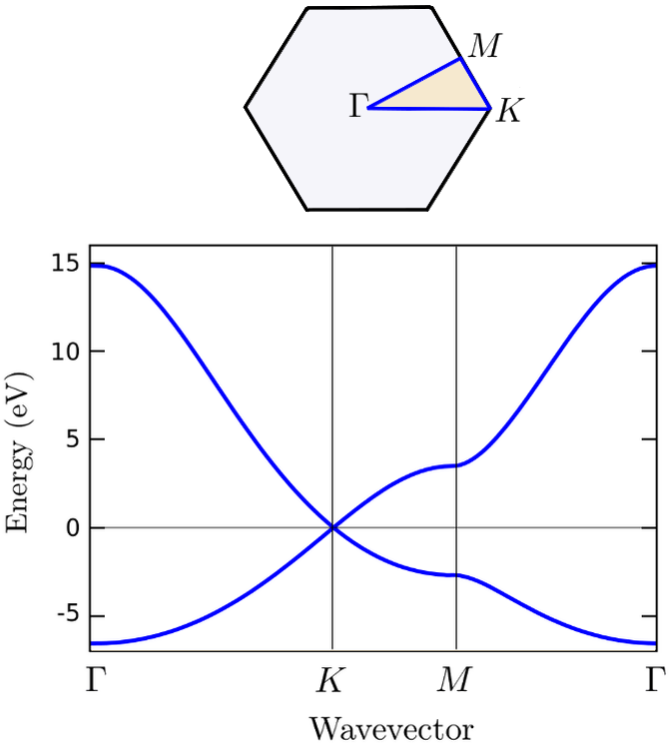
$$\phi_{i\mathbf{k}}(\mathbf{r}) = \mathrm{e}^{i\mathbf{k}\cdot\mathbf{r}} u_{i\mathbf{k}}(\mathbf{r}) \text{ with } u_{i\mathbf{k}}(\mathbf{r}) \text{ periodic}$$

$$-\frac{1}{2}(\nabla+i\mathbf{k})^2 u_{i\mathbf{k}}(\mathbf{r}) + V_{\mathrm{tot}}(\mathbf{r}) u_{i\mathbf{k}}(\mathbf{r}) = \boxed{\varepsilon_{i\mathbf{k}}}\, u_{i\mathbf{k}}(\mathbf{r})$$

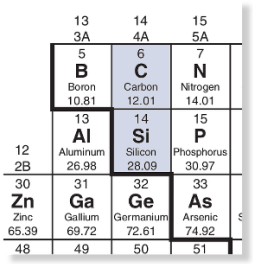
$\mathbf{k}$-dependent
Kohn-Sham eigenvalue

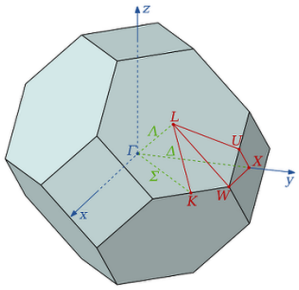
Example: graphene (simplified tight-binding model)

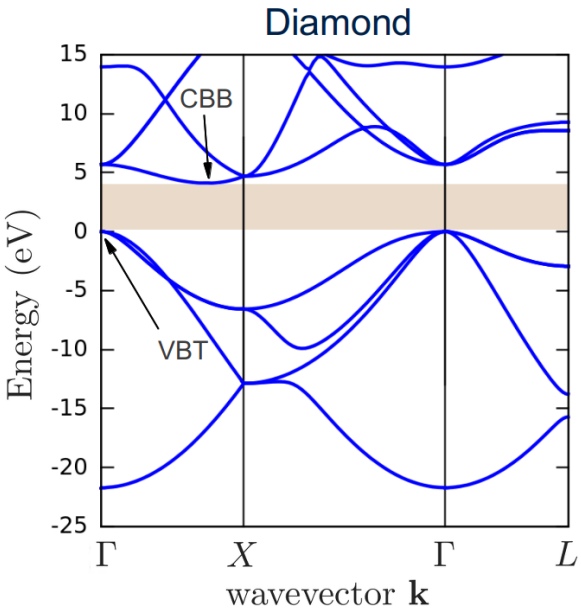


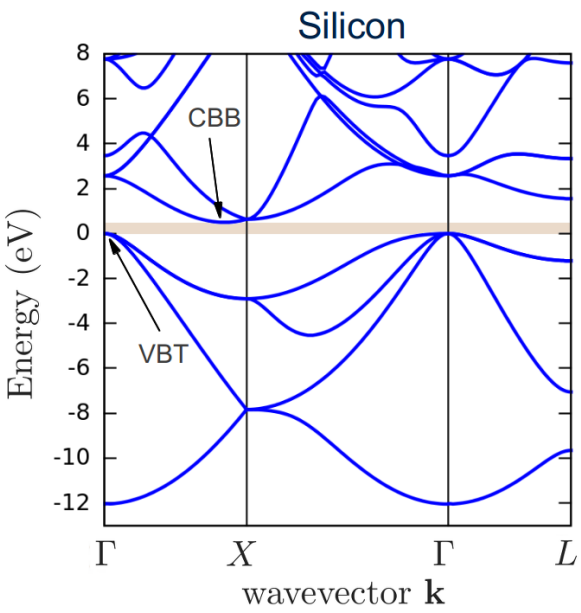


Example: DFT/LDA band structures of common semiconductors

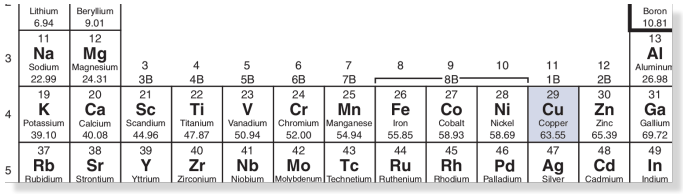


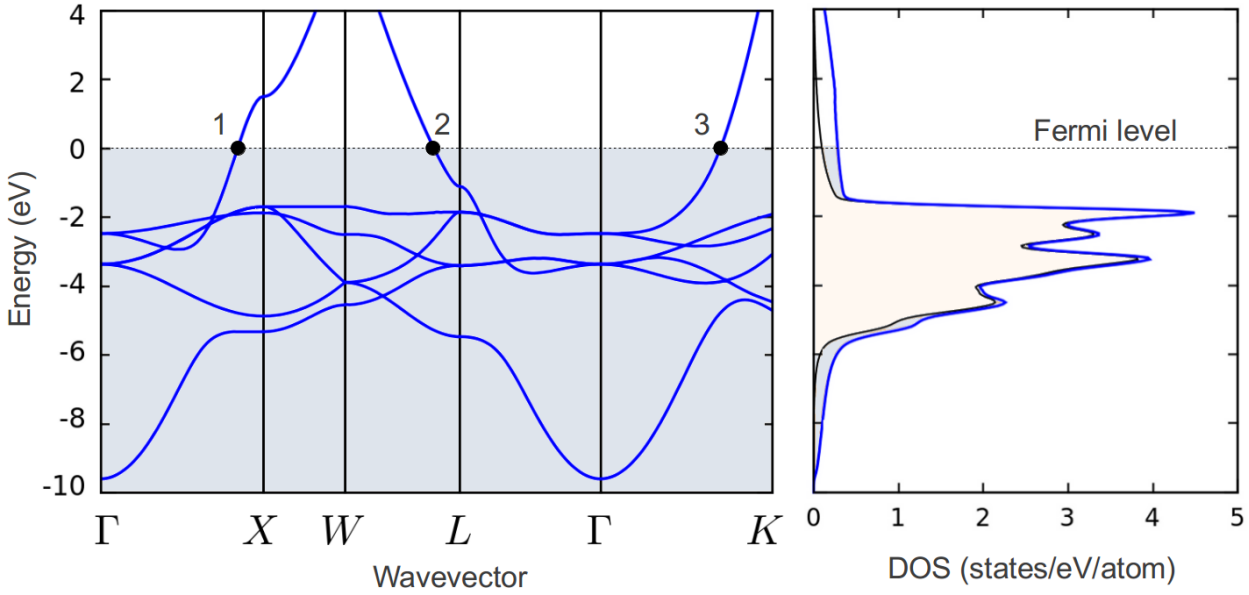




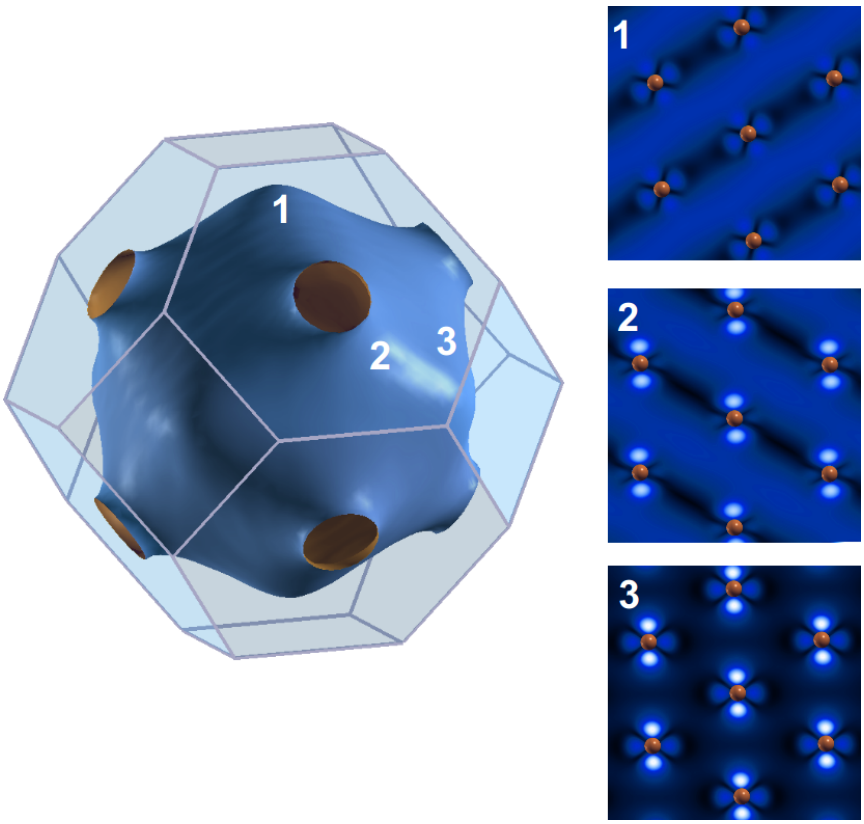


Example: DFT/LDA band structure and Density of States of copper



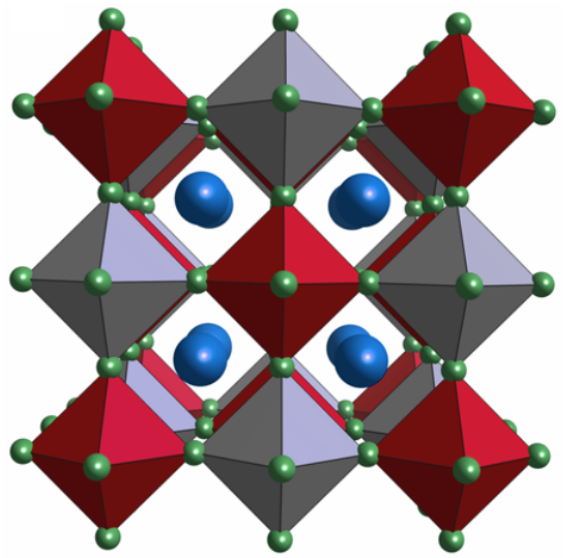


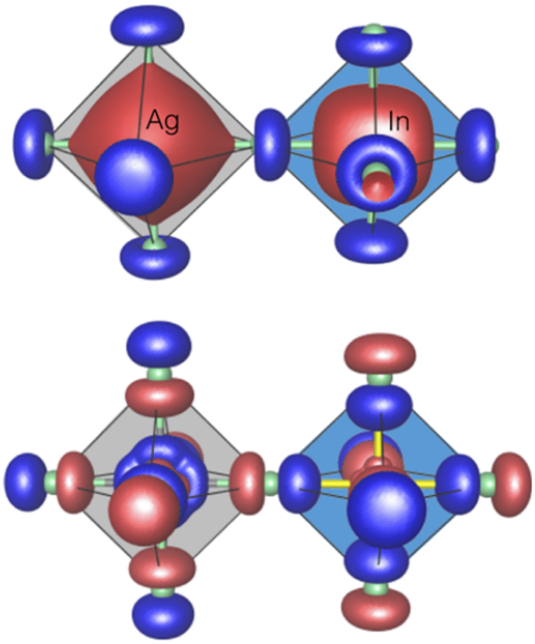
Example: DFT/LDA Fermi surface and wavefunctions of copper



Kohn-Sham wavefunctions

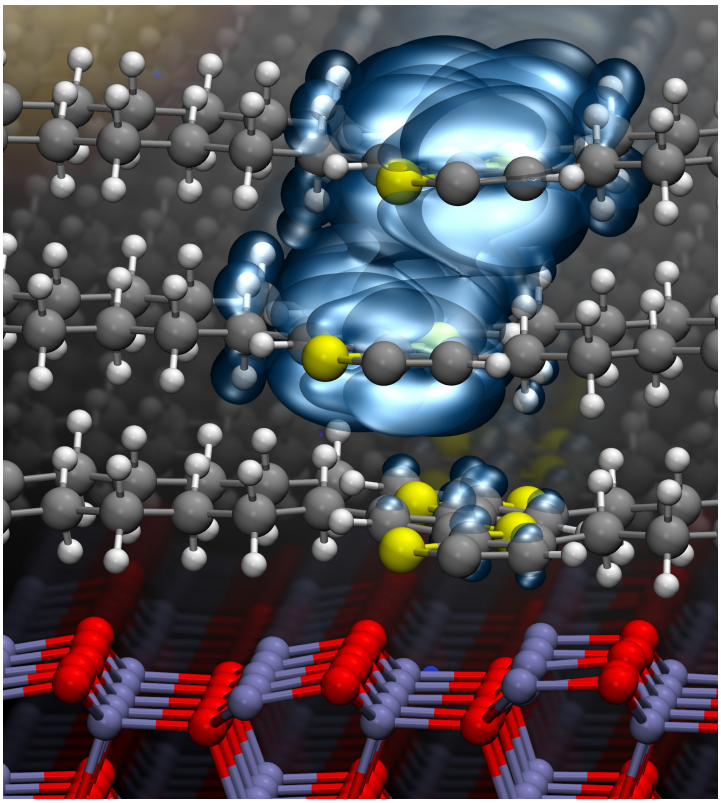
Example: Elpasolite $Cs_2InAgCl_6$





Volonakis et al, J. Phys. Chem. Lett., 8, 772 (2017)

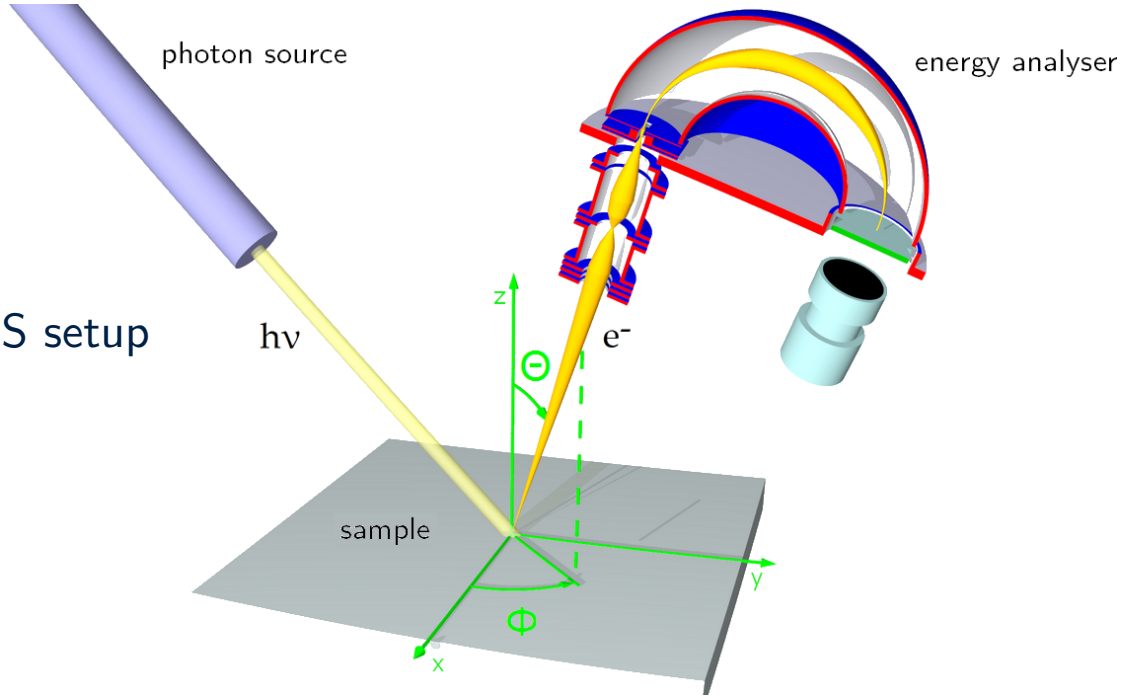
Example: Highest-occupied orbital of P3HT on ZnO



Noori & Giustino, Adv. Func. Mater. 22, 5089 (2012)

Meaning of band structures

- Are band structures real?

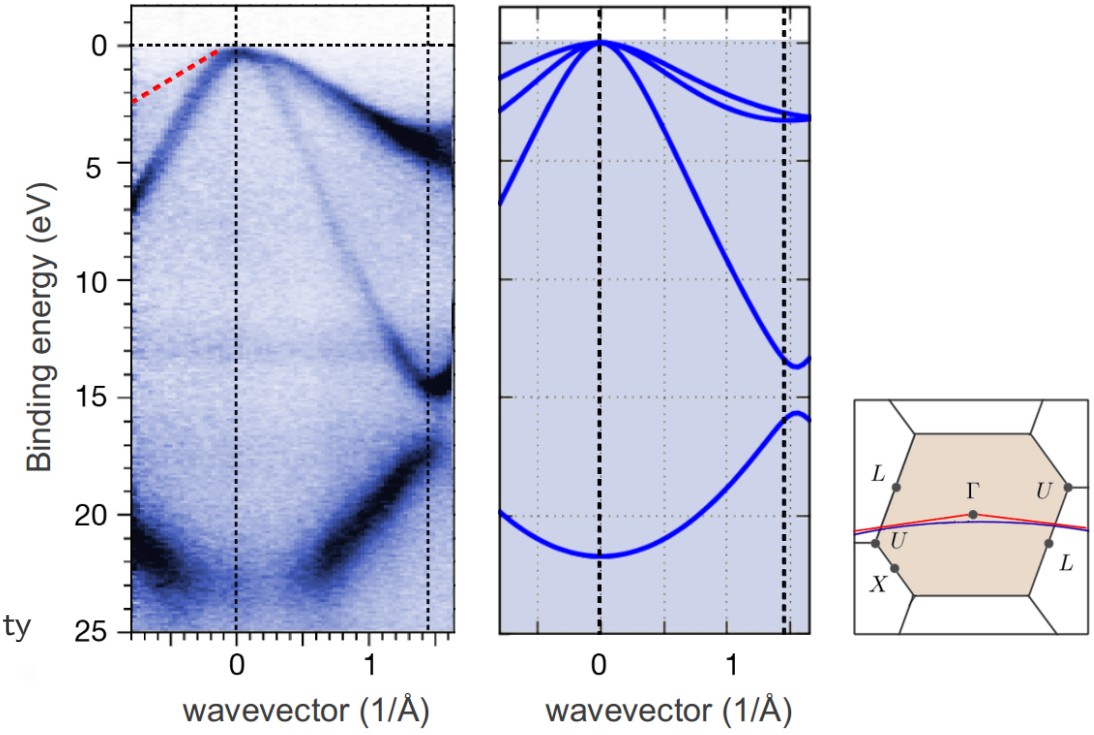


commons.wikimedia.org/wiki/File:ARPESgeneral.png

Meaning of band structures

- Are band structures real?

ARPES spectrum of diamond and DFT/LDA bands



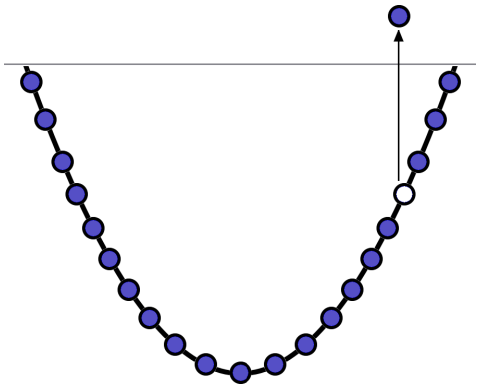
Experiment by
Prof. Yokoya
Okayama University

Meaning of band structures

The DFT total energy can be rewritten as

$$E[n] = \underbrace{\sum_i \frac{d\mathbf{k}}{\Omega_{\rm BZ}} f_{i\mathbf{k}}\,\varepsilon_{i\mathbf{k}}}_{\text{band structure term}} - \underbrace{\left[E_{\rm H} + \int d\mathbf{r}\, V_{xc}(\mathbf{r})n(\mathbf{r}) - E_{xc}\right]}_{\text{double counting term}}$$

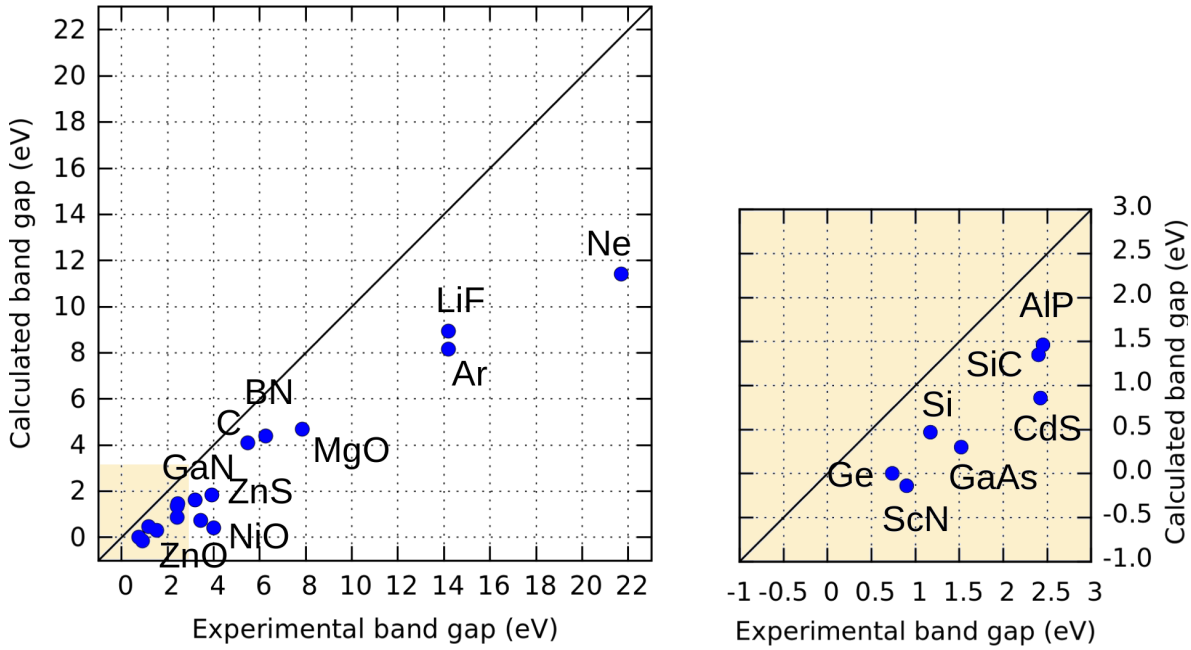
$f_{i\mathbf{k}}$ electron occupation



- If the double-counting term vanished, the eigenvalues would give the change of total energy upon adding or removing one electron
- The KS levels can be thought of as very rough approximations to addition or removal energies

Band gap problem

- DFT/LDA typically underestimates the band gaps of insulators and semiconductors
- Major challenge in materials design



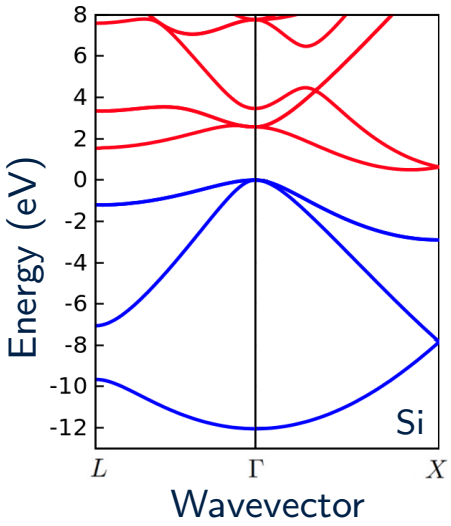
Dielectric function and optical spectra

From band structure & wavefunctions we can calculate UV/Vis spectra

$$\epsilon_2(\omega) = \frac{\pi\, e^2}{\varepsilon_0 m_e^2 \Omega} \frac{1}{\omega^2} \sum_{cv} \int \frac{d\mathbf{k}}{\Omega_{\rm BZ}} |\langle u_{c\mathbf{k}} | p_x | u_{v\mathbf{k}} \rangle|^2 \delta(\varepsilon_{c\mathbf{k}} - \varepsilon_{v\mathbf{k}} - \hbar\omega)$$

Independent-particle approximation

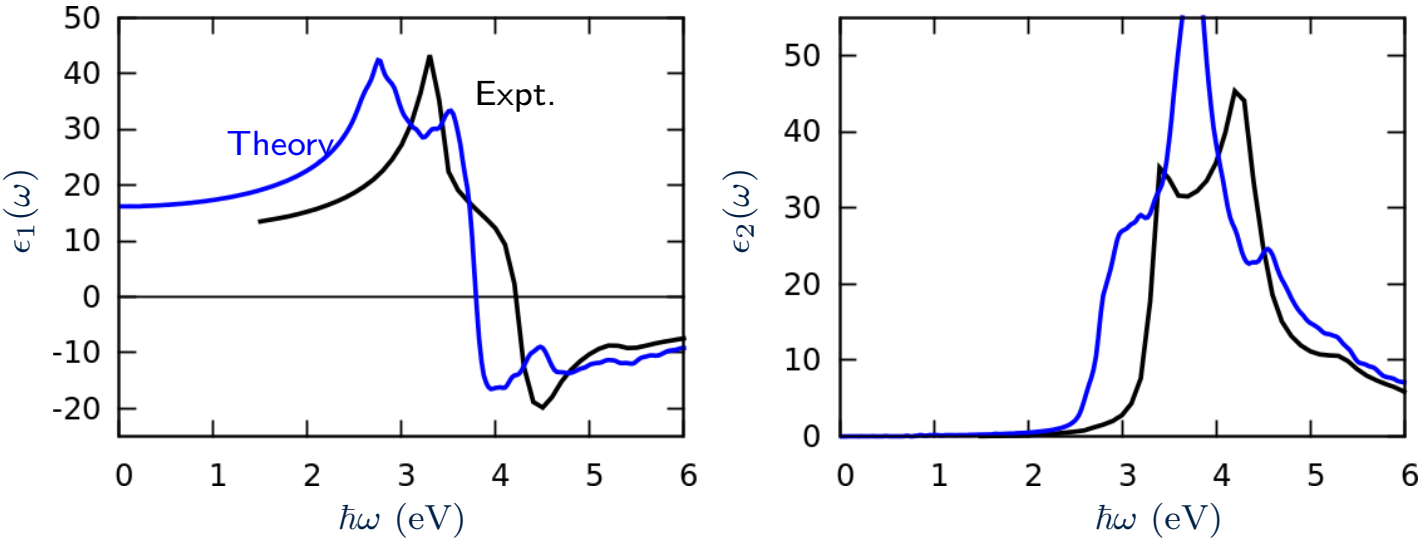
(derivation in Chap. 10 of DFT book)



- Transitions from occupied to empty states
- Energy conservation
- Momentum conservation
- Oscillator strength given by electric dipole matrix element $\langle u_{c\mathbf{k}} | x | u_{v\mathbf{k}} \rangle$

Dielectric function and optical spectra

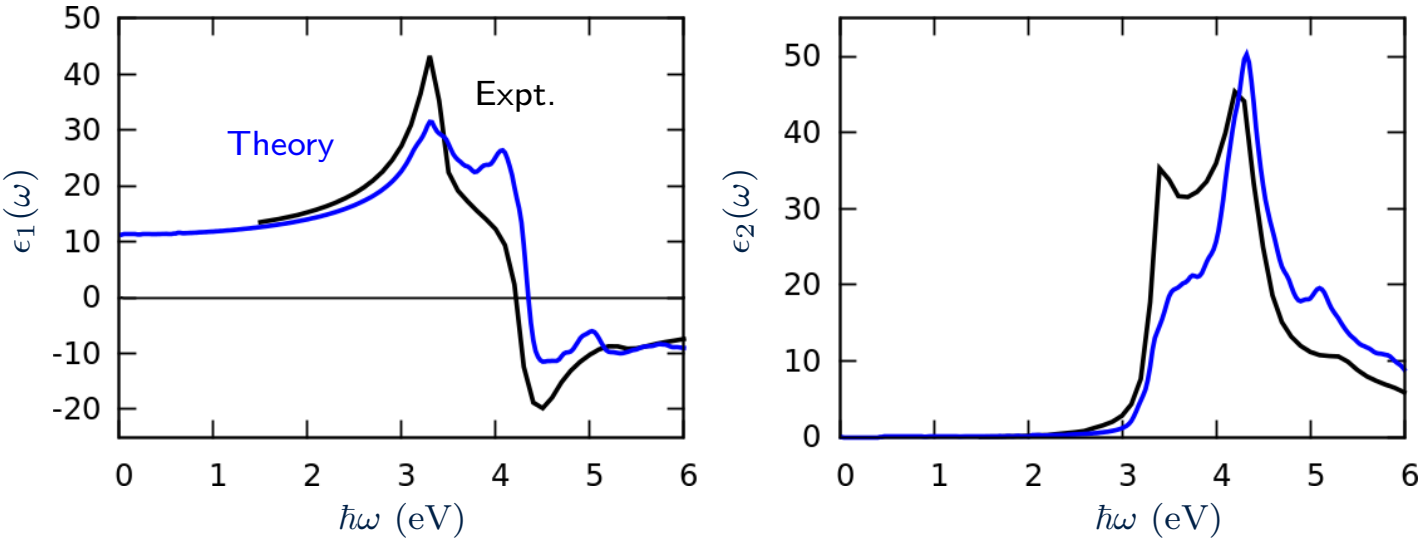
Example: Dielectric function of silicon



- DFT/LDA in the independent-particle approximation

Dielectric function and optical spectra

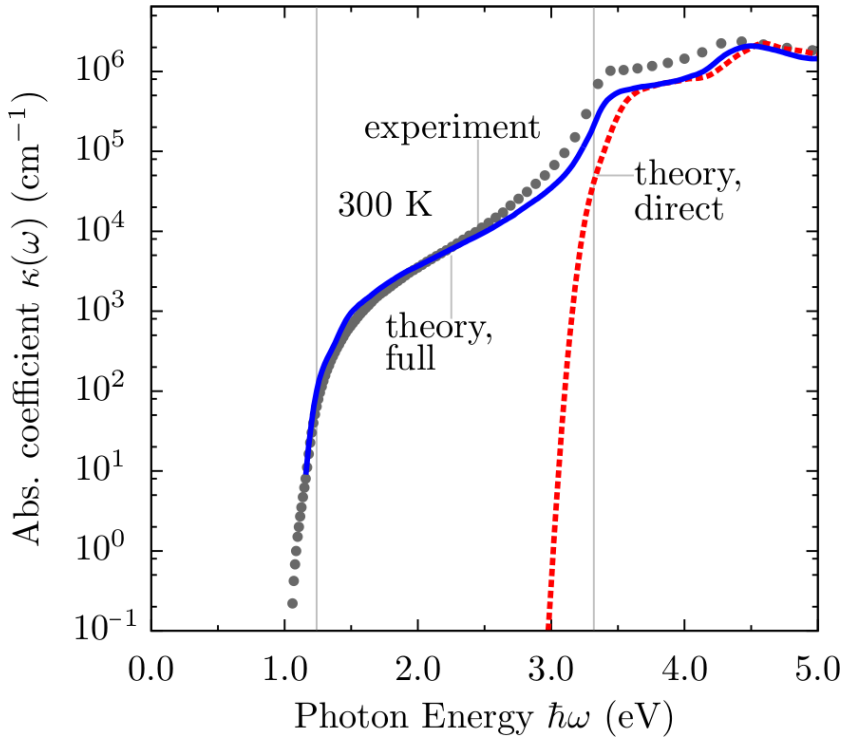
Example: Dielectric function of silicon



- DFT/LDA in the independent-particle approximation
- 'Scissor correction' of 0.55 eV
- Excitonic peak observed at 3.2 eV is missing in the calculation
- Phonon-assisted absorption below 3 eV is missing in the calculation

Dielectric function and optical spectra

Example: Optical absorption coefficient of silicon including phonon-assisted processes



Zacharias, Patrick & Giustino, Phys. Rev. Lett. 115, 177401 (2015)

Which of the following statements is true ?

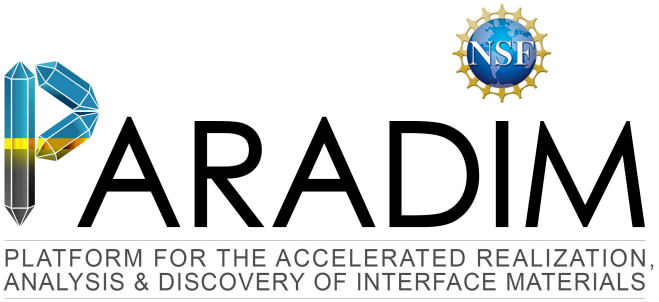
A DFT band structures are generally good for sp semiconductors

B DFT predicts accurate band gaps in semiconductors and insulators

C DFT performs poorly for metals

D The DFT band gap problem has to do with pseudopotentials

E The Kohn-Sham eigenvalues correspond to electron addition or removal energies



An Introduction to Density Functional Theory for Experimentalists

Feliciano Giustino

University of Oxford & Cornell University

Ithaca, 8-14 July 2018

Lecture 5.2

DFT beyond the LDA

PBE exchange and correlation

PBE Perdew, Burke & Ernzerhof, Phys. Rev. Lett. 77, 3865 (1996)

- Within the LDA the XC energy is approximated using the local density

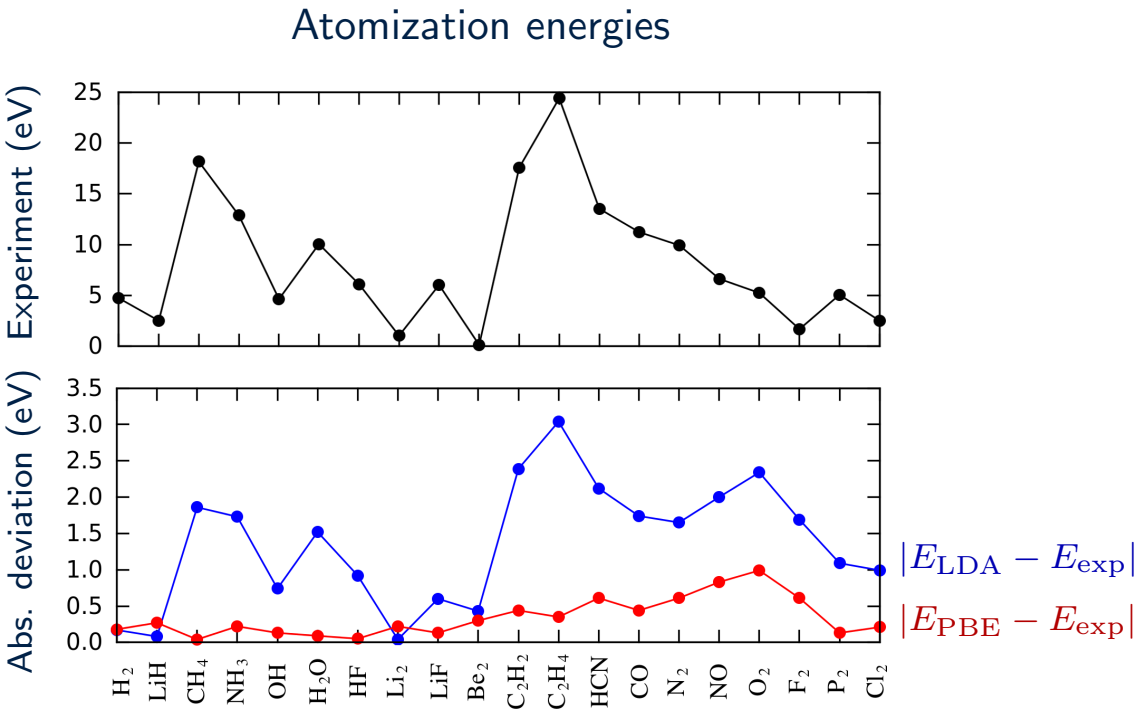
$$E_{xc}^{\rm LDA} = \int d\mathbf{r}\, n(\mathbf{r})\, \epsilon_{xc}^{\rm HEG}[n(\mathbf{r})]$$

- Generalized gradient approximations (GGA) like PBE incorporate also information about the density gradient

$$E_{xc}^{\rm GGA} = \int d\mathbf{r}\, f[n(\mathbf{r}), \nabla n(\mathbf{r})]$$

PBE exchange and correlation

PBE Perdew, Burke & Ernzerhof, Phys. Rev. Lett. 77, 3865 (1996)



From Table 1 of Perdew, Burke, Ernzerhof, PRL 1996

DFT+U Anisimov, Zaanen & Andersen, Phys. Rev. B 44, 943 (1991)

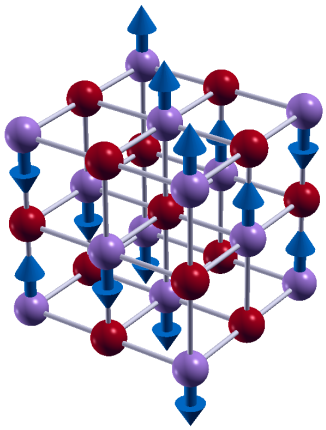
1 H Hydrogen 1.01																	2 He Helium 4.00
3 Li Lithium 6.94	4 Be Beryllium 9.01											5 B Boron 10.81	6 C Carbon 12.01	7 N Nitrogen 14.01	8 O Oxygen 16.00	9 F Fluorine 19.00	10 Ne Neon 20.18
11 Na Sodium 22.99	12 Mg Magnesium 24.31											13 Al Aluminum 26.98	14 Si Silicon 28.09	15 P Phosphorus 30.97	16 S Sulfur 32.07	17 Cl Chlorine 35.45	18 Ar Argon 39.95
19 K Potassium 39.10	20 Ca Calcium 40.08	21 Sc Scandium 44.96	22 Ti Titanium 47.87	23 V Vanadium 50.94	24 Cr Chromium 52.00	25 Mn Manganese 54.94	26 Fe Iron 55.85	27 Co Cobalt 58.93	28 Ni Nickel 58.69	29 Cu Copper 63.55	30 Zn Zinc 65.39	31 Ga Gallium 69.72	32 Ge Germanium 72.61	33 As Arsenic 74.92	34 Se Selenium 78.96	35 Br Bromine 79.90	36 Kr Krypton 83.80
37 Rb Rubidium 85.47	38 Sr Strontium 87.62	39 Y Yttrium 88.91	40 Zr Zirconium 91.22	41 Nb Niobium 92.91	42 Mo Molybdenum 95.94	43 Tc Technetium (98)	44 Ru Ruthenium 101.07	45 Rh Rhodium 102.91	46 Pd Palladium 106.42	47 Ag Silver 107.87	48 Cd Cadmium 112.41	49 In Indium 114.82	50 Sn Tin 118.71	51 Sb Antimony 121.76	52 Te Tellurium 127.60	53 I Iodine 126.90	54 Xe Xenon 131.29
55 Cs Cesium 132.91	56 Ba Barium 137.33	57 La Lanthanum 138.91	72 Hf Hafnium 178.49	73 Ta Tantalum 180.95	74 W Tungsten 183.84	75 Re Rhenium 186.21	76 Os Osmium 190.23	77 Ir Iridium 192.22	78 Pt Platinum 195.08	79 Au Gold 196.97	80 Hg Mercury 200.59	81 Tl Thallium 204.38	82 Pb Lead 207.2	83 Bi Bismuth 208.98	84 Po Polonium (209)	85 At Astatine (210)	86 Rn Radon (222)
87 Fr Francium (223)	88 Ra Radium (226)	89 Ac Actinium (227)	104 Rf Rutherfordium (261)	105 Db Dubnium (262)	106 Sg Seaborgium (266)	107 Bh Bohrium (264)	108 Hs Hassium (269)	109 Mt Meitnerium (268)									

58 Ce Cerium 140.12	59 Pr Praseodymium 140.91	60 Nd Neodymium 144.24	61 Pm Promethium (145)	62 Sm Samarium 150.36	63 Eu Europium 151.96	64 Gd Gadolinium 157.25	65 Tb Terbium 158.93	66 Dy Dysprosium 162.50	67 Ho Holmium 164.93	68 Er Erbium 167.26	69 Tm Thulium 168.93	70 Yb Ytterbium 173.04	71 Lu Lutetium 174.97
90 Th Thorium 232.04	91 Pa Protactinium 231.04	92 U Uranium 238.03	93 Np Neptunium (237)	94 Pu Plutonium (244)	95 Am Americium (243)	96 Cm Curium (247)	97 Bk Berkelium (247)	98 Cf Californium (251)	99 Es Einsteinium (252)	100 Fm Fermium (257)	101 Md Mendelevium (258)	102 No Nobelium (259)	103 Lr Lawrencium (262)

- $3d$ transition metal and $4f$ rare earth
- LDA and PBE underestimate on-site Coulomb energy
- DFT+U adds Hubbard-like correction to remedy this deficiency
- Important for strongly-correlated materials

DFT+U Anisimov, Zaanen & Andersen, Phys. Rev. B 44, 943 (1991)

FeO (AF-II phase)



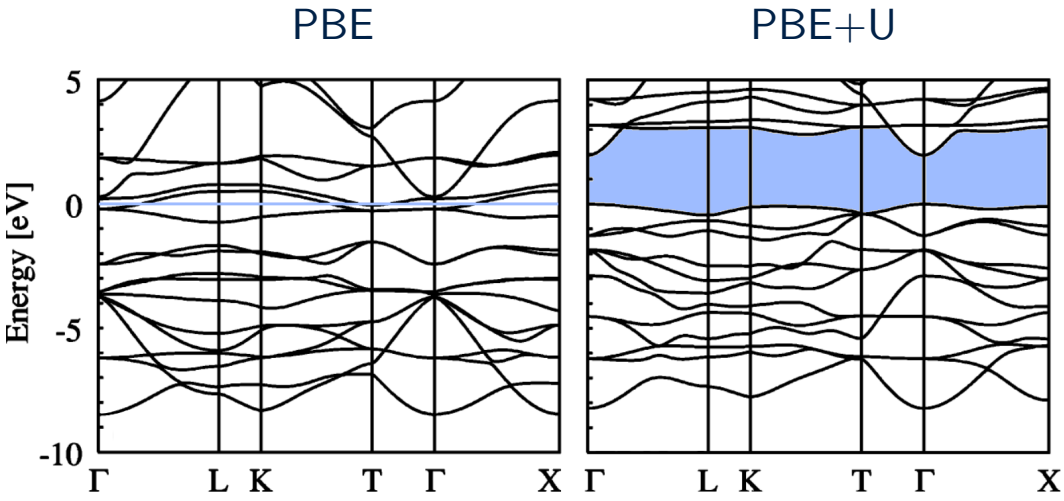


Figure from: Cococcioni & De Gironcoli, PRB 2005

- Computationally efficient
- Hubbard U often used as an adjustable parameter
- Results can be very sensitive to U

Hybrid functionals

PBE0 Perdew, Burke & Ernzerhof, J. Chem. Phys. 105 (1996)
HSE Heyd, Scuseria & Ernzerhof, J. Chem. Phys. 118, 8207 (2003)

- Improve upon semilocal GGA by including fully non-local Fock exchange

$$E_x^{\rm HF} = -\sum_{j\in{\rm occ}} \int d\mathbf{r}_1 d\mathbf{r}_2 \frac{\phi_i^*(\mathbf{r}_1)\phi_j^*(\mathbf{r}_2)\phi_i(\mathbf{r}_2)\phi_j(\mathbf{r}_1)}{|\mathbf{r}_1-\mathbf{r}_2|}$$

- PBE0 prescription

$$E_{xc}^{\rm PBE0} = \frac{3}{4}E_x^{\rm PBE} + \frac{1}{4}E_x^{\rm F} + E_c^{\rm PBE}$$

- Requires the evaluation of the **non-local** Fock exchange potential (expensive)

$$V_{\rm X}(\mathbf{r},\mathbf{r}') = -\sum_{j\in{\rm occ}} \frac{\phi_j(\mathbf{r})\phi_j^*(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}$$

Hybrid functionals

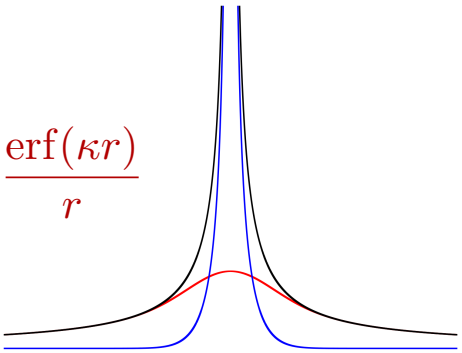
PBE0 Perdew, Burke & Ernzerhof, J. Chem. Phys. 105 (1996)
HSE Heyd, Scuseria & Ernzerhof, J. Chem. Phys. 118, 8207 (2003)

- HSE prescription: separate short-range and long-range coulomb interactions

$$E_{xc}^{\rm HSE} = \left[\frac{3}{4}E_x^{\rm PBE,sr} + \frac{1}{4}E_x^{\rm F,sr}\right] + E_x^{\rm PBE,lr} + E_c^{\rm PBE}$$

- The separation is carried our by breaking the Coulomb potential in two parts

$$\frac{1}{r} = \frac{\mathrm{erfc}(\kappa r)}{r} + \frac{\mathrm{erf}(\kappa r)}{r}$$



Hybrid functionals

PBE0 Perdew, Burke & Ernzerhof, J. Chem. Phys. 105 (1996)
HSE Heyd, Scuseria & Ernzerhof, J. Chem. Phys. 118, 8207 (2003)

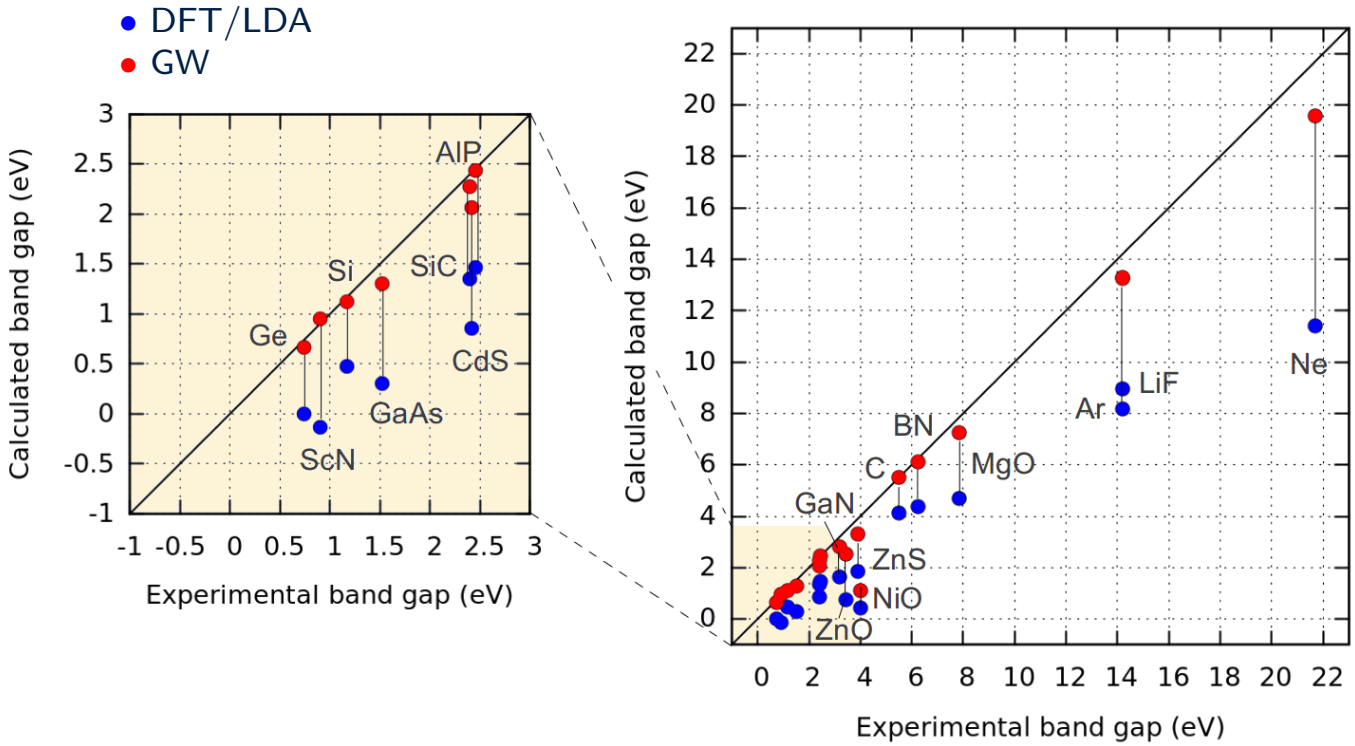
- **Band gaps**
  Typically PBE underestimates band gaps while Hartree-Fock overestimates. Mixing PBE and HF yields values closer to experiments.

- **Exchange fraction**
  Widespread practice of using the mixing fraction as an adjustable empirical parameter. Loss of predictive power.

- **Correlation**
  The correlation energy is still described at the PBE level.
  Misses van der Waals effects and dynamical renormalization effects.

GW method

Hedin & Lundqvist, Solid State Physics 23, 1 (1969)
Hybertsen & Louie, Phys Rev B 34, 5390 (1986)



GW method

Hedin & Lundqvist, Solid State Physics 23, 1 (1969)
Hybertsen & Louie, Phys Rev B 34, 5390 (1986)

KS $$-\frac{1}{2}\nabla^2\phi_i(\mathbf{r}) + [V_{\rm n}(\mathbf{r}) + V_{\rm H}(\mathbf{r})]\,\phi_i(\mathbf{r}) + \boxed{V_{xc}(\mathbf{r})}\phi_i(\mathbf{r}) = \varepsilon_i\phi_i(\mathbf{r})$$

GW $$-\frac{1}{2}\nabla^2\phi_i(\mathbf{r}) + [V_{\rm n}(\mathbf{r}) + V_{\rm H}(\mathbf{r})]\,\phi_i(\mathbf{r}) + \boxed{\int d\mathbf{r}'\Sigma(\mathbf{r},\mathbf{r}',\varepsilon_i)}\phi_i(\mathbf{r}') = \varepsilon_i\phi_i(\mathbf{r})$$

$$\Sigma = GW$$

Screened Coulomb interaction

Electron Green's function

van der Waals interaction

Example: Two H atoms far away from each other

- Electronic ground-state wavefunction for non-interacting atoms

$$\Psi_0(\mathbf{r}_1, \mathbf{r}_2) = \phi_{1s}^A(\mathbf{r}_1)\, \phi_{1s}^B(\mathbf{r}_2)$$

- Electron density

- This is (roughly) what we would obtain within DFT/LDA

van der Waals interaction

Example: Two H atoms far away from each other

- We can obtain a lower-energy wavefunction by using
  (Exercise 4.5 of DFT book)

$$\Psi(\mathbf{r}_1,\mathbf{r}_2) = \Psi_0(\mathbf{r}_1,\mathbf{r}_2) + \frac{\alpha/4\pi\epsilon_0}{d^3}\,\phi^A_{2p_x}(\mathbf{r}_1)\,\phi^B_{2p_x}(\mathbf{r}_2)$$

  α = atomic polarizability

- The electron density for this wavefunction is polarized

van der Waals interaction

Example: Two H atoms far away from each other

- The density redistribution leads to an additional attractive potential energy between the nuclei, the van der Waals interaction

$$U = -\frac{C}{d^6}$$

- Not included in standard DFT
- Important for graphitic materials, biological systems, molecular crystals

Andersson, Langreth & Lundqvist, Phys. Rev. Lett. 76, 102 (1996)

Dion & al, Phys. Rev. Lett. 92, 246401 (2004)

Grimme, J. Comp. Chem. 27, 1787 (2006)

Tkatchenko & Scheffler, Phys. Rev. Lett. 102, 073005 (2009)

Which of the following statements is true ?

A I can calculate accurate band gaps using the GW method

B DFT/LDA cannot be used to study organic crystals

C To obtain the best results I should always combine DFT+U, GW, hybrid functionals, and van der Waals corrections

D DFT+U is a predictive *ab initio* method

E I know everything about DFT

An introduction to density functional theory for experimentalists

Connecting to MARCC

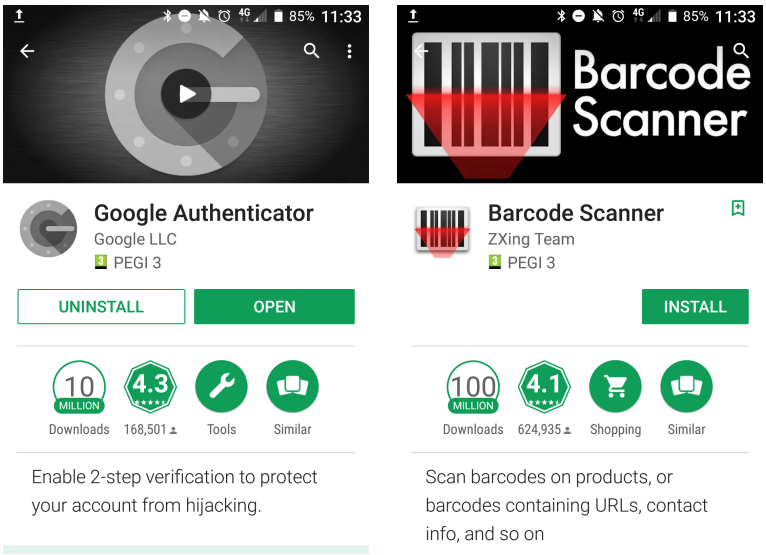
Two-factor authentication

During this school we will perform calculations at the Maryland Advanced Research Computing Center (MARCC). In order to connect to MARCC we need to setup the 'two-factor authentication' protocol.

For this you will need:

- The username sent to you by MARCC via email
- The temporary password sent to you by MARCC via email
- The Google Authenticator

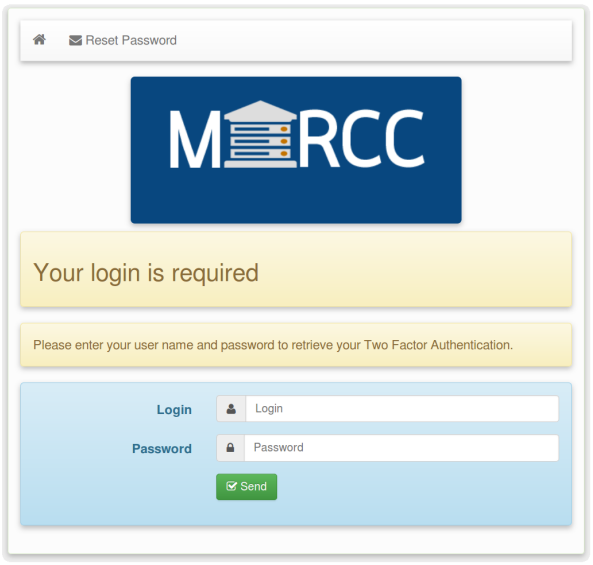
The Google Authenticator can be installed from the Play Store on your mobile device. You will also need a Barcode Scanner. The two apps look as follows:



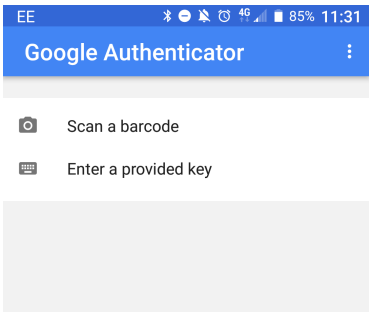
Now open this webpage (just follow the hyperlink):

https://password.marcc.jhu.edu/?action=qrretrieve

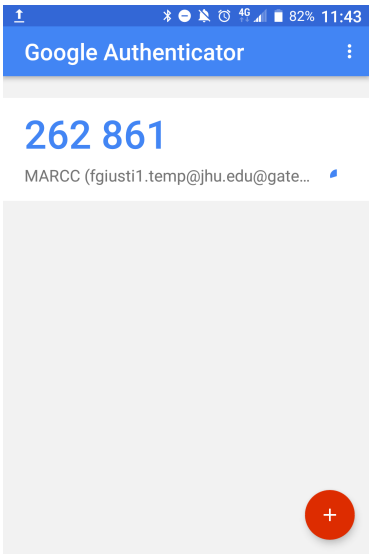
You will be prompted for username and password, here we need to enter the `username` and `password` that we have received by email:



After entering our username and password, the webpage shows a barcode. Here we link the Google Authenticator to our MARCC account by scanning the barcode:



The above steps are performed only once. From now on we can login into MARCC using our username, password, and verification code. The verification code is the number provided by the Google Authenticator, as in the following snapshot:



The standard procedure for logging in is described in Tutorial T1.1.

Connecting from a Windows machine

If you are using Windows on your laptop/desktop, then in order to connect to MARCC you will need a software that can handle a secure shell (SSH) connection.

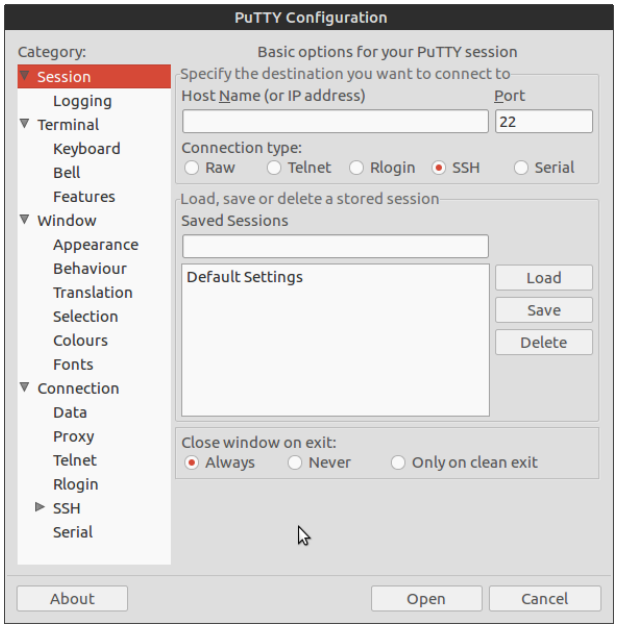
A popular choice is Putty, which can be downloaded from:

http://www.putty.org

If you are unsure whether you have a 32 bit or 64 bit architecture, then the safest option is to download the following executable:

https://the.earth.li/~sgtatham/putty/latest/w32/putty.exe

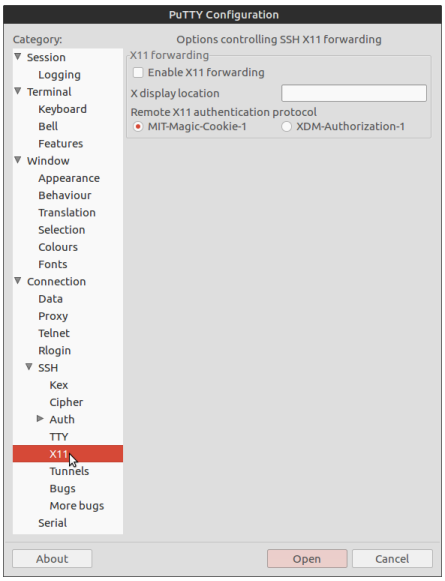
When you execute Putty you will see something like the following:

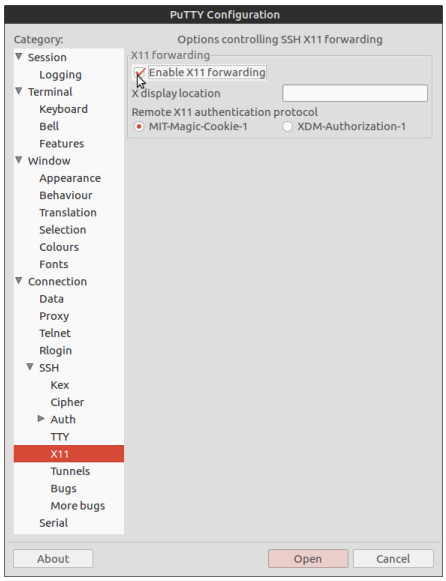


In the field 'Host Name' we enter:

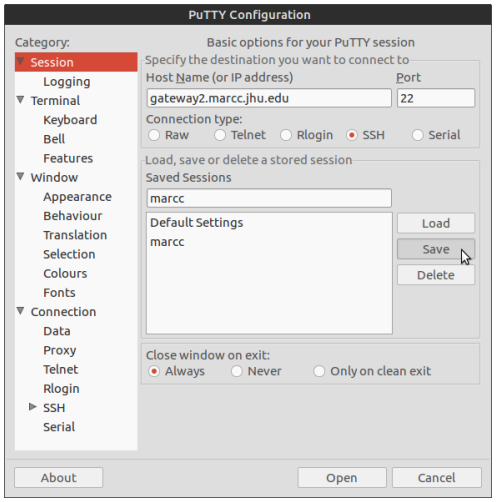
`gateway2.marcc.jhu.edu`

In order to be able to see graphics over this connection, we need to enable 'X11 forwarding'.
For this we proceed as indicated below:





Now we can save these settings, so that next time we will just click on the session name, say 'marcc':



Visualizing graphics from a Windows laptop

In order to visualize graphics when using Putty, your machine must be able to understand the X11 protocol. This can be done by downloading the program Xming.

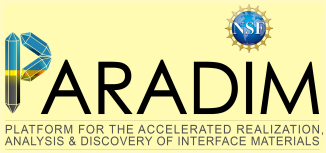
The installation file can be found at the following link:

https://sourceforge.net/projects/xming/files/Xming/6.9.0.31/Xming-6-9-0-31-setup.exe/download

After installing `Xming` the procedure for running calculations and visualizing graphics on MARCC is as follows:

- Start `Xming`. This application will now run in the background.
- Start `Putty` and open a session.

From this point onward everything works exactly in the same way as for users of Linux or Mac.



An introduction to density functional theory for experimentalists

Tutorial 1.1

Login shell and compilation

We will perform calculations on the Blue Crab Linux cluster of MARCC. Blue Crab hosts 676 Intel Haswell dual socket 12-core processors, and for this tutorial we will be using between 4 and 24 cores at a time. In order to work on Blue Crab we need to establish a secure connection. We first open a terminal (on Ubuntu/Mac machines; instead we launch Putty from Windows), then we type:

```
$ ssh -Y gateway2.marcc.jhu.edu -l fgiusti1.temp@jhu.edu
```

where `fgiusti1.temp@jhu.edu` must be replaced by the username that you have been assigned. After entering your verification code from the Google Authenticator and your password you will see something like:

```
[fgiusti1.temp@jhu.edu@login-node01 ~]$
```

We can customize the Unix 'shell' environment by shortening the prompt, creating a couple of 'aliases', and adding modules that we will need later on. We copy/paste the following into the terminal (it is important to copy/paste exactly as it is, since the `bash` shell is very strict with spaces):

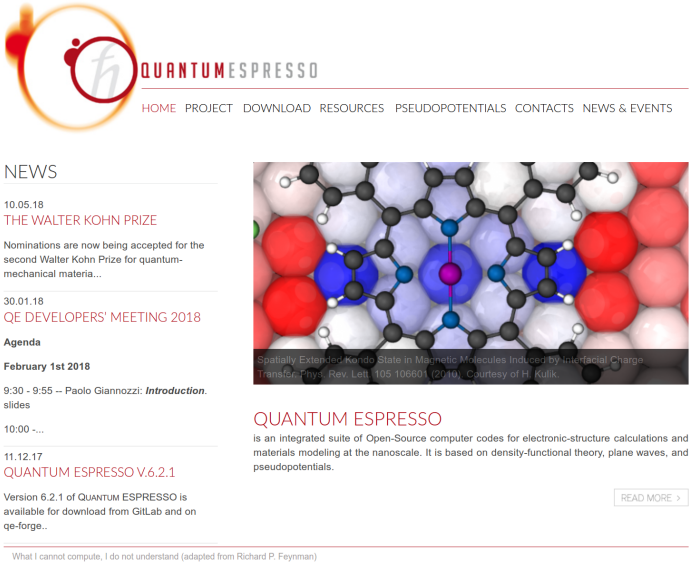
```
cat >> .bashrc << EOF
PS1="$ "
alias c="clear"
alias l="ls -lh --color=none"
module load gcc/4.8.2
module load openmpi/intel/2.0.1
module load fftw3/intel/3.3.7
module load xcrysden/1.5.53
module load gnuplot/5.0.0
EOF
source ~/.bashrc
```

From now on the prompt will be simply '$' and the command 'c' and 'l' will clear the screen and list the content of a directory, respectively. We can now create our working directory for this school (`mkdir`) and we move inside this directory (`cd`):

```
$ mkdir scratch/summerschool ; cd scratch/summerschool
```

In this school we will be using the `Quantum ESPRESSO` (QE) software package. QE is an open-source suite of *ab initio* electronic structure codes based on pseudopotentials and planewaves.

The project website can be found at `www.quantum-espresso.org`



In order to use QE we download the latest release as a compressed archive, and we install it in our working directory. The latest snapshot can be found at http://www.quantum-espresso.org/download. To make sure that we all run the same version, we download a snaphot from 15 May 2018 that I temporarily placed in a private directory:

```
$ wget http://giustino.materials.ox.ac.uk/q-e-master.tar.gz
```

Then we unzip and unpack the compressed archive (`tar xfz`):

```
$ tar xfz q-e-master.tar.gz
```

QE is now unpacked. It is useful to take a look inside the directory:

```
$ cd q-e-master ; l
```

```
total 228K
drwxr-xr-x  2 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 archive
drwxr-xr-x  6 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 atomic
drwxr-xr-x  2 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 clib
-rwxr-xr-x  1 fgiusti1.temp@jhu.edu paradim 2.3K May 11 13:02 configure
-rw-r--r--  1 fgiusti1.temp@jhu.edu paradim 2.2K May 11 13:02 CONTRIBUTING.md
drwxr-xr-x  6 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 COUPLE
drwxr-xr-x  5 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 CPV
drwxr-xr-x  3 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 dev-tools
drwxr-xr-x  2 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 dft-d3
drwxr-xr-x  4 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 Doc
-rw-r--r--  1 fgiusti1.temp@jhu.edu paradim 3.3K May 11 13:02 environment_variables
drwxr-xr-x  6 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 EPW
drwxr-xr-x  3 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 FFTXlib
drwxr-xr-x  5 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 GUI
drwxr-xr-x  9 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 GWW
drwxr-xr-x  2 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 include
drwxr-xr-x  3 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 install
drwxr-xr-x  5 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 KS_Solvers
drwxr-xr-x  2 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 LAXlib
-rw-r--r--  1 fgiusti1.temp@jhu.edu paradim  18K May 11 13:02 License
-rw-r--r--  1 fgiusti1.temp@jhu.edu paradim  42K May 11 13:02 logo.jpg
```

```
drwxr-xr-x  2 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 LR_Modules
-rw-r--r--  1 fgiusti1.temp@jhu.edu paradim  14K May 11 13:02 Makefile
drwxr-xr-x  2 fgiusti1.temp@jhu.edu paradim  12K May 11 13:02 Modules
drwxr-xr-x  6 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 NEB
drwxr-xr-x  7 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 PHonon
drwxr-xr-x  6 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 PlotPhon
drwxr-xr-x  7 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 PP
drwxr-xr-x  2 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 pseudo
drwxr-xr-x  7 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 PW
drwxr-xr-x  5 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 PWCOND
drwxr-xr-x  8 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 QHA
-rw-r--r--  1 fgiusti1.temp@jhu.edu paradim 1.4K May 11 13:02 README.md
drwxr-xr-x  7 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 TDDFPT
drwxr-xr-x 48 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 test-suite
drwxr-xr-x  3 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 upftools
drwxr-xr-x  2 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 UtilXlib
drwxr-xr-x  6 fgiusti1.temp@jhu.edu paradim 4.0K May 11 13:02 XSpectra
```

In this school we will be mainly interested in the program `pw.x`, which is contained in the directory `PW`. In order to use this program we need to compile the fortran source into an executable. This operation is performed by the script `Makefile`. `Makefile` in turn needs to know where to look for the compilers and numerical libraries. This information is retrieved by the program `configure`, which probes the host system for available software. To make this happen we issue:

```
$ ./configure ; make pw
```

This operation will require approximately 7 min.

Note `pw.x` and the other QE programs are available as precompiled executables on many HPC platforms, including MARCC. The reason for downloading and compiling this program by ourselves is to learn how to get started in case QE was not already available. This will allow you to use this tutorial on your own computer after the user accounts on MARCC will have been deleted.

While we wait we might as well check the page of MARCC where the cluster Blue Crab is described: Maryland Advanced Research Computing Center.

At the end of the compilation we should find a pointer to the newly-created executable `pw.x` inside the directory `bin`:

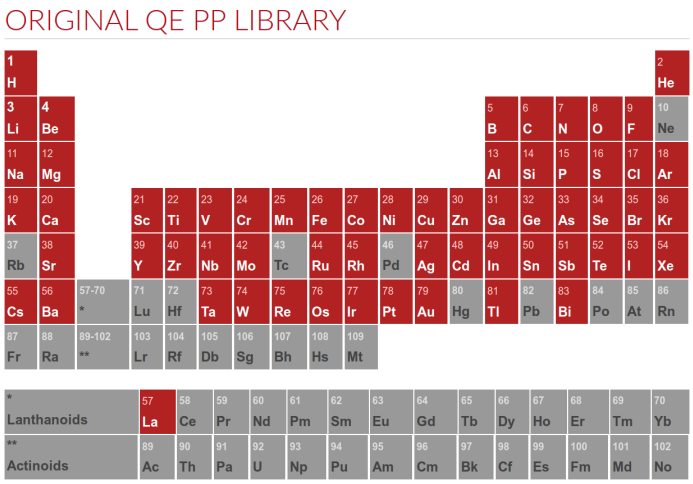
```
$ l bin
```

```
lrwxrwxrwx 1 fgiusti1.temp@jhu.edu paradim 14 May 15 14:39 pw.x -> ../PW/src/pw.x
```

Test run

Now we want to execute a simple job on the cluster. The goal of this operation is to make sure that everything runs smoothly.

We will consider a simple total energy calculation for **silicon** in the diamond structure. In order to proceed we first need a 'pseudopotential'. The concept of pseudopotentials is covered in the theory lectures; for now it suffices to know that we need one pseudopotential for each atomic species, and that the pseudopotential describes the atomic nucleus and all the electrons except the outermost (valence) shell. The QE pseudopotential libraries can be found at `http://www.quantum-espresso.org/pseudopotentials`. Using the menu on the left we can select the 'Original QE PP table', which looks like the following:



If we click on silicon we see a list of available pseudopotentials. In this example we will use the pseudopotential labelled `Si.pz-vbc.UPF`. By hovering on this link with the mouse we can copy/paste the web link, and we can use it to download the file directly in our working directory:

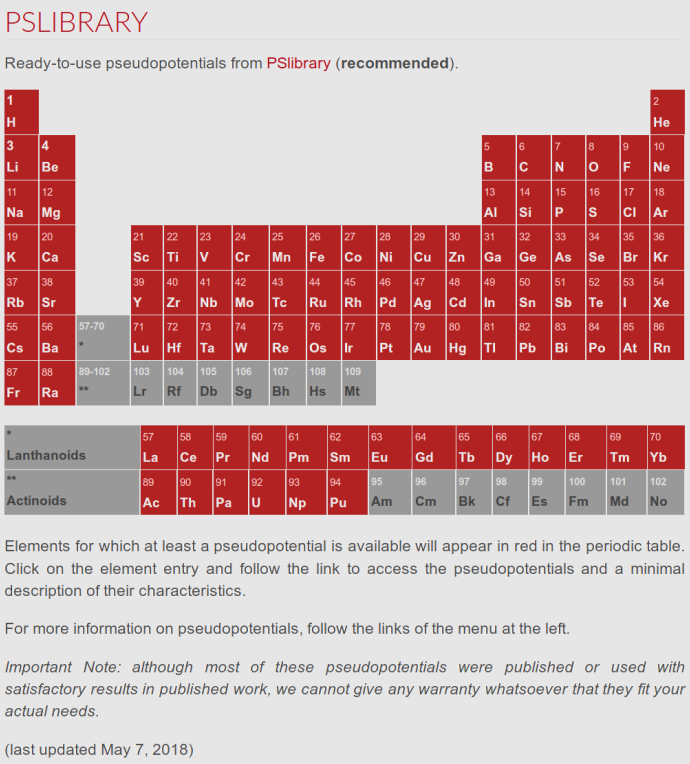
```
$ cd ..
$ mkdir tutorial-1.1 ; cd tutorial-1.1
$ wget https://www.quantum-espresso.org/upf_files/Si.pz-vbc.UPF
```

Now we should have the pseudopotential file inside the directory `tutorial-1.1`. We can look inside this plain text file by using the command `more`:

```
$ more Si.pz-vbc.UPF
```

```
<PP_INFO>
...
Info: Si LDA 3s2 3p2 VonBarth-Car, l=2 local
    0        The Pseudo was generated with a Non-Relativistic Calculation
  0.00000000000E+00    Local Potential cutoff radius
nl pn  l   occ               Rcut            Rcut US             E pseu
3S  0  0  2.00      0.00000000000      0.00000000000      0.00000000000
3P  0  1  2.00      0.00000000000      0.00000000000      0.00000000000
</PP_INFO>

<PP_HEADER>
   0                  Version Number
  Si                  Element
   NC                 Norm - Conserving pseudopotential
    F                 Nonlinear Core Correction
 SLA  PZ   NOGX NOGC  PZ   Exchange-Correlation functional
    4.00000000000     Z valence
    0.00000000000     Total energy
  0.0000000  0.0000000 Suggested cutoff for wfc and rho
    1                 Max angular momentum component
  431                 Number of points in mesh
    2    2            Number of Wavefunctions, Number of Projectors
 Wavefunctions        nl  l   occ
                      3S  0  2.00
                      3P  1  2.00
</PP_HEADER>
...
```

Note For all production calculations it is recommended to use pseudopotentials from the 'PSlibrary table'. This library reports the most up-to-date and carefully validated pseudopotentials. The table is accessed at http://www.quantum-espresso.org/pseudopotentials and looks as follows:



PSLIBRARY

Ready-to-use pseudopotentials from PSlibrary (**recommended**).

1 H																	2 He
3 Li	4 Be											5 B	6 C	7 N	8 O	9 F	10 Ne
11 Na	12 Mg											13 Al	14 Si	15 P	16 S	17 Cl	18 Ar
19 K	20 Ca		21 Sc	22 Ti	23 V	24 Cr	25 Mn	26 Fe	27 Co	28 Ni	29 Cu	30 Zn	31 Ga	32 Ge	33 As	34 Se	35 Br	36 Kr
37 Rb	38 Sr		39 Y	40 Zr	41 Nb	42 Mo	43 Tc	44 Ru	45 Rh	46 Pd	47 Ag	48 Cd	49 In	50 Sn	51 Sb	52 Te	53 I	54 Xe
55 Cs	56 Ba	57-70 *	71 Lu	72 Hf	73 Ta	74 W	75 Re	76 Os	77 Ir	78 Pt	79 Au	80 Hg	81 Tl	82 Pb	83 Bi	84 Po	85 At	86 Rn
87 Fr	88 Ra	89-102 **	103 Lr	104 Rf	105 Db	106 Sg	107 Bh	108 Hs	109 Mt									

* Lanthanoids	57 La	58 Ce	59 Pr	60 Nd	61 Pm	62 Sm	63 Eu	64 Gd	65 Tb	66 Dy	67 Ho	68 Er	69 Tm	70 Yb
** Actinoids	89 Ac	90 Th	91 Pa	92 U	93 Np	94 Pu	95 Am	96 Cm	97 Bk	98 Cf	99 Es	100 Fm	101 Md	102 No

Elements for which at least a pseudopotential is available will appear in red in the periodic table. Click on the element entry and follow the link to access the pseudopotentials and a minimal description of their characteristics.

For more information on pseudopotentials, follow the links of the menu at the left.

Important Note: although most of these pseudopotentials were published or used with satisfactory results in published work, we cannot give any warranty whatsoever that they fit your actual needs.

(last updated May 7, 2018)

For simplicity we also copy (cp) the executable inside the current working directory (this is not standard practice but it makes things easier to understand at the beginning):

```
$ cp ../q-e-master/bin/pw.x ./
```

At this point we have the executable `pw.x` and the pseudopotential for silicon `Si.pz-vbc.UPF`. We are missing the input file for the executable.
We can create the simplest possible input file, `silicon-1.in`, as follows:

```
$ cat << EOF > silicon-1.in
&control
 calculation = 'scf',
 prefix = 'silicon',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 2,
 celldm(1) = 10.28,
 nat = 2,
 ntyp = 1,
 ecutwfc = 18.0,
/
&electrons
/
```

```
ATOMIC_SPECIES
 Si  28.086  Si.pz-vbc.UPF
ATOMIC_POSITIONS alat
 Si 0.00 0.00 0.00
 Si 0.25 0.25 0.25
K_POINTS automatic
 4 4 4 1 1 1
EOF
```

In order to run our jobs we will mostly use *interactive* sessions on Blue Crab. An interactive session is launched by issuing the following command:

```
$ interact -p shared -n 12 -t 360 -r paradim
```

This command is requesting 12 CPUs for a duration of 360 min.

Important Note: We run this command **only once** at the beginning of the session.

When the interactive session starts, we can run `pw.x` by issuing:

```
$ mpirun -n 4 pw.x -npool 4 < silicon-1.in > silicon-1.out
```

This job will take less than a second to complete. The output file is `silicon-1.out` and should look like the following:

```
$ more silicon-1.out
```

```
     Program PWSCF v.6.2.2 starts on 15May2018 at 14:42:20

     This program is part of the open-source Quantum ESPRESSO suite
     for quantum simulation of materials; please cite
         "P. Giannozzi et al., J. Phys.:Condens. Matter 21 395502 (2009);
         "P. Giannozzi et al., J. Phys.:Condens. Matter 29 465901 (2017);
          URL http://www.quantum-espresso.org",
     in publications or presentations arising from this work. More details at
     http://www.quantum-espresso.org/quote

     Parallel version (MPI), running on     4 processors

     MPI processes distributed on     1 nodes
     K-points division:     npool     =       4
     Waiting for input...
     Reading input from standard input

...
     PWSCF        :     0.40s CPU         0.88s WALL

   This run was terminated on:  14:42:23  15May2018

=------------------------------------------------------------------------------=
   JOB DONE.
=------------------------------------------------------------------------------=
```

Note Interactive sessions are mostly used for software development purposes. For most production jobs we do not run interactive sessions, instead we use a *batch queuing system*. In this case we prepare a 'job' submission script that instructs the machine about the resources that we need, and then we place it in a queue with all the other jobs pending. A job scheduler (SLURM in Blue Crab) monitors the available resources on the cluster and allocates them in order of priority.
This route is not worth the effort for small, fast jobs, but it becomes a necessity for jobs that are expected to require days or weeks of HPC time.
If we want to follow this route we need to create a submission script `job-1.pbs` as follows:

```
$ cat << EOF > job-1.pbs
#!/bin/bash -l
#SBATCH --reservation=paradim
#SBATCH --job-name=job-1
#SBATCH --time=00:30:00
#SBATCH --partition=shared
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=12
#SBATCH --mem-per-cpu=1000MB
mpirun -np 12 pw.x -npool 4 < silicon-1.in > silicon-1.out
EOF
```

The SBATCH directives are used to request the appropriate resources, as we did for the interactive session. We submit this job to the queue by issuing:

```
$ qsub job-1.pbs
```

We can check the status of this job in the queue using the command `qstat`. If we do not remember our username we can find this information using:

```
$ whoami
```

```
fgiusti1.temp@jhu.edu
```

```
$ qstat -u fgiusti1.temp@jhu.edu
```

```
login-node01.cm.cluster:
                                                                  Req'd Req'd Elap
Job id          Username Queue    Name      SessID NDS   TSK    Memory Time Use S Time
--------------- -------- -------- --------- ------ ----- ----- ------ ----- - -----
27175243        fgiusti1 parallel job-1     --          1    12     -- 00:30 Q 00:00
```

At the end of the execution we will find the results in the file `silicon-1.out`, as for the interactive session.

Generating new input files

In the following we will prepare input files by modifying the file `silicon-1.in`.
Instead of using the `cat` command as we did in the previous section, we first create a new file by just copying the previous one:

```
$ cp silicon-1.in silicon-2.in
```

Now we use `vi` to modify the as-created file:

```
$ vi silicon-2.in
```

This command will open the file inside the current terminal window. The rules for using `vi` are simple:

1. We move around using ↑ ↓ → ←
2. In order to change the text we press `i` and modify as we wish
3. When we are done making changes we press `ESC`
4. We write the modified file and exit by typing `:` `w` `q` (`w` is for write, `q` is for quit)

As an example we now change the parameter `ecutwfc` from 18 Ry to 40 Ry. This parameter represents the kinetic energy cutoff of the planewaves used in the Fourier expansion of the electron wavefunctions (see theory lectures). We can also change the Brillouin zone sampling from `4 4 4 1 1 1` to something more accurate, say `8 8 8 1 1 1`.
We execute once again `pw.x` using the new input file:

```
$ mpirun -n 4 pw.x -npool 4 < silicon-2.in > silicon-2.out
```

The calculation parameters and runtime flags will become more clear as we progress with the tutorials.

Documentation

A comprehensive description of the input variables accepted by `pw.x` can be found here:

https://www.quantum-espresso.org/Doc/INPUT_PW.html


Input File Description
Program: pw.x / PWscf / [sc]Quantum ESPRESSO[/sc] (version: 6.2)

TABLE OF CONTENTS

INTRODUCTION

&CONTROL

calculation | title | verbosity | restart_mode | wf_collect | nstep | iprint | tstress | tprnfor | dt | outdir | wfcdir | prefix | lkpoint_dir | max_seconds | etot_conv_thr | forc_conv_thr | disk_io | pseudo_dir | tefield | dipfield | lelfield | nberrycyc | lorbm | lberry | gdir | nppstr | lfcpopt | gate

&SYSTEM

ibrav | celldm | A | B | C | cosAB | cosAC | cosBC | nat | ntyp | nbnd | tot_charge | starting_charge | tot_magnetization | starting_magnetization | ecutwfc | ecutrho | ecutfock | nr1 | nr2 | nr3 | nr1s | nr2s | nr3s | nosym | nosym_evc | noinv | no_t_rev | force_symmorphic | use_all_frac | occupations | one_atom_occupations | starting_spin_angle | degauss | smearing | nspin | noncolin | ecfixed | qcutz | q2sigma | input_dft | exx_fraction | screening_parameter | exxdiv_treatment | x_gamma_extrapolation | ecutvcut | nqx1 | nqx2 | nqx3 | lda_plus_u | lda_plus_u_kind | Hubbard_U | Hubbard_J0 | Hubbard_alpha | Hubbard_beta | Hubbard_J(i,ityp) | starting_ns_eigenvalue(m,ispin,t) | U_projection_type | edir | emaxpos | eopreg | eamp | angle1 | angle2 | constrained_magnetization | fixed_magnetization | lambda | report | lspinorb | assume_isolated | esm_bc | esm_w | esm_efield | esm_nfit | fcp_mu | vdw_corr | london | london_s6 | london_c6 | london_rvdw | london_rcut | ts_vdw_econv_thr | ts_vdw_isolated | xdm | xdm_a1 | xdm_a2 | space_group | uniqueb | origin_choice | rhombohedral | zgate | relaxz | block | block_1 | block_2 | block_height

A comprehensive description of the QE project and the most recent developments is provided in the following manuscript:

P. Giannozzi et al., Advanced capabilities for materials modelling with Quantum ESPRESSO, J. Phys.: Condens. Matter 29, 465901 (2017).

Information about the job scheduling system of MARCC can be found at:

http://www.marcc.jhu.edu/getting-started/running-jobs

In order to find out which 'queues' or 'partitions' are available we can use the command

```
$ sinfo -s
```

Unix cheat sheet

Below is a summary of useful Unix commands that we will use throughout this school:

`ls`	List content of current folder
`clear`	Clear the screen
`pwd`	Print working directory
`cd thisfolder`	Enter the folder called *thisfolder*
`cd ..`	Go up one folder
`cd`	Go to home directory
`more thisfile`	Show the content of the file called *thisfile*
`rm thisfile`	Remove the file called *thisfile*
`cp file1 file2`	Copy the file *file1* into the file *file2*
`mv thisfile thisfolder/`	Move the file *thisfile* into the folder *thisfolder*
`vi thisfile`	Open the file *thisfile* using the Vi text editor
`grep thisword thisfile`	Search for the word *thisword* inside the file *thisfile*
`man thiscommand`	Show the manual page for the command *thiscommand*
`scp myfile username@remoteip:~/`	Copy the file *myfile* to the home directory of user *username* in the remote computer identified by the IP address *remoteip*.

An introduction to density functional theory for experimentalists

Tutorial 1.2

Exercise 1

▶ Repeat the steps illustrated during Tutorial T1.1, in particular:

1. Login into your account, set the modules in the file `.bashrc`, and create your working directory
2. Download the QE package, unzip, configure, and make the executable `pw.x`
3. Download the pseudopotential for silicon `Si.pz-vbc.UPF`
4. Create the input files `silicon-1.in`
5. Execute `pw.x` and check the output file `silicon-1.out`

You should be able to directly copy/paste the instructions from the PDF document of Tutorial T1.1, or you can type everything if you are patient.

Exercise 2

Now we want to familiarize ourselves with one important convergence parameter of DFT calculations based on pseudopotentials and planewaves, the planewave kinetic energy cutoff `ecutwfc`.
To this aim we create a new directory:

```
$ cd ~/scratch/summerschool ; mkdir tutorial-1.2 ; cd tutorial-1.2
```

and we copy over the executable, pseudopotential, and input files generated in the previous exercise:

```
$ cp ../tutorial-1.1/pw.x ./
$ cp ../tutorial-1.1/Si.pz-vbc.UPF ./
$ cp ../tutorial-1.1/silicon-1.in ./silicon-3.in
```

Using `vi` we modify the input variable `ecutwf` to 5.0 Ry (note 1 Ry = 13.6058 eV). After this change, line #12 of `silicon-3.in` should read:

```
ecutwfc    = 5.0,
```

Now we execute `pw.x` as usual:

```
$ mpirun -n 4 pw.x -npool 1 < silicon-3.in > silicon-3.out
```

When the job is completed we can analyze the output file `silicon-3.out`. This output file contains the most important information regarding your run. During the tutorials we will become familiar with the various sections of this file. For now we concentrate only on some key info.

First of all we can check that we are using the local density approximation (LDA) to DFT. To see this, open the output file using `vi`, and search for the words `Exchange-correlation`. To activate the search function in `vi` we simply press `/` and enter the search word. You will find:

```
Exchange-correlation      =  SLA  PZ   NOGX NOGC ( 1   1   0   0 0 0)
```

Here SLA stands for 'Slater exchange', PZ stands for Perdew-Zunger parametrization of the LDA, NOGX and NOGC say that density gradients are not taken into account (the meaning of this will become clear from the theory lectures). The numbers are internal codes of pw.x.

Now we search for the words kinetic-energy cutoff. This should be indentical to the value of ecutwfc set in the input file. This parameter is the kinetic energy cutoff of the planewaves basis set, and sets the number of planewaves in which every Kohn-Sham wavefunction is expanded (i.e. the number of Fourier components used to represent electron wavefunctions). The number of planewaves corresponding to the cutoff ecutwfc can be found by searching for the keyword Parallelization info. You will see the following line:

```
Parallelization info
--------------------
sticks:   dense  smooth     PW     G-vecs:    dense   smooth      PW
...
Sum          73      73     37                  411      411     137
```

This means that each wavefunction is expressed as a linear combination of 137 planewaves. The number of electrons and wavefunctions is found by searching for the keywords number of electrons and number of Kohn-Sham states. We find:

```
number of electrons       =         8.00
number of Kohn-Sham states=            4
```

meaning that we have 8 electrons in 4 Kohn-Sham states. This is consistent with the fact that in silicon we have no spin-polarization, so each spatial wavefunction describes two electrons (spin-up and spin-down).

Now we want to look at the most important quantity in the output file, the DFT total energy. Search for the marker ! in the output file. You should find:

```
!    total energy              =     -15.60437807 Ry
```

This value should be taken with caution: it contains an offset which depends on the chosen pseudopotentials and on conventions in the code. This value is not referred to vacuum, since there is no vacuum reference when we perform a calculation in infinitely-extended crystals. This means that the absolute value of DFT total energies in extended solids is not meaningful; what is meaningful is any total energy difference where the artificial offset cancels out.

Finally we look at the timing: search for the word 'WALL'. You should find:

```
init_run     :      0.03s CPU      0.03s WALL (       1 calls)
...
PWSCF        :      0.08s CPU      0.11s WALL
```

The number on the left is the CPU-time, that is the execution time as measured on each individual CPU. The number on the right is the 'wall-clock' time, and indicates the actual time elapsed from the beginning to the end of the run. The code provides a breakdown of the time spent in each key subroutine, and the grand total at the end.

Now we want to study how the total energy, the number of planewaves, and the timing vary as a function of the planewaves cutoff `ecutwfc`.

▶ Repeat the above steps by setting `ecutwfc` to 5, 10, 15, 20, 25, 30, 35, 40 in the input file. It is convenient to generate separate input/output files and then search for the energy, the number of planewaves, and the CPU time in each output file. You can collect the results for example by creating a text file using `vi exercise2.txt` and entering your results one by one. You should be able to construct a file looking like this:

```
$ more excercise2.txt
# ecutwfc (Ry)   planewaves   energy (Ry)   time (s)
    5              137        -15.60437807    0.08
   10              283        -15.77558549    0.09
   15              459        -15.83422231    0.14
   20              645        -15.84721985    0.22
   25              893        -15.85087566    0.21
   30             1139        -15.85182958    0.23
   35             1363        -15.85235148    0.31
   40             1639        -15.85268613    0.41
```

Note: If you do not want to change each input file manually, you can use the following to generate the files:

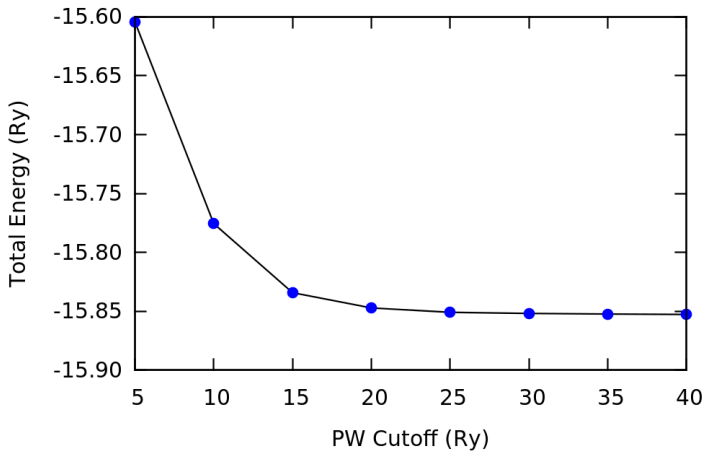
```
$ cat > loop.tcsh << EOF
foreach ECUTWFC ( 5 10 15 20 25 30 35 40 )
  sed "s/5\\.0/\${ECUTWFC}/g" silicon-3.in > silicon-\${ECUTWFC}.in
end
EOF
$ tcsh loop.tcsh
```

Furthermore you can use the command `grep` in order to extract the information that you are looking for automatically. For example:

```
$ grep "\!" silicon-5.out

!    total energy              =     -15.60437814 Ry
```

At this point we can analyze our results. For this you can either use `gnuplot` directly on the cluster, or you can transfer the file `exercise2.txt` using the command `scp` or the program `filezilla`, and then plot the data using your favourite software (e.g. `Origin` or `Excel`).

For the total energy you should find something like this:



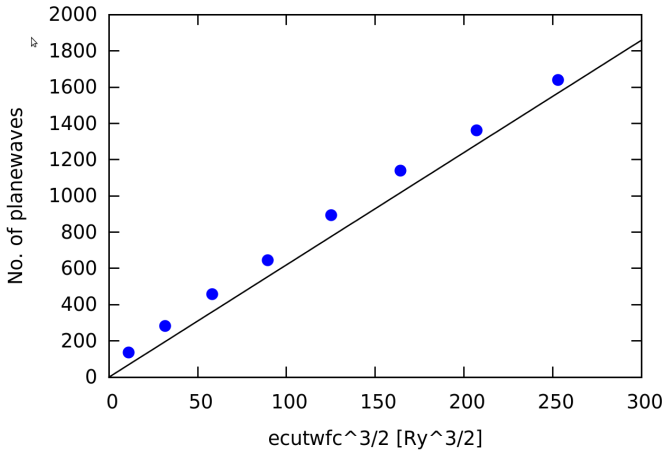
Here we see that, using a cutoff of 25 Ry, we obtain a total energy which is only 12 meV/atom higher than our best-converged value at 40 Ry. In principle we should test even higher values of `ecutwfc` (the exact result corresponds to the limit of infinite cutoff), but in practice 25 Ry is good enough for this tutorial. Most quantities that can be computed using DFT depend critically on this cutoff, therefore we should always perform this test when running DFT calculations. In general, different properties (total energy, equilibrium structure, band structures, vibrations, etc.) will exhibit a slightly different convergence behavior, therefore it is *very important* to always check that a given property is converged with respect to the planewaves cutoff.

Since the planewaves cutoff is so important, can we not use a very large value to be on the safe side? The answer is that the higher the cutoff, the more time-consuming the calculation. You can test this directly by plotting the CPU time vs. the cutoff using the values inside the file `exercise2.txt`.
In this example the runtime is below 1 sec, therefore we can use a very high cutoff without problems. However, in most DFT calculations a careful choice of cutoff can save us weeks of computer time.

The longer times required for higher cutoffs relate to the fact that we are performing linear algebra operations using larger arrays to describe the electron wavefunctions.

▶ Verify that the number of planewaves increases with the cutoff.

▶ Verify that a plot of the number of planewaves vs. $\texttt{ecutwfc}^{3/2}$ yields approximately a straight line:



▶ Can you explain the origin of this relation between the cutoff and the number of planewaves? (To answer this question you will need to look at Lecture 2.2).

Note: If you want to plot your data using `gnuplot`, here is the standard command line:

```
$ gnuplot
> plot "exercise2.txt" u 1:3 w lp pt 6 ps 2 lc 3
```

Here `u 1:3` indicates that we want to plot column 1 as x-coordinate and column 2 as y-coordinate. `w lp` means that we want lines and points, `pt` and `ps` are the type and size of the points, respectively, and `lc 3` sets the line color to blue.

More information about `gnuplot` can be found at http://gnuplot.sourceforge.net/demo_cvs/simple.html

Exercise 3

We now want to explore one other convergence parameter of DFT calculations for crystals, the Brillouin zone sampling K_POINTS. This is important for calculations on extended (periodic) systems, e.g. crystals.

In the input file `silicon-3.in` we had requested a uniform sampling of Bloch wavevectors **k** by setting `4 4 4 1 1 1`. This means that we want to slice the Brillouin zone in a $4\times4\times4$ grid, and we shift this grid by half a grid spacing in each direction (`1 1 1`). This shift is used because it usually provides a better sampling. So now we expect the code to work with exactly $4\times4\times4 = 64$ **k**-vectors.

▶ Now search for '`number of k points`' in the output file `silicon-3.in`. You should find:

```
     number of k points=    10
```

Therefore the code is using only 10 **k**-points instead of the expected 64 points. The reason for this difference is that many points in our grid are equivalent by symmetry. The code recognizes these symmetries and only performs explicit calculations for the inequivalent points.

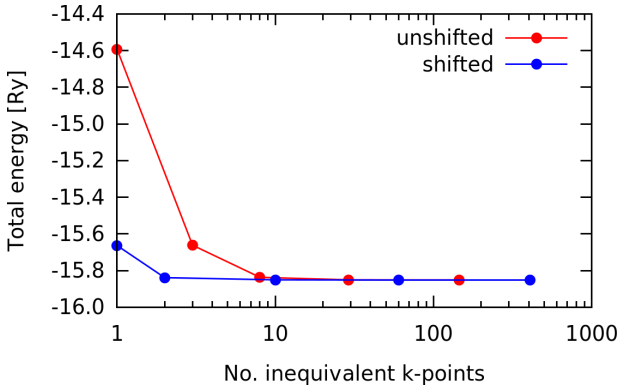
▶ Determine how the total energy of silicon varies with the number of **k**-points, using the same procedure as in Exercise 2. Consider the following situations for the input parameters
K_POINTS: `1 1 1 0 0 0` / `2 2 2 0 0 0` / `4 4 4 0 0 0` / `8 8 8 0 0 0` / `16 16 16 0 0 0`. For these calculations you can use our 'converged' cutoff `ecutwfc = 25.0` Ry.

Note. For these calculations it is convenient to set the execution flag `-npool` inside the command line to 1, `mpirun pw.x -n 12 pw.x -npool 1`. This flag tells the code to distribute the **k**-points among groups of CPUs. For example if we use `-n 12 -npool 4` we are telling the code to use 12 CPUs, and to distribute the **k**-points in 4 groups of 12/4=3 CPUs.
The number of 'pools' cannot be smaller than the total number of **k**-points.

▶ Repeat the last operation, this time using nonzero shifts, eg `1 1 1 1 1 1` / `2 2 2 1 1 1` / `4 4 4 1 1 1` and so on.
You should be able to construct two files similar to the ones below:

```
$ more excercise3a.txt

#  grid     shift    energy (Ry)  inequiv. k-points   time (s)
  1  1  1   0 0 0   -14.59239644     1                 0.08
  2  2  2   0 0 0   -15.66198419     3                 0.15
  4  4  4   0 0 0   -15.83707677     8                 0.23
  8  8  8   0 0 0   -15.85079063    29                 0.56
 16 16 16   0 0 0   -15.85108165   145                 2.34


$ more excercise3b.txt

#  grid     shift    energy (Ry)  inequiv. k-points   time (s)
  1  1  1   1 1 1   -15.66368661     1                 0.12
  2  2  2   1 1 1   -15.83894842     2                 0.10
  4  4  4   1 1 1   -15.85087567    10                 0.26
  8  8  8   1 1 1   -15.85108289    60                 0.88
 16 16 16   1 1 1   -15.85108127   408                 5.64
```

▶ Plot the total energy as a function of the number of inequivalent **k**-points in each calculation, both for the case of the unshifted grid (`0 0 0`) and the shifted grid (`1 1 1`). You should obtain something similar to the following (note the logarithmic scale in the horizontal axis):



Here we can see that by using the `4 4 4 1 1 1` grid we obtain a total energy which is already very good, < 2 meV/atom away from our best value at `8 8 8 1 1 1`. We also see that the shifted grid converges faster (in terms of **k**-point count and CPU time) than the unshifted grid.

▶ Plot the CPU time vs. the number of inequivalent **k**-points and verify that the time scales approximately linearly with the number of such points. Can you explain why this is the case?

Exercise 4

In this exercise we want to explore how the time required to perform DFT calculations scales as a function of system size.

The input file `silicon-1.in` has been modified to generate 5 new input files which you can download and unpack as follows:

```
$ wget http://giustino.materials.ox.ac.uk/tutorial_1.2_exercise_4.tgz
$ tar xfz tutorial_1.2_exercise_4.tgz ; ls tutorial_1.2_exercise_4
```

```
silicon-4.1.in  silicon-4.2.in  silicon-4.3.in  silicon-4.4.in  silicon-4.5.in
```

These files correspond to **supercells** of silicon, containing one primitive unit cell (`silicon-4.1.in`), a $2\times2\times2$ supercell (`silicon-4.2.in`), and so on, up to $5\times5\times5$ primitive unit cells (`silicon-4.5.in`). By looking inside these input files you can check that we have a number of Si atoms ranging from 2 to 250.

▶ Now run `pw.x` using these five different input files, and extract the CPU time in each case. The procedure for running jobs is the same as in the previous exercises.
Your data should look similar to the following. Note that the timing of each job depends on the number of cores, the following data were generated using 12 cores:

```
# atoms   cputime (s)
     2        0.08
    16        0.69
    54        8.04
   128       61.78
   250      349.12
```

▶ Plot the CPU time as a function of the number of atoms. Can you identify a simple law relating these two quantities?

Generally speaking DFT calculations scale with the cube of the number of atoms, $T_{\mathrm{CPU}} = \mathrm{const} \cdot N^3$ (N = number of atoms). This can be verified directly by plotting the above data using the cube of the first column: this plot should give an approximately straight line.
The take-home message here is that if we double the size of our system, then our DFT calculation will require approximately 8 times longer to complete (eg 1 week$\rightarrow$ 8 weeks).

An introduction to density functional theory for experimentalists

Tutorial 2.1

As usual we create a new folder on the HPC cluster:

```
$ cd ~/scratch/summerschool ; mkdir tutorial-2.1 ; cd tutorial-2.1
```

Equilibrium structure of a diatomic molecule

In this tutorial we are going to learn how to calculate the equilibrium structure of simple systems. The formal theory required for these calculations will be discussed in Lecture 3.1, for now we can just use the following rule:

Among all possible structures, the equilibrium structure at zero temperature and zero pressure is found by minimizing the DFT total energy.

The DFT total energy is the same quantity that we have been using during Tutorial 1.1 and Tutorial 1.2 (e.g. when we were doing `grep "\!" silicon-1.out`). This quantity includes all the terms appearing in the electron-ion Hamiltonian, except the kinetic energy of the ions. This quantity is also called the *potential energy surface*.

Let us calculate the equilibrium structure of the Cl_2 molecule.

The Cl_2 molecule has only 2 atoms, and the structure is completely determined by the Cl−Cl bond length. Therefore we can determine the equilibrium structure by calculating the total energy as a function of the Cl−Cl distance.

The first step is to find a suitable pseudopotential for Cl. As in Tutorial 1.1 we go to http://www.quantum-espresso.org/pseudopotentials/original-qe-pp-library and look for Cl. We recognize LDA pseudopotential by the label pz in the filename. Let us go for the following:

```
$ wget http://www.quantum-espresso.org/upf_files/Cl.pz-bhs.UPF
```

We also copy the executable and the input file from the previous tutorial:

```
$ cp ../tutorial-1.1/pw.x ./
$ cp ../tutorial-1.1/silicon-1.in ./cl2.in
```

Now we modify the input file in order to consider the Cl_2 molecule:

```
$ more cl2.in
```

```
&control
 calculation = 'scf'
 prefix = 'Cl2',
 pseudo_dir  = './',
 outdir = './'
/
```

```
&system
 ibrav = 1,
 celldm(1) = 20.0,
 nat = 2,
 ntyp = 1,
 ecutwfc = 100,
/
&electrons
/
ATOMIC_SPECIES
 Cl  1.0  Cl.pz-bhs.UPF
ATOMIC_POSITIONS bohr
 Cl 0.00 0.00 0.00
 Cl 2.00 0.00 0.00
K_POINTS gamma
```

Using `ibrav = 1` we select a simulation box which is simple cubic, with lattice parameter `celldm(1)`. Here we are choosing a cubic box of side 20 bohr (1 bohr = 0.529167 Å). The keyword `gamma` means that we will be sampling the Brillouin zone at the Γ point, that is $\mathbf{k} = 0$. This is fine since we want to study a molecule, not an extended crystal. Note that we increased the planewaves cutoff to 100 Ry: this number was obtained from separate convergence tests.

We can now execute `pw.x` in order to check that everything will run smoothly:
`mpirun -n 12 pw.x < Cl2.in > Cl2.out`.

Incidentally, from the output file we can see the various steps of the DFT self-consistent cycle (SCF). For example if we look for the total energy:

```
$ grep "total energy" Cl2.out
     total energy              =     -55.44319995 Ry
     total energy              =     -55.73171512 Ry
     total energy              =     -55.83944540 Ry
     total energy              =     -55.84156819 Ry
     total energy              =     -55.84161959 Ry
!    total energy              =     -55.84162097 Ry
     The total energy is the sum of the following terms:
```

Here we see that the energy reaches its minimum in 6 iterations. The iterative procedure stops when the energy difference between two successive iterations is smaller than `conv_thr = 1.0d-6`.

Now we proceed to calculate the total energy as a function of the Cl−Cl bond length. In the reference frame chosen for the above input file, we have one Cl atom at (0,0,0) and one at (2,0,0) in atomic units. Therefore we can vary the bond length by simply displacing the second atom along the x axis. In order to automate the procedure we can use the following script:

```
$ cat > loop2.tcsh << EOF
sed "s/2.00/NEW/g" Cl2.in > tmp
foreach DIST ( 2.2 2.4 2.6 2.8 3.0 3.2 3.4 3.6 3.8 4.0 4.2 4.4 4.6 )
  sed "s/NEW/\${DIST}/g" tmp > Cl2_\${DIST}.in
end
EOF
$ tcsh loop2.tcsh
```

This script generates identical files which will differ only by the Cl−Cl bond length:

```
$ ls Cl2_*
Cl2_2.2.in  Cl2_2.4.in  Cl2_2.6.in  Cl2_2.8.in  Cl2_3.0.in  Cl2_3.2.in
Cl2_3.4.in  Cl2_3.6.in  Cl2_3.8.in  Cl2_4.0.in  Cl2_4.2.in  Cl2_4.4.in
Cl2_4.6.in
```

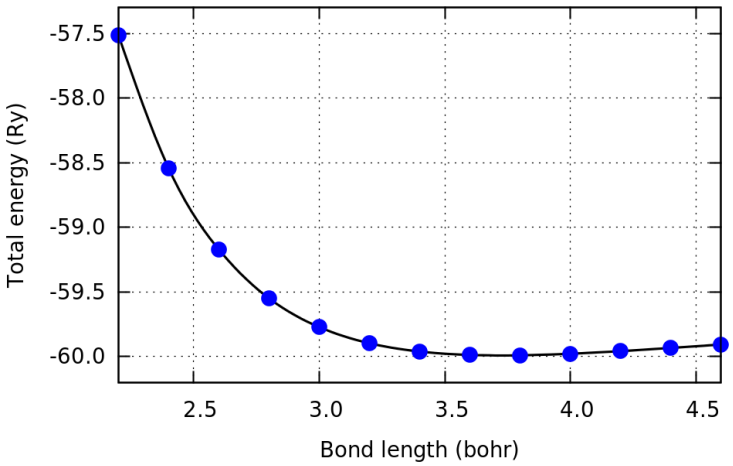
At this point we can execute `pw.x` for each of these input files:

```
mpirun -n 12 pw.x < Cl2_2.2.in > Cl2_2.2.out
...
mpirun -n 12 pw.x < Cl2_4.6.in > Cl2_4.6.out
```

After running these calculations we can look for the total energy using `grep`:

```
$ grep "\!" Cl2_*.out
Cl2_2.2.out:!    total energy              =     -57.51390374 Ry
Cl2_2.4.out:!    total energy              =     -58.54440028 Ry
Cl2_2.6.out:!    total energy              =     -59.17256598 Ry
Cl2_2.8.out:!    total energy              =     -59.55021726 Ry
Cl2_3.0.out:!    total energy              =     -59.77201201 Ry
Cl2_3.2.out:!    total energy              =     -59.89671103 Ry
Cl2_3.4.out:!    total energy              =     -59.96077568 Ry
Cl2_3.6.out:!    total energy              =     -59.98691033 Ry
Cl2_3.8.out:!    total energy              =     -59.98945927 Ry
Cl2_4.0.out:!    total energy              =     -59.97764674 Ry
Cl2_4.2.out:!    total energy              =     -59.95749215 Ry
Cl2_4.4.out:!    total energy              =     -59.93293892 Ry
Cl2_4.6.out:!    total energy              =     -59.90658481 Ry
```

By extracting the bond length and the energy from this data we can obtain the plot shown below:



In this plot the blue dots are the calculated datapoints, and the blue line is a spline interpolation. In `gnuplot` this interpolation is obtained using the flag `smooth csplines` at the end of the `plot` command.

By zooming in the figure we find that the bond length at the minimum is 3.725 bohr = 1.97 Å. The calculated bond length is 1.5% shorter than the measured value 1.99 Å.

Binding energy of a diatomic molecule

The total energy of Cl_2 at the equilibrium bond length can be used to calculate the dissociation energy of this molecule.

The dissociation energy is defined as the difference $E_{\rm diss} = E_{\rm Cl_2} - 2E_{\rm Cl}$, with $E_{\rm Cl}$ the total energy of an isolated Cl atom.

In order to evaluate this quantity we first calculate $E_{\rm Cl_2}$ using the equilibrium bond length determined in the previous section. For this we create a new input file `Cl2-2.in` by copying `Cl.in` and making the following change:

```
...
ATOMIC_POSITIONS bohr
 Cl 0.00 0.00 0.00
 Cl 3.725 0.00 0.00
...
```

A calculation with this modified input file yields the total energy

$E_{\rm Cl_2}$ = -59.99059540 Ry

Next we consider the isolated Cl atom.

The slight complication in this case is that the outermost ($3p$) electronic shell of Cl has one unpaired electron: $\boxed{\uparrow\downarrow}\ \boxed{\uparrow\downarrow}\ \boxed{\uparrow\ }$. In order to take this into account we perform a **spin-polarized** calculation using the following modification of the previous input file:

```
$ cat > Cl.in << EOF
&control
 calculation = 'scf'
 prefix = 'Cl',
 pseudo_dir = './',
 outdir = './'
/
 &system
 ibrav = 1,
 celldm(1) = 20.0,
 nat = 1,
 ntyp = 1,
 ecutwfc = 100,
 nspin = 2,
 tot_magnetization = 1.0,
 occupations = 'smearing',
 degauss = 0.001,
/
&electrons
/
ATOMIC_SPECIES
 Cl 1.0 Cl.pz-bhs.UPF
ATOMIC_POSITIONS
 Cl 0.00 0.00 0.00
K_POINTS gamma
EOF
```

After running `pw.x` with this input file, we obtain a total energy

E_{Cl} = -29.86386034 Ry

By combining the last two results we find

E_{diss} = 0.262875 Ry = 3.58 eV

This result should be compared to the experimental value of 2.51 eV from https://en.wikipedia.org/wiki/Bond-dissociation_energy. We can see that DFT/LDA overestimates the dissociation energy of Cl_2 by about 1 eV: interatomic bonding is slightly too strong in LDA.

Equilibrium structure of a bulk crystal

In this section we study the equilibrium structure of a bulk crystal. We consider again a silicon crystal, since we already studied the convergence parameters in Tutorial 1.2.
Before starting we can clean up the folder by removing all files and subfolders referring to the previous example. We can also copy the pseudopotential from the folder `tutorial-1.2`.

```
$ cd ~/scratch/summerschool/tutorial-2.1 ; rm -rf Cl*
$ cp ../tutorial-1.2/Si.pz-vbc.UPF ./
$
```

In this case the input file with the converged parameters for planewaves cutoff and Brillouin-zone sampling is:

```
$ cat > silicon.in << EOF
&control
 calculation = 'scf'
 prefix = 'silicon',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 2,
 celldm(1) = 10.28,
 nat = 2,
 ntyp = 1,
 ecutwfc = 25.0,
/
&electrons
/
ATOMIC_SPECIES
 Si  28.086  Si.pz-vbc.UPF
ATOMIC_POSITIONS
 Si 0.00 0.00 0.00
 Si 0.25 0.25 0.25
K_POINTS automatic
 4 4 4 1 1 1
EOF
```

In the case of bulk crystals we often have information about the structure from XRD measurements. This information simplifies considerably the determination of the equilibrium structure. For example,

in the case of silicon, the diamond structure is uniquely determined by the lattice parameter a, therefore the minimization of the total energy is really a one-dimensional optimization problem, precisely as for the Cl_2 molecule.

In Tutorial 2.2 we will explore the slightly more complicated situation where we need to decide which one among several possible crystal structures is the most stable.

To find the equilibrium lattice parameter of silicon we perform total energy calculations for a series of plausible parameters. We can generate multiple input files at once by using the following script:

```
$ cat > loop3.tcsh << EOF
sed "s/10.28/NEW/g" silicon.in > tmp
foreach ALAT ( 9.9 10.0 10.1 10.2 10.3 10.4 10.5 10.6 )
sed "s/NEW/\${ALAT}/g" tmp > silicon_\${ALAT}.in
end
EOF
```

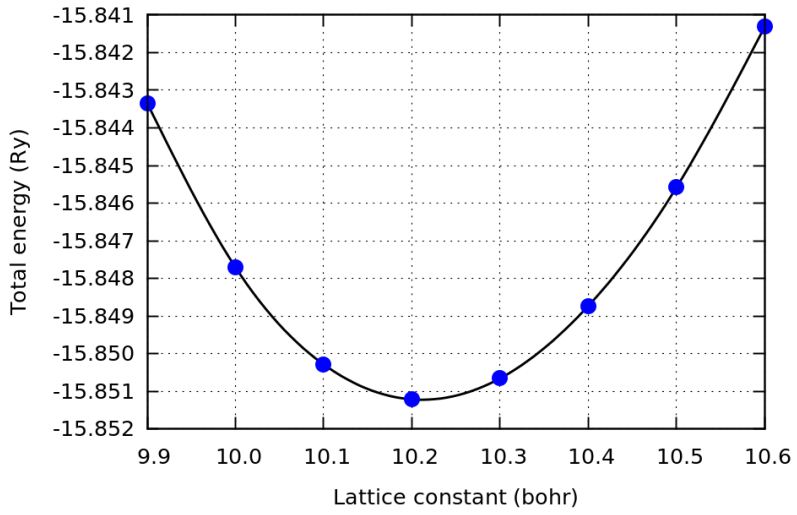
Now we can execute `pw.x` using the generated input files:

```
mpirun -n 12 pw.x -npool 4 < silicon_9.9.in > silicon_9.9.out
...
...
mpirun -n 12 pw.x -npool 4 < silicon_10.6.in > silicon_10.6.out
```

At the end of the execution we should be able to see the output files `silicon_9.9.out`, ··· `silicon_10.6.out`, and we can extract the total energies as follows:

```
$ grep "\!" silicon_*.out
silicon_10.0.out:!    total energy              =     -15.84770895 Ry
silicon_10.1.out:!    total energy              =     -15.85028960 Ry
silicon_10.2.out:!    total energy              =     -15.85121712 Ry
silicon_10.3.out:!    total energy              =     -15.85065978 Ry
silicon_10.4.out:!    total energy              =     -15.84873486 Ry
silicon_10.5.out:!    total energy              =     -15.84558104 Ry
silicon_10.6.out:!    total energy              =     -15.84131398 Ry
silicon_9.9.out:!    total energy              =     -15.84334817 Ry
```

A plot of the total energy vs. lattice parameter is shown below:



Also in this case the blue dots are the calculated datapoints, and the black line is a smooth interpolating function (obtained using 'smooth csplines' in gnuplot).

By zooming near the bottom we see that the equilibrium lattice parameter is $a = 10.2116$ bohr $=$ 5.40 Å. This calculated value is very close to the measured equilibrium parameter of 5.43 Å; DFT/LDA underestimates the measured lattice constant by 0.5%.

Cohesive energy of a bulk crystal

The **cohesive energy** is defined as the heat of sublimation of a solid into its elements.

In practice the calculation is identical to the case of the dissociation energy of the Cl_2 molecule: we need to take the difference between the total energy at the equilibrium lattice parameter and the total energy of each atom in isolation.

For the energy at equilibrium we just repeat one calculation using the same input files as above, this time by setting

```
...
 celldm(1) = 10.2116,
...
```

This calculation yields:

$E_{\text{bulk}} =$ -15.85121828 Ry

(this is the total energy per unit cell, and each unit cell contains 2 Si atoms)

To determine the total energy of one Si atom in isolation we consider a fictitious cubic crystal with a large lattice parameter and one Si atom per unit cell. In this case we need to consider that Si has 4 valence electrons in the configuration $3s^2p^2$. According to Hund's rules the spins in the p shell must be arranged as follows: $\boxed{\uparrow}\ \boxed{\uparrow}\ \boxed{\ }$.
We can modify the input file as follows:

```
$ cat > Si.in << EOF
&control
 calculation = 'scf'
 prefix = 'silicon',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 1,
 celldm(1) = 20,
 nat = 1,
 ntyp = 1,
 ecutwfc = 25.0,
 nspin = 2,
 tot_magnetization = 2.0,
 occupations = 'smearing',
 degauss = 0.001,
/
&electrons
```

```
/
ATOMIC_SPECIES
 Si  28.086  Si.pz-vbc.UPF
ATOMIC_POSITIONS
 Si 0.00 0.00 0.00
K_POINTS gamma
EOF
```

Here the input variable `tot_magnetization = 2` is used to request that the code finds the lowest-energy electronic configuration with two electron spins pointing in the same direction.
The calculation for the isolated atom gives:

E_{Si} = -7.53721924 Ry

By combining the last two results we obtain:

$E_{cohes} = (E_{bulk} - 2E_{Si})/2 = 0.38839$ Ry $= 5.28$ eV

The measured heat of sublimation of silicon is 4.62 eV (see pag. 71 of the DFT book), therefore our DFT/LDA calculation overestimates the experimental value by 14%.
The underestimation of lattice parameters and the overestimation of cohesive energies is quite typical with DFT/LDA. We can summarize these observations by stating that *DFT/LDA tends to overbind* molecules and solids.

Note While the DFT/LDA tends to overbind, another popular approximation to the exchange and correlation functional, the PBE functional [Perdew, Burke, Ernzerhof, PRL 77, 3865 (1996)] tends to *underbind*. For example, DFT/PBE usually yields lattice parameter slightly larger than in experiments.

An introduction to density functional theory for experimentalists

Tutorial 2.2

We create a new folder as usual:

```
$ cd ~/scratch/summerschool; mkdir tutorial-2.2 ; cd tutorial-2.2
```

In this hands-on session we will study the equilibrium structure of simple crystals, namely **silicon** (as in Tutorial 2.1), **diamond**, and **graphite**.

Exercise 1

▶ Familiarize yourself with the steps of Tutorial 2.1, in particular:

1. Calculate the equilibrium lattice parameter of silicon
2. Calculate the cohesive energy of silicon

Exercise 2

In this exercise we will study the equilibrium structure of diamond.

The crystal structure of diamond is almost identical to the one that we used for silicon in Exercise 1. The two important differences are (i) this time we need a pseudopotential for diamond, and (ii) we expect the equilibrium lattice parameter to be considerably smaller than in silicon.

▶ Find a suitable LDA pseudopotential for diamond. It is recommended to use the pseudopotential `C.pz-vbc.UPF`. The link to the pseudopotential library can be found in the PDF document of Tutorial 1.1.

▶ Download this pseudopotential, and copy the executable `pw.x` from `tutorial-2.1` into the current directory. Create an input file for diamond, `diamond.in`, by modifying the input file `silicon-1.in` from Tutorial T1.1. For the time being we can set the lattice parameter to the experimental value, 3.56 Å.

▶ Execute `pw.x` to make sure that everything goes smoothly.

▶ Determine the planewaves kinetic energy cutoff `ecutwfc` required for this pseudopotential. You can generate the input files for various cutoff energies manually, or by using the following script (adapted from pag. 3 of Tutorial 1.2):

```
$ cat > loop-diamond.tcsh << EOF
foreach ECUTWFC ( 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 150 200)
  sed "s/18\\.0/\${ECUTWFC}/g" diamond.in > diamond-\${ECUTWFC}.in
end
EOF
$ tcsh loop-diamond.tcsh
```

You should find that the total energy per atom is converged to within 10 meV when using a cutoff `ecutwfc` = 100 Ry (we take as 'exact' result the value for the highest cutoff considered, 200 Ry).

▶ Using the planewaves cutoff determined in the previous step, determine the equilibrium lattice parameter of diamond, by performing total energy calculations similar to those for silicon in Exercise 1. Compare the calculated lattice parameter with the experimental value.

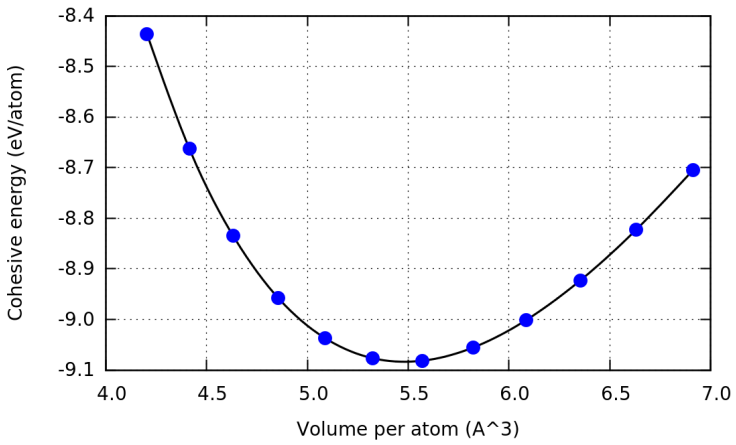
You should find an equilibrium lattice parameter of 6.66404 bohr = 3.53 Å.

▶ Using the equilibrium lattice parameter determined in the previous exercise, calculate the cohesive energy of diamond and compare your value with experiments.
For this calculation we use the same strategy employed in Tutorial 2.1 for silicon. The C atom in its ground state has a valence electronic configuration $2s$ $\boxed{\uparrow\downarrow}$ $2p$ $\boxed{\uparrow}$ $\boxed{\uparrow}$ $\boxed{}$, therefore we need to run a spin-polarized calculation. To this aim we can adapt the input file `Si.in` used in T2.1.

As a reference, the cohesive energy calculated using these settings should be 9.08 eV. The experimental value is 7.37 eV.

▶ Plot the cohesive energy vs. volume/atom for all the lattice parameters that you considered, using units of eV for the energy and Å^3 for the volume.

The plot should look like the following.



Exercise 3

In this exercise we study the equilibrium structure of graphite. We consider the structure of graphite in the Bernal stacking (AB), as obtained from the Inorganic Crystal Structure Database (ICSD).
From the ICSD we know that the unit cell of graphite is hexagonal, with lattice vectors

$$\begin{aligned}\mathbf{a}_1 &= a\,(\quad 1 \quad 0 \quad 0 \quad)\\ \mathbf{a}_2 &= a\,(\quad -1/2 \quad \sqrt{3}/2 \quad 0 \quad)\\ \mathbf{a}_3 &= a\,(\quad 0 \quad 0 \quad c/a \quad)\end{aligned}$$

(a =2.464 Å and $c/a = 2.724$), and with 4 C atoms per primitive unit cell, with fractional coordinates:

$$\begin{aligned}C_1 &: (\quad 0 \quad 0 \quad 1/4 \quad)\\ C_2 &: (\quad 0 \quad 0 \quad 3/4 \quad)\\ C_3 &: (\quad 1/3 \quad 2/3 \quad 1/4 \quad)\\ C_4 &: (\quad 2/3 \quad 1/3 \quad 3/4 \quad)\end{aligned}$$

▶ Starting from the input file that you used for diamond in Exercise 2, build an input file for calculating the total energy of graphite, using the experimental crystal structure given above.

Here you will need to pay attention to the entries `ibrav` and `celldm()` in the input file. Search for these entries in the documentation page:
https://www.quantum-espresso.org/Doc/INPUT_PW.html
Here you should find the following:

Namelist: &SYSTEM

ibrav INTEGER
Status: REQUIRED

Bravais-lattice index. If ibrav /= 0, specify EITHER
[celldm(1)-celldm(6)] OR [A, B, C, cosAB, cosAC, cosBC]
but NOT both. The lattice parameter "alat" is set to
alat = celldm(1) (in a.u.) or alat = A (in Angstrom);
see below for the other parameters.
For ibrav=0 specify the lattice vectors in CELL_PARAMETERS,
optionally the lattice parameter alat = celldm(1) (in a.u.)
or = A (in Angstrom), or else it is taken from CELL_PARAMETERS

ibrav	structure	celldm(2)-celldm(6) or: b,c,cosbc,cosac,cosab
0	free crystal axis provided in input: see card CELL_PARAMETERS	
1	cubic P (sc) v1 = a(1,0,0), v2 = a(0,1,0), v3 = a(0,0,1)	
2	cubic F (fcc) v1 = (a/2)(-1,0,1), v2 = (a/2)(0,1,1), v3 = (a/2)(-1,1,0)	
3	cubic I (bcc) v1 = (a/2)(1,1,1), v2 = (a/2)(-1,1,1), v3 = (a/2)(-1,-1,1)	
-3	cubic I (bcc), more symmetric axis: v1 = (a/2)(-1,1,1), v2 = (a/2)(1,-1,1), v3 = (a/2)(1,1,-1)	
4	Hexagonal and Trigonal P v1 = a(1,0,0), v2 = a(-1/2,sqrt(3)/2,0), v3 = a(0,0,c/a)	celldm(3)=c/a

Based on this information we must use `ibrav = 4` and set `celldm(1)` to a, `celldm(3)` to c/a.

As a sanity check, if you run a calculation with `ecutwfc = 100` and `K_POINTS gamma` you should obtain a total energy of -44.581847 Ry.

▶ After performing convergence test with respect to the number of **k**-points, we found that the total energy is converged to 4 meV/atom when using a shifted $6\times6\times2$ grid (`6 6 2 1 1 1` with `K_POINTS automatic`).

Using this setup for the Brillouin-zone sampling, calculate the lattice parameters of graphite a and c/a at equilibrium. Note that this will require a minimization of the total energy in a **two-dimensional** parameter space.

For this calculation it is convenient to automatically generate input files as follows, assuming that your input file is called `graphite.in`:

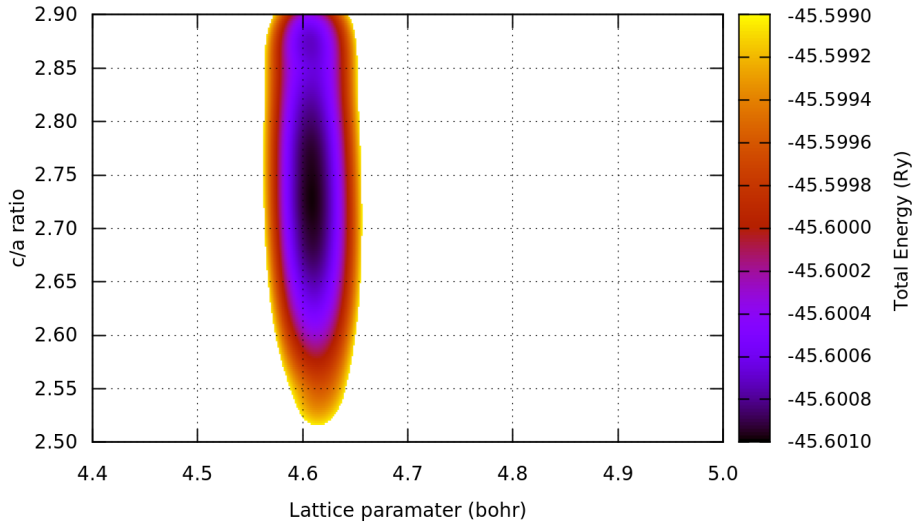
- Replace the values of `celldm(1)` and `celldm(3)` by the placeholders `ALAT` and `RATIO`, respectively;
- Create a script `graphite.tcsh` as follows:

```
$ cat > graphite.tcsh << EOF
foreach A ( 4.4 4.5 4.6 4.7 4.8 4.9 5.0 )
foreach CBYA ( 2.50 2.55 2.60 2.65 2.70 2.75 2.80 2.85 2.90 )
 sed "s/ALAT/\${A}/g" graphite.in > tmp
 sed "s/RATIO/\${CBYA}/g" tmp > graphite_\${A}_\${CBYA}.in
```

```
 mpirun -n 12 pw.x -npool 12 < graphite_\${A}_\${CBYA}.in > graphite_\${A}_\${CBYA}.out
end
end
EOF
```

- By running `tcsh graphite.tcsh` you will be able to generate input files for all these combinations of a and c/a, execute `pw.x` for each file, and store the output in the corresponding `.out` files. Note that this will produce $7 \times 9 = 63$ input files, but the total execution time on 12 cores should be around 1 min.
- At the end you will be able to extract the total energies by using
  `grep "\!" graphite_*_*.out > graphite.txt`

If you plot the total energies that you obtained as a function of a and c/a you should be able to get something like the following:



This plot was generated using the following commands in `gnuplot` (the file `graphite.txt` must first be cleaned up in order to obtain only three columns with the values of a, c/a, and energy):

```
set dgrid3d splines 100,100
set pm3d map
splot  [] [] [:-45.599] "graphite.txt"
```

The '`splines`' keyword provides a smooth interpolation between our discrete set of datapoints. The plotting range along the energy axis is restricted in order to highlight the location of the energy minimum.
Here we see that the energy minimum is very shallow along the direction of the c/a ratio, while it is very deep along the direction of the lattice parameter a. This corresponds to the intuitive notion that the bonding in graphite is very strong within the carbon planes, and very weak in between planes.
By zooming in a plot like the one above you should be able to find the following equilibrium lattice parameters:

$a = 2.439$ Å, $c/a = 2.729$

From these calculations we can see that the agreement between DFT/LDA and experiments for the structure of graphite is excellent. This result is somewhat an artifact: most DFT functionals cannot

correctly predict the interlayer binding in graphite due to the lack of van der Waals corrections. Since LDA generally tends to overbind (as we have seen in all examples studied so far), but it does not contain van der Waals corrections, this functional works well for graphite owing to a cancellation of errors.

For future reference let us just note that the total energy calculated using these optimized lattice parameters is -45.60104546 Ry.

Exercise 4

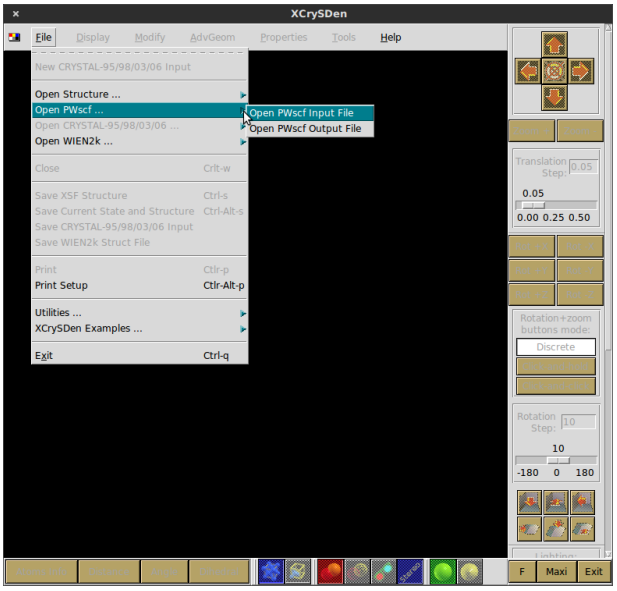
In this excercise we want to see how the structure of graphite that we are using in our input file looks like in a ball-and-stick model.

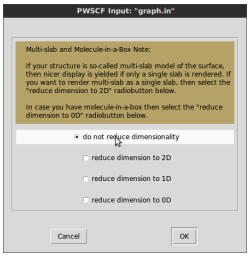
The software `xcryden` can import QE input files and visualize the atomistic structures. General info about this project can be found at `http://www.xcrysden.org`.

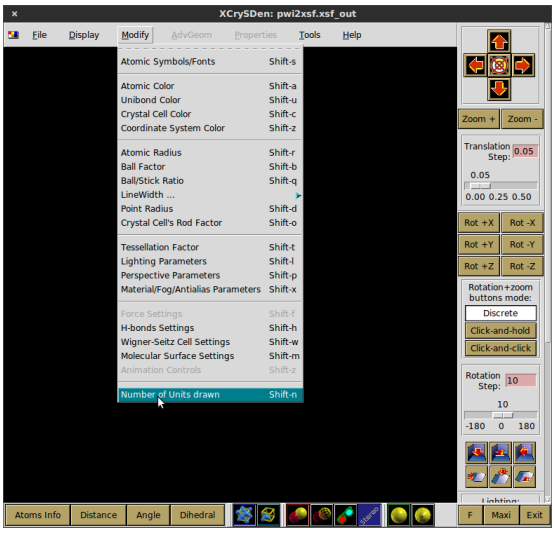
We launch `xcryden` by typing:

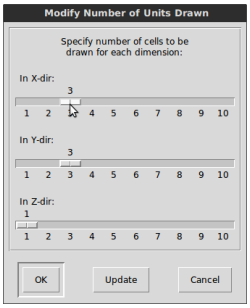
```
$ xcrysden
```

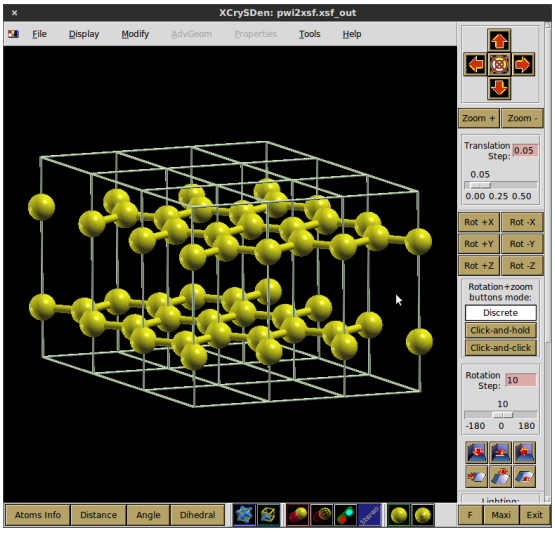
The user interface is very simple and intuitive. The following snapshots may be helpful to get started with the visualization.











Exercise 5

▶ Which carbon allotrope is more stable at ambient conditions, diamond or graphite?

Note The answer to this question is rather delicate. In nature graphite is more stable than diamond by 40 meV/atom at ambient pressure and low temperature.

Using the calculations of Exercises 2 and 3 we find that the cohesive energy of diamond is 8 meV lower than in graphite. Therefore DFT/LDA would predict diamond to be more stable, contrary to experiments. Our result is in agreement with the following stuy by Janotti et al, http://dx.doi.org/10.1103/PhysRevB.64.174107.

An introduction to density functional theory for experimentalists

Tutorial 3.1

We begin with a new folder as usual:

```
$ cd ~/scratch/summerschool ; mkdir tutorial-3.1 ; cd tutorial-3.1
```

In the first part of this tutorial we will say more on the determination of equilibrium structures of molecules and solids. In the second part we will attempt a calculation of elastic properties.

Automatic optimization of atomic coordinates

In Tutorial 2.1 we discussed how to calculate the potential energy surface of a molecule, and how to determine equilibrium structures by locating the minima of that surface.

In this section we consider an alternative route for finding the equilibrium geometry of Cl_2.

To begin with we copy over the input files that we used for the exercise on Cl_2 during Tutorial 2.1:

```
$ cp ../tutorial-2.1/Cl2.in ./
$ cp ../tutorial-2.1/pw.x ./
$ cp ../tutotial-2.1/Cl.pz-bhs.UPF ./
```

We check that everything is in place and that everything goes smoothly by performing a test run:

```
$ mpirun -n 12 pw.x < Cl2.in > Cl2.out
```

If everything is still fine, we should find `Cl2.out` in the current folder, indicating that the run completed successfully.

Now let us take a look at the documentation of `pw.x`. As a reminder we need to go to:
https://www.quantum-espresso.org/Doc/INPUT_PW.html

We look for the input variable `calculation`; we should find the following:

Namelist: &CONTROL

calculation	CHARACTER
Default:	'scf'

A string describing the task to be performed. Options are:

- 'scf'
- 'nscf'
- 'bands'
- 'relax'
- 'md'
- 'vc-relax'
- 'vc-md'

(vc = variable-cell).

Until now we have used only one type of calculation, namely `calculation = 'scf'`. This means that we required the code to perform only a self-consistent DFT calculation with the ions **clamped** at the coordinates specified below the keyword `ATOMIC_POSITIONS`.

Another possibility is to set this variable to `relax` inside `Cl2.in`:

```
&control
 calculation = 'relax'
 prefix = 'Cl2',
 ...
```

This choice instructs `pw.x` to automatically determine the equilibrium structure, starting from the coordinates given below the keyword `ATOMIC_POSITIONS`. In practice the code calculates the **forces** acting on the ions, and updates the ionic positions in such a way as to minimize those forces. The equilibrium configuration will correspond to the situation where all forces are smaller than a given threshold, and where the total potential energy has changed less than a given threshold with respect to the previous iteration.

The threshold on the forces is given by the variable `forc_conv_thr`:

forc_conv_thr	REAL
Default:	1.0D-3

Convergence threshold on forces (a.u) for ionic minimization:
the convergence criterion is satisfied when all components of
all forces are smaller than forc_conv_thr.
See also etot_conv_thr - both criteria must be satisfied

The threshold on the total potential energy is specified by the variable `etot_conv_thr`:

etot_conv_thr	REAL
Default:	1.0D-4

Convergence threshold on total energy (a.u) for ionic
minimization: the convergence criterion is satisfied
when the total energy changes less than etot_conv_thr
between two consecutive scf steps. Note that etot_conv_thr
is extensive, like the total energy.
See also forc_conv_thr - both criteria must be satisfied

In principle we could perform calculations without specifying these parameters; in this case `pw.x` will use some preset default values. For the sake of completeness let us specify some rather stringent criteria in the input file `Cl2.in`:

```
&control
 calculation = 'relax'
 prefix = 'Cl2',
 forc_conv_thr = 1.d-5,
 etot_conv_thr = 1.d-8,
 ...
```

When we perform the automatic optimization of the atomic coordinates we also need to add a 'card' for the ions, as follows:

```
...
&electrons
/
&ions
/
ATOMIC_SPECIES
..
```

This extra card is to specify runtime parameters, but we can leave it empty for the time being.

As a starting point for the atomic coordinates let us use a very large Cl-Cl separation:

```
...
ATOMIC_POSITIONS bohr
 Cl 0.00 0.00 0.00
 Cl 5.00 0.00 0.00
...
```

We now execute `pw.x` and look at the output file `Cl2.out` directly using `vi`. As a reminder, in order to search for a word in `vi` we simply press `/` and type the word. We search for '`Forces`' and obtain:

```
     Forces acting on atoms (cartesian axes, Ry/au):

     atom    1 type  1   force =     0.12373290    0.00000000    0.00000000
     atom    2 type  1   force =    -0.12373290    0.00000000    0.00000000

     Total force =     0.174985     Total SCF correction =     0.000552
```

These lines are telling us that, in the initial configuration, the two Cl atoms experience forces directed along the Cl-Cl axis, and pointing towards the other Cl atom. This was to be expected since we started with the atoms at a distance much larger than the equilibrium bond length.

Immediately below the forces we see the **updated** atomic positions which will be used at the next iteration:

```
ATOMIC_POSITIONS (bohr)
Cl       0.123732900   0.000000000   0.000000000
Cl       4.876267100   0.000000000   0.000000000
```

Clearly the atoms are being displaced towards each other. At the end of the iterations we can see something like the following:

```
     Forces acting on atoms (cartesian axes, Ry/au):

     atom    1 type  1   force =    -0.00000486    0.00000000    0.00000000
     atom    2 type  1   force =     0.00000486    0.00000000    0.00000000

     Total force =     0.000007     Total SCF correction =     0.000115
...
     Final energy   =     -59.9905957134 Ry
Begin final coordinates
```

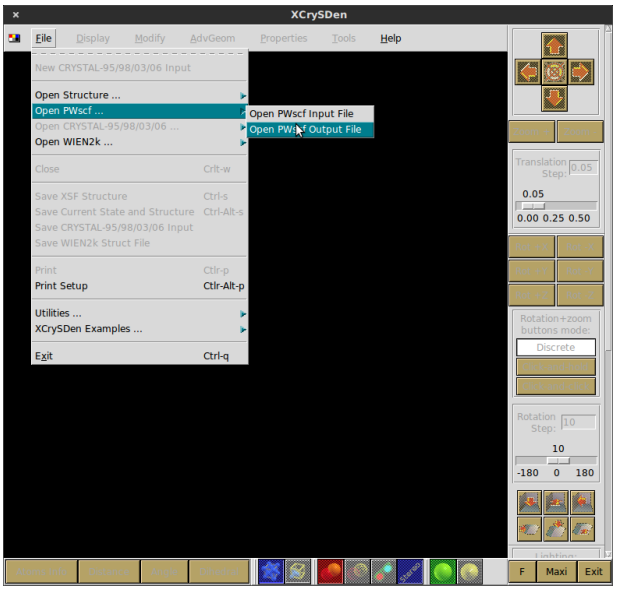
```
ATOMIC_POSITIONS (bohr)
Cl       0.637708469   0.000000000   0.000000000
Cl       4.362291531  -0.000000000   0.000000000
End final coordinates
```

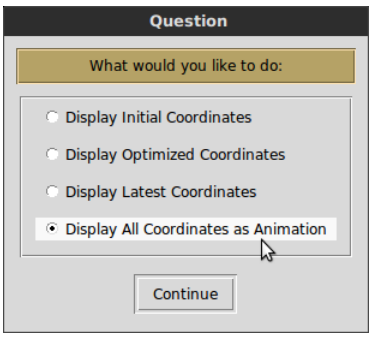
Here we see that the forces are practically vanishing, therefore we reached the equilibrium configuration. The total energy at equilibrium is -59.9905957134 Ry, and the bond length is 3.7246 bohr = 1.97 Å. These values are in agreement with what we had found in Tutorial 2.1 (pag. 3) by explicitly looking for the minimum of the potential energy surface.

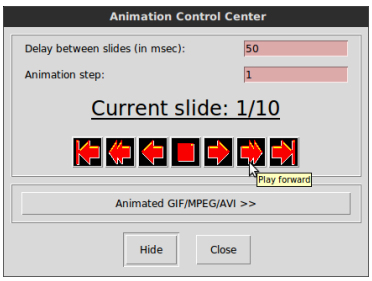
If we want to see how the atomic coordinates evolved towards the equilibrium configuration, we can simply issue:

```
$ grep "Cl" Cl2.out
```

There is also a way to directly visualize the evolution of the atomic coordinates: we can open `xcrysden` and go through the following steps:







Automatic optimization of atomic coordinates and unit cell

In addition to the optimization of atomic coordinates, it is also possible to optimize the vectors of the primitive unit cell. This feature is activated in `pw.x` by setting the calculation type to `vc-relax`:

Namelist: &CONTROL

calculation	CHARACTER
Default:	'scf'

A string describing the task to be performed. Options are:

- 'scf'
- 'nscf'
- 'bands'
- 'relax'
- 'md'
- 'vc-relax'
- 'vc-md'

(vc = variable-cell).

Let us consider the case of **graphite** as in Tutorial 2.2. We can copy over the corresponding input file and the carbon pseudopotential:

```
$ cp ../tutorial-2.2/graphite.in ./
$ cp ../tutorial-2.2/C.pz-vbc.UPF ./
```

Let us modify the input file is such a way as to start from a highly-compressed unit cell of graphite:

```
&control
 calculation = 'vc-relax'
 prefix = 'graphite',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 4,
 celldm(1) = 4.0,
 celldm(3) = 2.0,
 nat = 4,
 ntyp = 1,
 ecutwfc = 100,
/
&electrons
/
&ions
/
&cell
/
ATOMIC_SPECIES
 C 1.0 C.pz-vbc.UPF
ATOMIC_POSITIONS crystal
 C 0.00 0.00 0.25
 C 0.00 0.00 0.75
 C 0.333333 0.666666 0.25
```

```
 C 0.666666 0.333333 0.75
K_POINTS automatic
6 6 2 1 1 1
```

In the above input file we should note the 'cards' `&ions` and `&cell` which are required when running this kind of calculation. In this file the cards are left empty; generally they can be used to fine-tune the optimization procedure.

When instructed to execute a calculation of type `vc-relax`, `pw.x` evaluates the **stress tensor** of the system in the initial configuration, and updates the unit cell vectors so as to reduce the stress.

Let us execute `pw.x` using the above input file. At the end of the run we can look inside the output file using `vi` and search for the following words:

`/` `stress`

We will see something like:

```
     Computing stress (Cartesian axis) and pressure

          total   stress  (Ry/bohr**3)                   (kbar)     P= 4044.41
   0.03445748  -0.00000000   0.00000000       5068.87     -0.00      0.00
  -0.00000000   0.03445748  -0.00000000         -0.00   5068.87     -0.00
   0.00000000  -0.00000000   0.01356517          0.00     -0.00   1995.50
```

This indicates that, as expected, in the first iteration the system is under very high pressure, precisely 4.04 Mbar. Following this initial iteration, `pw.x` modifies the lattice vectors in the direction of lower pressure.

We can note that in this case the **forces** acting on each atom are all vanishing by symmetry:

```
     Forces acting on atoms (cartesian axes, Ry/au):

     atom    1 type  1   force =     0.00000000    0.00000000    0.00000000
     atom    2 type  1   force =     0.00000000    0.00000000    0.00000000
     atom    3 type  1   force =     0.00000000    0.00000000    0.00000000
     atom    4 type  1   force =     0.00000000    0.00000000    0.00000000
```

As a result this procedure will not modify the Wickoff positions of the 4 C atoms in the unit cell. The final optimized structure is found by looking for

`/` `Begin final coordinates`

```
Begin final coordinates
     new unit-cell volume =    200.62004 a.u.^3 (    29.72882 Ang^3 )
     density =      0.22342 g/cm^3

CELL_PARAMETERS (alat=  4.00000000)
   1.149453897   0.000000000   0.000000000
  -0.574726949   0.995456275   0.000000000
   0.000000000   0.000000000   2.739558552
```

```
ATOMIC_POSITIONS (crystal)
C       0.000000000   0.000000000   0.250000000
C       0.000000000   0.000000000   0.750000000
C       0.333333000   0.666666000   0.250000000
C       0.666666000   0.333333000   0.750000000
End final coordinates
```

In this output file we should note that the lattice vectors are given in units of the original lattice parameter in input, that is `alat = 4.0` bohr. Therefore the optimized lattice parameter is now

$a = 4.00000000 \cdot 1.149453897 = 4.59782\,\text{bohr} = 2.433\,\text{Å}$

The optimized c/a ratio is

$c/a = 2.739558552/1.149453897 = 2.384.$

It is immediate to see that the lattice vectors correspond to an hexagonal lattice; for example the second line of `CELL_PARAMETERS` is $a(-1/2, \sqrt{3}/2, 0)$.

The total energy in the optimized configuration is -45.59331698 Ry.

Note. From this calculation we have obtained a c/a ratio which is much smaller than the one determined in Tutorial 2.2 by studying the potential energy surface (2.383 here vs. 2.729 in T2.2). The interlayer separation is approximately 15% shorter in the present calculation. This result is a calculation artifact. What is happening here is that the code modifies the crystal structure so as to minimize the energy. Now, the Hamiltonian describing the system is expressed in a basis of planewaves, and the wavevectors of these planewaves along the c axis are multiples of $2\pi/c$. If, during the optimization, the c parameter undergoes a significant change (as it is the case here, since we start from $c/a = 2$ and we end up with $c/a = 2.729$), then we are effectively reducing our planewaves cutoff at each iteration. As a result the calculation becomes less and less accurate. In order to avoid this problem, `pw.x` performs one additional calculation at the very end, after having redefined all the reciprocal lattice vectors according to the optimized structure. We can see that this step yields a residual pressure of 92.7 kbar along the c-axis. This indicates that the structure is not yet fully optimized. In order to avoid this problem we should run a new calculation, starting from the latest lattice parameters.

Elastic constants of diamond

Now we want study the elastic constants of diamond. As we have seen in Lecture 3.2, for a cubic system like diamond there are only three independent elastic constants, namely C_{11}, C_{12}, and C_{44}.

We can determine these constants by using the relations discussed in that lecture. In particular:

- We consider an **isotropic** deformation of the diamond structure. This corresponds to a uniform stretch of the lattice vectors:

  $$\mathbf{a}_i' = (1+\eta)\,\mathbf{a}_i \text{ for } i = 1, 2, 3$$

  The resulting change in the total potential energy from the equilibrium value U_0 is:

  $$U - U_0 = \Omega\,\frac{3}{2}\,(C_{11} + 2C_{12})\,\eta^2 \qquad (1)$$

The calculation of U and U_0 can be performed by using the following input file for diamond (taken from Tutorial 2.2):

```
$ cat > diamond-0.in << EOF
&control
 calculation = 'scf'
 prefix = 'diamond',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 0,
 celldm(1) = 6.66405,
 nat = 2,
 ntyp = 1,
 ecutwfc = 100.0,
/
&electrons
/
ATOMIC_SPECIES
 C 1.0 C.pz-vbc.UPF
ATOMIC_POSITIONS
 C 0.00 0.00 0.00
 C 0.25 0.25 0.25
K_POINTS automatic
 6 6 6 1 1 1
CELL_PARAMETERS
 -0.50 0.00 0.50
  0.00 0.50 0.50
 -0.50 0.50 0.00
EOF
```

In this version of the input file we are specifying that the unit cell vectors are given manually, `ibrav = 0`; these vectors are provided in units of the lattice parameter, `celldm(1)`, after the keywoord `CELL_PARAMETERS`.

By running `pw.x` with the above input file we obtain the total energy in the ground state:

U_0 = -22.80164823 Ry.

Furthermore we can read the volume of the unit cell by searching for: / `volume`

This gives $\Omega = 73.9869$ bohr^3.

Now we can modify this input file in order to set in an isotropic deformation with $\eta = 0.002$ (let us call the new file `diamond-1.in`):

```
$ more diamond-1.in

...
CELL_PARAMETERS
 -0.501 0.000 0.501
  0.000 0.501 0.501
 -0.501 0.501 0.000
```

With this new input file we find the total energy:

U = -22.80159961 Ry

Using Eq. (1) with $\eta = 0.002$ we obtain

$C_{11} + 2C_{12} = 0.109524$ Ry/bohr^3 = 1611.1 GPa 1 Ry/bohr^3 = 14710.5 GPa

- We now consider a **tetragonal** deformation.

From Lecture 3.2 we have the relation:

$$U - U_0 = \Omega\, 3\, (C_{11} - C_{12})\, \eta^2 \tag{2}$$

and the tetragonal distortion of the unit cell can be realized by considering an expansion $(1+\eta)$ along x and y, and a contraction $(1-2\eta)$ along z. We copy the input file `diamond-0.in` into `diamond-2.in` and we modify this new file as follows:

```
$ more diamond-2.in
...
CELL_PARAMETERS
 -0.501 0.000 0.498
  0.000 0.501 0.498
 -0.501 0.501 0.000
```

This calculation gives:

U =-22.80156420 Ry

Using Eq. (2) with $\eta = 0.002$ we obtain

$C_{11} - C_{12}$ =0.0946451 Ry/bohr^3 = 1392.6 GPa

By combining the two relations for C_{12} and C_{12} we obtain:

C_{11} = 1465.4 GPa, C_{12} = 72.8 GPa.

The corresponding experimental values are C_{11} = 1079 GPa, C_{12} = 124 GPa, from McSkimin & Andreatch, J. Appl. Phys. 43, 2944 (1972).

Note. These calculations are not fully converged, and by refining our setup we can obtain better agreement with experiment. In particular, we are determining elastic constants using only 2 calculations in each case. Since elastic constants are second derivatives of the total energy, a much more acurate approach is to evaluate such derivatives using 3 total energy calculations. See Exercise 6.4 of the DFT book for how to perform more refined calculations.

- Finally we can consider a **trigonal** deformation.

From Lecture 3.2 we have the relation:

$$U - U_0 = \Omega\, \frac{1}{2}\, C_{44}\, \eta^2 \tag{3}$$

and the trigonal distortion of the unit cell can be realized by considering the following distortion for $\eta = 0.002$:

```
$ more diamond-3.in

...
CELL_PARAMETERS
 -0.5000 -0.0005 0.5000
  0.0005  0.5000 0.5000
 -0.4995  0.4995 0.0000
```

This calculation gives:

U =-22.80164336 Ry

Using Eq. (3) with $\eta = 0.002$ we obtain

$C_{44} = 0.0329112$ Ry/bohr^3 = 484.1 GPa

The corresponding experimental value is $C_{44} = 578$ GPa [McSkimin & Andreatch, J. Appl. Phys. 43, 2944 (1972)].

Note. These calculations are not fully converged: in order to obtain accurate results we need to use higher-order finite difference formulas. See Exercise 6.4 of the DFT book.

An introduction to density functional theory for experimentalists

Tutorial 3.2

We create a new folder as usual:

```
$ cd ~/scratch/summerschool; mkdir tutorial-3.2 ; cd tutorial-3.2
```

In this hands-on session we will first familiarize ourselves with calculations of elastic constants, using **diamond** as a test case. Then we will try to set up an entirely new calculation on **$SrTiO_3$**; here we will use the Materials Project database to find the initial geometry.

Exercise 1

▶ Calculate the elastic constants C_{11}, C_{12}, and C_{44} of diamond, by following the steps illustrated in Tutorial 3.1.

The bulk modulus B is a measure of the resistance of a materials to hydrostatic compression. This quantity can be obtained from the elastic constants as:

$$B = \frac{1}{3}(C_{11} + 2C_{12}).$$

▶ Calculate the bulk modulus of diamond, using the data obtained in the previous step, and compare your result with experiment.

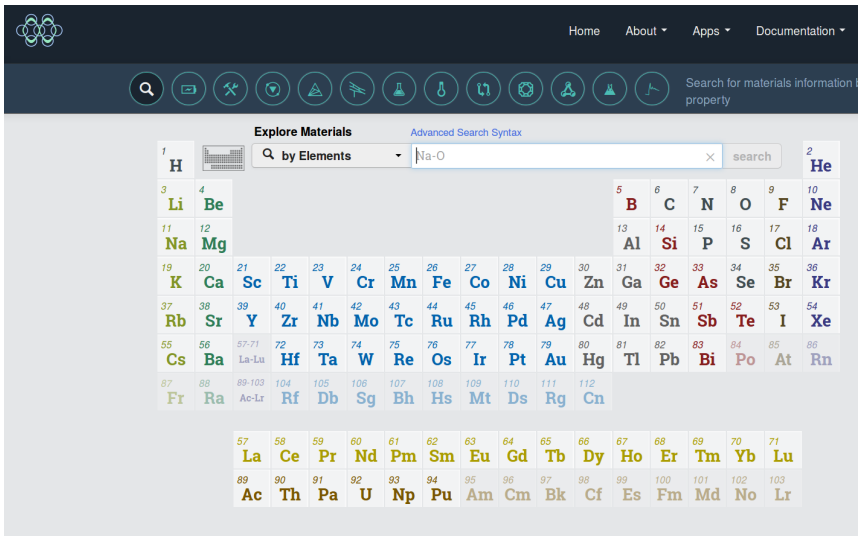
▶ In Tutorial 3.1 we used a 'deformation' parameter $\eta = 0.002$ in all our calculations.
Investigate the **sensitivity** of the calculated bulk modulus of diamond to the choice of η, by repeating the calculations for $\eta = 0.1$, 0.01, and 0.001.

Exercise 2

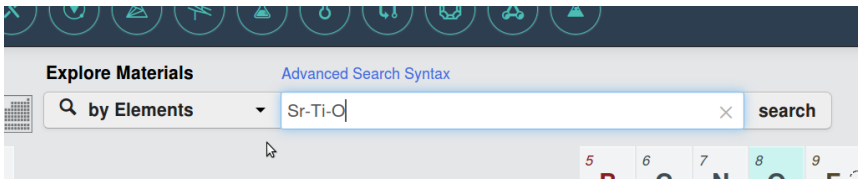
In this exercise we want to set up a simple input file to study $SrTiO_3$. In order to find an initial guess for the unit cell and atomic coordinates, we search the Materials Project database.

If you do not already have an account on the Materials Project, please go to `https://www.materialsproject.org` and click on Sign in or Register. The registration process only requires an email address and a password, this should take less than a minute to complete.

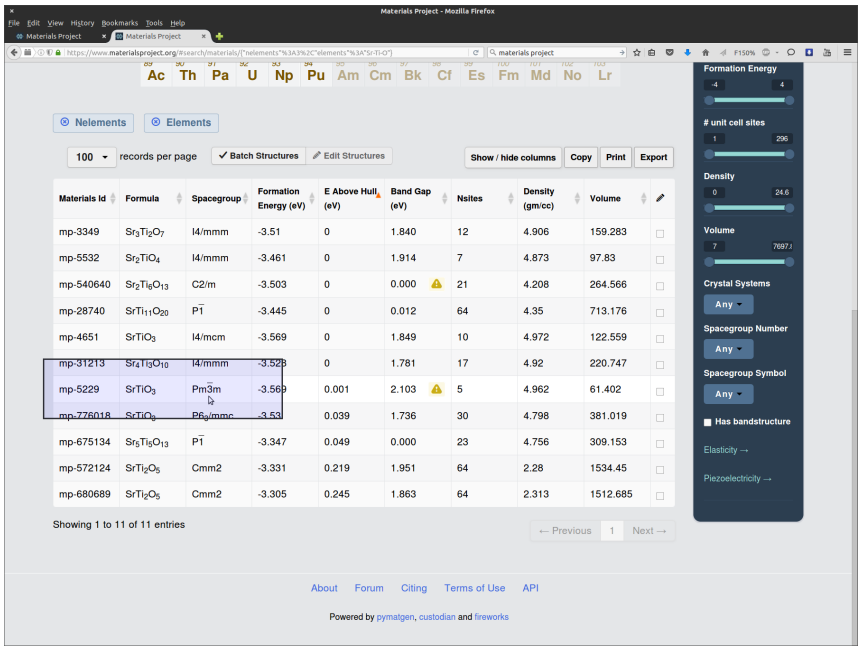
After logging-in you should be able to see a periodic table like the following:



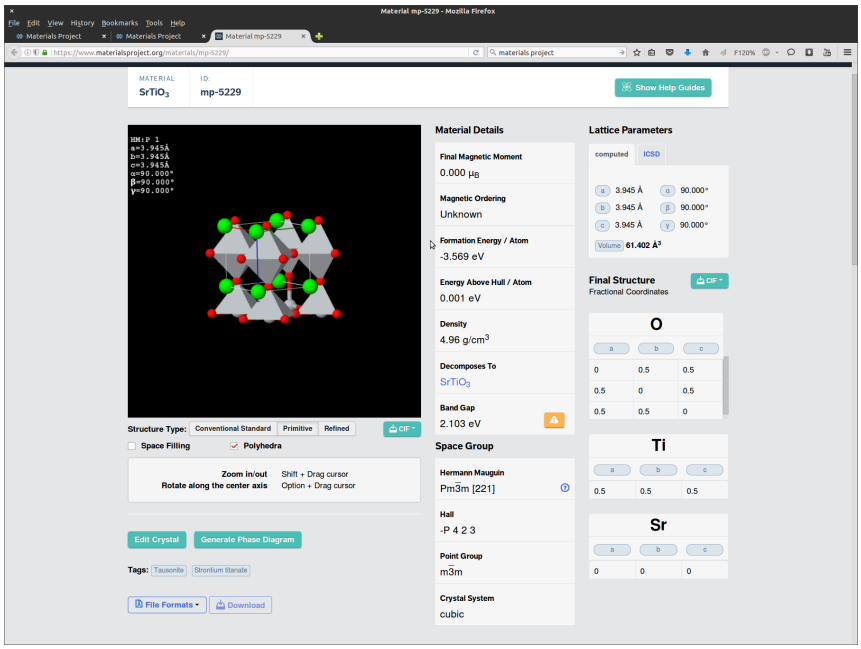
Now we can search for $SrTiO_3$ by entering Sr, Ti, and O in the search bar:



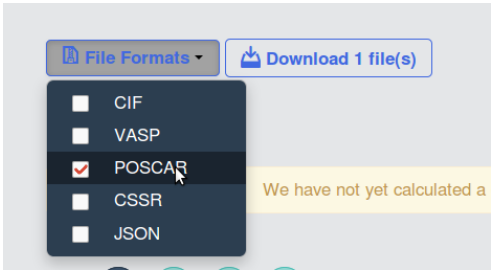
We are looking for the $Pm\bar{3}m$ structure of $SrTiO_3$, which is the high-temperature cubic phase. We will study the cubic phase since it only contains 5 atoms, therefore calculations are relatively easy.



By clicking on the $Pm\bar{3}m$ field we are shown the properties of this structures that have been uploaded in the database:



All we need from this page is the structural data, which can be found in the `poscar` file:



The 'poscar' format is the standard format of VASP, another widely used sofwtare packages for DFT calculations using planewaves and pseudopotentials.

The 'poscar' file thus downloaded should look like the following:

```
1   Sr1 Ti1 O3
2   1.0
3   3.945130 0.000000 0.000000
4   0.000000 3.945130 0.000000
5   0.000000 0.000000 3.945130
6   Sr Ti O
7   1 1 3
8   direct
9   0.000000 0.000000 0.000000 Sr
10  0.500000 0.500000 0.500000 Ti
11  0.500000 0.000000 0.500000 O
12  0.500000 0.500000 0.000000 O
13  0.000000 0.500000 0.500000 O
```

Here the first line is a comment field, the second line contains the lattice parameter a in Å. Lines 3–5 contain the lattice vectors, scaled by the lattice parameter a: $\mathbf{a}_1/a$, $\mathbf{a}_2/a$, $\mathbf{a}_3/a$ (note that in this

example the authors decided to set $a = 1$ so as to give the lattice vectors directly in Å). Line 6 contains the list of atoms in the unit cell, followed by the number of atoms of each type on line 7, in the same order. The keyword 'direct' on line 8 specifies that the atomic coordinates in lines 9-13 are expressed as fractional coordinates (units of 'direct' lattice), eg the Ti atom is at $(\mathbf{a}_1+\mathbf{a}_2+\mathbf{a}_3)/2$.

▶ Construct the input file for a total energy calculation of $SrTiO_3$ using `pw.x`. For the time being we can leave the pseudopotential field blank. You can start by modifying an input file from a previous exercise.
If anything is unclear, please consult the documentation at https://www.quantum-espresso.org/Doc/INPUT_PW.html. Remember that `celldm(1)` is to be given in atomic units. To specify that the atomic positions are in crystal coordinates we use the keyword: `ATOMIC_POSITIONS crystal`.

At this point we need pseudopotentials for Sr, Ti, O. For this exercise we want to use the LDA exchange and correlation functional, therefore we need to identify LDA pseudopotentials. We are looking for pseudos with the label `pz` (Perdew-Zunger) in the filename.
In order to make sure that we all obtain the same results, let us decide that we consider only pseudopotentials generated by Hartwigsen, Goedeker & Hutter. These pseudos can be found here: https://www.quantum-espresso.org/pseudopotentials/hartwigesen-goedecker-hutter-pp

▶ Download the required pseudopotential files from the QE library, using `wget` as in Tutorial 1.1.

▶ Using the input file just created, say `sto-1.in`, perform a test run using `calculation = 'scf'`. This test is only to make sure that everything goes smoothly. For this test we can use some arbitrary convergence parameters, say `ecutwfc = 40` and a Brillouin-zone sampling `4 4 4 1 1 1`.

Exercise 3

Now that we have a basic setup for $SrTiO_3$, we need to perform convergence tests.

▶ Determine the planewaves kinetic energy cutoff that is required to have the total energy converged to within 50 meV/atom. For this calculation you can use the same `K_POINTS` settings as in Exercise 2. As a reminder, we performed a similar operation in Tutorial 1.2, Exercise 2.

▶ Using the cutoff just obtanied, determine the sampling of the Brillouin zone required to have the total energy converged to within 10 meV/atom.
You can find an example of such a test in Tutorial 1.2/Exercise 3.

Exercise 4

▶ Using the convergence parameters obtained in Exercise 3, determine the optimized lattice parameter of $SrTiO_3$ by using a calculation of type `vc-relax` (see Tutorial 3.1 for an example).

In this calculation you will note that the residual pressure at the end of the run is still nonzero. In order to fully optimize the lattice parameter it is convenient to perform one or two additional runs using the optimized parameter as a starting point.

▶ Compare your optimized lattice parameter with the experimental value from Cao et al, PSSA 181, 387 (2000).

Exercise 5

As a sanity check, at the end of Exercise 3 and 4 we should have obtained the following parameters:

```
...
celldm(1) = 7.18720,
ecutfwc = 210,
...
K_POINTS
4 4 4 1 1 1
```

▶ Use these parameters to calculate the bulk modulus of cubic $SrTiO_3$.

Note: In order to obtain reasonably accurate results it is convenient to use Eq. (1) of Tutorial 3.1, after rewriting as follows:

$$B = \frac{1}{3}(C_{11} + 2C_{12}) = \frac{1}{9}\frac{1}{\Omega}\frac{\partial^2 U}{\partial \eta^2} \simeq \frac{1}{9}\frac{1}{\Omega}\frac{U(+\eta) - 2U(0) + U(-\eta)}{\eta^2}$$

This expression shows that we can calculate the bulk modulus by using the second derivative of the total energy with respect to the deformation parameter. The second derivative is then approximated using a finite-difference formula involving 3 points $(+\eta,\, 0,\, -\eta)$.

For your reference, using $\eta = 0.01$ and considering the unit cell volume $\Omega = 360.5033\,\text{bohr}^3$, the value $B = 190$ GPa is obtained.

▶ Following the same lines as in the last step, calculate the elastic constants C_{11} and C_{12} of cubic $SrTiO_3$.

Hint: We already have the bulk modulus, therefore we know that $C_{11} + 2C_{12} = 3B$. What we still need is the difference $C_{11} - C_{12}$, as discussed in Tutorial 3.1. Also in this case we can rewrite Eq. (2) of Tutorial 3.1 by taking the second derivative and then using finite differences:

$$C_{11} - C_{12} \simeq \frac{1}{6}\frac{1}{\Omega}\frac{U(+\eta) - 2U(0) + U(-\eta)}{\eta^2}.$$

In this case we need to perform three calculations corresponding to a tetragonal deformation of the lattice.

As a reference, you should obtain values in the range of $C_{11} = 364$ GPa and $C_{12} = 103$ GPa.

▶ Compare your calculated constants B, C_{11}, and C_{12} with the experimental values of Bell & Ruprecht, Phys. Rev. 129, 90 (1963).
You should find that the deviation from experiment is smaller than 10%.

▶ Can you think of possible strategies to improve your results?

We create a new folder:

```
$ cd ~/scratch/summerschool ; mkdir tutorial-4.1 ; cd tutorial-4.1
```

In this tutorial we will learn how to calculate the vibrational frequencies of molecules and solids, phonon dispersion relations, LO-TO splitting, IR activity, and low-frequency dielectric constants.

Stretching frequency of a diatomic molecule

We start from the simplest possible system, the diatomic molecule Cl_2 studied in Tutorial 2.1.

We copy the pseudopotential and executable from Tutorial T2.1:

```
$ cp ../tutorial-2.1/Cl.pz-bhs.UPF ./
$ cp ../tutorial-2.1/pw.x ./
```

For the input file we use the following, from Tutorial 2.1. This file already contains the optimized geometry and convergence parameters.

```
$ cat > Cl2.in << EOF
&control
 calculation = 'scf'
 prefix = 'Cl2',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 1,
 celldm(1) = 20.0,
 nat = 2,
 ntyp = 1,
 ecutwfc = 100,
/
&electrons
/
ATOMIC_SPECIES
 Cl 1.0 Cl.pz-bhs.UPF
ATOMIC_POSITIONS bohr
 Cl 0.000 0.00 0.00
 Cl 3.725 0.00 0.00
K_POINTS gamma
EOF
```

As usual we perform a test run to make sure that everything goes smoothly:

```
$ mpirun -n 12 pw.x < Cl2.in
```

In Lecture 4.1 we have seen that the vibrational frequency of a diatomic molecule can be calculated using:

$$\omega = \sqrt{\frac{2K}{M}}, \qquad K = \left.\frac{\partial^2 U}{\partial d^2}\right|_{d_0}$$

where M is the mass of the Cl nucleus, U is the total potential energy surface, d is the Cl-Cl distance, and d_0 is the equilibrium bond length.
By approximating the second derivative using finite differences we have:

$$\hbar\omega \simeq \hbar\sqrt{\frac{2}{M}\frac{U(d_0+\delta) - 2U(d_0) + U(d_0-\delta)}{\delta^2}}$$

where δ is a small number, say $\delta = 0.001$ bohr.

We now calculate $U(d_0)$, $U(d_0+\delta)$, and $U(d_0-\delta)$ by creating two new input files where the coordinates of the second Cl atom are modified.
We can do this as usual using `vi`. Alternatively we can use the following strategy, which is slightly faster:

```
$ sed "s/3.725/3.726/g" Cl2.in > Cl2_plus.in
$ sed "s/3.725/3.724/g" Cl2.in > Cl2_minus.in
```

Here we are replacing the distance 3.725 bohr by 3.726 and 3.724, respectively. It is convenient to extract the corresponding total energies from the output files on the fly. This can be done as follows:

```
$ mpirun -np 12 pw.x < Cl2.in | grep "\!" > U0.txt
$ mpirun -np 12 pw.x < Cl2_plus.in | grep "\!" > U_plus.txt
$ mpirun -np 12 pw.x < Cl2_minus.in | grep "\!" > U_minus.txt
```

In these expressions the vertical bar (`|`) 'pipes' the output from the command on the left (`mpirun -np 12 pw.x < Cl2.in`) into the input of the following command (`grep "\!"`). The output of `grep "\!"` is then 'redirected' (`>`) into the file on the right (`U0.txt`).

After executing these commmands we should see the following:

```
$ more U*.txt

::::::::::::::
U0.txt
::::::::::::::
!    total energy              =     -59.99059540 Ry
::::::::::::::
U_minus.txt
::::::::::::::
!    total energy              =     -59.99059536 Ry
::::::::::::::
U_plus.txt
::::::::::::::
!    total energy              =     -59.99059502 Ry
```

At this point we can combine our results, considering that the mass of Cl is 35.45 amu (1 amu = 1822.8885 m_e). We find:

$$\hbar\omega = 69.4\,\text{meV}$$

to be compared to the experimental value of 66.7 meV.

Stretching frequency of a diatomic molecule, using DFPT

The calculation method of the previous section is very general and widely used, however there exists a faster alternative based on density-functional perturbation theory (DFPT).

In DFPT the vibrational frequency is calculated directly by working with the equilibrium structure, using perturbation theory.

In the Quantum Espresso package DFPT for vibrations is implemented in a code named `ph.x`. In order to use this code we need to go back to the root directory `summerschool/q-e-master`, and execute:

```
$ make ph
$ cp bin/ph.x ../tutorial-4.1/
```

We can build a simple input file for Cl_2 as follows:

```
$ cat > Cl2_ph.in << EOF
vibrations of Cl2
&inputph
 prefix = 'Cl2',
 amass(1) = 35.45,
 outdir = './',
 fildyn = 'Cl2.dyn',
/
0.0 0.0 0.0
EOF
```

Here the first line is just a comment field; `prefix` must be the same as that used by `pw.x`; `amass` is the atomic mass in amu (atomic mass units); `fildyn` specifies the output file that will contain the dynamical matrix. The last line specifies that we want a calculation at the Γ point, that is $\mathbf{q} = (0, 0, 0)$. This is appropriate since we are considering an isolated molecule.
In order to execute `ph.x` we first need to calculate the ground state properties of the system using `pw.x`. In this case we must modify the input file `Cl2.in` as follows:

```
$ cat > Cl2_pw.in << EOF
&control
 calculation = 'scf'
 prefix = 'Cl2',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 1,
 celldm(1) = 20.0,
 nat = 2,
 ntyp = 1,
 ecutwfc = 100,
/
&electrons
/
ATOMIC_SPECIES
 Cl 1.0 Cl.pz-bhs.UPF
```

```
ATOMIC_POSITIONS bohr
 Cl 0.000 0.00 0.00
 Cl 3.725 0.00 0.00
K_POINTS tpiba
1
0.0 0.0 0.0 1.0
EOF
```

With this modification `pw.x` is still instructed to calculate wavefunctions at Γ, that is $\mathbf{k} = 0$. The difference between this file and the version that we have used on pag. 1 of this tutorial is that now we are instructing `pw.x` to treat wavefunctions as complex quantities; in the previous version, the keyword `gamma` was instructing the code to treat wavefunctions as real quantities.
The results do not change, but this modification is needed because `ph.x` only recognizes complex wavefunctions.

We can now execute `pw.x` and `ph.x` as usual:

```
$ mpirun -np 12 pw.x < Cl2_pw.in > Cl2_pw.out
$ mpirun -np 12 ph.x < Cl2_ph.in > Cl2_ph.out
```

After completion of this job we should find the file `Cl2.dyn` in our working directory:

```
$ more Cl2.dyn
```

```
Dynamical matrix file
vibrations of Cl2
  1    2  1 20.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
           1  'Cl  '    32310.6983835891
    1    1       0.0000000000       0.0000000000       0.0000000000
    2    1       0.1862500000       0.0000000000       0.0000000000

     Dynamical  Matrix in cartesian axes

     q = (    0.000000000   0.000000000   0.000000000 )

    1    1
  0.39652684  0.00000000    0.00000000  0.00000000    0.00000000  0.00000000
  0.00000000  0.00000000    0.00262702  0.00000000    0.00000000  0.00000000
  0.00000000  0.00000000    0.00000000  0.00000000    0.00262702  0.00000000
    1    2
 -0.42777126  0.00000000    0.00000000  0.00000000    0.00000000  0.00000000
  0.00000000  0.00000000    0.00470897  0.00000000    0.00000000  0.00000000
  0.00000000  0.00000000    0.00000000  0.00000000    0.00470897  0.00000000
    2    1
 -0.42777827  0.00000000    0.00000000  0.00000000    0.00000000  0.00000000
  0.00000000  0.00000000    0.00470985  0.00000000    0.00000000  0.00000000
  0.00000000  0.00000000    0.00000000  0.00000000    0.00470985  0.00000000
    2    2
  0.39665156  0.00000000    0.00000000  0.00000000    0.00000000  0.00000000
  0.00000000  0.00000000    0.00260188  0.00000000    0.00000000  0.00000000
  0.00000000  0.00000000    0.00000000  0.00000000    0.00260188  0.00000000

     Diagonalizing the dynamical matrix

     q = (    0.000000000   0.000000000   0.000000000 )

 **************************************************************************
     freq (    1) =      -3.232057 [THz] =    -107.809807 [cm-1]
 ( -0.707158  0.000000 -0.000373  0.000000 -0.000850  0.000000 )
 ( -0.707055  0.000000  0.000350  0.000000  0.000619  0.000000 )
     freq (    2) =      -0.838883 [THz] =     -27.982116 [cm-1]
 ( -0.000246  0.000000  0.697984  0.000000 -0.107011  0.000000 )
```

```
( -0.000245  0.000000 -0.699763  0.000000  0.108160  0.000000 )
    freq (    3) =      -0.836519 [THz] =     -27.903284 [cm-1]
( -0.000775  0.000000  0.107884  0.000000  0.697896  0.000000 )
( -0.000786  0.000000 -0.107289  0.000000 -0.699851  0.000000 )
    freq (    4) =       1.565695 [THz] =      52.225979 [cm-1]
(  0.000015  0.000000  0.699984  0.000000 -0.106713  0.000000 )
( -0.000003  0.000000  0.698208  0.000000 -0.105548  0.000000 )
    freq (    5) =       1.566887 [THz] =      52.265737) [cm-1]
( -0.000089  0.000000  0.105842  0.000000  0.700073  0.000000 )
( -0.000144  0.000000  0.106421  0.000000  0.698119  0.000000 )
    freq (    6) =      16.617335 [THz] =     554.294618 [cm-1]
( -0.707055  0.000000  0.000014  0.000000  0.000031  0.000000 )
(  0.707158  0.000000  0.000012  0.000000  0.000020  0.000000 )
 **************************************************************************
```

Here the blue lines represent the calculated dynamical matrix: we have 2 atoms and 3 Cartesian coordinates, therefore the size of this matrix is 6×6. The blue lines correspond to precisely 36 numbers, presented as pairs of real and imaginary part.
The numbers in red are the vibrational frequencies obtained by diagonalizing the dynamical matrix.

Here we see that some frequencies are negative. This is only a **convention**, and it is meant to indicate that the diagonalization of the dynamical matrix led to a negative eigenvalue: $\omega^2 < 0$. In these cases the code prints the quantity $-\sqrt{|\omega^2|}$, and the minus sign is just a flag to warn us that something is not right. In other DFT codes you may find the imaginary unit next to these frequencies, eg $107.889470\,\mathbf{i}$.

In this example we were expecting to obtain $\omega = 0$ for 5 modes (3 translations of Cl_2 and 2 rotations), and one high-frequency stretching mode. Clearly this is not the case in the above output file. What is happening here is that our Cl_2 molecule is in a **periodic** supercell, therefore a global rotation of all the molecules must involve some small amount of energy. Furthermore, in these calculations the 3D space is not exactly 'isotropic', because we are using a finite planewaves cutoff. Together these two effect lead to nonzero frequencies in modes 1–5.

These artifacts can be corrected by imposing so-called **acoustic sum rules**. This procedure corresponds to modifying the dynamical matrix in such a way as to make sure that the molecule will not experience any restoring force when translated or rotated. We can perform this operation by calling a **post-processing** program, `dynmat.x`:

```
$ cp ../q-e-master/bin/dynmat.x ./
$ cat > Cl2.dynmat.in << EOF
&input
 fildyn = 'Cl2.dyn',
 asr = 'zero-dim',
/
EOF
$ ./dynmat.x < Cl2.dynmat.in
```

Here `asr` is a flag that instructs the code to impose the acoustic sum rule (a.s.r.). Note that we are executing this small program in serial on the current node, without requesting many cores. This program will produce the following frequencies:

```
# mode    [cm-1]    [THz]      IR
    1      0.00    0.0000    0.0000
    2      0.00    0.0000    0.0000
    3      0.00    0.0000    0.0000
    4      0.00    0.0000    0.0000
    5      0.00    0.0000    0.0000
    6    554.29   16.6173    0.0000
```

We see that now the system has only one nonzero vibrational frequency, as expected. The calculated value 68.7 meV (1 meV = 8.0655 cm^{-1}) is close to our result from the previous section, 69.4 meV. The two values are not identical for two reasons: (1) The acoustic sum rule modifies the potential energy surface, and (2) the present calculations correspond to taking the second derivative of U in the limit $\delta \to 0$.

Note: The documentation about the phonon code `ph.x` can be found at the following link:
http://www.quantum-espresso.org/Doc/INPUT_PH.html
http://www.quantum-espresso.org/Doc/INPUT_DYNMAT.html

An extensive set of examples on how to use `ph.x` is located inside the the directory:
q-e-master/PHonon/examples/

Phonon dispersion relations of diamond

In this section we calculate the phonon dispersion relations of diamond. We begin by setting up the usual input file for diamond, from Tutorial 2.2:

```
$ wget http://www.quantum-espresso.org/upf_files/C.pz-vbc.UPF
$ cat > diamond_scf.in << EOF
&control
 calculation = 'scf'
 prefix = 'diamond',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 2,
 celldm(1) = 6.66405,
 nat = 2,
 ntyp = 1,
 ecutwfc = 100.0,
/
&electrons
 conv_thr = 1.0d-12
/
ATOMIC_SPECIES
 C 1.0 C.pz-vbc.UPF
ATOMIC_POSITIONS crystal
 C 0.00 0.00 0.00
 C 0.25 0.25 0.25
K_POINTS automatic
 4 4 4 1 1 1
EOF
```

Here the lattice constant, the Brillouin-zone sampling, and the planewaves cutoff are set to the same values that we obtained in Tutorial 2.2. We are now using a threshold for the self-consistent cycle, `conv_thr`, which is more stringent than the default value. This is important since phonon calculations are quite sensitive to the accuracy of the ground-state DFT calculation.

In order to calculate phonon frequencies along some high-symmetry paths in the Brillouin zone, we need to go through three separate steps:

1) Calculate the frequencies on a uniform grid of **q**-points;

2) Calculate the real-space interatomic force constants associated with the grid of step 1;

3) Calculate the frequencies along the chosen path of **q**-points, using a Fourier interpolation of the interatomic force constants of step 2.

The **first step** is performed using `ph.x`:

```
$ cat > diamond_ph.in << EOF
-comment line
&inputph
 prefix = 'diamond',
 ldisp = .true.
 amass(1) = 12.0107,
 fildyn = 'dyn',
 nq1 = 2,
 nq2 = 2,
 nq3 = 2,
 tr2_ph = 1.0d-14,
/
EOF
```

Here the flag ldisp = .true. specifies that we are requesting a calculation on a uniform grid. The size of this grid is specified by the variables `nq1`, `nq2`, and `nq3`. Standard grids are of the order of $4 \times 4 \times 4$ to $8 \times 8 \times 8$ points; here we use a modest $2 \times 2 \times 2$ grid only to save time.

This calculation can be performed by running `pw.x` and `ph.x` as usual:

```
$ mpirun -np 12 pw.x -npool 4 < diamond_scf.in > diamond_scf.out
$ mpirun -np 12 ph.x -npool 4 < diamond_ph.in > diamond_ph.out
```

The **second step** is performed using a program called `q2r.x`. This is a small post-processing program which is found in the directory `../espresso-5.4.0/bin`. The input file is very simple, and we can execute this program with a single core:

```
$ cp ../q-e-master/bin/q2r.x ./
$ cat > diamond_q2r.in << EOF
&input
 fildyn = 'dyn',
 flfrc = 'diam.fc'
/
EOF
$ ./q2r.x < diamond_q2r.in
```

At the end of the execution the file `diam.fc` will contain the interatomic force constants.

For the **third step** we need a program called `matdyn.x`. This is also a small post-processing program located in `../q-e-master/bin`.

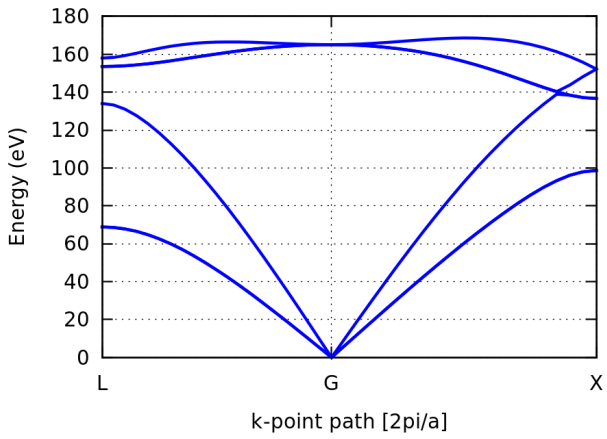
```
$ cp ../q-e-master/bin/matdyn.x ./
```

The input file is as follows:

```
$ cat > diamond_matdyn.in << EOF
&input
 asr = 'simple',
 flfrc = 'diam.fc',
 flfrq = 'diam.freq'
 q_in_band_form = .true.,
/
 3
 0.500 0.500 0.500 20
 0.000 0.000 0.000 20
 1.000 0.000 0.000 20
EOF
$ ./matdyn.x < matdyn.in
```

Here the keyword `q_in_band_form = .true.` indicates that we want the code to calculate vibrational frequencies along a path with 3 vertices in the Brillouin zone, from $(1/2, 1/2, 1/2)2\pi/a$ to $(0, 0, 0)$, and from $(0, 0, 0)$ to $(1, 0, 0)2\pi/a$. Along each segment we will have 20 **q**-points. The Cartesian coordinates of these points are specified in units of $2\pi/a$. In this example we are considering the path $L \to \Gamma \to X$. L is $(1/2, 1/2, 1/2)2\pi/a$, X is $(1, 0, 0)2\pi/a$, and Γ is $(0, 0, 0)$.

The calculated frequencies can be found in the file `diam.freq.gp`. A plot of these data using `gnuplot` gives the following phonon dispersion relations:



In case you are using `gnuplot`, the above was obtained as follows:

```
unset key
set xlabel "k-point path [2pi/a]"
set xtics ("L" 0.0, "G" 0.866, "X" 1.866)
set ylabel "Energy (eV)"
plot [0:1.866] for [i=2:7] "diam.freq.gp" u 1:(column(i)/8.0655) w l lc 3 lw 2
```

LO-TO splitting, IR activity, and dielectric constant of GaAs

In this section we consider GaAs as an example of **polar** semiconductor. The atoms of polar semiconductors exhibit nonzero Born effective charges. The main consequences of nonzero Born charges are: 1) The vibrational frequencies of longitudinal and transverse optical phonons at long wavelength ($\mathbf{q} \to 0$) do not coincide. This is called LO-TO splitting. 2) The system exhibits infrared (IR) activity. 3) The ionic vibrations provide an additional contribution to the dielectric constant at low frequency.

Let us create a basic input file for `pw.x`, for the case of GaAs:

```
$ wget http://www.quantum-espresso.org/upf_files/Ga.pz-bhs.UPF
$ wget http://www.quantum-espresso.org/upf_files/As.pz-bhs.UPF
$ cat > GaAs_scf.in << EOF
&control
 calculation = 'scf'
 prefix = 'gaas',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 2,
 celldm(1) = 10.4749,
 nat = 2,
 ntyp = 2,
 ecutwfc = 40.0,
/
&electrons
/
ATOMIC_SPECIES
 Ga 1.0 Ga.pz-bhs.UPF
 As 1.0 As.pz-bhs.UPF
ATOMIC_POSITIONS crystal
 Ga 0.00 0.00 0.00
 As 0.25 0.25 0.25
K_POINTS automatic
6 6 6 1 1 1
EOF
```

All the parameters in this input file have been optimized separately. We can perform a test run to make sure that everything is in place: as usual we call `pw.x`:

```
$ mpirun -np 12 pw.x -npool 12 < GaAs_scf.in > GaAs_scf.out
```

Now we calculate vibrational frequencies at $\mathbf{q} = 0$. The input file for `ph.x` is similar to what we have seen in the previous section. The only differences are the two additional flags `epsil` and `zeu`:

```
$ cat > GaAs_ph.in << EOF
phonons of GaAs
&inputph
 prefix = 'gaas',
 amass(1) = 69.723,
 amass(2) = 74.9216,
 epsil = .true.,
 zeu = .true.,
 fildyn = 'dyn',
 tr2_ph = 1.0d-14,
/
0.0 0.0 0.0
EOF
```

If we visit the documentation page, http://www.quantum-espresso.org/Doc/INPUT_PH.html, and look for these flags we find:

epsil	LOGICAL
Default:	.false.

If .true. in a q=0 calculation for a non metal the macroscopic dielectric constant of the system is computed. Do not set epsil to .true. if you have a metallic system or q/=0: the code will complain and stop.

zeu	LOGICAL
Default:	zeu=epsil

If .true. in a q=0 calculation for a non metal the effective charges are computed from the dielectric response. This is the default algorithm. If epsil=.true. and zeu=.false. only the dielectric tensor is calculated.

Therefore these flags instruct `ph.x` to also evaluate the high-frequency (electronic) dielectric constant tensor of the system, as well as the Born effective charges. As we have seen in Lecture 4.2, these quantities are needed for calculating the IR activity of each mode and the static dielectric constant.

We execute `ph.x` using this input file from our batch script:

```
$ mpirun -np 12 ph.x -npool 12 < GaAs_ph.in > GaAs_ph.out
```

Near the end of the output file we find the following information:

```
     Number of q in the star =    1
     List of q in the star:
          1   0.000000000   0.000000000   0.000000000

          Dielectric constant in cartesian axis

          (      11.559429679       0.000000000       0.000000000 )
          (       0.000000000      11.559429679       0.000000000 )
          (       0.000000000       0.000000000      11.559429679 )

          Effective charges (d Force / dE) in cartesian axis

           atom      1   Ga
      Ex   (        2.03189        0.00000        0.00000 )
      Ey   (        0.00000        2.03189        0.00000 )
      Ez   (        0.00000        0.00000        2.03189 )
           atom      2   As
      Ex   (       -2.04609        0.00000        0.00000 )
      Ey   (        0.00000       -2.04609        0.00000 )
      Ez   (        0.00000        0.00000       -2.04609 )

     Diagonalizing the dynamical matrix

     q = (    0.000000000   0.000000000   0.000000000 )

 **************************************************************************
     freq (    1) =       0.142309 [THz] =       4.746905 [cm-1]
     freq (    2) =       0.142309 [THz] =       4.746905 [cm-1]
     freq (    3) =       0.142309 [THz] =       4.746905 [cm-1]
     freq (    4) =       8.264350 [THz] =     275.669027 [cm-1]
     freq (    5) =       8.264350 [THz] =     275.669027 [cm-1]
     freq (    6) =       8.264350 [THz] =     275.669027 [cm-1]
 **************************************************************************
```

Here we recognize the high-frequency dielectric constant of GaAs, $\epsilon_\infty = 11.56$, and the Born effective charges of Ga ans As, respectively $Z^*_{\text{Ga}} = 2.03$ and $Z^*_{\text{As}} = -2.04$ (in principle these two values should add up to zero, but we have some numerical error).
The calculated dielectric constant is about 6% higher than the experimental value, $\epsilon^{\text{exp}}_\infty = 10.89$. This overestimation is related to the band gap problem of DFT, which will be discussed in Lecture 5.1.

In the above output of `ph.x` we can see that the optical modes exhibit three identical frequencies, while we were expecting two degenerate TO modes and one LO mode at a higher frequency. The reason why the three optical modes are degenerate is that the calculation performed by `ph.x` missed

a contribution, called the non-analytical part of the dynamical matrix. This contribution can be calculated by using the dielectric constant and Born charges. Let us see how:

We create a new input file for `dynmat.x`:

```
$ cat > GaAs_dynmat.in << EOF
&input
 fildyn = 'dyn',
 asr = 'simple',
 lperm = .true.,
 q(1)=1.0,
 q(2)=0.0,
 q(3)=0.0
/
EOF
```

Here `q(1)`, `q(2)`, and `q(3)` specify the direction along which we approach $\mathbf{q} \to 0$ (the LO-TO splitting is direction-dependent). When one of these numbers is nonzero, `dynmat.x` understands that it must read the dielectric constant and Born charges, calculate the LO-TO correction, and determine IR activities. The additional flag `lperm` specifies that we also want the static dielectric permittivity. After running `dynmat.x` we should obtain the following:

```
$ ./dynmat.x < dynmat.in

...
     IR activities are in (D/A)^2/amu units

# mode   [cm-1]    [THz]      IR
    1      0.00    0.0000    0.0000
    2      0.00    0.0000    0.0000
    3      0.00    0.0000    0.0000
    4    275.63    8.2632    2.6559
    5    275.63    8.2632    2.6559
    6    295.77    8.8670    2.6559

Electronic dielectric permittivity tensor (F/m units)
        11.559430    0.000000    0.000000
         0.000000   11.559430    0.000000
         0.000000    0.000000   11.559430

 ... with zone-center polar mode contributions
        13.080194    0.000000    0.000000
         0.000000   13.310576    0.000000
         0.000000    0.000000   13.310576
```

We can see that now we have 2 degenerate TO modes at 34.17 meV, and 1 LO mode at 36.67 meV. The corresponding LO-TO splitting is 2.5 meV, in good agreement with the experimental value of 2.72 meV obtained by Strauch & Dorner, J. Phys. Condens. Matter 2, 1457 (1990).

The dielectric constants are highlighted in magenta. Here we see that $\epsilon_0 = 13.08$. For comparison the experimental value is $\epsilon_0^{\text{exp}} = 12.9$, therefore the relative deviation is of only 1.4%.

In the output file we also see the IR activities in units of debye/Å^2/amu (highlighted in red above).

A word of caution about the comparison of calculated phonon frequencies with IR experiments in the case of polar materials: IR measurements on thick films only detect TO modes; LO modes are seen in thin films at oblique incidence (Berreman effect), and the relative intensities of LO and TO modes depend on the angle of incidence and film thickness. Some illustrative measurements on III-V semiconductors can be found in Ibáñez et al, J. Appl. Phys. 104, 033544 (2008).

Note. The procedure that we used for calculating the LO-TO splitting can be bypassed by performing a direct calculation of phonon frequencies using a small but nonzero wavevector. For example we could use:

```
$ cat > GaAs_ph_2.in << EOF
phonons of GaAs near Gamma
&inputph
 prefix = 'gaas',
 amass(1) = 69.723,
 amass(2) = 74.9216,
 fildyn = 'dyn',
 tr2_ph = 1.0d-14,
/
0.01 0.0 0.0
EOF
```

This gives two degenerate TO phonons at 34.17 meV and one LO phonon at 36.67 meV, in agreement with our previous calculation.
This alternative procedure is perfectly legitimate, but it does not provide us with Born charges, dielectric constants, and IR activities.

An introduction to density functional theory for experimentalists

Tutorial 4.2

We create a new folder as usual:

```
> cd ~/scratch/summerschool; mkdir tutorial-4.2 ; cd tutorial-4.2
```

In this tutorial we study the phonon dispersion relations of **diamond**, **GaAs** and **$SrTiO_3$**.

Exercise 1

▶ Familiarize yourself with the calculation of the phonon dispersion relations of diamond, following step-by-step the procedure outlined in Tutorial 4.1.

Exercise 2

▶ Repeat the calculations illustrated in Tutorial 4.1 for GaAs. Note that all these calculations refer to zone-center phonons, $\mathbf{q} \to 0$.

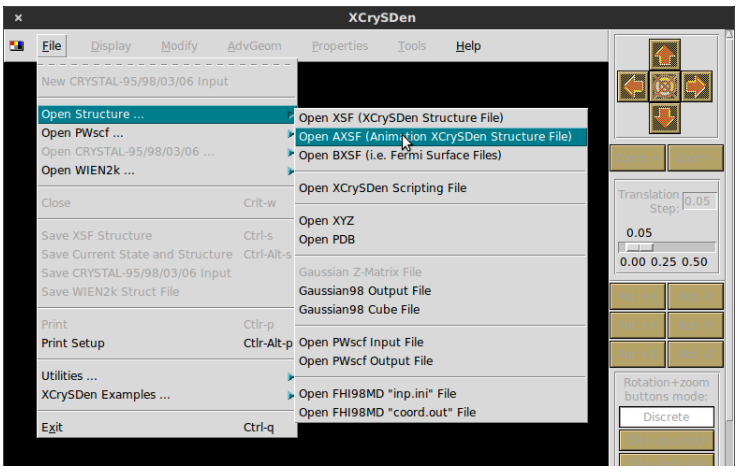
Exercise 3

If you succeeded to complete Exercise 2, you should now see in your working directory the file `dynmat.axsf`. This file has been produced by the code `dynmat.x` that was invoked at the very end of the exercise.

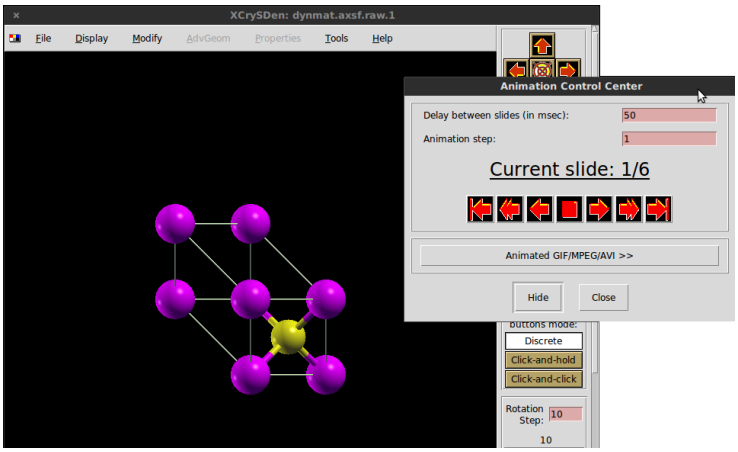
The file `dynmat.axsf` contains the vibrational eignemodes in a format which can be read and visualized by `xcrysden`. Let us recall that these modes correspond to the atomic displacement patterns associated with a wavevector $\mathbf{q} \to 0$ along the direction x (this was specified in the input file `GaAs_dynmat.in`).

▶ In this exercise we want to visualize these modes.

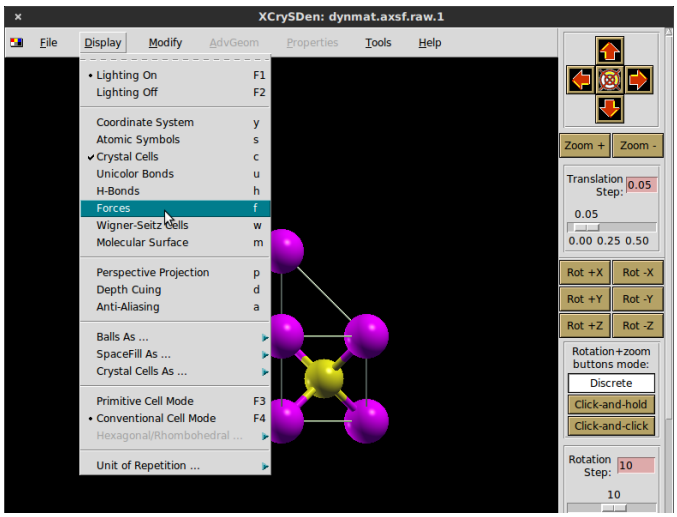
Let us call `xcrysden` and go through the following steps. We open the `dynmat.axsf` file:



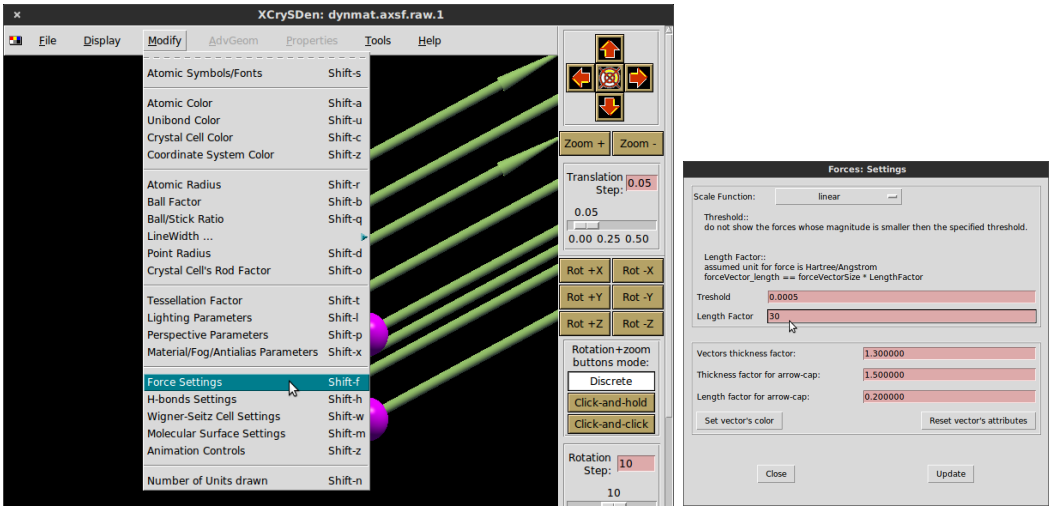
For the time being we ignore the window indicating the 'current slide'. This window will be used later to select the vibrational mode to be visualized.



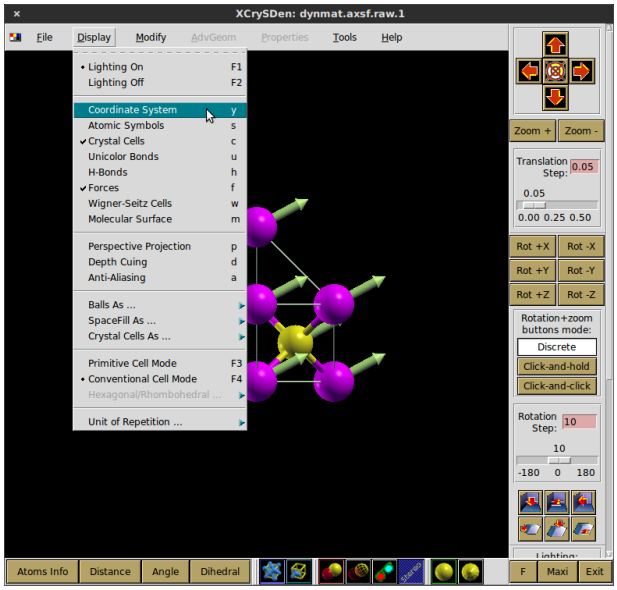
We activate the visualization of 'forces' (in our case the arrows will represent displacements, not forces, but the file format and the naming conventions are the same).



We adjust the length of the arrows by a uniform scaling:

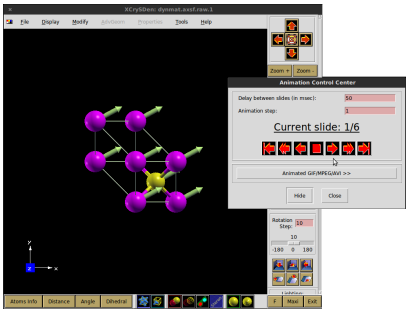


We ask the program to show the Cartesian axes:

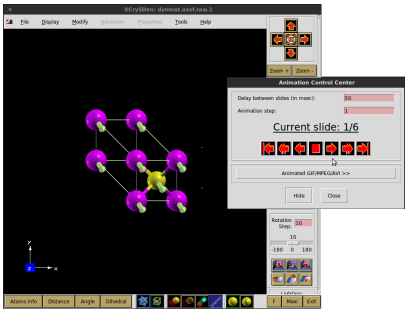


At this point we can use the window entitled 'Current Slide' to inspect each vibrational modes. The result should look similar to the following:

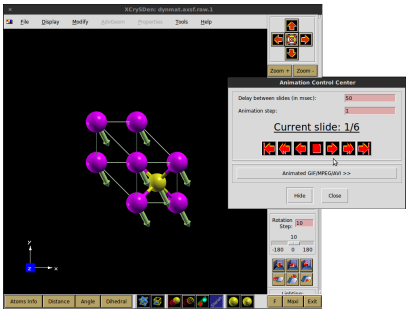
mode 1



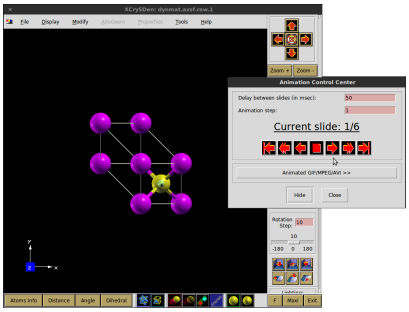
mode 2



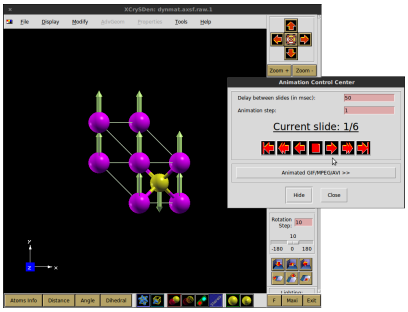
mode 3



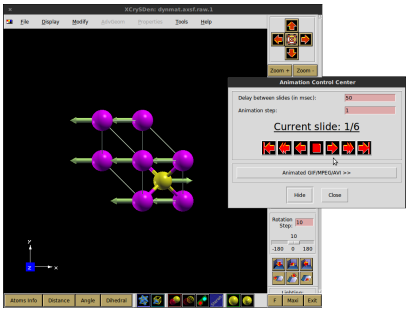
mode 4



mode 5



mode 6



Here we recognize the three translational modes at $\mathbf{q} = 0$ (modes 1–3), which should have $\omega = 0$. We can also recognize the optical phonons (modes 4–6): in these modes Ga and As atoms move in opposite directions. Furthermore, we see that in modes 4 and 5 the atomic displacements are along y and z, while in mode 6 the atoms displace along x. Since we have $\mathbf{q}$ along the x axis (see `GaAs_dynmat.in`), we can conclude that mode 6 is LO, while modes 4–5 are TO.

Exercise 4

▶ Calculate the phonon dispersion relations of GaAs, including the LO-TO splitting.

In this exercise we need to combine what we have learned when we calculated the phonon dispersion relations of diamond, and what we did for calculating the LO-TO splitting in GaAs. The correct input file for `ph.x` is:

```
$ cat > GaAs_disp_ph.in << EOF
phonons of GaAs
&inputph
 prefix = 'gaas',
 amass(1) = 69.723,
 amass(2) = 74.9216,
 epsil = .true.,
 zeu = .true.,
 fildyn = 'dyn',
 tr2_ph = 1.0d-14,
 ldisp = .true.,
 nq1 = 4,
 nq2 = 4,
 nq3 = 4,
/
EOF
```

In this input file the flags in blue instruct `ph.x` to calculate the electronic dielectric permittivity tensor and the Born effective charges. These quantities are needed in order to correctly describe the LO-TO splitting. The lines in red instruct the code to calculate phonons on a uniform grid of $4\times4\times4$ **q**-points.

In order to obtain the dispersion relations, you will need to call `pw.x`, `ph.x`, `q2r.x`, and `matdyn.x` as we already did for diamond:

```
$ mpirun -np 12 pw.x -npool 4 < GaAs_scf.in > GaAs_scf.out
$ mpirun -np 12 ph.x -npool 4 < GaAs_disp_ph.in > GaAs_disp_ph.out
$ ./q2r.x < GaAs_q2r.in
$ ./matdyn.x < GaAs_matdyn.in
```

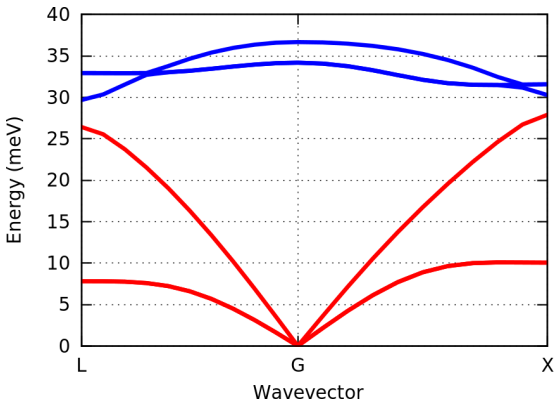
The input files `GaAs_q2r.in` and `GaAs_matdyn.in` must be prepared in the same way as for diamond. The input file `GaAs_scf.in` for GaAs is the same as the one that we used in Tutorial 4.1.

▶ Plot the phonon dispersion relations of GaAs along the Brillouin zone path $L\Gamma X$ (as we did for diamond in Tutorial 4.1).

▶ Verify that the LO-TO splitting at Γ is the same as that calculated in Exercise 2.

▶ Compare your phonon dispersion relations with the inelastic neutron scattering data of Strauch & Dorner, J. Phys. Condens. Matter 2, 1457 (1990).

As a sanity check, you should be able to obtain dispersion relations resembling the following:



Exercise 5

In this exercise we want to calculate the phonon dispersion relations of cubic $SrTiO_3$, including the LO-TO splitting.

▶ We will use the input file for the self-consistent calculation from Tutorial 3.2 (with the optimized parameters given at the beginning of T3.2/Exercise 5), and the pseudopotentials from the same tutorial. Try to perform a test run using this setup, in order to make sure that everything goes smoothly.

▶ We now adapt to the case of $SrTiO_3$ the input file `ph.in` prepared in Exercise 4 for GaAs:

```
$ cat > sto_disp_ph.in << EOF
 phonons of STO
 &inputph
 prefix = 'sto',
 amass(1) = 87.62,
 amass(2) = 47.867,
 amass(3) = 15.9994,
 epsil = .true.,
 zeu = .true.,
 fildyn = 'dyn',
 tr2_ph = 1.0d-14,
 ldisp = .true.,
 nq1 = 2,
 nq2 = 2,
 nq3 = 2,
/
EOF
```

Now you can execute `ph.x` using this input file. You should find out that the execution takes much longer than in the case of GaAs.

Calculations on $SrTiO_3$ are more time-consuming than for GaAs since we now have 24 electrons per unit cell, and we are using a cutoff of 210 Ry. This means that we have to work with 12 Kohn-Sham wavefunctions per each **k**-point, and each wavefunction is expanded in a basis of 20,000+ planewaves. In these cases it is convenient to perform calculations in two steps:

1) We reduce massively the kinetic energy cutoff and the Brillouin zone sampling, and carry out the calculations until we manage to obtain our phonon dispersion relations. These results will be inaccurate and unreliable, but they will allow us to test every step very quickly.

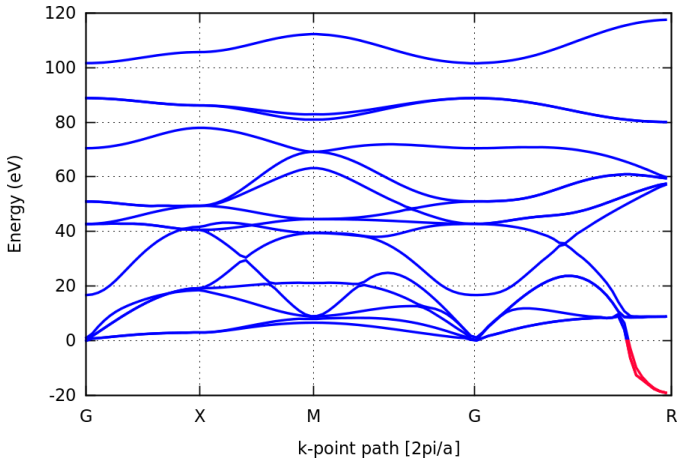
2) Once we are confident about the complete procedure, we launch a production calculation with all the optimized convergence parameters. This may take up to 30min on 12 cores, but now we can be confident that the calculation will complete successfully.

▶ Following this two-step procedure, reduce the cutoff to 50 Ry and the Brillouin zone sampling to 2 2 2 1 1 1, and calculate the phonon dispersion relations of $SrTiO_3$. The procedure is identical to what was done for GaAs on page 4.

In this case we can plot the dispersions along the high-symmetry path $\Gamma X M \Gamma R$. The Cartesian coordinates of these points are $\Gamma : (0,0,0)$, $X : (0.5,0,0)$, $M : (0.5,0.5,0)$, $R : (0.5,0.5,0.5)$ in units of $2\pi/a$. Therefore the `sto_matdyn.in` file will be:

```
$ cat > sto_matdyn.in << EOF
&input
 asr = 'simple',
 flfrc = 'sto.fc',
 flfrq = 'sto.freq',
 q_in_band_form = .true.,
/
5
 0.000 0.000 0.000 20
 0.500 0.000 0.000 20
 0.500 0.500 0.000 20
 0.000 0.000 0.000 20
 0.500 0.500 0.500 20
EOF
```

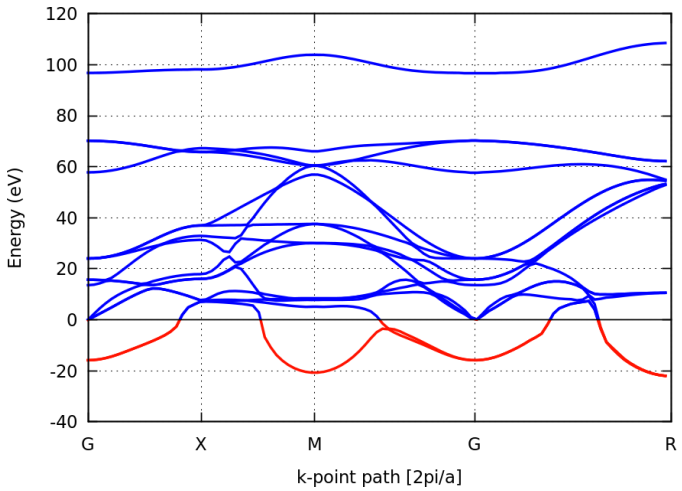
You should obtain something like the following:



▶ Now that are are confident that we can successfully complete the entire procedure, we move on to perform a production run and obtain the final dispersion relations. You can repeat the entire procedure by using the optimized parameters `ecutwfc = 210` and `K_POINTS automatic 4 4 4 1 1 1`.

For this calculation it is recommended to use 24 cores. If you started your interactive session with 12 cores, then you need to exit the session (`exit`), and start a new one with 24 cores: `interact -p shared -n 20 -t 360 -r paradim`.

The final result should look as follows:



▶ Compare your result with those reported by Ghosez et al, AIP Conf. Proc. 535, 102 (2000) and by Cancellieri et al, Nat. Commun. 7, 10386 (2016).

▶ In the plot at the top of this page the curves in red denote imaginary frequencies, that is modes for which $\omega^2 < 0$. Since ω^2 represents the curvature of the potential energy surface, negative values imply that the system is in a local **maximum** of the energy surface, and is therefore **unstable** with respect to those vibrational modes.
In order to better understand the origin of these instabilities, use `xcrysden` to visualize the displacement patterns of the soft modes at Γ, M, and R.

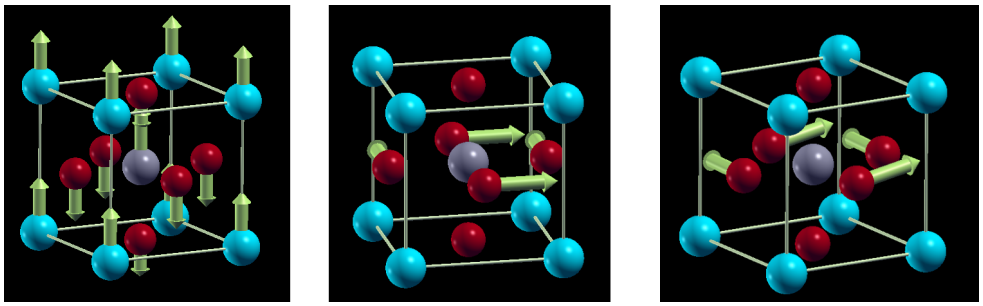
In order to use `xcrysden` you will need to generate `dynmat.axsf` files as in Exercise 3, using `dynmat.x`. The `dynmat.in` files should look as follows:

```
$ cat > dynmat.in << EOF
&input
 fildyn = 'dyn1'
/
EOF
```

where `dyn1` is for the Γ point, `dyn3` is for the M point, and `dyn4` is for the R point.

The soft modes, that is the modes with imaginary frequency, should look as follows:

Γ (modes 1–3) M (mode 1) R (modes 1–3)



The soft modes at Γ indicate the presence of a ferroelectric (quantum paraelectric) instability, the soft modes at M and R indicate instabilities against rotations of the TiO_6 octahedra.

Note: The displacements provided by `dynmat.x` correspond to the Bloch-periodic part of the vibrational eigenmodes. These displacements correspond to the real atomic displacements only when $\mathbf{q} = \Gamma$. In order to see the complete eigendisplacements for $\mathbf{q} \neq 0$ we would need to add the Bloch wave $\exp(i\mathbf{q} \cdot \mathbf{r})$ manually.

The origin of these soft modes lies in the fact that we are performing ground-state calculations (i.e. at 0 K) for the cubic structure of $SrTiO_3$, but the cubic structure is only stable above 110 K. These soft modes can be taken as an indication of the tendency of the system to lower its symmetry. The soft modes disappear when performing calculations on larger orthorhombic unit cells (containing 20 atoms per cell).

▶ Calculate the dielectric permittivity and IR activities of cubic $SrTiO_3$, using `dynmat.x`.

In this case the appropriate input file is

```
$ cat > dynmat_sto_eps.in << EOF
&input
 fildyn = 'dyn1',
 asr = 'simple',
 lperm = .true.,
 q(1)=1.0,
 q(2)=0.0,
 q(3)=0.0
/
EOF
```

Note that we are instructing `dynmat.x` to read data from `dyn1`, which corresponds to the Γ point.

Here we should be careful in interpreting our data: the static permittivity ϵ_0 is **not reliable** since we have soft modes with a large IR activity. Generally speaking, calculations for the high-temperature cubic phase of $SrTiO_3$ necessitate including temperature and quantum nuclear effects.

An introduction to density functional theory for experimentalists

Tutorial 5.1

```
$  cd ~/scratch/summerschool ; mkdir tutorial-5.1 ; cd tutorial-5.1
```

In this tutorial we will see how to calculate the band structures and the optical absorption spectra of semiconductors.

Band structures

We start from the band structure of **silicon**. First we copy the setup from Tutorial 2.1:

```
$ cp ../q-e-master/bin/pw.x ./
$ wget http://www.quantum-espresso.org/upf_files/Si.pz-vbc.UPF
$ cat > si_scf.in << EOF
&control
 calculation = 'scf'
 prefix = 'silicon',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 2,
 celldm(1) = 10.2094,
 nat = 2,
 ntyp = 1,
 ecutwfc = 25.0,
/
&electrons
/
ATOMIC_SPECIES
 Si  28.086  Si.pz-vbc.UPF
ATOMIC_POSITIONS
 Si 0.00 0.00 0.00
 Si 0.25 0.25 0.25
K_POINTS automatic
 4 4 4 1 1 1
EOF
```

We test that everything is in place by performing the usual test run:

```
mpirun -np 12 pw.x < si_scf.in > si_scf.out
```

Now we want to calculate the band structure. This calculation is *non self-consistent*, in the sense that we use the ground-state electron density, Hartree, and exchange and correlation potentials determined in the previous run. In a non self-consistent calculation the code `pw.x` determines the Kohn-Sham eigenfunctions and eigenvalues without upgrading the Kohn-Sham Hamiltonian at every step. This is achieved by using the keyword `calculation = 'bands'` and by specifying the **k**-points for which we want the eigenvalues:

```
$ cat > si_nscf.in << EOF
&control
 calculation = 'bands'
 prefix = 'silicon',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 2,
 celldm(1) = 10.2094,
 nat = 2,
 ntyp = 1,
 ecutwfc = 25.0,
 nbnd = 8,
/
&electrons
/
ATOMIC_SPECIES
 Si  28.086  Si.pz-vbc.UPF
ATOMIC_POSITIONS
 Si 0.00 0.00 0.00
 Si 0.25 0.25 0.25
K_POINTS tpiba_b
3
0.500 0.500 0.500 20
0.000 0.000 0.000 20
1.000 0.000 0.000 20
EOF
```

In this input file we are using the same path in the Brillouin zone that we used for the phonon dispersion relations of diamond in Tutorial 4.1. The keyword `tpiba_b` after `K_POINTS` specifies that we want `pw.x` to generate a path going through the points specified in the list. The following number (3) is the number of vertices, and the integer following the coordinates (20) is the number of points in each segment. So in this case we will have 20 points from $L = (1/2, 1/2, 1/2)2\pi/a$ to $\Gamma = (0, 0, 0)$ and 20 points from $\Gamma = (0, 0, 0)$ to $X = (1, 0, 0)2\pi/a$. The points are given in Cartesian coordinates and in units of $2\pi/a$. In this input file we also specify the number of bands that we want to calculate: in order to see the 4 valence bands of silicon and the 4 lowest conduction bands we are setting `nbnd = 8`.

After executing `pw.x`:

```
mpirun -np 12 pw.x -npool 12 < si_nscf.in > si_nscf.out
```

we can find the Kohn-Sham eigenvalues in the output file `si_nscf.out` (`vi si_nscf.out` and / band):

```
...
    End of band structure calculation

          k = 0.5000 0.5000 0.5000 (   568 PWs)   bands (ev):

    -3.4470  -0.8438   4.9915   4.9915   7.7615   9.5415   9.5415  13.7948

          k = 0.4750 0.4750 0.4750 (   568 PWs)   bands (ev):

    -3.4841  -0.7911   4.9961   4.9961   7.7666   9.5469   9.5469  13.8032
...
```

Here, for each **k**-point in the input file, we have the coordinates of the point (blue) and the calculated eigenvalues in eV (red). We see 8 eigenvalues because we have requested 8 bands. In order to plot the bands along the chosen path, we must extract these eigenvalues, and calculate the distance covered as we move along the path $L \rightarrow \Gamma \rightarrow X$. We can use the following script which automates this procedure (copy/paste this file into `extract.tcsh`):

```
$ more extract.tcsh
```

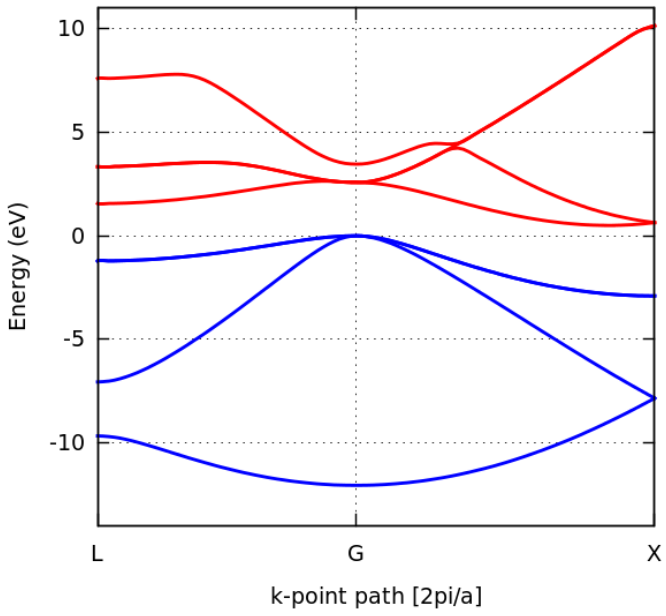
```
set filename = si_nscf.out
set tmp = `grep Kohn-Sham $filename`
set bands = $tmp[5]
set nrows = `awk "BEGIN{ print int (($bands+8-1)/8) }"`
set klines = `grep -nr " k =" $filename | cut -d : -f 1`
set k0 = `head -$klines[1] $filename | tail -1`
set len = 0
foreach nline ( $klines )
 set k = `head -$nline $filename | tail -1`
 @ nline = $nline + $nrows + 1
 set eig = `head -$nline $filename | tail -$nrows`
 set len=`awk "BEGIN{print $len +sqrt(($k[3]-$k0[3])^2+($k[4]-$k0[4])^2+($k[5]-$k0[5])^2)}"`
 set k0 = `echo $k`
 echo $len $eig
end
```

```
$ tcsh extract.tcsh > si_bands.txt
```

At this point the file `bands.txt` contains the distance along the path in reciprocal space, and the eigenvalues in each column. We can plot this file using the `gnuplot` instructions similar to those of Tutorial 4.1:

```
ezero = 6.2057
unset key
set ylabel "Energy (eV)"
set xtics ("L" 0, "G" 0.866, "X" 1.866)
set xlabel "k-point path [2pi/a]"
plot for [i=2:9] "si_bands.txt" using 1:(column(i)-ezero) w l lc 3 lw 2 smooth csplines
```

The result should look as follows:



Here the conduction bands have also been colored in red and the zero of the energy axis has been set manually to the top of the valence band at Γ (`ezero = 6.2057`).

By looking for the valence band top at Γ and the conduction band bottom along the Γ-X line, we find that the band gap of silicon in DFT/LDA is $E_g = 0.502$ eV. The calculated band gap is much smaller than the experimental value of 1.2 eV.

Visualizing Kohn-Sham wavefunctions

Following the calculation of the band structure of silicon, we can visualize the wavefunctions corresponding to selected Kohn-Sham eigenvalues.

In order to plot a wavefunction we must use a post-processing code named `pp.x`. We compile this code as we already did for `pw.x` and `pp.x`:

```
$ cd ../q-e-master ; make pp
$ cd ../tutorial-5.1 ; cp ../q-e-master/bin/pp.x ./
```

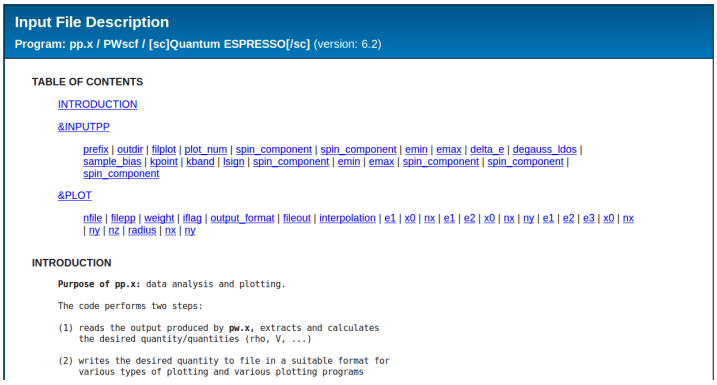
This small post-processing code reads the output of a `pw.x` run, and rewrites it in a format compatible with standard visualization software. The structure of the input file of `pp.x` is:

```
$ cat > si_pp.in << EOF
&inputpp
  prefix = 'silicon'
  outdir = './',
  filplot = 'wavefc'
  plot_num = 7
  lsign = .true.
  kpoint = 21
  kband = 4
/
&plot
  iflag = 3
  output_format = 5
  fileout = 'silicon.xsf'
/
EOF
```

The important input variables are shown in color. `plot_num = 7` specifies that we want to plot the square modulus of Kohn-Sham wavefunctions, and the flag `lsign = .true.` is to keep track of the sign of the wavefunction. The variables `kpoint` and `kband` indicate the **k**-point and band that we want to plot. In this case we are choosing the 21-st point from the list in `si_nscf.out` and the band number 4. This is precisely the valence band top at $\mathbf{q} = 0$, i.e. at Γ. The flags `iflag = 3` and `output_format = 5` specify that we want a 3D plot and that this must be in `xcrysden` format, respectively.

There are many other options for plotting other quantities of interest, for the complete range please see the documentation page:

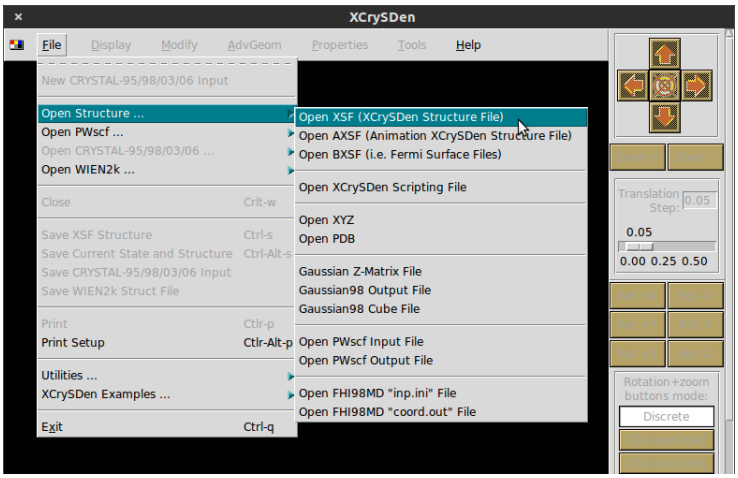
http://www.quantum-espresso.org/Doc/INPUT_PP.html

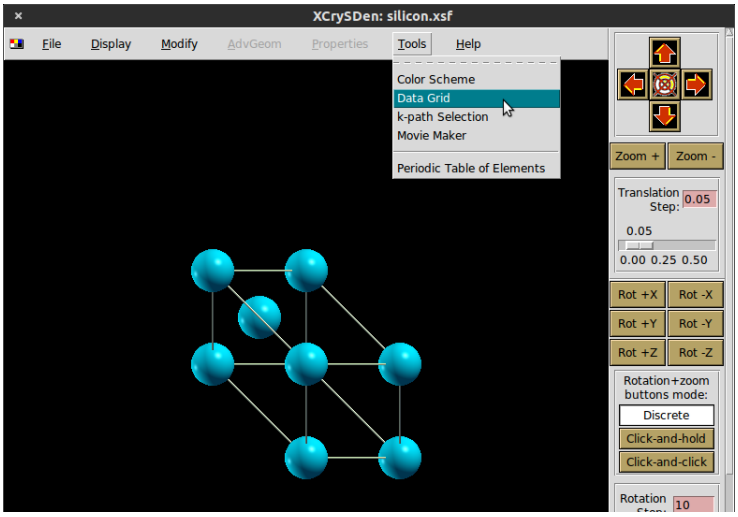


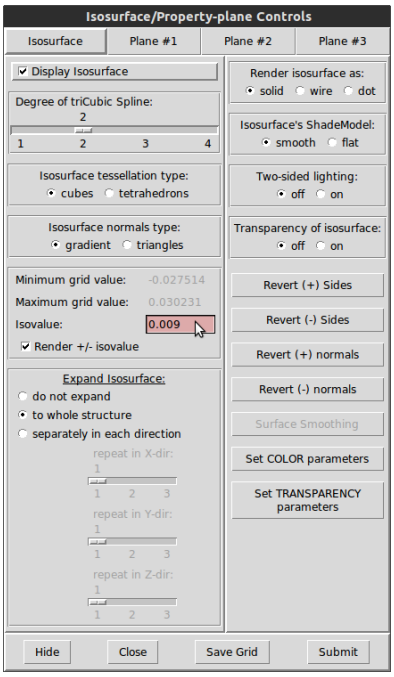
In order to obtain our wavefunction, first we execute `pw.x` from the previous section, and then we run `pp.x`:

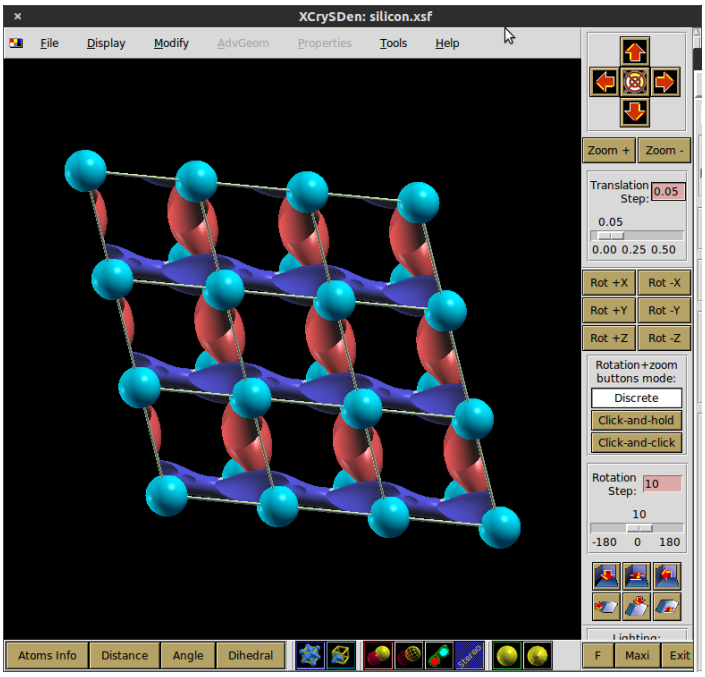
```
$ mpirun -np 12 pw.x < si_nscf.in
$ mpirun -np 12 pp.x < si_pp.in
```

After this operation we should have in our directory the file `silicon.xsf`. We visualize the wavefunctions by launching `xcrysden` and following the steps below:









In this example we can see that the electrons at the valence band top concentrate around the Si–Si bonds, as expected from tight-binding models.

Calculation of UV/Vis spectra

Now we consider the band structure and the optical absorption spectrum of **GaAs**. We already studied GaAs in Tutorial 4.1, therefore we can start from the same `scf.in` file:

```
$ wget http://www.quantum-espresso.org/upf_files/Ga.pz-bhs.UPF
$ wget http://www.quantum-espresso.org/upf_files/As.pz-bhs.UPF
$ cat > GaAs_scf.in << EOF
&control
 calculation = 'scf'
 prefix = 'gaas',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 2,
 celldm(1) = 10.4749,
 nat = 2,
 ntyp = 2,
 ecutwfc = 40.0,
/
&electrons
/
ATOMIC_SPECIES
 Ga 1.0 Ga.pz-bhs.UPF
 As 1.0 As.pz-bhs.UPF
ATOMIC_POSITIONS crystal
 Ga 0.00 0.00 0.00
 As 0.25 0.25 0.25
K_POINTS automatic
 6 6 6 1 1 1
EOF
```

We perform a test run to make sure that everything works:

```
$ mpirun -np 12 pw.x -npool 12 < GaAs_scf.in > GaAs_scf.out
```

Now we can take a look at the band structure of GaAs. The procedure is identical to what we just did for silicon, and we can recycle most of the input file `si_nscf.in` from page 2. The colored lines below indicate the modifications required to work with GaAs instead of Si. These parameters are taken directly from the input file `GaAs_scf.in` above:

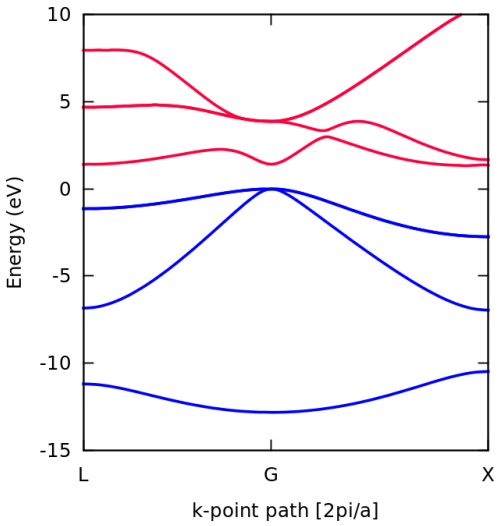
```
$ cat > GaAs_nscf.in << EOF
&control
 calculation = 'bands'
 prefix = 'gaas',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 2,
 celldm(1) = 10.4749,
 nat = 2,
 ntyp = 2,
 ecutwfc = 40.0,
 nbnd = 8,
/
&electrons
/
ATOMIC_SPECIES
 Ga 1.0 Ga.pz-bhs.UPF
 As 1.0 As.pz-bhs.UPF
ATOMIC_POSITIONS
 Ga 0.00 0.00 0.00
 As 0.25 0.25 0.25
K_POINTS tpiba_b
3
0.500 0.500 0.500 20
0.000 0.000 0.000 20
1.000 0.000 0.000 20
EOF
```

We can now run the band structure calculation, precisely as we did for silicon:

```
$ mpirun -np 12 pw.x < GaAs_nscf.in > GaAS_nscf.out
```

At the end of the run we extract the **k**-point path and the Kohn-Sham eigenvalues again using the script `extract.tcsh` from page 3, and plot these data using `plot.gp`. Note that we need to change the first line of `extract.tcsh` into `set filename = GaAs_nscf.out`.

The resulting band structure should look as follows:



In this calculation we see that the direct gap at Γ is E_g =1.42 eV. This value is unusually close to experiment (1.52 eV) for a DFT/LDA calculation. We can also see that in this calculation we have an indirect gap of 1.35 eV along the ΓX line. This is an artifact of the DFT/LDA approximation (GaAs is a direct-gap semiconductor).

Now we calculate the imaginary part of the dielectric function, $\epsilon_2(\omega)$. This quantity is related to the optical absorption coefficient $\kappa(\omega)$ by $\kappa(\omega) = \omega\,\epsilon_2(\omega)/c\,n(\omega)$, where $\hbar\omega$ is the photon energy, c the speed of light, and n the refractive index.

In order to calculate $\epsilon_2(\omega)$ we use the post-processing code `epsilon.x`. We already compiled this program when we issued `make pp` on page 4, therefore we only need to copy the code inside the current directory:

```
$ cp ../q-e-master/bin/epsilon.x ./
```

The manual of this post-processing code can be found in the directory `../q-e-master/PP/Doc`. To obtain a PDF version we simply issue:

```
$ cd ../q-e-master/PP ; make doc
```

The as-compiled PDF file `eps_man.pdf` will be found in the directory `q-e-master/PP/Doc`.

The input file for `epsilon.x` is as follows:

```
$ cat > GaAs_eps.in << EOF
&inputpp
  outdir = './'
  prefix = 'gaas'
  calculation = 'eps'
/
&energy_grid
  smeartype = 'gauss'
  intersmear = 0.2
  wmin =  0.0
  wmax = 30.0
  nw = 500
/
EOF
```

This file instructs `epsilon.x` to calculate the real and the imaginary parts of the dielectric function, $\epsilon_1(\omega)$ and $\epsilon_2(\omega)$. The variables `smeartype` and `intersmear` define the numerical approximation used to represent the Dirac delta functions in the expression that we have seen in Lecture 5.1. The variables `wmin`, `wmax` and `nw` define the energy grid for the dielectric function. All the energy variables are in eV.

Before executing `epsilon.x` we need to perform a new run with `pw.x`, using a slightly modified input file:

```
$ cat > GaAs_nscf_eps.in << EOF
&control
calculation = 'nscf'
prefix = 'gaas',
pseudo_dir = './',
outdir = './'
verbosity = 'high',
/
&system
ibrav = 2,
celldm(1) = 10.4749,
nat = 2,
ntyp = 2,
ecutwfc = 40.0,
nbnd = 16,
nosym = .true.
noinv = .true.
/
&electrons
/
ATOMIC_SPECIES
Ga 1.0 Ga.pz-bhs.UPF
As 1.0 As.pz-bhs.UPF
ATOMIC_POSITIONS
Ga 0.00 0.00 0.00
As 0.25 0.25 0.25
K_POINTS automatic
10 10 10 1 1 1
EOF
```

The modifications brought to our standard input file are as follows:

- We perform a non-self-consistent calculation
- We use a uniform grid of **k**-points
- We turn off the automatic reduction of **k**-points that `pw.x` does by using crystal symmetries (`nosym = .true.` and `noinv = .true.`).
- We request a larger number of bands (16), since we are interested in interband transitions.

The first two modifications are related to the fact that `epsilon.x` is a fairly basic post-processing code and does not recognize crystal symmetries.

The grid used in the above input file includes 10^3 (reducible) points.
We can now execute `pw.x` and `epsilon.x`:

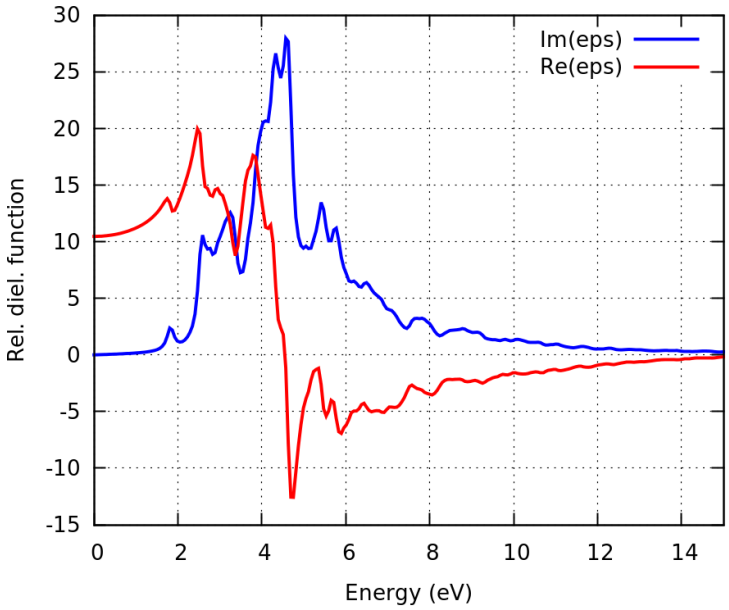
```
$ mpirun -np 12 pw.x -npool 12 < GaAs_nscf_eps.in > GaAs_nscf_eps.out
$ mpirun -np 12 epsilon.x -npool 12 < GaAs_eps.in > GaAs_eps.out
```

At the end of the execution we will find the output files:

```
$ more epsi.dat

# energy grid [eV]     epsi_x  epsi_y  epsi_z
#
    0.000000000    0.000000000    0.000000000    0.000000000
    0.060120240    0.007992650    0.007992651    0.007992651
...

$ more epsr.dat

# energy grid [eV]     epsr_x  epsr_y  epsr_z
#
    0.000000000   10.460661055   10.460660612   10.460662360
    0.060120240   10.463052313   10.463051870   10.463053619
...
```

The first file contains the real part of the dielectric function, for an electric field polarized along x, y, or z. The second file is the corresponding imaginary part. In this case the system is cubic, therefore the x, y and z components will be identical.

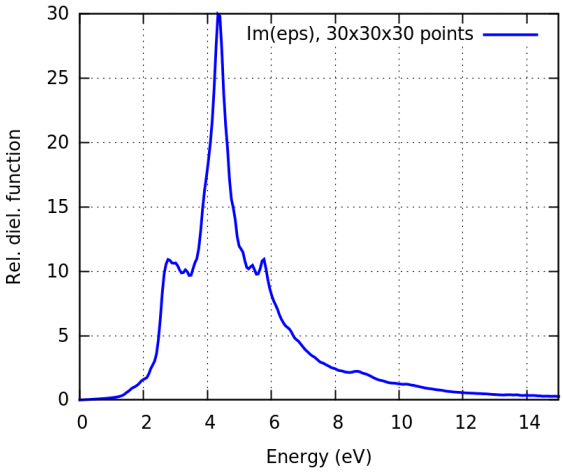
A plot of these quantities using `gnuplot` yields:

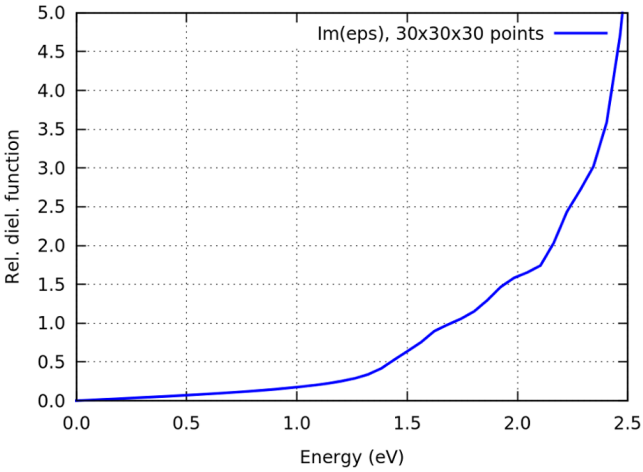


Note: From this figure we can read the high-frequency dielectric constant of GaAs, $\epsilon_\infty = 10.46$. This value is quite different from what we had obtained in Tutorial 4.1 (11.56). This discrepancy arises from the fact that `epsilon.x` makes two approximations, the 'independent-particle approximation' and the neglect of the 'nonlocal component' of the pseudopotentials. As a result, while the gross structure of the spectrum is reasonably accurate, subtle features such as the intensity and energy of the peaks are not very reliable, and the same goes for the value of ϵ_∞. A more comprehensive discussion of dielectric functions in DFT can be found in Chapter 11 of the DFT book.

In the above plot we can see that the curves are not very smooth. This phenomenon is related to the sampling of the Brillouin zone: in this calculation we used a $10 \times 10 \times 10$ mesh, and this is definitely not enough for studying the dielectric function. The meshes required for calculations of dielectric functions may need to contain as many as $50 \times 50 \times 50$ points. With our coarse $10 \times 10 \times 10$ mesh it is difficult to identify the optical absorption onset around the direct band gap at 1.4 eV.

A more accurate calculation using a 30×30×30 mesh is shown below, together with a zoom where we can see the onset around 1.4 eV (broadened by a Gaussian smearing via the input variable `intersmear = 0.2` eV):





An introduction to density functional theory for experimentalists

Tutorial 5.2

```
$ cd ~/scratch/summerschool; mkdir tutorial-5.2 ; cd tutorial-5.2
```

In this tutorial we will first calculate the band structures of **GaAs**, **$SrTiO_3$**, and **graphite**. Then we will test a calculation of dielectric functions for the case of GaAs. In the last exercise we will examine the performance of the LDE and PBE exchange and correlation in predicting the interlayer binding energy and the interlayer distance in graphite.

Exercise 1

▶ Familiarize yourself with the calculation of band structures, following step-by-step the example of GaAs discussed in Tutorial 5.1.

Exercise 2

In this exercise we want to calculate the band structure of $SrTiO_3$.

▶ Perform a test run for $SrTiO_3$ using exactly the same setup as in Exercise 5 of Tutorial 3.2.

As a sanity check, you should obtain a total energy of -105.4136 Ry.

▶ Now we run a band structure calculation for $SrTiO_3$. Prepare the input file `sto_nscf.in` for the non-selfconsistent calculation, using the high-symmetry path $\Gamma X M \Gamma R$. The Cartesian coordinates of these points are $\Gamma : (0, 0, 0)$, $X : (0.5, 0, 0)$, $M : (0.5, 0.5, 0)$, $R : (0.5, 0.5, 0.5)$ in units of $2\pi/a$.

As a reference, the input file should look like this:

```
&control
 calculation = 'bands'
 prefix = 'sto',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 1,
 celldm(1) = 7.18899,
 nat = 5,
 ntyp = 3,
 ecutwfc = 210.0,
 nbnd = 20,
/
&electrons
/
ATOMIC_SPECIES
 Sr 1.0 Sr.pz-hgh.UPF
```

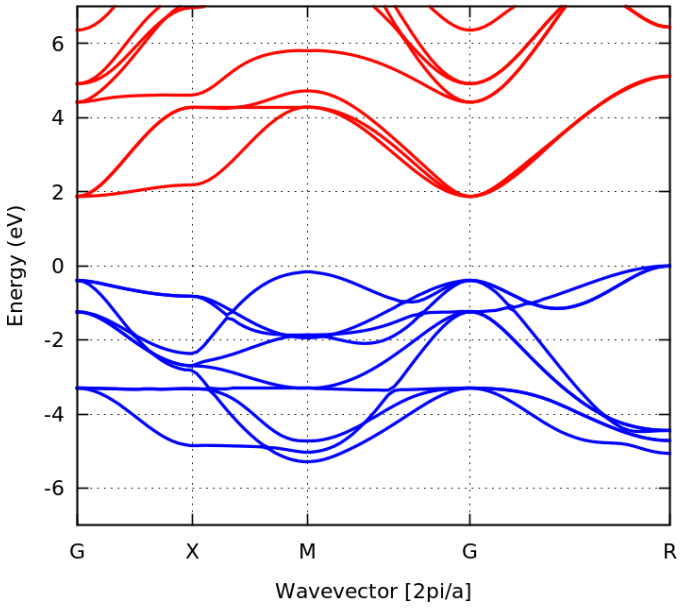
```
 Ti 1.0 Ti.pz-hgh.UPF
 O  1.0  O.pz-hgh.UPF
ATOMIC_POSITIONS crystal
 Sr 0.0 0.0 0.0
 Ti 0.5 0.5 0.5
 O  0.5 0.0 0.5
 O  0.5 0.5 0.0
 O  0.0 0.5 0.5
K_POINTS tpiba_b
5
0.0 0.0 0.0 20
0.5 0.0 0.0 20
0.5 0.5 0.0 20
0.0 0.0 0.0 20
0.5 0.5 0.5 20
```

Note that the number of bands must be increased in order to calculate conduction states (here we have set `nbnd = 20`).

▶ After executing `pw.x` with this input file, you can extract the Kohn-Sham eigenvalues and the coordinate along the **k**-path using the same script `extract.tcsh` on page 3 of Tutorial 5.1. In that script we only need to change the file name in the first line (`set filename = si_nscf.out` → `set filename = sto_nscf.out`).
After obtaining the band structure via `tcsh extract.tcsh > sto_bands.txt`, we can plot the result using the following instructions in `gnuplot`:

```
ezero = 5.9668
unset key
set xlabel "Wavevector [2pi/a]"
set xtics ("G" 0.0, "X" 0.500, "M" 1.000, "G" 1.707, "R" 2.573)
set ylabel "Energy (eV)"
plot [] [-7:7] for [i=2:21] "sto_bands.txt" u 1:(column(i)-ezero) w l lw 2 lc 3 smooth csplines
```

As a reference, the plot should look similar to the following:



▶ Using the band structure just calculated, determine the lowest direct and indirect band gaps of $SrTiO_3$ in DFT/LDA.

▶ Compare your band gaps and band structure with previous experimental and theoretical work from Benthem et al, J. Appl. Phys. 90, 6156 (2001) and from Benrekia et al, Physica B 407, 2632 (2012).

Exercise 3

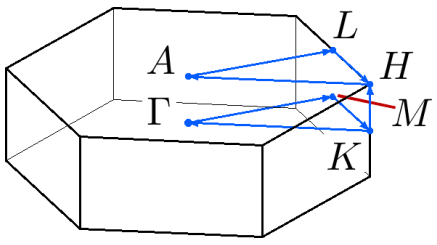
In this exercise we calculate the band structure of graphite.

▶ Perform a test run for the total energy of graphite in the ground state, using the optimized setup determined in Exercise 3 of Tutorial 2.2.

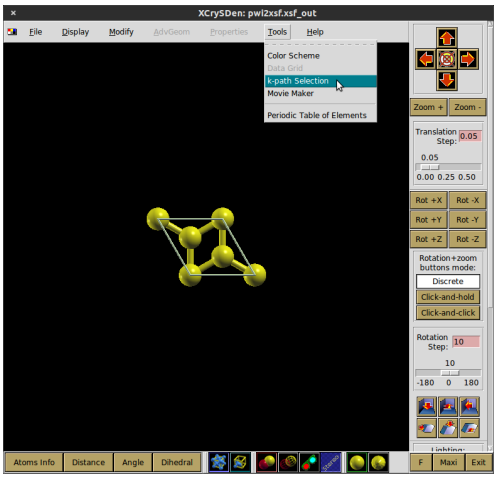
As a reminder, the optimized input file is:

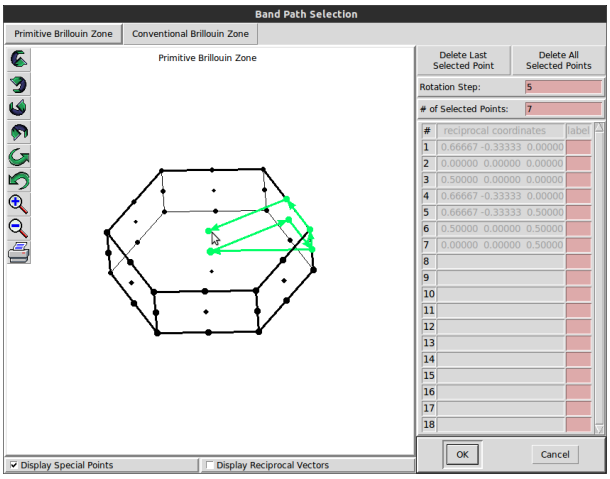
```
&control
 calculation = 'scf'
 prefix = 'graphite',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 4,
 celldm(1) = 4.60913,
 celldm(3) = 2.729,
 nat = 4,
 ntyp = 1,
 ecutwfc = 100,
/
&electrons
/
ATOMIC_SPECIES
C 1.0 C.pz-vbc.UPF
ATOMIC_POSITIONS crystal
C 0.00 0.00 0.25
C 0.00 0.00 0.75
C 0.333333 0.666666 0.25
C 0.666666 0.333333 0.75
K_POINTS automatic
6 6 2 1 1 1
```

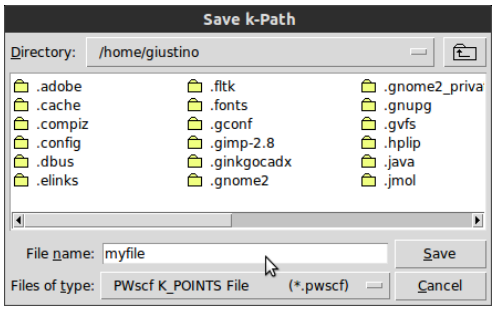
▶ Prepare the input file `nscf.in` for the band structure calculation along the high-symmetry path $K \to \Gamma \to M \to K \to H \to A \to L \to H$.
In this case we want to perform calculations for 16 bands.
The location of the high-symmetry points in the Brillouin zone of graphite is given below:



In order the generate the **k**-point path we can use `xcrysden`. After loading the input file inside `xcrysden`, we go through the following steps:







At the end of the file `myfile.kpf` we will find the desired path. Here each vertex is expressed in crystal coordinates, that is in terms of the primitive vectors of the reciprocal lattice.

```
$ more myfile.kpf
...
#
# #---------------------------------#
# # REAL FORM of k-point COORDINATES #
# #---------------------------------#
#
Real form of k-point coordinates (kx,ky,kz,label):
   0.6666666667    -0.3333333333     0.0000000000     K.1
   0.0000000000     0.0000000000     0.0000000000     K.2
   0.5000000000     0.0000000000     0.0000000000     K.3
   0.6666666667    -0.3333333333     0.0000000000     K.4
   0.6666666667    -0.3333333333     0.5000000000     K.5
   0.0000000000     0.0000000000     0.5000000000     K.6
   0.5000000000     0.0000000000     0.5000000000     K.7
   0.6666666667    -0.3333333333     0.5000000000     K.8
...
```

Since the coordinates of the vertices are expressed in terms of the reciprocal lattice vectors, in our input file we will need to replace `K_POINTS tpiba_b` by `K_POINTS crystal_b`. For more details you can look at the documentation page: http://www.quantum-espresso.org/Doc/INPUT_PW.html.

As a reference, the correct input file is:

```
&control
calculation = 'bands'
prefix = 'graphite',
```

```
pseudo_dir = './',
outdir = './',
/
&system
ibrav = 4,
celldm(1) = 4.60913,
celldm(3) = 2.729,
nat = 4,
ntyp = 1,
ecutwfc = 100,
nbnd = 16,
/
&electrons
/
ATOMIC_SPECIES
C 1.0 C.pz-vbc.UPF
ATOMIC_POSITIONS crystal
C 0.00 0.00 0.25
C 0.00 0.00 0.75
C 0.333333 0.666666 0.25
C 0.666666 0.333333 0.75
K_POINTS crystal_b
8
   0.6666666667    -0.3333333333     0.0000000000     10
   0.0000000000     0.0000000000     0.0000000000     10
   0.5000000000     0.0000000000     0.0000000000     10
   0.6666666667    -0.3333333333     0.0000000000     10
   0.6666666667    -0.3333333333     0.5000000000     10
   0.0000000000     0.0000000000     0.5000000000     10
   0.5000000000     0.0000000000     0.5000000000     10
   0.6666666667    -0.3333333333     0.5000000000     10
```

Once executed `pw.x` using this input file, we can extract the bands using exactly the same script `extract.tcsh` as in Exercise 2 (keep in mind that we need to update the first line of that script with the current file name).

▶ Plot the band structure of graphite using `gnuplot`.

Possible `gnuplot` instructions for this are:

```
unset key
set xlabel "Wavevector [2pi/a]"
set xtics ("K" 0.0, "G" 0.6667, "M" 1.24406, "K" 1.57744, "H" 1.76064, "A" 2.42734, "L" 3.00469, "H" 3.33807)
set ylabel "Energy (eV)"
plot [0:3.33807] [:20] for [i=2:17] "graphite_bands.txt" u 1:(column(i)) w l lw 2 lc 3
```

▶ Compare your calculated band structure with the results of Marinopoulos et al, Phys. Rev. B 69, 245419 (2004).

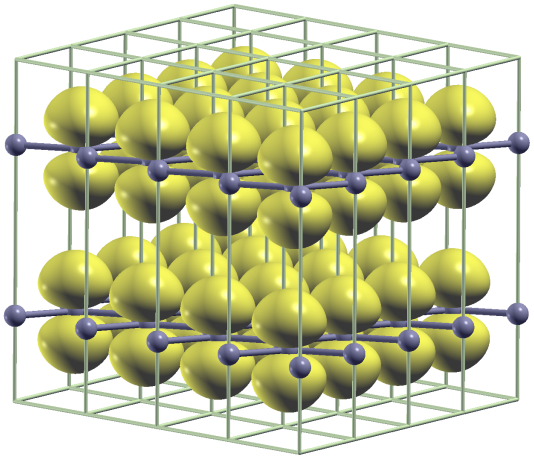
Exercise 4

▶ Using the post-processing program `pp.x`, verify that the wavefunctions near the Dirac point are p_z orbitals as expected (the Dirac point is the K point, at the intersection between valence and conduction bands).

We can run `pp.x` using the following input file, after having specified the appropriate **k**-point and band:

```
&inputpp
 prefix = 'graphite'
 outdir = './',
 filplot = 'wavefc'
 plot_num = 7
 kpoint = ...
 kband = ...
/
&plot
 iflag = 3
 output_format = 5
 fileout = 'graphite.xsf'
/
```

By tuning the isosurface value you should be able to obtain something similar to the following:



Exercise 5

In this exercise we want to calculate the optical absorption spectrum of GaAs, precisely as we did in Tutorial 5.1.

▶ Repeat the steps on pagg. 6–11 of Tutorial 5.1 in order to calculate the absorption spectrum of GaAs.

▶ Run `epsilon.x` once again, this time using a smearing parameter `intersmear = 0.02` eV.

▶ Compare the imaginary part of the dielectric function of GaAs, $\epsilon_2(\omega)$, obtained above using the two smearing values 0.1 eV and 0.2 eV.

In `gnuplot` we can do this by using the usual `plot` command:

```
plot "epsi_gaas_0.02.dat" u 1:2 w l, "epsi_gaas_0.2.dat" u 1:2 w l
```

where `epsi_gaas_0.02.dat` and `epsi_gaas_0.2.dat` are the output files of the two separate runs.

Here you should see that, as we reduce the artificial smearing, the curve turns into a series of discrete peaks. This is because we are discretizing the Brillouin zone using only 10^3 points. An accurate calculation requires a much finer sampling.

Exercise 6

In this exercise we want to explore the sensitivity of the structure of **graphite** to the exchange and correlation functional used in the calculation.

The structure of graphite and the various calculation parameters have already been optimized in Tutorial 2.2, and the optimized input file was given earlier, see pag. 3.

▶ Calculate the total energy of graphite within DFT/LDA as a function of interlayer distance, for ratios c/a ranging between 2.2 and 3.2.

For this exercise you can simply use the input file on page 3 and perform several calculations for different values of `celldm(3)`.

▶ Now convert the results of the previous step into a cohesive energy, and plot the cohesive energy as a function of the c/a ratio.

From Tutorial 2.2 we already know that the total energy of an isolated C atom using the same parameters as for graphite is -10.73321495 Ry.

▶ Now we repeat the same calculations as in the previous steps, this time using the PBE exchange and correlation functional. Determine the cohesive energy of graphite within DFT/PBE, as a function of the ratio c/a, for the same range considered above.

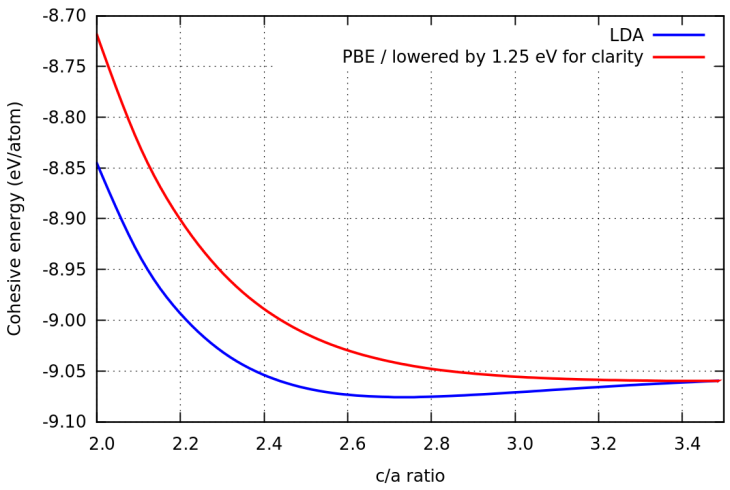
For this exercise we need a new pseudopotential, constructed within the PBE approximation. The following pseudopotential was tested separately:

```
$ wget http://www.quantum-espresso.org/upf_files/C.pbe-hgh.UPF
```

It was found that the kinetic energy cutoff required to have total energies converged to within 10 meV is `ecutwfc = 130` Ry. Furthermore, the total energy of an isolated C atom using this pseudopotetial was calculated to be -10.82157608 Ry.

▶ Plot the cohesive energies just obtained, and compare the predictions of LDA and PBE. What should we conclude from this comparison?

For your reference, the plot should look as follows:



Graphite is a prototypical system where van der Waals interactions play an important role in the interlayer binding.