Compute the refractive index using first-principles

First-principles calculations are based on quantum mechanics in which the time-independent Schrödinger equation is solved. Performing these calculations is best done on Linux (we recommend Ubuntu) but you can use a virtual machine.
Do do so:

- Download and install Virtual Box version virtualbox-7.1.6
- We will use the Quantum Mobile.
- However, for this class, we will use a special version that you can download here.
- Launch Virtual box and load QM: `File --> Import Appliance` and choose the `MSE468Spring2023.ova` file that you have downloaded.
- Launch the Quantum Mobile (check the cpu and memory you are allocating to it).
- Once Ubuntu is launched (it may take some time to start), start the terminal
- **The system may ask for update. Do NOT update.**
- Type `cd /usr/local/bin` to go where the executable are located.

We will obtain the refractive index by computing the dielectric function $\varepsilon_1 + i\varepsilon_2$:

$$n(\omega) = \left[\frac{1}{2}\left(\frac{\varepsilon_1^2 + \varepsilon_2^2}{2}\right)^{1/2} + \varepsilon_1\right]^{1/2} \tag{1}$$

$$\kappa(\omega) = \left[\frac{1}{2}(\varepsilon_1^2 + \varepsilon_2^2)^{1/2} - \varepsilon_1\right]^{1/2} \tag{2}$$

For this, we will use the Quantum ESPRESSO code, installed directly with then Quantum Mobile.
First, go to the Materials Cloud and use the periodic table to find your materials.
Then click on it and download the structure in CIF format.
WARNING: your material should contain less than 6 atoms per primitive cell. Else the computational cost will be too high for your computer.
Go on the Quantum ESPRESSO input generator and structure visualizer and upload your CIF file by selecting the CIF file for the file format. Then click on "Generate the PWscf input file.
Click on "Download zip of input file and pseudopotentials".
Unzip. This creates a "PWscf" folder with the input and pseudopotential.
We need a norm-conserving pseudopotential. You can find one on Pseudo Dojo. Select the `.upf` format when downloading them. You can rename the pseudopotential such as `Cu.upf`. You can unzip `unzip PWscf.zip`.
You need to update the path/name for the pseudopotential in your input file. Here is an example input for the case of Cu:

```
&CONTROL
  calculation = 'scf'
  etot_conv_thr =   1.0000000000d-05
  forc_conv_thr =   1.0000000000d-04
  outdir = './out/'
  prefix = 'aiida'
  pseudo_dir = './pseudo/'
  tprnfor = .true.
  tstress = .true.
  verbosity = 'high'
/
&SYSTEM
  degauss =   2.0000000000d-02
  ecutrho =   7.2000000000d+02
```

```
  ecutwfc =   9.0000000000d+01
  ibrav = 0
  nat = 1
  nosym = .false.
  ntyp = 1
  occupations = 'smearing'
  smearing = 'cold'
/
&ELECTRONS
  conv_thr =   2.0000000000d-10
  electron_maxstep = 80
  mixing_beta =   4.0000000000d-01
/
ATOMIC_SPECIES
Cu     63.546 Cu.upf
ATOMIC_POSITIONS crystal
Cu           0.0000000000       0.0000000000       0.0000000000
K_POINTS automatic
21 21 21 0 0 0
CELL_PARAMETERS angstrom
      2.5189362786       0.0000000000       0.0000000000
      1.2594681393       2.1814628078       0.0000000000
      1.2594681393       0.7271542693       2.0567028591
```

Run the self-consistent calculation with
`pw.x < pwscf.in | tee pwscf.out`
or if you want to run with multiple cores:
`mpirun -np 4 pw.x < pwscf.in | tee pwscf.out`
to compute the ground-state electronic density.
Then make a non-self-consistent calculation on a denser grid. To do this, copy the file and edit it (change the pseudopotential name).
`cp pwscf.in pwnscf.in`
In the file, change the calculation type to nscf:
`calculation = 'nscf'`
Change nosym to true and add the two new line in the SYSTEM block (you need `nband` to be sufficiently large for your system):

```
nosym = .TRUE.
noinv = .TRUE.
nbnd = 20
```

Increase the k-point grid to a large value:

```
K_POINTS automatic
24 24 24 0 0 0
```

Then run the calculation
`mpirun -np 4 pw.x < pwnscf.in | tee pwnscf.out`
Note that this step can be time-consuming, e.g. took about 10 min for Cu. If the calculation is too slow, try reducing the k-point grid or talk with us.
Then create the input for the `epsilon.x` code and call it `epsilon.in`:

```
&inputpp
  outdir = './out/',
  prefix = 'aiida',
calculation = 'eps'
/

&energy_grid
```

```
smeartype = 'gauss',
intersmear = 0.2,
wmin = 0.0,
wmax = 30.0,
nw = 500
/
```

Run it:
`epsilon.x < epsilon.in | tee epsilon.out`
The code will then produce the `epsr_aiida.dat` and `ieps_aiida.dat` files.
Open the file `epsr_aiida.dat`. If you have too big number, you will have '*****', remove these lines. The column should not touch as well.
You can plot them using the following python code:

```
import matplotlib.pyplot as plt
from matplotlib import rcParamsDefault
import numpy as np

plt.rcParams["figure.dpi"]=150
plt.rcParams["figure.facecolor"]="white"

data_r = np.loadtxt('epsr_aiida.dat')
data_i = np.loadtxt('epsi_aiida.dat')
energy_r, epsilon_r = data_r[:, 0], data_r[:, 2]
energy_i, epsilon_i = data_i[:, 0], data_i[:, 2]

plt.plot(energy_r, epsilon_r, lw=1, label="$\\epsilon_1$")
plt.plot(energy_i, epsilon_i, lw=1, label="$\\epsilon_2$")
plt.xlim(0, 15)
plt.ylim(-40,40)
plt.xlabel("Energy (eV)")
plt.ylabel("$\\epsilon_1~/~\\epsilon_2$")
plt.legend(frameon=False)
plt.show()
```

You may need to install matplotlib:
`pip install matplotlib`
Then from this, you can plot the refractive index and compare it to the experimental data.
Do you see difference? Try to explain why if there are.

Additional information can be found here.

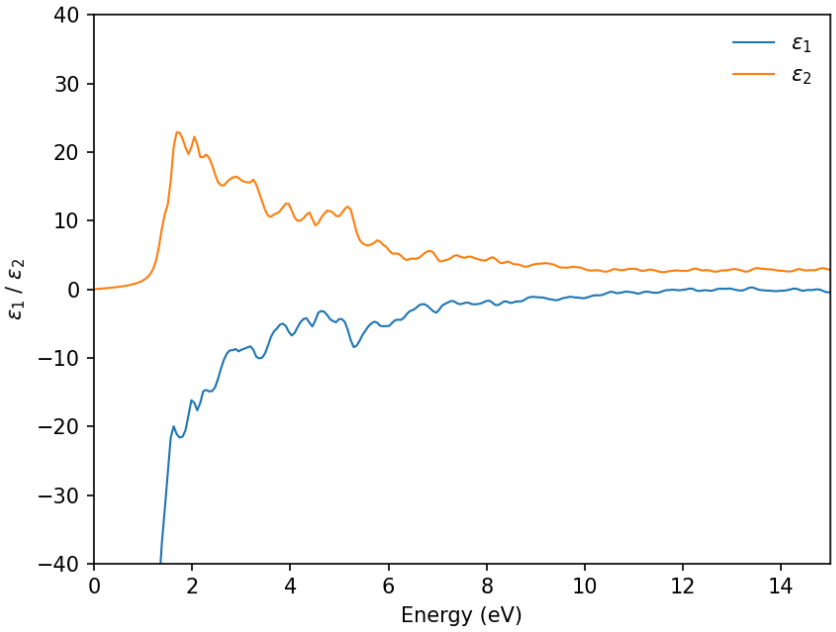

Figure 1: Refractive index and reflectivity bulk copper using Quantum ESPRESSO calculations.