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 ../espresso-5.4.0/bin/pw.x ./
$ wget http://www.quantum-espresso.org/wp-content/uploads/upf_files/Si.pz-vbc.UPF
$ cat > 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
 conv_thr = 1.0d-8
/
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. For this we submit a batch job with the line:

```
mpirun -np 12 pw.x -npool 1 < scf.in > 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 > 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
 conv_thr = 1.0d-8
/
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 10
0.000 0.000 0.000 10
1.000 0.000 0.000 10
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 (10) is the number of points in each segment.
So in this case we will have 10 points from $L = (1/2, 1/2, 1/2)2\pi/a$ to $\Gamma = (0, 0, 0)$ and 10 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` using our batch script:

```
mpirun -np 12 pw.x -npool 12 < nscf.in > nscf.out
```

we can find the Kohn-Sham eigenvalues in the output file `nscf.out` (`vi nscf.out` and / `band`):

```
...
     End of band structure calculation

          k = 0.5000 0.5000 0.5000 (   568 PWs)   bands (ev):

    -3.4467  -0.8437   4.9918   4.9918   7.7616   9.5416   9.5416  13.7947

          k = 0.4500 0.4500 0.4500 (   571 PWs)   bands (ev):

    -3.5870  -0.6409   5.0097   5.0097   7.7820   9.5628   9.5628  13.8281
...
```

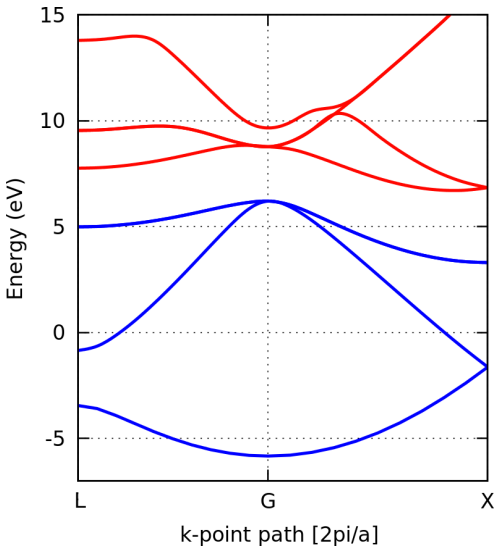
Here, for each **k**-point in the input file, we have the coordinates of the point (blue) and the calculated eigenvalues in eV (red). We see 8 eigenvalues because we have requested 8 bands.
In order to plot the bands along the chosen path, we must extract manually these eigenvalues, and calculate the distance along the path as we move from L to Γ to X. We can do this quickly using the following `tcsh` script:

```
$ more extract.tcsh

set klines = `grep -nr " k =" nscf.out | cut -d : -f 1`
set k0 = `head -$klines[1] nscf.out | tail -1`
set kk = 0
foreach NLINE ( $klines )
 set k = `head -$NLINE nscf.out | tail -1`
 @ NLINE = $NLINE + 2
 set eigenv = `head -$NLINE nscf.out | tail -1`
 set kk=`awk "BEGIN{print $kk +sqrt(($k[3]-$k0[3])^2+($k[4]-$k0[4])^2+($k[5]-$k0[5])^2)}"`
 set k0 = `echo $k`
 echo $kk $eigenv
end

$ tcsh extract.tcsh > bands.txt
```

At this point the file `bands.txt` will contain the distance along the **k**-point path and the eigenvalues in each column. We can plot this file using the following `gnuplot` instructions:

```
$ cat > plot.gp << EOF
unset key
set xlabel "k-point path [2pi/a]"
set xtics ("L" 0, "G" 0.866, "X" 1.866)
set ylabel "Energy (eV)"
set style line 1 lc 3 lw 2
plot "bands.txt" u 1:2 w l ls 1, \
     "bands.txt" u 1:3 w l ls 1, \
     "bands.txt" u 1:4 w l ls 1, \
     "bands.txt" u 1:5 w l ls 1, \
     "bands.txt" u 1:6 w l ls 1, \
     "bands.txt" u 1:7 w l ls 1, \
     "bands.txt" u 1:8 w l ls 1, \
     "bands.txt" u 1:9 w l ls 1
EOF
$ gnuplot
gnuplot> load "plot.gp"
```

The result should look similar to the following plot:



Here the conduction bands have been colored in red and a smoothing has been used via the option `smooth csplines` of `gnuplot`.

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 $E_g = 0.5128$ 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 ../espresso-5.4.0 ; make pp
$ cd ../tutorial-5.1 ; cp ../espresso-5.4.0/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:

```
&inputpp
  prefix = 'silicon'
  outdir = './'
  filplot = 'wavefc'
  plot_num = 7
  lsign = .true.
  kpoint = 11
  kband = 4
/
&plot
  iflag = 3
  output_format = 5
  fileout = 'silicon.xsf'
/
```

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 11-th point from the list on page 2 and the band number 4. This is precisely the valence band top 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.

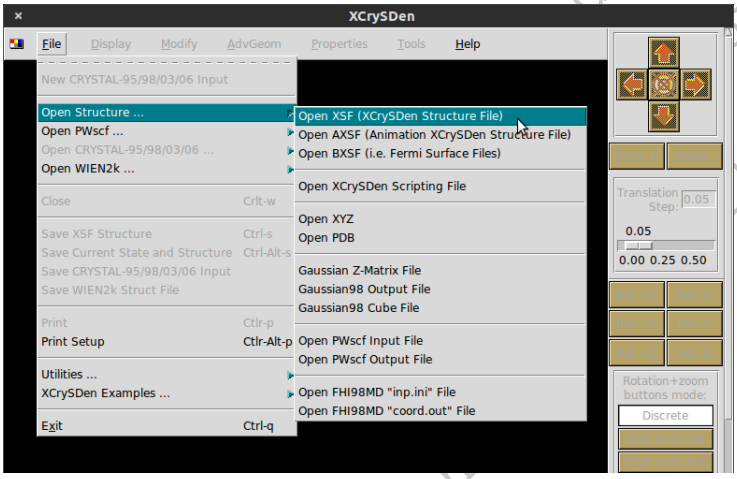
There are many other options for plotting other quantities of interest, for the complete range please see the documentation page:

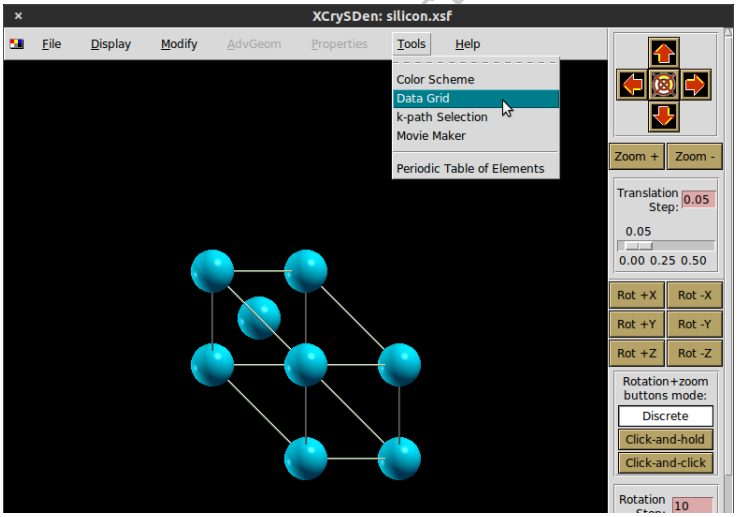
http://www.quantum-espresso.org/wp-content/uploads/Doc/INPUT_PP.html

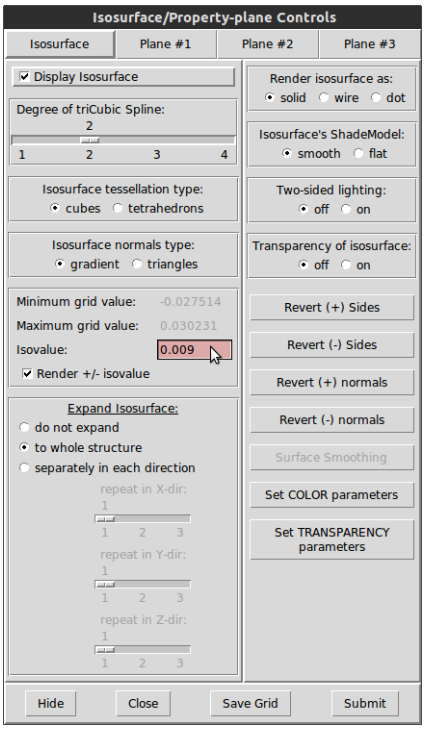
In order to obtain our wavefunction, first we execute `pw.x` from the previous section, and then we run `pp.x` using the same number of CPUs and in the same batch script:

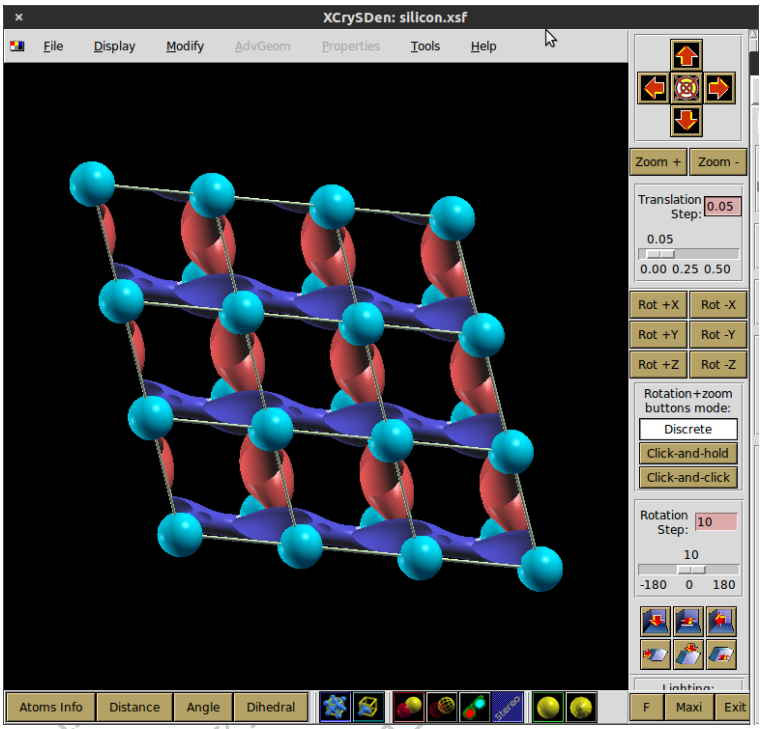
```
mpirun -np 12 pw.x < nscf.in
mpirun -np 12 pp.x < pp.in
```

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/wp-content/uploads/upf_files/Ga.pz-bhs.UPF
$ wget http://www.quantum-espresso.org/wp-content/uploads/upf_files/As.pz-bhs.UPF
$ cat > 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 < scf.in > scf.out
```

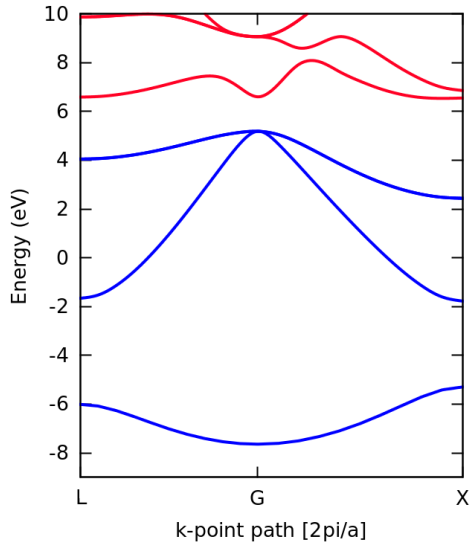
Now we can take a look at the band structure of GaAs. The procedure is identical to what we just did for silicon, and we can recycle most of the input file `nscf.in` from pag. 1. The colored lines below indicate the modifications required to work with GaAs instead of Si. These parameters are taken directly from the input file `scf.in` above:

```
$ cat > 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
 conv_thr = 1.0d-8
/
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 10
0.000 0.000 0.000 10
1.000 0.000 0.000 10
EOF
```


We can now run the band structure calculation, precisely as we did for silicon:

```
mpirun -np 12 pw.x < scf.in > scf.out
mpirun -np 12 pw.x < nscf.in > 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`, and plot these data using `plot.gp`.
The resulting band structure should look as follows (this plot has also been smoothed and the color of the conduction bands has been modified):



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, $\epsilon_2(\omega)$, which is related to the optical absorption coefficient $\kappa(\omega)$ by:

$$\kappa(\omega) = \frac{\omega\,\epsilon_2(\omega)}{c\,n(\omega)}$$

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 pag. 4, therefore we only need to copy the code inside the current directory:

```
$ cp ../espresso-5.4.0/bin/epsilon.x ./
```

The manual of this post-processing code can be found in the directory `../espresso-5.4.0/PP/Doc`. To obtain a PDF version we simply issue:

```
$ cd ../espresso-5.4.0/PP ; make doc
```

The as-compiled PDF file `eps_man.pdf` will be found in the directory `espresso-5.4.0/PP/Doc`.

The input file for `epsilon.x` is as follows:

```
$ cat > 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 > scf_eps.in << EOF
&control
 calculation = 'scf'
 prefix = 'gaas',
 pseudo_dir = './',
 outdir = './'
/
&system
 ibrav = 2,
 celldm(1) = 10.4749,
  nat = 2,
 ntyp = 2,
 ecutwfc = 40.0,
 nosym = .true.
 nbnd = 20
/
&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 crystal
 64
0.120.120.120.016
0.120.120.380.016
0.120.120.620.016
0.120.120.880.016
0.120.380.120.016
0.120.380.380.016
0.120.380.620.016
0.120.380.880.016
0.120.620.120.016
0.120.620.380.016
0.120.620.620.016
0.120.620.880.016
0.120.880.120.016
0.120.880.380.016
0.120.880.620.016
0.120.880.880.016
0.380.120.120.016
0.380.120.380.016
0.380.120.620.016
0.380.120.880.016
0.380.380.120.016
0.380.380.380.016
0.380.380.620.016
0.380.380.880.016
0.380.620.120.016
0.380.620.380.016
0.380.620.620.016
0.380.620.880.016
```

```
0.380.880.120.016
0.380.880.380.016
0.380.880.620.016
0.380.880.880.016
0.620.120.120.016
0.620.120.380.016
0.620.120.620.016
0.620.120.880.016
0.620.380.120.016
0.620.380.380.016
0.620.380.620.016
0.620.380.880.016
0.620.620.120.016
0.620.620.380.016
0.620.620.620.016
0.620.620.880.016
0.620.880.120.016
0.620.880.380.016
0.620.880.620.016
0.620.880.880.016
0.880.120.120.016
0.880.120.380.016
0.880.120.620.016
0.880.120.880.016
0.880.380.120.016
0.880.380.380.016
0.880.380.620.016
0.880.380.880.016
0.880.620.120.016
0.880.620.380.016
0.880.620.620.016
0.880.620.880.016
0.880.880.120.016
0.880.880.380.016
0.880.880.620.016
0.880.880.880.016
EOF
```


The modifications brought to our standard input file are as follows:

- We explicitly provide a uniform grid of **k**-points
- We turn off the automatic reduction of **k**-points that `pw.x` does by using crystal symmetries
- We request a number of bands much larger than the number of valence bands, 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 is a uniform and shifted 4×4×4 mesh in the Brillouin zone (64 points).
We can now execute `pw.x` and `epsilon.x` using the following lines in our batch script:

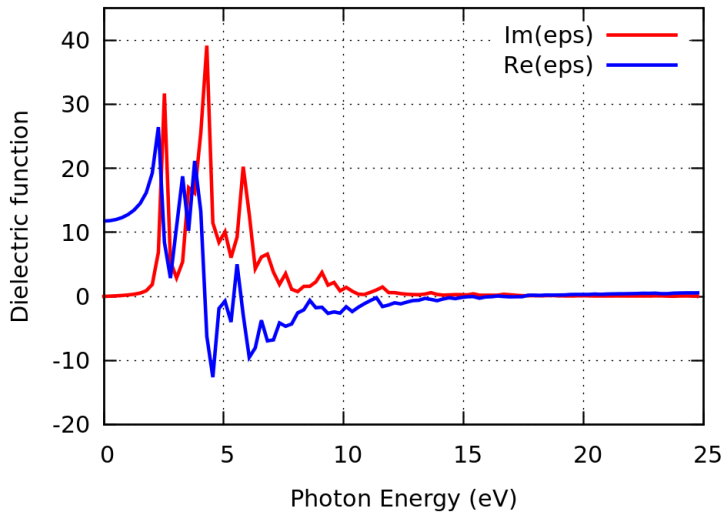
```
mpirun -np 12 pw.x -npool 12 < scf_eps.in > scf_eps.out
mpirun -np 12 epsilon.x -npool 12 < eps.in > eps.out
```

At the end of the execution we wil find the output files:

```
$ more epsr.dat

  # energy grid [eV]     epsr_x  epsr_y  epsr_z
       0.000000       11.787553       11.787537       11.787546
       0.060120       11.790677       11.790662       11.790670
...

$ more epsi.dat

  # energy grid [eV]     epsi_x  epsi_y  epsi_z
       0.000000        0.000000        0.000000        0.000000
       0.060120        0.010446        0.010446        0.010446
...
```

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.

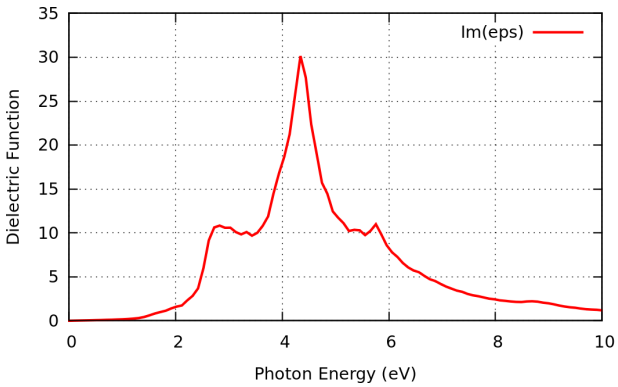
A plot of these quantities using `gnuplot` yields:

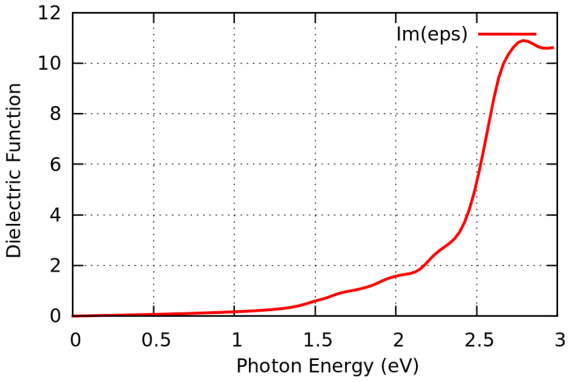


From this figure we can read the high-frequency dielectric constant of GaAs, $\epsilon_\infty = 11.8$. This value is similar but not identical to what we obtained in Tutorial 4.1 (11.6). This difference is due to the different sampling of the Brillouin zone and the different computational technique employed here.

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 4×4×4 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 50×50×50 points. With our coarse 4×4×4 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 the Gaussian smearing, which is set to `intersmear = 0.2` eV):





Note. The calculation of dielectric functions by means of `epsilon.x` suffers from two important approximations, namely 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.
A more comprehensive discussion of dielectric functions in DFT can be found in Chapter 11 of the Book.