2023 Virtual School on Many-Body Calculations using EPW and BerkeleyGW

Self-trapped polarons with EPW

Hands-on session (Wed.3)

Hands-on based on QE-v7.2 and EPW-v5.7

Introduction

In this session, you will learn to perform calculations of self-trapped polarons with the `EPW` code. In this short introduction, we briefly present the main concepts and equations relevant for this tutorial. The complete theory and computational method can be found in Phys. Rev. Lett. **122**, 246403 (2019) and Phys. Rev. B **99**, 235139 (2019).
The ground state wave function $\psi(\mathbf{r})$ and atomic displacements $\Delta\tau_{\kappa\alpha p}$ forming a polaron can be found by minimizing the total DFT energy functional of an excess electron added to a crystal, which translates into the solution of the following coupled system of equations:

$$\hat{H}^0_{\mathrm{KS}}\,\psi(\mathbf{r}) + \sum_{\kappa\alpha p}\frac{\partial V^0_{\mathrm{KS}}}{\partial\tau_{\kappa\alpha p}}\Delta\tau_{\kappa\alpha p}\,\psi(\mathbf{r}) = \varepsilon\,\psi(\mathbf{r}) \ , \tag{1}$$

$$\Delta\tau_{\kappa\alpha p} = -\sum_{\kappa'\alpha' p'}(C^0)^{-1}_{\kappa\alpha p,\kappa'\alpha' p'}\int d\mathbf{r}\,\frac{\partial V^0_{\mathrm{KS}}}{\partial\tau_{\kappa'\alpha' p'}}\,|\psi(\mathbf{r})|^2 \ . \tag{2}$$

In these expressions, $\tau_{\kappa\alpha p}$ represents the Cartesian coordinate of the atom κ in the unit cell p along the direction α, $C^0_{\kappa\alpha p,\kappa'\alpha' p'}$ is the matrix of interatomic force constants, and $\hat{H}^0_{\mathrm{KS}}$ and V^0_{KS} represent the Kohn-Sham Hamiltonian and the self-consistent potential, respectively. The superscript ^{0} indicates that the quantities are evaluated in the ground state without extra electron. The real space integral is performed over a Born-Von Karman supercell of the crystal containing N_p unit cells. We will refer to ε as the polaron eigenvalue.
By expanding the polaron wave function in terms of the single-particle Kohn-Sham states $\psi_{n\mathbf{k}}$ with eigenvalues $\varepsilon_{n\mathbf{k}}$:

$$\psi(\mathbf{r}) = \frac{1}{\sqrt{N_p}}\sum_{n\mathbf{k}}A_{n\mathbf{k}}\psi_{n\mathbf{k}} \ , \tag{3}$$

and the atomic displacements in terms of the phonon eigenmodes $e_{\kappa\alpha,\nu}(\mathbf{q})$ with frequencies $\omega_{\mathbf{q}\nu}$:

$$\Delta\tau_{\kappa\alpha p} = -\frac{2}{N_p}\sum_{\mathbf{q}\nu}B^*_{\mathbf{q}\nu}\left(\frac{\hbar}{2M_\kappa\omega_{\mathbf{q}\nu}}\right)^{1/2}e_{\kappa\alpha,\nu}(\mathbf{q})e^{i\mathbf{q}\cdot\mathbf{R}_p}, \tag{4}$$

where M_κ is the mass of the atom κ and $\mathbf{R}_p$ is the lattice vector of the unit cell p, we can transform Eqs. (1) and (2) into a coupled set of equations for the expansion coefficients in reciprocal space:

$$\frac{2}{N_p}\sum_{\mathbf{q}m\nu}B_{\mathbf{q}\nu}\,g^*_{mn\nu}(\mathbf{k},\mathbf{q})\,A_{m\mathbf{k}+\mathbf{q}} = (\varepsilon_{n\mathbf{k}}-\varepsilon)A_{n\mathbf{k}} \ , \tag{5}$$

$$B_{\mathbf{q}\nu} = \frac{1}{N_p}\sum_{mn\mathbf{k}}A^*_{m\mathbf{k}+\mathbf{q}}\frac{g_{mn\nu}(\mathbf{k},\mathbf{q})}{\hbar\omega_{\mathbf{q}\nu}}A_{n\mathbf{k}} \ . \tag{6}$$

This coupled set of equations can be rewritten more conveniently as follows:

$$\sum_{n'\mathbf{k}'} H_{n\mathbf{k},n'\mathbf{k}'} A_{n'\mathbf{k}'} = \varepsilon A_{n\mathbf{k}} \ , \tag{7}$$

where

$$H_{n\mathbf{k},n'\mathbf{k}'} = \delta_{n\mathbf{k},n'\mathbf{k}'}\varepsilon_{n\mathbf{k}} - \frac{2}{N_p}\sum_{\nu} B^*_{\mathbf{k}-\mathbf{k}'\nu}\, g_{nn'\nu}(\mathbf{k}',\mathbf{k}-\mathbf{k}'). \tag{8}$$

In practice, Eqs. (6)-(8) will be solved iteratively until reaching self-consistency. The final atomic displacements associated with the polaron will be obtained from Eq. (4), and the wave function in real space can be conveniently obtained from the maximally localized wannier functions w as:

$$\psi(\mathbf{r}) = \sum_{mp} A_m(\mathbf{R}_p) w_m(\mathbf{r}-\mathbf{R}_p) \ , \tag{9}$$

having defined

$$A_m(\mathbf{R}_p) = \frac{1}{N_p}\sum_{n\mathbf{k}} e^{i\mathbf{k}\cdot\mathbf{R}_p} U^\dagger_{mn\mathbf{k}} A_{n\mathbf{k}} \ , \tag{10}$$

and

$$\psi_{n\mathbf{k}} = \frac{1}{\sqrt{N_p}}\sum_{mp} e^{i\mathbf{k}\cdot\mathbf{R}_p} U^\dagger_{mn\mathbf{k}} w_m(\mathbf{r}-\mathbf{R}_p) \ , \tag{11}$$

where $U^\dagger_{mn\mathbf{k}}$ is the unitary matrix that generates the smooth Bloch gauge (see Rev. Mod. Phys. **84**, 1419 (2012)).
The polaron formation energy ΔE_f, defined as the energy required to trap a conduction band state with eigenvalue $\varepsilon_{\mathrm{CBM}}$ into a localized polaron, can be obtained from the expansion coefficients that solve Eqs. (5) and (6) by:

$$\Delta E_f = \frac{1}{N_p}\sum_{n\mathbf{k}} |A_{n\mathbf{k}}|^2(\varepsilon_{n\mathbf{k}} - \varepsilon_{\mathrm{CBM}}) - \frac{1}{N_p}\sum_{\mathbf{q}\nu} |B_{\mathbf{q}\nu}|^2 \hbar\omega_{\mathbf{q}\nu} \ . \tag{12}$$

We will refer to the first and second terms on the right hand side as the electron and phonon parts of the formation energy, respectively.
From Eqs. (5) and (6) we observe that the necessary ingredients to solve the polaron equations are the Kohn-Sham eigenvalues, phonon frequencies, and electron-phonon matrix elements on the $\mathbf{k}$- and $\mathbf{q}$-grids corresponding to the equivalent Born-Von Karman supercell in which Eqs. (1) and (2) are defined. In order to obtain the formation energy of an isolated polaron, we will need to solve Eqs. (5) and (6) in increasingly denser grids and extrapolate the results to the infinite supercell limit.
Note:
You are advised to run the calculations in your scratch folder ($SCRATCH). To do so, you can go into your scratch folder and copy the tar file which contains all necessary files for this tutorial, and then extract it:

```
$ cd $SCRATCH
$ cp /work2/06868/giustino/SCHOOL/tutorials/Wed.3.LafuenteLian.tar .
$ tar -xvf Wed.3.LafuenteLian.tar; cd Wed.3.LafuenteLian
```

A quick inspection of the folder will show the two directories containing the files needed in the two exercises, a directory with the pre-completed output files (for reference), and a copy of the pdf of this hands-on:

```
$ ls
exercise1  exercise2  reference  Wed.3.LafuenteLian.pdf
```

Note: in this tutorial it will be shown how to obtain all the input files and related files. The folder that you downloaded should be used as a reference or to speed up the process.

Exercise 1

In this exercise you will calculate and analyze the formation of **hole polarons in LiF**. To start, go to the directory of the first exercise:

```
$ cd exercise1
```

The initial steps of the calculation are similar to the ones followed in the previous tutorials. First, we will need to perform an electronic ground state calculation using `Quantum Espresso`, and a phonon calculation using the `PHonon` code. The corresponding input files (`lif.scf.in` and `lif.ph.in`) are the following:

```
--
&control
    calculation     = 'scf'
    prefix          = 'lif'
    pseudo_dir      = './'
    outdir          = './'
    tprnfor         = .true.
    tstress         = .true.
/
&system
    ibrav           = 2
    celldm(1)       = 7.67034
    nat             = 2
    ntyp            = 2
    ecutwfc         = 80.0
/
&electrons
    conv_thr        = 1.0d-12
/
ATOMIC_SPECIES
 Li 6.941  Li.upf
 F 18.9984 F.upf
ATOMIC_POSITIONS crystal
 Li 0.0000 0.0000 0.0000
 F  0.5000 0.5000 0.5000
K_POINTS automatic
 6 6 6 0 0 0
```

```
--
&inputph
  prefix    = 'lif'
  outdir    = './'
  epsil     = .true.
  zeu       = .true.
  ldisp     = .true.
  fildyn    = 'lif.dyn'
  fildvscf  = 'dvscf'
  tr2_ph    =  1.0d-14,
  nq1       = 6,
  nq2       = 6,
  nq3       = 6
/
```

You can use the following submission job (`submit1.sh`) to perform the calculation in Frontera:

```
#!/bin/bash
#SBATCH -J myjob                # Job name
#SBATCH -p small                # Queue (partition) name
#SBATCH -N 1                    # Total # of nodes
#SBATCH --ntasks-per-node 24
#SBATCH -t 01:00:00             # Run time (hh:mm:ss)
#SBATCH -A EPSchool2022
#SBATCH --reservation EPW-SCHOOL-Wed

# echo loaded modules, current directory, and starting time
module list
pwd
date

# export the path which contains executable file
export PATHQE=/work2/06868/giustino/SCHOOL/q-e-qe-7.2

# Launch MPI code...
# Total # of parallel tasks
MPIOPT="-np "$SLURM_NTASKS
# kpoint parallel groups
KPTPRL="-npool 8"

# run jobs
# scf calculation
ibrun ${MPIOPT} $PATHQE/bin/pw.x ${KPTPRL} -input lif.scf.in > lif.scf.out
# ph calculation
ibrun ${MPIOPT} $PATHQE/bin/ph.x ${KPTPRL} -input lif.ph.in > lif.ph.out

# echo finishing time
date
```

▶ Run a self-consistent calculation and a phonon calculation on a homogeneous $6 \times 6 \times 6$ $\mathbf{k}$ and $\mathbf{q}$-point grid.

```
$ sbatch submit1.sh
```

The calculation should take about 5 minutes to complete.
The keyword fildvscf tells the code to write on file the change of the self-consistent potential due to phonon perturbations, $\partial_{\mathbf{q}\nu} V^{\text{scf}}$, that is needed to compute the electron-phonon matrix elements.
In the output file `lif.dyn0`, you can find the list of 16 irreducible $\mathbf{q}$ points in the Brillouin Zone (IBZ).
If you type `ls`, you can see the `lif.dynX` files containing the dynamical matrix for each irreducible $\mathbf{q}$

point. The dvscf files are all named lif.dvscf1 and are located inside the _ph0/lif.q_X/ folders, except for the one corresponding to the first $\mathbf{q}$ point (Γ) that is located in _ph0/.

▶ Gather the .dyn and .dvscf files into a new save/ directory which EPW will read.

The files in _ph0/lif.phsave/ containing the displacement patterns are also needed. This can easily be done using the pp.py python script which is already included in the EPW distribution:

```
$ python3 /work2/06868/giustino/SCHOOL/q-e-qe-7.2/EPW/bin/pp.py
```

The script will ask you to prompt the prefix of your calculation (in this case lif):

```
Enter the prefix used for PH calculations (e.g. diam)
lif
```

▶ Run a non self-consistent calculation on a full homogeneous $\mathbf{k}$-point grid, and an EPW calculation to obtain the electron-phonon matrix elements in the Wannier representation.

To proceed with the calculation, we need to obtain the Wannier functions corresponding to the valence band manifold using wannier90 as an internal library, and then calculate the electron-phonon matrix elements in the Wannier representation with EPW.
First we need to perform a non self-consistent calculation on a full homogeneous $6 \times 6 \times 6$ $\mathbf{k}$ grid (lif.nscf.in):

```
--
&control
    calculation     = 'bands'
    prefix          = 'lif'
    pseudo_dir      = './'
    outdir          = './'
/
&system
    ibrav           = 2
    celldm(1)       = 7.67034
    nat             = 2
    ntyp            = 2
    ecutwfc         = 80.0
    nbnd            = 15
/
&electrons
    conv_thr        = 1.0d-12
/
ATOMIC_SPECIES
 Li 6.941  Li.upf
 F 18.9984 F.upf
ATOMIC_POSITIONS crystal
 Li 0.0000 0.0000 0.0000
 F  0.5000 0.5000 0.5000
K_POINTS crystal
 216
 0.00000000  0.00000000  0.00000000  4.629630e-03
 0.00000000  0.00000000  0.16666667  4.629630e-03
 0.00000000  0.00000000  0.33333333  4.629630e-03
 ...
```

Note: The **k**-point list under `K_POINTS crystal` in the file above is for illustrative purposes and is not complete. The complete positive-definite homogeneous $6 \times 6 \times 6$ **k**-point between 0 and 1 can be generated by using the script of kmesh.pl included in the `wannier90` package:

```
$ /work2/06868/giustino/SCHOOL/q-e-qe-7.2/wannier90-3.1.0/utility/kmesh.pl 6 6 6
```

You can find the complete list in the input file `lif.nscf.in`.

Next, we need to prepare the `EPW` input file to calculate the electron-phonon matrix elements in the Wannier representation (`lif.epw1.in`):

```
--------
&inputepw
  prefix        = 'lif'
  outdir        = './'

  elph          = .true.
  epwwrite      = .true.
  epwread       = .false.

  lpolar        = .true.
  nbndsub       =  3
  dvscf_dir     = './save/'

  bands_skipped = 'exclude_bands = 1:2, 6:15'
  wannierize    = .true.
  num_iter      = 500
  iprint        = 2

  proj(1)       = 'F:p'

  wannier_plot  = .true.
  wannier_plot_supercell = 6 6 6

  nk1           = 6
  nk2           = 6
  nk3           = 6
  nq1           = 6
  nq2           = 6
  nq3           = 6

  band_plot     = .true.
  filkf         = './path.kpt'
  filqf         = './path.kpt'
/
```

As you can observe in the input file, we will consider three Wannier functions corresponding to the three bands forming the isolated valence band manifold below the Fermi level. In accordance with their orbital character, we will use three p-orbitals centered at the F-atom as the initial projections. You can now submit the calculation:

```
$ sbatch submit2.sh
```

Note: The submission script `submit2.sh` can be obtained from the previous `submit1.sh` by replacing the two lines containing `ibrun` by the following ones:

```
# nscf calculation
ibrun ${MPIOPT} $PATHQE/bin/pw.x ${KPTPRL} -input lif.nscf.in > lif.nscf.out
# epw calculation
ibrun ${MPIOPT} $PATHQE/bin/epw.x -npool 24 -input lif.epw1.in > lif.epw1.out
```

Note: Similar to what you did in the tutorial Tue.5.Lee, you can assess the quality of the Wannierization by comparing the interpolated electron and phonon band structures with the ones calculated using `pw.x` and `matdyn.x`, respectively, as well as by checking the decay of the matrix elements in the Wannier representation as contained in the `decay.*` files.

▶ Interpolate the electron-phonon matrix elements to a fine **k**- and **q**-point grid and solve the self-consistent polaron equations.

First we need to prepare the `EPW` input file to perform the interpolation and solve the polaron equations (`lif.epw2.in`):

```
--------
&inputepw
  prefix        = 'lif'
  outdir        = './'

  elph          = .true.
  epwread       = .true.
  lpolar        = .true.
  nbndsub       =  3
  dvscf_dir     = './save/'

  wannierize    = .false.

  plrn              = .true.
  restart_plrn      = .false.
  type_plrn         = 1
  init_plrn         = 1
  init_sigma_plrn   = 1.0
  niter_plrn        = 500
  conv_thr_plrn     = 1E-4
  ethrdg_plrn       = 1E-5

  nk1           = 6
  nk2           = 6
  nk3           = 6
  nq1           = 6
  nq2           = 6
  nq3           = 6
  nkf1         = 4
  nkf2         = 4
  nkf3         = 4
  nqf1         = 4
  nqf2         = 4
  nqf3         = 4
/
```

You can observe that we will use a $4 \times 4 \times 4$ **k**- and **q**- point grid in this example. The `plrn = .true.` tag activates the polaron calculations, and the `type_plrn = 1` tag indicates that we are dealing with a hole polaron. The self-consistent polaron equations will be initialized with a Gaussian of width $\sigma = 10.0\,\mathrm{bohr}$ (`init_sigma_plrn`), and the solution will be considered converged when the greatest difference in the magnitude of the atomic displacements between two subsequent iterations is lower than $10^{-3}\,\mathrm{bohr}$ (`conv_thr_plrn`)
You can now submit the calculation:

```
$ sbatch submit3.sh
```

Note: The submission script `submit3.sh` can be obtained from the previous `submit2.sh` by replacing the two lines containing `ibrun` by the following one:

```
ibrun ${MPIOPT} $PATHQE/bin/epw.x -npool 24 -input lif.epw2.in > lif.epw2.out
```

You can analyze the iterative self-consistent process in the output:

```
Starting the self-consistent process
-------------------------------------------------------------------------------
 iter      Eigval/eV      Phonon/eV    Electron/eV   Formation/eV       Error/eV
    1     0.6870E+00    -0.2404E+00    -0.3840E+00     0.1435E+00     0.2559E+00
    2     0.2702E+01    -0.1387E+01    -0.7373E+00    -0.6493E+00     0.3938E+00
    3     0.3612E+01    -0.2171E+01    -0.8146E+00    -0.1357E+01     0.1767E+00
    4     0.3712E+01    -0.2268E+01    -0.8162E+00    -0.1452E+01     0.2851E-01
    5     0.3718E+01    -0.2273E+01    -0.8150E+00    -0.1458E+01     0.9334E-02
    6     0.3719E+01    -0.2273E+01    -0.8152E+00    -0.1457E+01     0.3438E-02
    7     0.3721E+01    -0.2274E+01    -0.8153E+00    -0.1459E+01     0.3735E-02
    8     0.3723E+01    -0.2275E+01    -0.8154E+00    -0.1459E+01     0.2395E-02
    9     0.3724E+01    -0.2275E+01    -0.8155E+00    -0.1460E+01     0.1606E-02
   10     0.3724E+01    -0.2276E+01    -0.8155E+00    -0.1460E+01     0.1152E-02
   11     0.3725E+01    -0.2276E+01    -0.8155E+00    -0.1461E+01     0.7113E-03
   12     0.3725E+01    -0.2276E+01    -0.8155E+00    -0.1461E+01     0.4235E-03
   13     0.3725E+01    -0.2276E+01    -0.8155E+00    -0.1461E+01     0.2483E-03
   14     0.3725E+01    -0.2276E+01    -0.8155E+00    -0.1461E+01     0.1449E-03
-------------------------------------------------------------------------------
End of self-consistent cycle
```

All the information related to the energetics of the converged polaron solution is printed at the end of the self-consistent process:

```
           Eigenvalue (eV):        3.7251434
          Phonon part (eV):       -2.2763735
        Electron part (eV):        0.8155442
     Formation Energy (eV):       -1.4608292
```

The breakdown of the formation energy in the different terms is briefly explained in the introduction. More details and further discussion can be found in Phys. Rev. B **99**, 235139 (2019).
A quick inspection of the output files will show that several files have been written with the `.plrn` extension. The most relevant output files for this exercise are `Amp.plrn` and `dtau.plrn`, which contain the polaron wave function coefficients in the Wannier basis and the atomic displacements, respectively. These files are needed to postprocess and analyze the polaron solution in more detail.

▶ Plot and analyze polaron wave function.

The post-processing of the polaron solution can be activated by the following input file (`lif.epw3.in`):

```
--------
&inputepw
  prefix        = 'lif'
  outdir        = './'

  elph          = .true.
  epwread       = .true.
  lpolar        = .true.
  nbndsub       =  3
  dvscf_dir     = './save/'

  wannierize    = .false.

  plrn              = .true.
  restart_plrn      = .true.
  type_plrn         = 1

  cal_psir_plrn     = .true.
  interp_Ank_plrn   = .true.
  interp_Bqu_plrn   = .true.
  filkf             = './path.kpt'
  filqf             = './path.kpt'

  nk1           = 6
  nk2           = 6
  nk3           = 6
  nq1           = 6
  nq2           = 6
  nq3           = 6

  nkf1          = 4
  nkf2          = 4
  nkf3          = 4
  nqf1          = 4
  nqf2          = 4
  nqf3          = 4
/
```

The input tag `restart_plrn = .true.` skips the self-consistent solution of the polaron equations and triggers the post-processing of the previously converged solution by reading the `Amp.plrn` and `dtau.plrn` files.
The input tag `cal_psir_plrn = .true.` calls the subroutine to plot the polaron wave function in real space, which is contained in the `psir_plrn.xsf` file. This file can be plotted with `VESTA`. First submit the calculation:

```
$ sbatch submit4.sh
```

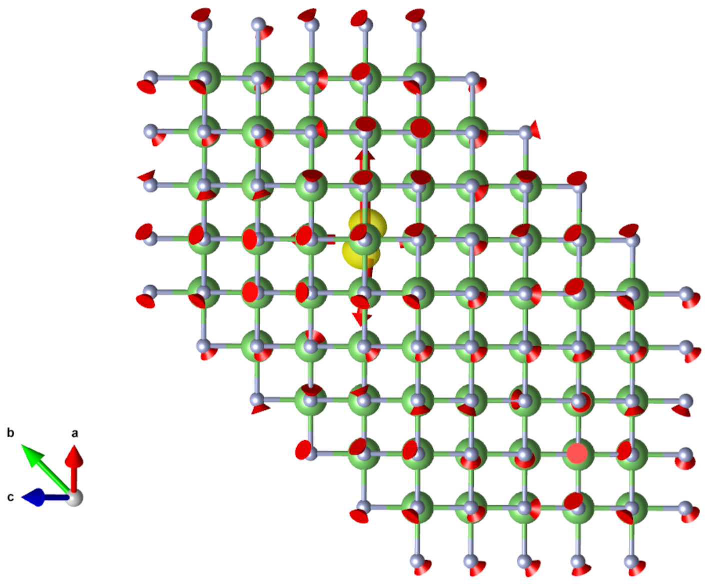
and then copy the file containing the wave function to your own computer:

```
$ scp username@frontera.tacc.utexas.edu:path_to_exercise/psir_plrn.xsf .
```

As when logging in, enter your TACC password at the password prompt. At the TACC Token prompt, enter your 6-digit code. Then you can visualize the polaron wave function by running `VESTA` (https://jp-minerals.org/vesta/en/download.html) locally on your computer:

```
(local)$ VESTA psir_plrn.xsf
```

You should obtain something similar to this:



The polaron wave function is plotted as an isosurface, and the red arrows indicate the magnitude and direction of the atomic displacements in the polaronic configuration. You can observe that the $4 \times 4 \times 4$ **k**-grid transforms to a $4 \times 4 \times 4$ supercell in real space, with periodic boundary conditions. From this figure it is also appreciated that holes form small polarons in LiF, localized in one unit cell, and have the shape of a p-orbital centered on a F-atom.
Next we will analyze the polaron wave function coefficients in the single-particle basis of the DFT-Kohn Sham states. This calculation is activated by the `interp_Ank_plrn = .true.` tag. Similarly, the calculation of the expansion coefficients of the atomic displacements in the phonon basis is

activated by `interp_Bqu_plrn = .true.` . The files containing the **k**- and **q**-points in which the polaron coefficients will be interpolated have to be indicated in the `filkf` and `filqf` input tags, respectively. In this example we will use a path along the high-symmetry W-L-Γ-X-W-K points, as contained in the `path.kpt` file.
The output files containing the interpolated coefficients are `Ank.band.plrn` and `Bmat.band.plrn`. You can visualize your results by using, for example, the following script (`plot_Ank.py`):

```
## Script to plot Ank coefficients
import matplotlib as mpl
mpl.use('pdf')
import matplotlib.pyplot as plt
import numpy as np
# change font on mathematical expressions on plots
mpl.rcParams['mathtext.fontset'] = 'cm'

# Read kpath file
kpath = np.loadtxt('path.kpt', skiprows=1)

# Read Ank file
ik, ibnd, ek0, ReAnk, ImAnk, AbsAnk = np.loadtxt('Ank.band.plrn',
                                                  unpack=True, skiprows=1)

# Separate data in different bands
nbnd=np.int(max(ibnd))
maxik=np.int(max(ik))
Ak=np.zeros((maxik,nbnd))
ek=np.zeros((maxik,nbnd))
iklist=np.zeros((maxik,nbnd))
for i in range(maxik):
    for ibnd in range(nbnd):
        Ak[i][ibnd]=AbsAnk[i*nbnd+ibnd]
        ek[i][ibnd]=ek0[i*nbnd+ibnd]
        iklist[i][ibnd]=i

# Get vbm and bandwidth
vbm=max(ek[:,nbnd-1])
bandwidth=vbm-min(ek[:,0])

## Plot bands
f, ax = plt.subplots(figsize=(10,6))
ax.plot(iklist, ek, color='blue')

# Plot Ank
ax.scatter(iklist, ek, 100*Ak, color='gold', edgecolors='gray', alpha=0.8,
           label=r'$|A_{n\mathbf{k}}|$')

# Set tick params etc.
ax.set_ylim(-bandwidth-0.1*bandwidth,0.1*bandwidth)
ax.set_xlim((min(iklist[:,0]),max(iklist[:,0])))
ax.set_xticklabels([])
ax.tick_params(axis='x', color='black', labelsize='0', pad=0, length=0, width=0)
ax.tick_params(axis='y', color='black', labelsize='18', pad=5, length=5, width=1)
ax.set_ylabel(r'$E-E_{\mathrm{VBM}} ~ (\mathrm{eV})$', fontsize=25, labelpad=10)
ax.legend(loc='upper right', fontsize=25)

plt.savefig('Ank.pdf')
#plt.show()
```

Note: You can find a similar script to plot the $B_{\mathbf{q}\nu}$ coefficients in the example directory (`plot_Bqv.py`).

Executing these scripts:

```
$ python3 plot_Ank.py
$ python3 plot_Bqv.py
```

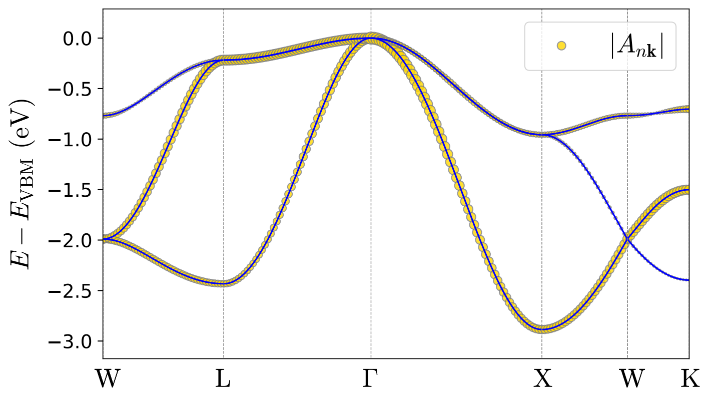
should generate two `.pdf` output files (`Ank.pdf` and `Bqv.pdf`) containing the corresponding figures. To visualize the plots, copy the output files to your own computer (enter your TACC password and token as before):

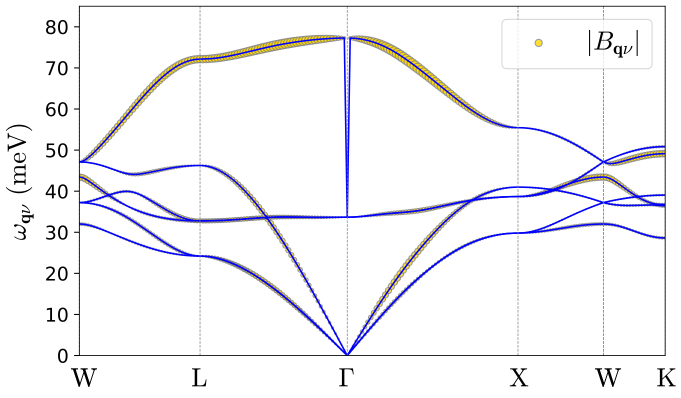
```
$ scp username@frontera.tacc.utexas.edu:path_to_exercise/*.pdf .
```

and open them with your preferred `.pdf` file reader. For example, in Linux you can type:

```
(local)$ evince Ank.pdf
(local)$ evince Bqv.pdf
```

Your figures should be similar to:





As you can observe in these results, due to the strongly localized nature of the hole polaron in LiF, its wave function coefficients in the Bloch basis are spread across the entire Brillouin zone (see Eqs. (3) and (4) in the introduction).

▶ Study the change of the polaron formation energy as a function of the supercell size, and extrapolate results to isolated polaron in an infinite supercell.

In the previous calculation we have seen that the $4 \times 4 \times 4$ **k**- and **q**-point grid in momentum space corresponds to a $4 \times 4 \times 4$ supercell in real space. Due to the interaction between replicated polarons in the neighbouring supercells, the formation energy of the polaron will depend on the momentum grid (supercell) size.
You can analyze the dependence of the polaron formation energy on the supercell size by using the following script (`submit5.sh`):

```
#!/bin/bash
#SBATCH -J myjob                # Job name
#SBATCH -p small                # Queue (partition) name
#SBATCH -N 1                    # Total # of nodes
#SBATCH --ntasks-per-node 48
#SBATCH -t 00:30:00             # Run time (hh:mm:ss)
#SBATCH -A EPSchool2022
#SBATCH --reservation EPW-SCHOOL-Wed

# echo loaded modules, current directory, and starting time
module list
pwd
date

# export the path which contains executable file
export PATHQE=/work2/06868/giustino/SCHOOL/q-e-qe-7.2

# Launch MPI code...

echo "#    nk      Eform" > "E_vs_nk.dat"

for i in 6 8 10 12

do

sed -e "30s/.*/  nkf1          = $i/" \
    -e "31s/.*/  nkf2          = $i/" \
    -e "32s/.*/  nkf3          = $i/" \
    -e "33s/.*/  nqf1          = $i/" \
    -e "34s/.*/  nqf2          = $i/" \
    -e "35s/.*/  nqf3          = $i/" \
    "lif.epw4.in" > "lif.epw4.$i.in"

# run jobs
j=$(( 4*$i ))
ibrun -np $j $PATHQE/bin/epw.x -npool $j -input lif.epw4.$i.in > lif.epw4.$i.out
```

```
grep 'Formation Energy (eV):' "lif.epw4.$i.out" >> "E_vs_nk.dat"

sed -i "s/.Formation Energy (eV):/ $i/" "E_vs_nk.dat"

done

# echo finishing time
date
```

You can submit this job by:

```
$ sbatch submit5.sh
```

For each of the desired **k**-point grids (in this case 6^3, 8^3, 10^3 and 12^3), this script creates a copy of the input file `lif.epw4.in` with modified `nkf` and `nqf` input variables into `lif.epw4.*.in`, launches the calculation, and copies the resulting formation energy to the `E_vs_nk.dat` file:

```
#    nk      Eform
     6       -1.3921241
     8       -1.6052681
     10      -1.7364374
     12      -1.8107382
```

You can plot the formation energy as a function of the inverse supercell size (here defined as L^{-1} where L^3 is the supercell volume) and extrapolate to the infite supercell by using, for example, the following script (`plot_E_s_nk.py`):

```
import matplotlib as mpl
mpl.use('pdf')
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator,FormatStrFormatter
import numpy as np

# Read E vs nk
Nk, Eform = np.genfromtxt('E_vs_nk.dat', unpack=True)

# Set unit cell volume to convert Nk to inverse supercell size
ucell_volume = 112.8044

# Start figure
fig, ax = plt.subplots(figsize=(12,8))

# Plot Eform
ax.scatter(1/(Nk*ucell_volume**(1/3)), Eform, s=50, marker='o',
           color='darkred', edgecolors='black')

# Perform linear fit and plot
mf, bf = np.polyfit(1/(Nk*ucell_volume**(1/3)), Eform, 1)
```

```
xlist = np.linspace(0.0, np.max(1/(Nk*ucell_volume**(1/3))), 100)
print("Extrapolation to isolated polaron formation energy = ", "%.3f" % bf, "eV")
ax.plot(xlist, mf*xlist+bf, '--', color='gray')

# Set tick params etc.
ax.set_xlabel(r'Inverse supercell size ($\mathrm{bohr}^{-1}$)',fontsize=20)
ax.set_ylabel('Formation energy (eV)',fontsize=20, labelpad=5)
ax.tick_params(axis='x', color='black', labelsize='20', pad=5, length=5, width=2)
ax.tick_params(axis='y', color='black', labelsize='20', pad=5, length=5, width=2,
               right=True)
ax.yaxis.set_minor_locator(MultipleLocator(0.1))
ax.tick_params(axis='y', which='minor', color='black', labelsize='20', pad=5,
               length=3, width=1.2, right=True)
ax.set_xlim(0.0, np.max(1/(Nk*ucell_volume**(1/3)))+0.01)
ax.set_title('LiF hole polaron', fontsize=20)

plt.savefig("E_vs_nk.pdf")
#plt.show()
```

Executing this script :

```
$ python3 plot_E_vs_nk.py
```

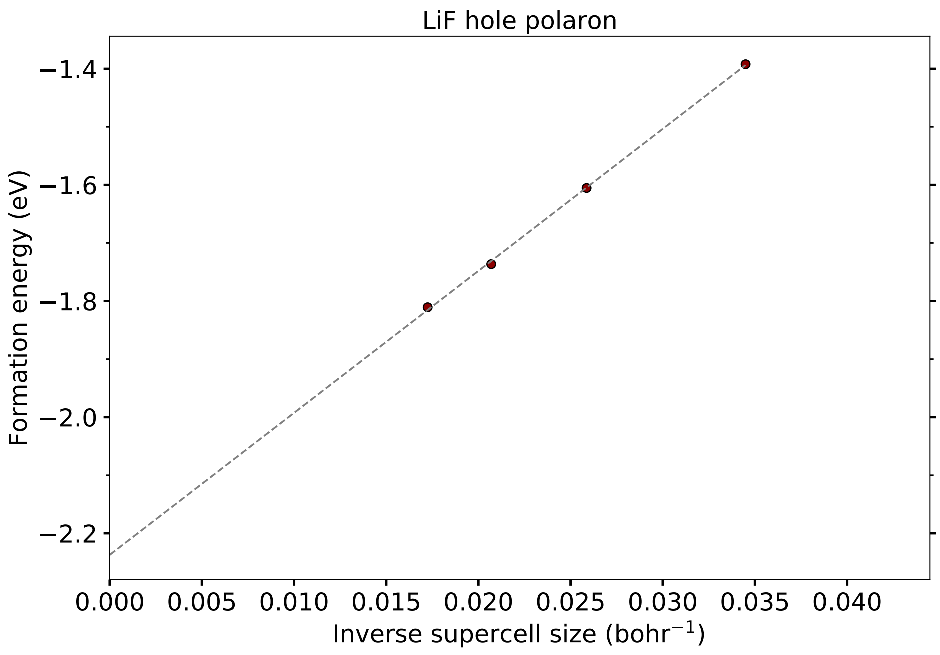
copying the output file to your own computer:

```
$ scp username@frontera.tacc.utexas.edu:path_to_exercise/E_vs_nk.pdf .
```

and opening the `.pdf` file locally:

```
(local)$ evince E_vs_nk.pdf
```

You should obtain something similar to this:



In this example, the extrapolation to infinite supercell gives a formation energy of $\Delta E_f = -2.237$ eV for the isolated hole polaron in LiF. Note that the present results are not fully converged (see Phys. Rev. B **99**, 235139 (2019) for fully converged calculation parameters).

Exercise 2

In this exercise you will repeat a similar procedure as before, but this time to calculate and analyze the formation of **electron polarons in LiF**.
Start by moving to the directory for this exercise, and creating symbolic links to the `lif.save` and `save` directories generated in the previous exercise, so that we don't need to run the scf, ph, and nscf calculations again:

```
$ cd ../exercise2/
$ ln -s ../exercise1/lif.save/ .
$ ln -s ../exercise1/save/ .
```

You can follow the calculation steps one by one as in Exercise 1, and analyze the intermediate results. Alternatively, you can submit the following script that will perform all the calculations automatically (`submit1.sh`):

```
#!/bin/bash
#SBATCH -J myjob                # Job name
#SBATCH -p small                # Queue (partition) name
#SBATCH -N 1                    # Total # of nodes
#SBATCH --ntasks-per-node 28
#SBATCH -t 01:00:00             # Run time (hh:mm:ss)
#SBATCH -A EPSchool2022
#SBATCH --reservation EPW-SCHOOL-Wed

# echo loaded modules, current directory, and starting time
module list
pwd
date

# export the path which contains executable file
export PATHQE=/work2/06868/giustino/SCHOOL/q-e-qe-7.2

# run jobs

# epw1: g(Re,Rp)
ibrun -np 24 $PATHQE/bin/epw.x -npool 24 -input lif.epw1.in > lif.epw1.out

# epw2: polaron for different nk
echo "#    nk       Eform" > "E_vs_nk.dat"

for i in 6 8 10 12 14

do

sed -e "31s/.*/  nkf1          = $i/" \
    -e "32s/.*/  nkf2          = $i/" \
    -e "33s/.*/  nkf3          = $i/" \
    -e "34s/.*/  nqf1          = $i/" \
    -e "35s/.*/  nqf2          = $i/" \
    -e "36s/.*/  nqf3          = $i/" \
    "lif.epw2.in" > "lif.epw2.$i.in"

# run jobs
j=$(( 2*$i ))
ibrun -np $j $PATHQE/bin/epw.x -npool $j -input lif.epw2.$i.in > lif.epw2.$i.out

grep 'Formation Energy (eV):' "lif.epw2.$i.out" >> "E_vs_nk.dat"

sed -i "s/.Formation Energy (eV):/ $i/" "E_vs_nk.dat"

done
```

Note: Analyze the input file `lif.epw2.in`. You will see that in this case we consider one conduction band for the Wannierization, and that the **electron** polaron calculation is set by the `type_plrn = -1` input tag.

▶ Study the change of the electron polaron formation energy as a function of the supercell size.

You can submit the calculation with:

```
$ sbatch submit1.sh
```

Note: This calculation is computationally more demanding and should take around 20 minutes to complete.
The dependence of the polaron formation energy with respect to the momentum grid (supercell) size in this case is (`E_vs_nk.dat`):

```
#    nk      Eform
     6        0.0000016
     8        0.0000056
     10       0.0127052
     12      -0.0568941
     14      -0.1043104
```

Exercise: Why do we obtain negative formation energies only for supercell sizes larger than $10 \times 10 \times 10$?

As in the previous exercise, we can postprocess and analyze the converged polaron solution by studying its wave function in real space and the distribution of the expansion coefficients in reciprocal space. We will analyze the polaron solution obtained in the last $14 \times 14 \times 14$ **k**-point grid calculation:

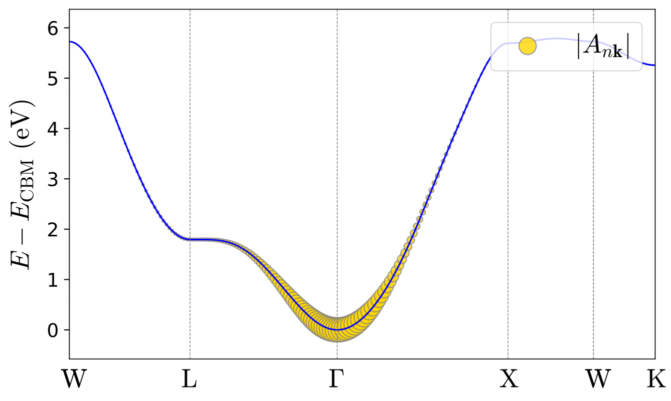
```
$ sbatch submit2.sh
```

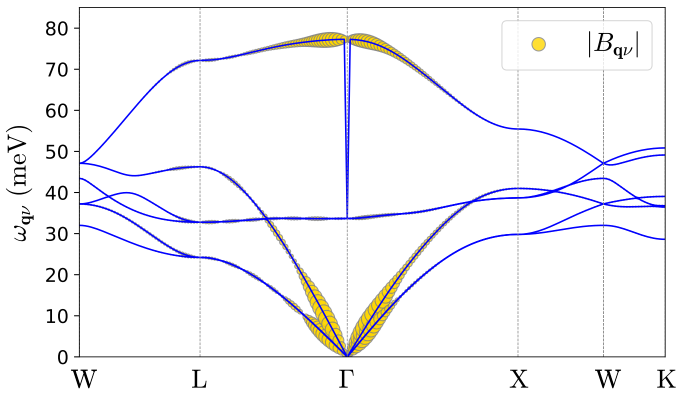
Note: The calculation of the polaron wave function in the large 14×14×14 real space supercell should take around 15 min to complete.
Note: You can obtain the `lif.epw3.in` input file by modifying `lif.epw2.14.in` in a similar way as we have done before for `lif.epw3.in` in Exercise 1. The input tag `step_wf_grid_plrn = 2` reduces the amount of real-space points

in which the polaron wave function is computed, and it is only used here to reduce the computational time. You can obtain the `submit2.sh` submission file by copying `submit4.sh` from `../exercise1/`

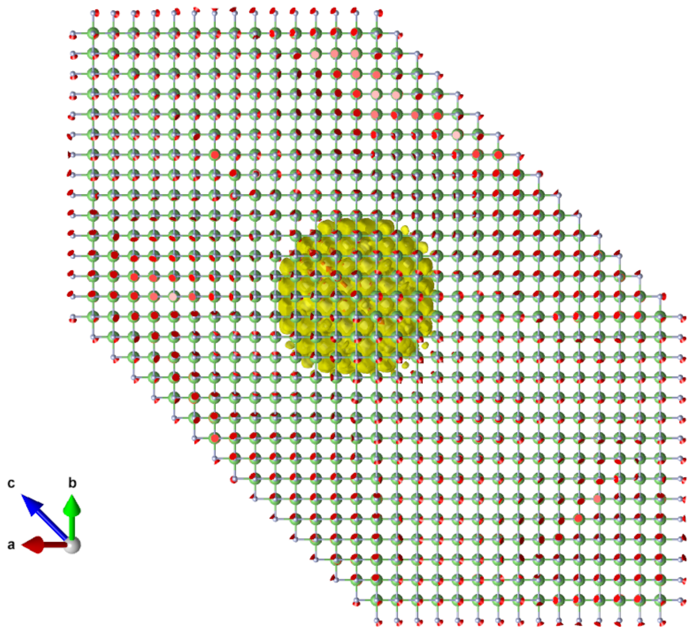
The distribution of the polaron wave function coefficients along the high-symmetry W-L-Γ-X-W-K points should be similar to:

Note: You can obtain plot the polaron wave function coefficients by copying `plot_Ank.py` and `plot_Bqv.py` from `../exercise1/`, and modifying the following lines in `plot_Ank.py`: `ax.set_ylim(-0.1*bandwidth,bandwidth+0.1*bandwidth)` `ax.set_ylabel(r'$E-E_{\mathrm{CBM}} ~ (\mathrm{eV})$', fontsize=25, labelpad=10)`
Note: The small oscillations that might appear in the plot for the $B_{\mathbf{q}\nu}$ coefficients are a spurious effect due to the fact that the $14 \times 14 \times 14$ $\mathbf{k}$-point grid in which the polaron equations are solved is not large enough. They vanish for denser $\mathbf{k}$-point grids.





And the real space plot of the polaron wave function should look similar to:



As you can observe in these results, electrons in LiF form **large** polarons in real space, which correspond to wave function coefficients localized around the conduction band bottom.