2024 School on Electron-Phonon Physics, Many-Body Perturbation Theory, and Computational Workflows

Self-trapped polarons with EPW

Hands-on session (Fri.6)

Hands-on based on QE-v7.3 and EPW-v5.8

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:

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

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

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 $\hat{H}^0_{\rm KS}$ and $V^0_{\rm KS}$ 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:

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

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

In these expressions, $\varepsilon_{n\mathbf{k}}$ are the Kohn-Sham eigenvalues, $\omega_{\mathbf{q}\nu}$ are the phonon frequencies and $g_{mn\nu}(\mathbf{k},\mathbf{q})$ are the electron-phonon matrix elements. All these quantities can be obtained by standard DFT and DFPT calculations.
Eqs. (5) and (6) 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.

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 /work2/05193/sabyadk/stampede3/EPWSchool2024/tutorials/Fri.6.Lafuente.tar .
$ tar -xvf Fri.6.Lafuente.tar; cd Fri.6.Lafuente
```

A quick inspection of the folder will show the two directories containing the files needed in the two exercises and a copy of the `pdf` of this hands-on:

```
$ ls
exercise1  exercise2  exercise3  exercise4  Fri.6.Lafuente.pdf
```

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-16
/
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-16,
```

```
  nq1       = 6,
  nq2       = 6,
  nq3       = 6
/
```

You can use the following submission job (`submit1.sh`) to perform the calculation in Stampede3:

```
#!/bin/bash
#SBATCH -J myjob              # Job name
#SBATCH -N 1                  # Total # of nodes
#SBATCH --ntasks-per-node 24  # Tasks/node
#SBATCH -t 00:30:00           # Run time (hh:mm:ss)
#SBATCH -A DMR23030           # Project/Allocation name
#SBATCH -p spr                # Queue (partition) name
##SBATCH --reservation=NSF_Summer_School_Fri

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

# export the path which contains executable file
export PATHQE=/work2/05193/sabyadk/stampede3/EPWSchool2024/q-e

# 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
```

Note1: These coarse $\mathbf{k}$ and $\mathbf{q}$ point grids have been chosen with a reasonable accuracy versus computational cost in mind. A fully converged calculation might require the use of more $\mathbf{k}$ and $\mathbf{q}$ points in the coarse grid.

This calculation should take about 3 minutes to complete.
The keyword `fildvscf` tells the code to write on file the change of the self-consistent potential due to phonon perturbations, $\partial_{\mathbf{q}\nu} V^{\mathrm{scf}}$, which 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 /work2/05193/sabyadk/stampede3/EPWSchool2024/heap/q-e/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-16
/
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:

```
$ /work2/05193/sabyadk/stampede3/EPWSchool2024/heap/q-e/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.

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

  etf_mem       = 0

  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 comprising 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. All the input variables for this and other EPW calculations can be found at https://docs.epw-code.org/doc/Inputs.html.

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.4.Lafuente, 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/'

  etf_mem        = 0

  wannierize     = .false.

  plrn               = .true.
  restart_plrn       = .false.
  type_plrn          = 1
  init_plrn          = 1
  init_sigma_plrn    = 1.0
  niter_plrn         = 500
```

```
  conv_thr_plrn     = 5E-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 $\sigma = 10.0$ 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} bohr (`conv_thr_plrn`). The input variable `ethrdg_plrn` controls the convergence threshold (Ry) used in the iterative diagonalization at each step in the self-consistent solution of the polaron equations.
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     0.3776E+00    -0.1363E+00    -0.2245E+00     0.8814E-01     0.1675E+00
    2     0.1745E+01    -0.8186E+00    -0.5818E+00    -0.2368E+00     0.2521E+00
    3     0.3271E+01    -0.1862E+01    -0.7846E+00    -0.1078E+01     0.3015E+00
    4     0.3675E+01    -0.2227E+01    -0.8173E+00    -0.1409E+01     0.1168E+00
    5     0.3731E+01    -0.2275E+01    -0.8172E+00    -0.1458E+01     0.3967E-01
    6     0.3742E+01    -0.2282E+01    -0.8158E+00    -0.1466E+01     0.1879E-01
    7     0.3747E+01    -0.2285E+01    -0.8165E+00    -0.1468E+01     0.1315E-01
    8     0.3748E+01    -0.2285E+01    -0.8169E+00    -0.1469E+01     0.1303E-02
    9     0.3750E+01    -0.2286E+01    -0.8145E+00    -0.1472E+01     0.2168E-02
   10     0.3750E+01    -0.2286E+01    -0.8149E+00    -0.1471E+01     0.8140E-02
```

```
   11      0.3751E+01     -0.2286E+01     -0.8151E+00     -0.1471E+01      0.5499E-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):        3.7507486
        Phonon part (eV):        -2.2863546
      Electron part (eV):         0.8150941
   Formation Energy (eV):        -1.4712605
```

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/'

  etf_mem       = 0

  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 will be printed in the `psir_plrn.xsf` file. This file can be plotted with `VESTA`. First submit the calculation:

```
$ sbatch submit4.sh
```

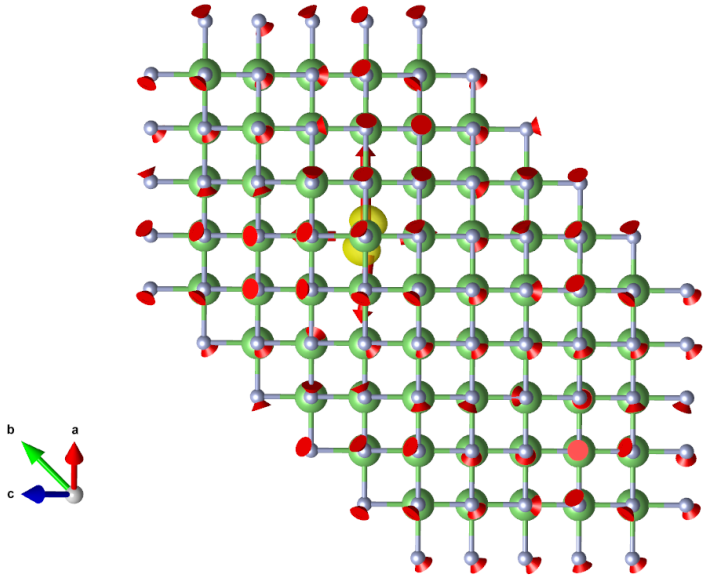
and then copy the file containing the wave function to your own computer:

```
$ scp username@stampede3.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=int(max(ibnd))
maxik=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:

```
$ module load python/3.9.18
$ pip install matplotlib
$ python3 plot_Ank.py
$ python3 plot_Bqv.py
```

should generate two `.pdf` output files (`Ank.pdf` and `Bqv.pdf`) containing the corresponding figures. You can visualize the plots in Stampede3 if you have accessed with X11 forwarding:

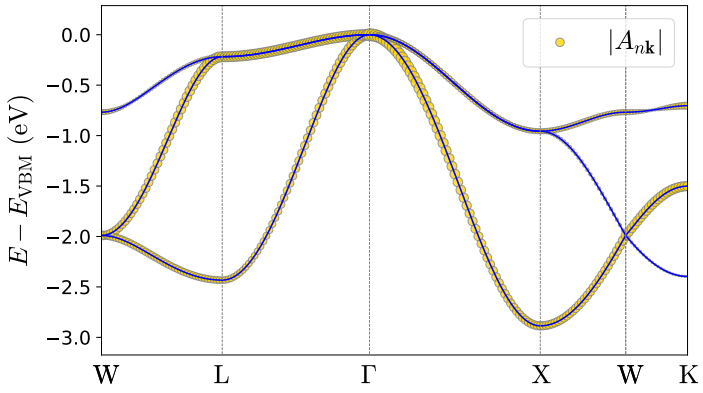
```
$ evince Ank.pdf
$ evince Bqv.pdf
```

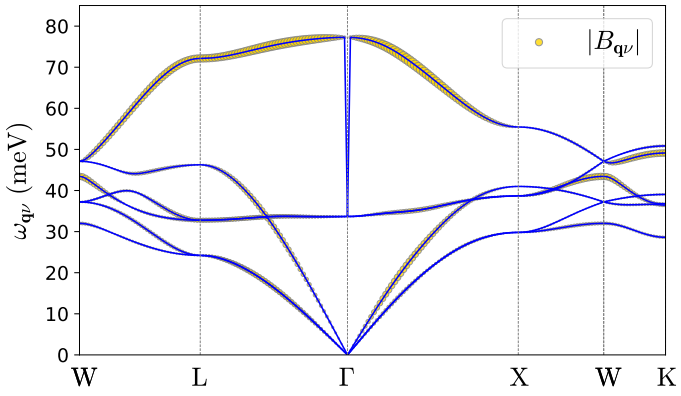
Note: Alternatively, you can copy the output files to your own computer (enter your TACC password and token as before):

```
$ scp username@stampede3.tacc.utexas.edu:path_to_exercise/*.pdf .
```

and open them with your preferred `.pdf` file reader.

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 an 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 -N 1                   # Total # of nodes
#SBATCH --ntasks-per-node 48   # Tasks/node
#SBATCH -t 00:30:00            # Run time (hh:mm:ss)
#SBATCH -A DMR23030            # Project/Allocation name
#SBATCH -p spr                 # Queue (partition) name
##SBATCH --reservation=NSF_Summer_School_Fri

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

# export the path which contains executable file
export PATHQE=/work2/05193/sabyadk/stampede3/EPWSchool2024/q-e

# Launch MPI code...

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

for i in 6 8 10 12
```

```
do

sed -e "32s/.*/  nkf1          = $i/" \
    -e "33s/.*/  nkf2          = $i/" \
    -e "34s/.*/  nkf3          = $i/" \
    -e "35s/.*/  nqf1          = $i/" \
    -e "36s/.*/  nqf2          = $i/" \
    -e "37s/.*/  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       -1.3827861
     8       -1.6075816
     10      -1.7361292
     12      -1.8077436
```

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()
```

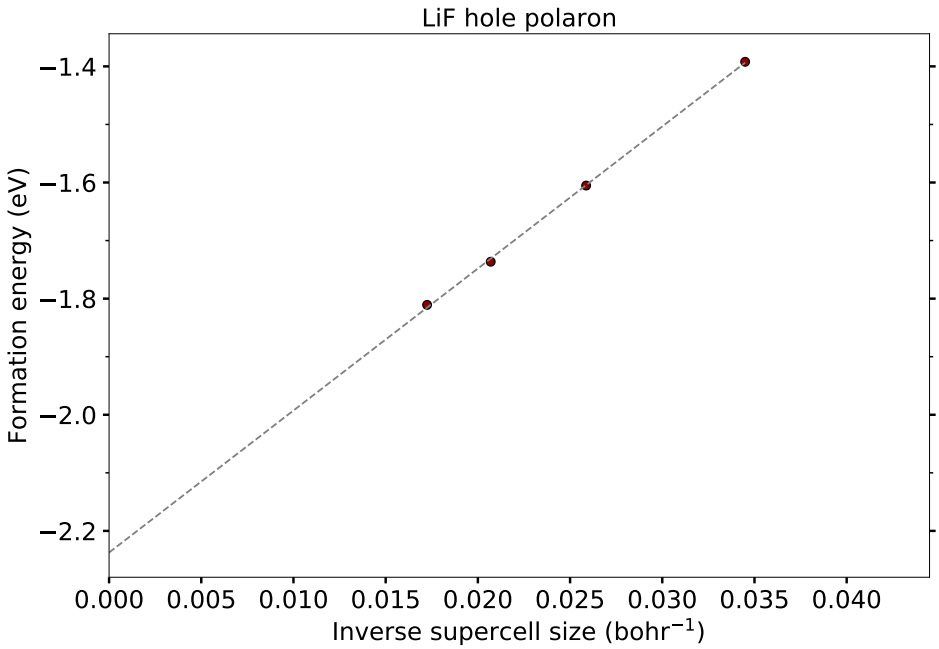
Executing this script :

```
$ python3 plot_E_vs_nk.py
```

and opening the .pdf file:

```
$ 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 $\Delta E_f = -2.244$ 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 do not 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 -N 1                   # Total # of nodes
#SBATCH --ntasks-per-node 24   # Tasks/node
#SBATCH -t 00:30:00            # Run time (hh:mm:ss)
#SBATCH -A DMR23030            # Project/Allocation name
#SBATCH -p spr                 # Queue (partition) name
##SBATCH --reservation=NSF_Summer_School_Fri

# echo loaded modules, current directory, and starting time
```

```
module list
pwd
date

# export the path which contains executable file
export PATHQE=/work2/05193/sabyadk/stampede3/EPWSchool2024/q-e

# 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 "33s/.*/  nkf1          = $i/" \
    -e "34s/.*/  nkf2          = $i/" \
    -e "35s/.*/  nkf3          = $i/" \
    -e "36s/.*/  nqf1          = $i/" \
    -e "37s/.*/  nqf2          = $i/" \
    -e "38s/.*/  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 10 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         0.0000037
8         0.0000057
10        0.0130390
12       -0.0565812
14       -0.1034906
```

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
```

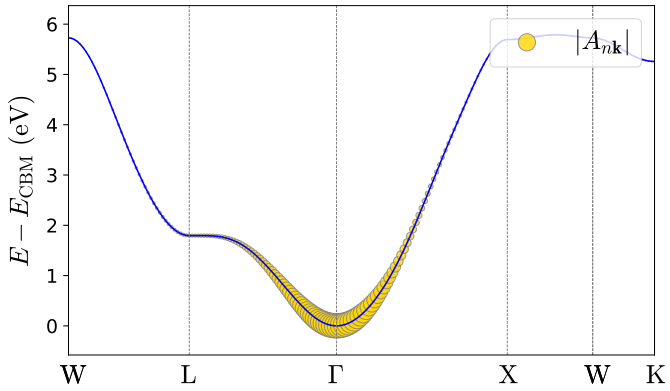
Note: The calculation of the polaron wave function in the large 14×14×14 real space supercell should take around 5 min to complete.

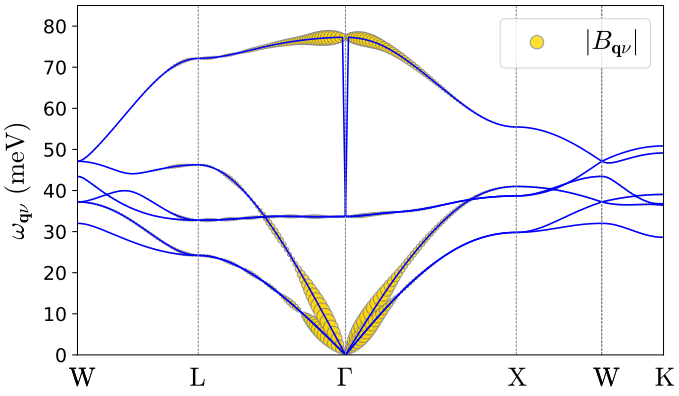
Note: You can obtain the `lif.epw3.in` input file by modifying `lif.epw2.14.in` in a similar way as we have done before for `lif.epw3.in` in Exercise 1. The input tag `step_wf_grid_plrn = 2` reduces the amount of real-space points in which the polaron wave function is computed, and it is only used here 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:

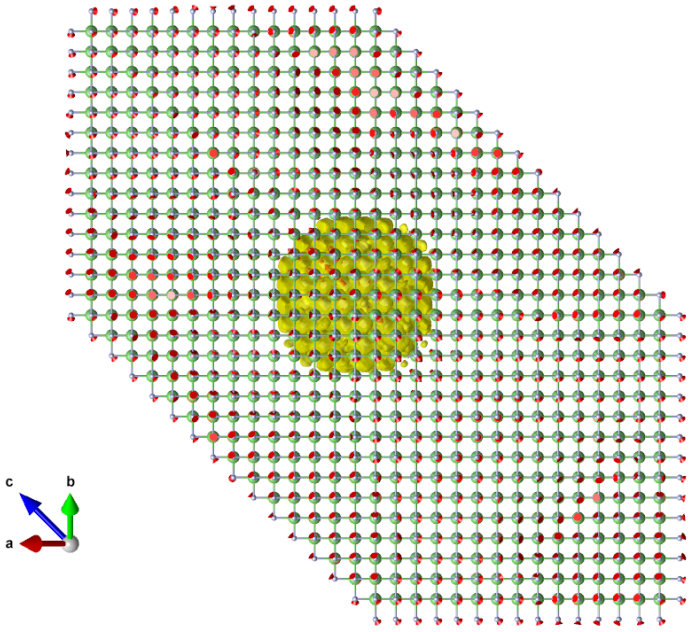
Note: You can obtain plot the polaron wave function coefficients by copying `plot_Ank.py` and `plot_Bqv.py` from `../exercise1/`, and modifying the following lines in `plot_Ank.py`: `ax.set_ylim(-0.1*bandwidth,bandwidth+0.1*bandwidth)` `ax.set_ylabel(r'$E-E_{\mathrm{CBM}} ~ (\mathrm{eV})$', fontsize=25, 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 $14 \times 14 \times 14$ **k**-point grid in which the polaron equations are solved is not large enough. They vanish for denser **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.

Exercise 3

In this exercise, we will illustrate how to perform **polaron calculations in non-diagonal supercells** with any shape. This will be beneficial in some cases, to better understand finite-size effects and to converge calculations in which the shape of the polaron is very different from the shape of the unit cell.

Background and notation

We denote by $\mathbf{a}_{s1}$, $\mathbf{a}_{s2}$, $\mathbf{a}_{s3}$ the primitive lattice vectors of a BvK supercell, and by $\mathbf{a}_{p1}$, $\mathbf{a}_{p2}$, $\mathbf{a}_3$ the primitive lattice vectors of the crystal unit cell. These vectors must be related as follows:

$$\begin{pmatrix} \mathbf{a}_{s1} \\ \mathbf{a}_{s2} \\ \mathbf{a}_{s3} \end{pmatrix} = \begin{pmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \\ S_{31} & S_{32} & S_{33} \end{pmatrix} \begin{pmatrix} \mathbf{a}_{p1} \\ \mathbf{a}_{p2} \\ \mathbf{a}_{p3} \end{pmatrix} , \tag{13}$$

where the matrix elements S_{ij} are integers. The transformation between the corresponding reciprocal lattice vectors is:

$$\begin{pmatrix} \mathbf{b}_{s1} \\ \mathbf{b}_{s2} \\ \mathbf{b}_{s3} \end{pmatrix} = \begin{pmatrix} \bar{S}_{11} & \bar{S}_{12} & \bar{S}_{13} \\ \bar{S}_{21} & \bar{S}_{22} & \bar{S}_{23} \\ \bar{S}_{31} & \bar{S}_{32} & \bar{S}_{33} \end{pmatrix} \begin{pmatrix} \mathbf{b}_{p1} \\ \mathbf{b}_{p2} \\ \mathbf{b}_{p3} \end{pmatrix} \tag{14}$$

where $\mathbf{b}_{s1}$, $\mathbf{b}_{s2}$, $\mathbf{b}_{s3}$ are the primitive reciprocal lattice vectors of the BvK supercell, and $\mathbf{b}_{p1}$, $\mathbf{b}_{p2}$, $\mathbf{b}_3$ are the primitive vectors of the reciprocal lattice of the crystal unit cell. The matrix elements of these two transformations are related by $\bar{S} = (S^{-1})^T$.

When the off-diagonal elements of the matrix S in Eq. (13) are non-zero, one obtains a so-called "non-diagonal" BvK supercell. A non-diagonal supercell differs from a standard supercell in that it does not have the same shape as the primitive unit cell. This observation was exploited in Phys. Rev. B **92**, 184301 (2015) to generate supercells for computing phonons at desired $\mathbf{q}$-vectors from finite-difference calculations. Here, we employ the same strategy but in reverse: first we choose the shape of the BvK supercell, then we use Eq. (14) to determine the Brillouin zone grid that we need in Eqs. (5)-(6) to obtain polarons in such a supercell. A more detailed discussion on polaron calculations in non-diagonal supercells can be found at Proc. Natl. Acad. Sci. USA 121, e2318151121.

Example: FCC primitive to conventional cell

The lattice vectors of the LiF primitive cell are:

$$\begin{cases} \mathbf{a}_{p1} = (-0.5, 0.0, 0.5) \\ \mathbf{a}_{p2} = (0.0, 0.5, 0.5) \\ \mathbf{a}_{p3} = (-0.5, 0.5, 0.0) \end{cases} \tag{15}$$

The lattice vectors corresponding to a conventional cubic supercell of LiF are given by:

$$\begin{cases} \mathbf{a}_{s1} = N(1, 0, 0) \\ \mathbf{a}_{s2} = N(0, 1, 0) \\ \mathbf{a}_{s3} = N(0, 0, 1) \end{cases} \tag{16}$$

where N is a positive integer. Such a supercell is obtained by using Eq. (13) with the transformation matrix:

$$S = \begin{pmatrix} -N & N & -N \\ -N & N & N \\ N & N & -N \end{pmatrix} \tag{17}$$

which contains $|\det S| = 4N$ crystal unit cells. The inverse transformation $\bar{S}$ is given by:

$$\bar{S} = \frac{1}{2N} \begin{pmatrix} -1 & 0 & -1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} . \tag{18}$$

The $\mathbf{k}$-point (and $\mathbf{q}$-point) grids to be used in the solution of (5) and (6) are given by the points $\mathbf{k} = i\mathbf{b}_{s1} + j\mathbf{b}_{s2} + k\mathbf{b}_{s3}$ that fall within the first Brillouin zone of the crystal unit cell, being i, j, k integer numbers and $\mathbf{b}_{s1}$, $\mathbf{b}_{s2}$, and $\mathbf{b}_{s3}$ the BvK supercell reciprocal lattice vectors obtained from (14).

▶ Solve the polaron equations for an electron polaron in a conventional $6 \times 6 \times 6$ cubic supercell of 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 do not need to run the scf, ph, and nscf calculations again:

```
$ cd ../exercise3/
$ ln -s ../exercise2/lif.save/ .
$ ln -s ../exercise2/save/ .
```

You can inspect the `lif.epw2.in` file:

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

  elph          = .true.
  epwread       = .true.
  lpolar        = .true.
  nbndsub       =  1
  dvscf_dir     = './save/'

  etf_mem       = 0

  wannierize    = .false.

  plrn              = .true.
  restart_plrn      = .false.
  niter_plrn        = 500
  type_plrn         = -1
  init_plrn         = 1
  init_sigma_plrn   = 10.0
  niter_plrn        = 500
  conv_thr_plrn     = 1E-4
  ethrdg_plrn       = 1E-6

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

  scell_mat_plrn = .true.
  scell_mat(1, 1:3) = -6,  6, -6
  scell_mat(2, 1:3) = -6,  6,  6
  scell_mat(3, 1:3) =  6,  6, -6
/
```

The calculation of the polaron equations using general non-diagonal supercells is activated by the input flag `scell_mat_plrn`. The size and shape of transformation matrix in Eq. (13) is determined by the input flags `scell_mat(:,:)`. You can notice that no `nkf1, nkf2, nkf3` are specified in the input file. The $\mathbf{k}$-point (and $\mathbf{q}$-point) grids to be used in the solution of (5) and (6) corresponding to the supercell specified in the input will be automatically generated. You can inspect such grid points

by looking at the output file `kgrid.scell.plrn`. Conversely, the list of lattice vectors forming the supercell in real space are written to the output file `Rp.scell.plrn`.

The job can be submitted by following a similar script as in the previous exercises:

```
$ sbatch submit1.sh
```

All the relevant information regarding the supercell transformation matrices is given in the output `lif.epw2.out`:

```
Supercell transformation activated (k), as=S*at
S(1, 1:3):   -6   6  -6
S(2, 1:3):   -6   6   6
S(3, 1:3):    6   6  -6
Transformed lattice vectors (alat units):
as(1, 1:3):     6.000000    0.000000    0.000000
as(2, 1:3):     0.000000    6.000000    0.000000
as(3, 1:3):     0.000000    0.000000    6.000000
Reciprocal lattice transformation matrix, Sbar = (S^{-1})^{t}:
Sbar(1, 1:3):    -0.083333    0.000000   -0.083333
Sbar(2, 1:3):     0.000000    0.083333    0.083333
Sbar(3, 1:3):     0.083333    0.083333    0.000000
Transformed reciprocal lattice vectors (2pi/alat units):
bs(1, 1:3):     0.166667    0.000000    0.000000
bs(2, 1:3):     0.000000    0.166667    0.000000
bs(3, 1:3):     0.000000    0.000000    0.166667

Number of unit cells within supercell:   864
Number of k-points needed:   864
```

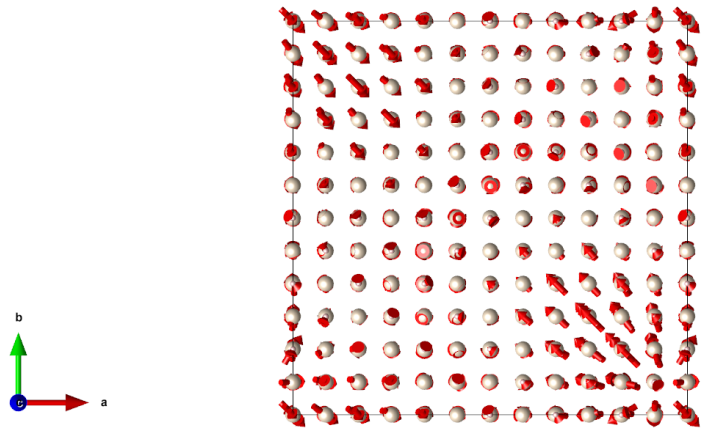
After setting the $\mathbf{k}$ and $\mathbf{q}$ point grids the polaron calculation proceeds as before. You can visualize the conventional cubic cell by plotting the final atomic displacements contained in the `dtau.plrn.xsf` file:

```
$ scp username@stampede3.tacc.utexas.edu:path_to_exercise/dtau.plrn.xsf .
```

Then you can visualize the displacements by running `VESTA` locally on your computer:

```
(local)$ VESTA dtau.plrn.xsf
```

You should obtain something similar to this:



Note that, due to the more symmetric shape of the conventional cubic supercell, we have obtained a localized large polaron solution in a reduced size as compared to the diagonal supercells employed in Exercise 2.

Exercise 4

In this exercise, we will illustrate how to perform calculations of **polaron energy landscapes**. The ground state energy of the polaron for a given configuration of the atomic displacements, $\{\Delta\tau_{\kappa\alpha p}\}$, is obtained by performing the variational minimization of the energy functional with respect to the polaron wave function, leading to (1). The dependence on the atomic displacements allows us to explore the adiabatic potential energy surface of the polaron. An explicit relation between the eigenvalue in (1) and the formation energy can be obtained by:

$$\Delta\tilde{E}_{\rm f} = \varepsilon - \varepsilon_{\rm CBM} + \frac{1}{N_p}\sum_{\mathbf{q}\nu} |B_{\mathbf{q}\nu}|^2 \hbar\omega_{\mathbf{q}\nu} \ , \tag{19}$$

where the $B_{\mathbf{q}\nu}$ coefficients and the atomic displacements are related by Eq. (4). In Eq. (19), we use the tilde symbol in $\Delta\tilde{E}_{\rm f}$ to emphasize that this energy is a minimum with respect to the eletronic degrees of freedom, but the forces on the atoms are generally nonvanishing in this configuration. If we minimize this quantity also with respect to the atomic displacements, then we obtain the formation energy $\Delta E_{\rm f}$ of (12).
In order to explore polaron energy surfaces we proceed as follows: First, we define a given displacement configuration $\{\Delta\tau_{\kappa\alpha p}\}$; then, we perform the transformation from $\{\Delta\tau_{\kappa\alpha p}\}$ to $\{B_{\mathbf{q}\nu}\}$ using (4); we diagonalize the effective Hamiltonian in (7); and we obtain the associated formation energy using (19).

▶ Generate a path along the configuration space.

In this exercise we will consider a path which corresponds to a hopping of hole polarons in LiF across nearest-neighbor sites. Start by moving to the directory for this exercise, and by copying or creating symbolic links to the relevant files and directories generated in Exercise 1:

```
$ cd ../exercise4/
$ ln -s ../exercise1/lif.save/ .
$ ln -s ../exercise1/save/ .
$ cp ../exercise1/path.kpt .
$ cp ../exercise1/lif.epw1.in .
```

The second input file already present in the directory, `lif.epw2.in`, can be substituting the last six lines in `../exercise1/lif.epw2.in` by:

```
  scell_mat_plrn = .true.
  scell_mat(1, 1:3) = -4,  4, -4
  scell_mat(2, 1:3) = -4,  4,  4
  scell_mat(3, 1:3) =  4,  4, -4
```

Now run a self-consistent polaron calculation on a conventional cubic $4 \times 4 \times 4$ supercell:

```
$ sbatch submit1.sh
```

We will use the displacement configuration obtained in this step, which is contained in the `dtau.plrn` output file, as the initial point in our configuration coordinate path. The final configuration will obtained by shifting the original small polaron configuration by one unit cell along the [1, -1, 0] direction, and a linear interpolation between the two configurations will be used to define the hopping path. An automated script to generate the displacement configurations along the hopping path is provided in the directory:

```
$ python3 hopping_path.py
```

This script will output several `dtau_disp.plrn_*` files, which contain the displacement configurations for each of the points along the path. It also outputs a `initial_final_displacements.pdf` file, which contains figures of the initial and final displacement configurations.

▶ Compute the polaron energy landscape along the configuration space.

The input file corresponding to the polaron energy landscape calculation is `lif.epw3.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         = 6
  init_ntau_plrn    = 21
  niter_plrn        = 1
  full_diagon_plrn  = .true.

  nk1           = 6
  nk2           = 6
```

```
  nk3           = 6
  nq1           = 6
  nq2           = 6
  nq3           = 6

  scell_mat_plrn = .true.
  scell_mat(1, 1:3) = -4,  4, -4
  scell_mat(2, 1:3) = -4,  4,  4
  scell_mat(3, 1:3) =  4,  4, -4
/
```

You can notice several differences with respect to the `lif.epw2.in` file in Exercise 1. Firstly, a non-diagonal supercell is employed as in the previous calculation in this Exercise 4. Secondly, the flag `init_plrn` is set to 6. This activates the initialization of the polaron equations using displacement configurations $\{\Delta\tau_{\kappa\alpha p}\}$, which need to be provided as input files (see below). The flag `init_ntau_plrn` indicates the number of displacement configurations to be considered. Following the discussion at the beginning of the exercise, in order to compute the polaron energy landscape we only need to diagonalize the effective polaron Hamiltonian once. Thus, the flag `niter_plrn` is set to 1 and no self-consistency will be sought. Additionally, the flag `full_diagon_plrn=.true.` activates the serial `LAPACK` diagonalization of the effective polaron Hamiltonian, which for small supercells improves accuracy although at a highly increased computational cost. This calculation can be launched by:

```
$ sbatch submit2.sh
```

A quick inspection of the output file `lif.epw3.out` shows all the relevant information about the polaron energetics for each displacement configuration along the configuration coordinate path:

```
--------------------------------------------------------------------------------
iter      Eigval/eV      Phonon/eV    Electron/eV   Formation/eV       Error/eV
    1     0.3818E+01    -0.2304E+01    -0.7999E+00    -0.1504E+01     0.8218E+00
      Eigenvalue (eV):        3.8177375
     Phonon part (eV):       -2.3036678
   Electron part (eV):        0.7998584
   Formation Energy at this \dtau (eV):       -1.5140697
    1     0.3643E+01    -0.2138E+01    -0.7902E+00    -0.1347E+01     0.4070E-01
      Eigenvalue (eV):        3.6428433
     Phonon part (eV):       -2.1376311
   Electron part (eV):        0.7901827
   Formation Energy at this \dtau (eV):       -1.5052122
    1     0.3468E+01    -0.1989E+01    -0.7798E+00    -0.1209E+01     0.4070E-01
      Eigenvalue (eV):        3.4684343
     Phonon part (eV):       -1.9890720
   Electron part (eV):        0.7797832
   Formation Energy at this \dtau (eV):       -1.4793623
    1     0.3295E+01    -0.1858E+01    -0.7686E+00    -0.1089E+01     0.4070E-01
      Eigenvalue (eV):        3.2945769
     Phonon part (eV):       -1.8579903
   Electron part (eV):        0.7685768
   Formation Energy at this \dtau (eV):       -1.4365865
```

...

As you can observe, the polaron energy increases as we move away from the original ground state configuration, and then decreases again as we get closer to the final configuration after the hopping. This information about the energy barrier can be extracted from the output file and plotted with the script `plot_barrier.py`:

```
$ python3 plot_barrier.py
```

The resulting hopping barrier height is given as an output:

```
Hopping barrier (meV): 617.03
```

and a figure with the energy landscape is written to the `hopping_barrier.pdf` file. To visualize the results, copy the output files to your own computer (enter your TACC password and token as before):

```
$ scp username@stampede3.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 initial_final_displacements.pdf
(local)$ evince hopping_barrier.pdf
```

Your figures should be similar to: