SSCHA Tutorial

In this tutorial we will:

- Generate a Fcc structure file (cif) for silver (Ag).
- Harmonic phonons using a EMT force-field with ASE.
- Anharmonic self-consistent phonons for silver.
- Relax the unit-cell at fixed temperature.
- Thermal expansion.

Generation of the structure of Silver.

We use the Atomic Simulation Environment (ASE) to build the cell of silver (FCC).

Run the file `create_structure.py` to generate the cif file of the Ag. This should be the same as `Ag.cif` provided.

Exercize 1. Compute harmonic phonons.

Harmonic phonons can be evaluated within DFPT as illustrated in the previous tutorial. Here we employ a different approach:

$$\Phi_{ab} = -\frac{df_a}{dR_b}$$

where f_a is the force along the a direction (encoding both the atom index and the Cartesian coordinate), while R_b is the position of the b coordinate (encoding both the atom index and component).

Exploiting symmetries, we can reduce the number of independent displacements to be evaluated. In the case of Fcc lattice (Fm$\bar{3}$m symmetry group), we have only 1 independent displacement.

We use cellconstructor to perform this calculation:

```python
import cellconstructor as CC, cellconstructor.Phonons

# Initialize the cellconstructor atomic structure
struct = CC.Structure.Structure()
struct.read_generic_file("Ag.cif")

ag_harmonic = CC.Phonons.compute_phonons_finite_displacements(
    struct,
    EMT(),
    supercell=(4,4,4))
```

The function `compute_phonons_finite_displacements` takes as input the structure, the ASE calculator (to compute the force on the structure), and the supercell on which to evaluate the dynamical matrix. The supercell is equivalent to the phonon q-mesh employed in a DFPT calculation.

To impose the symmetries and the acoustic sum rule:

```python
ag_harmonic.Symmetrize()
```

To save the dynamical matrix in quantum espresso format:

```python
ag_harmonic.save_qe("AgDyn")
```

You find the complete script in `scripts/compute_harmonic_phonons.py` .

As in the quantum espresso format, the dynamical matrix generates many files (13 in this case), each one encoding the dynamical matrix of a separate star of q-points. The total number of q-points matches with the q-mesh (4x4x4 = 64), but only 13 of these are independent by symmetry.

Exercize 2. Plot the phonon dispersion.

At the end of the harmonic calculation, you should have the harmonic dynamical matrix of silver. You can plot the phonon dispersion editing the script `scripts/plot_dispersion.py`

There, you have the PATH in the Brilluin zone, the number of k points and the dynamical matrix.

```
# Let us define the PATH in the brilluin zone and the total number of points
PATH = "GXWXKGL"
N_POINTS = 1000

# Here we define the position of the special points
SPECIAL_POINTS = {"G": [0,0,0],
             "X": [0, .5, .5],
             "L": [.5, .5, .5],
             "W": [.25, .75, .5],
             "K": [3/8., 3/4., 3/8.]}

# The two dynamical matrix to be compared
HARM_DYN = 'AgDyn'
SSCHA_DYN = 'relaxed_300_'

# The number of irreducible q points
# i.e., the number of files in which the phonons are stored
NQIRR = 13
```

This script is ment to compare the dispersion between two dynamical matrices whose name is in `HARM_DYN` and `SSCHA_DYN` . As a simple edit, pass to `SSCHA_DYN` the same file as `HARM_DYN` .

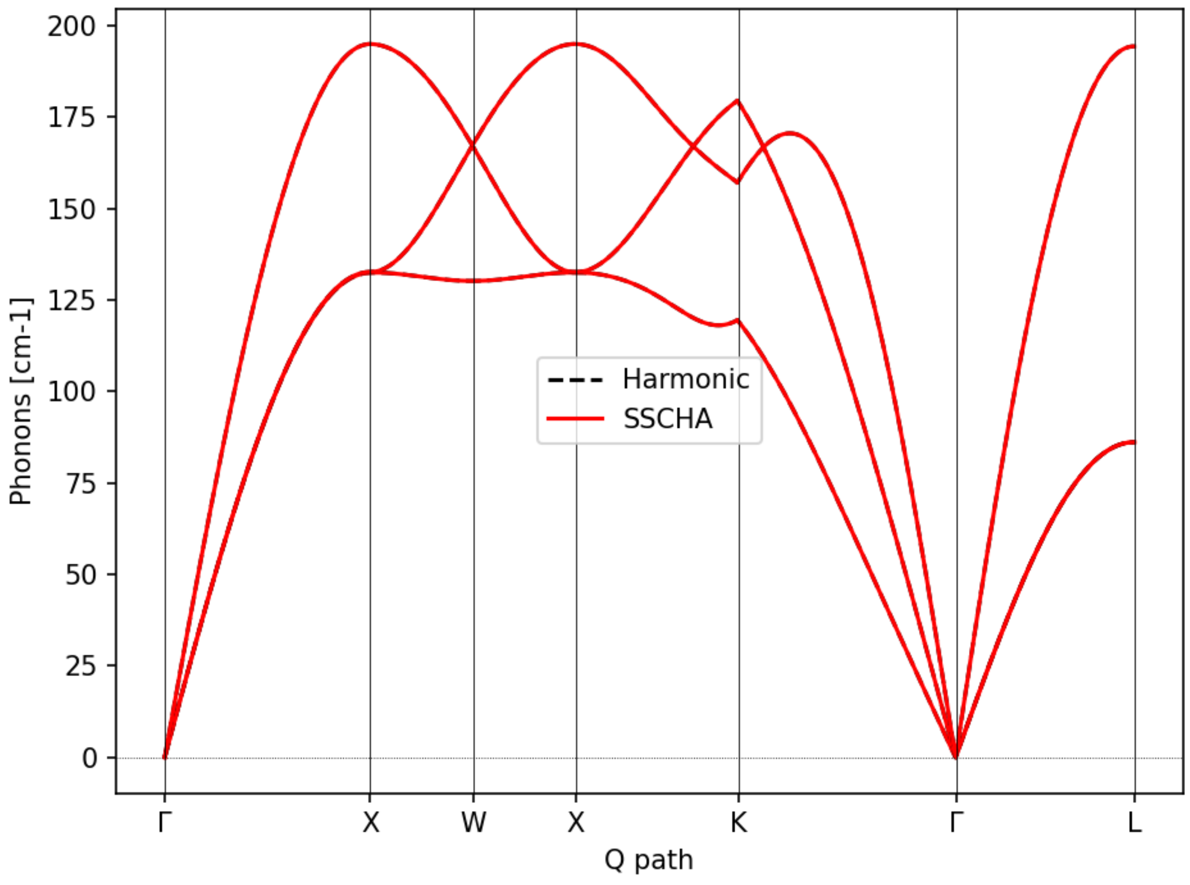

Figure 1: Harmonic phonon dispersion of Ag.

Exercize 3. The SSCHA calculation.

After the harmonic calculation, we can start the SSCHA run. This works similarly to the DFT self-consistent loop, but instead of optimizing the electronic bands, we optimize the phonon bands self-consistently.

To run the SSCHA, we need a positive definite harmonic phonons. Ag should have real harmonic phonons, so nothing to do, but if you simulate more complex materials that may not be the case.

To enforce the dynamical matrix to be positive definite, we use:

```
dyn.ForcePositiveDefinite()
```

The sscha calculation needs the following steps:

- Generate a random ensemble of ionic displaced configurations according to the original dynamical matrix
- Evaluate forces and energies (end stress tensors) for each of the configuration generated.
- Run the free energy minimization to obtain the self-consistent dynamical matrix

Ensemble generation

The ensemble generation is done with the module `sscha.Ensemble.Ensemble`, you need to load the starting dynamical matrix and the temperature.

```
import cellconstructor as CC, cellconstructor.Phonons
import sscha
import sscha.Ensemble, sscha.SchaMinimizer, sscha.Utilities

# Load the starting dynamical matrix
dyn = CC.Phonons.Phonons("AgDyn", nqirr=13)
TEMPERATURE=300 # Kelvin
N_CONFIGS = 100

# Generate the ensemble
ensemble = sscha.Ensemble.Ensemble(dyn, T0=TEMPERATURE)
ensemble.generate(N_CONFIGS)
```

Compute forces and energies.

We use the EMT force field for silver, as we did for the harmonic calculation, to evaluate energies, forces and stress tensors of the configurations in the ensemble.

```
from ase.calculators.emt import EMT

# Compute energies forces and stress for the ensemble
ensemble.compute_ensemble(calculator = EMT(),
        compute_stress=True)
```

If you want to use DFT, you can either configure the ASE calculator for quantum espresso (see this link, or configure a connection with a remote cluster to automatize the submission of all ensemble calculations (see this tutorial for more details).

Submit the Free energy minimization (SSCHA)

```
# Run SSCHA
minim = sscha.SchaMinimizer.SSCHA_Minimizer(ensemble)
minim.meaningful_factor = 0.001

# Setup utilities
ioinfo = sscha.Utilities.IOInfo()
ioinfo.SetupSaving("minimization_300")
```

```
minim.init()
minim.run(custom_function_post=ioinfo.CFP_SaveAll)
minim.finalize()
```