LibAFCC 1.0

User guide

Overview

Conventional plane-wave solvers determine the electrostatic potential of a charge density confined within a finite simulation box by implicitly assuming three-dimensional periodic-boundary conditions at the frontier of the simulation box. LibAFCC (library of auxiliary-function countercharge corrections) allows instead to solve electrostatic problems with plane waves without requiring three-dimensional periodicity.

In practical terms, LibAFCC can address Poisson problems for electrically neutral or charged electrostatic systems that exhibit:

- no periodicity (e.g., molecules and nanoparticles);
- one-dimensional periodicity (e.g., polymers and nanotubes);
- two-dimensional periodicity (e.g., slabs and atomic layers).

LibAFCC has been developed in the context of electronic-structure calculations but can be applied to any type of simulations that require the solution of Poisson's equation with arbitrary electrostatic periodicity. LibAFCC relies on the methodological framework of auxiliary-function corrections.

Citation

The following reference should be cited for any publication employing LibAFCC.

Li Y. L. and Dabo I., Electronic levels and electrical response of periodic molecular structures from plane-wave orbital-dependent calculations, *Physical Review B* **84**, 155127 (2011), DOI: 10.1103/PhysRevB.84.155127

Compilation

The following compilers and libraries can be employed to compile LibAFCC:

- Fortran90: GFORTRAN (recommended), IFORT
- Fourier Transforms: FFTW-3.0
- Mathematical library: LAPACK and BLAS

To compile, edit the Makefile and make.rules files and issue the command

```
$ make clean
$ make afcc
```

Input parameters

The LibAFCC executable is named afcc.x.

```
$ ./afcc.x < filename.in > filename.out
```

A typical input file is presented below:

```
&input
a(1,1)=100.d0
a(2,1)=0.d0
a(3,1)=0.d0
a(1,2)=0.d0
a(2,2)=100.d0
a(3,2)=0.d0
a(1,3)=0.d0
a(2,3)=0.d0
a(3,3)=100.d0
tperiodic(1)=.false.
tperiodic(2)=.false.
tperiodic(3)=.true.
npt(1)=200
npt(2)=200
npt(3)=200
spread=2.5d0
/
```

`a(i,j)` (array of type real) for the *i*th coordinate of the *j*th vector.
`t(i)` (vector of type logical) for indicating whether the *i*th space direction is periodic.
`npt(i)` (vector of type integer) for the grid resolution in the *i*th space direction.
`spread` (variable of type real) for the Gaussian spread to be used.

Output

The beginning of the output file summarizes calculations parameters, the re-indexing of space indexes, and reports the different potentials at the origin.

```
#lattice vectors
#  100.000000000000000      0.0000000000000000      0.0000000000000000
#   0.0000000000000000    100.000000000000000      0.0000000000000000
#   0.0000000000000000      0.0000000000000000    100.000000000000000
#volume
#  1000000.00000000000
#periodicity
# F F T
#grid
#       200       200       200
#periodic dimension
#         1
```

```
#Gaussian spread
#   2.5000000000000000
#lattice unit vectors
#  1.0000000000000000       0.0000000000000000       0.0000000000000000
#   0.0000000000000000      1.0000000000000000       0.0000000000000000
#   0.0000000000000000       0.0000000000000000      1.0000000000000000
#re-indexing
#         1          2          3
#phi1(0)  0.37507277591259530
#phi3(0)  0.42299832699748374
#afc(0)   7.3298529920970701
```

Then, the auxiliary-function correction in real space is written according to the convention.

```
do m=1,npt(1)
do n=1,npt(2)
do p=1,npt(3)
print *, afc(m,n,p)
enddo
enddo
enddo
```