README.md

Examples: Use with PSCF (C++/Cuda)

Examples in this collection illustrate how to use pscfFieldGen for C++/Cuda version calculations. This represents cases with the model file entry `software pscfpp`. Two examples are provided, one in two dimensions, and one in three.

Available Geometries

Morphology	Directory	Dim	Crystal Sys	Space Group
Hexagonally Packed Cylinders	`hex/`	2	hexagonal	p 6 m m
Body Centered Cubic Spheres	`bcc/`	3	cubic	I m -3 m

Running the Examples

In order to run these examples, first make sure that PSCF (C++/Cuda) is installed on your machine, and that `pscfFieldGen` is available as a module in your python environment. When that is done, navigate to the example's directory.

Each example has two automated execution options. The first of these focuses on the pscfFieldGen operation, generating only the KGRID initial guess file and converting it to an RGRID format for visualization. The second option does this and continues on to solve the SCFT equations and converge the field.

Quick Run

The first automated run option is a quick generation of the initial guess and conversion to a visualizer-friendly format. This execution option is automated by the executable script *quickrun* available in the directory, and can be launched by typing

```
./quickrun
```

on the command line from within the example's directory. The script acts as a shorthand for the following commands, which could be used directly

```
python -m pscfFieldGen -f model -t > logs/generation
[ pscf_pc2d  |  pscf_pc3d ] -e -p param -c command_kgrid > logs/conversion
```

In the second line, the bracketed term (`[ pscf_pc2d | pscf_pc3d ]`) should be replaced by the proper command (either `pscf_pc2d` or `pscf_pc3d`, depending on the dimensionality) when executed. The bracket notation is used to list the options to choose between.

The first line of the script uses the traced invocation of pscfFieldGen to generate a KGRID field file ('in/rho_kgrid'). The traced invocation (flag `-t`) causes the program to echo the input read from *model*. After echoing this data, the program subsequently prints the details of the Lattice it will use, the crystal structure it will use, and output notice of key steps in its calculation. The addition of `> logs/generation` at the end of the command redirects this output to a file 'generation' in the sub-directory 'logs' within the example's root directory.

The second line executes PSCF using the parameter file 'param' and command file 'command_kgrid'. This operation convers the KGRID field file data in 'in/rho_kgrid' to RGRID field file format as the file 'in/rho_rgrid'. This file can be used with the visualizer to check the structure of the resulting concentration field. The output from this invocation of pscf is similarly redirected into a log file called `conversion` within the 'logs/' sub-directory.

Full Run

The second automated run option expands on the first and solves the SCFT equations from the generated initial guess. This execution option is automated by the *fullrun* script present in the example directory. It can be invoked by entering

```
./fullrun
```

on the command line from within the example's directory. This script is equivallent to the following commands

```
python -m pscfFieldGen -f model -t > logs/generation
[ pscf_pc2d  |  pscf_pc3d ] -e -p param -c command_kgrid > logs/conversion
[ pscf_pc2d  |  pscf_pc3d ] -e -p param -c command_converge > logs/solution
```

The first two lines in this execution are identical to those in the quick run, and behave identically. The last command executes PSCF using the parameter file 'param' and command file 'command_converge'. This first converts the concentration field in 'in/rho_rgrid' to a basis format file called 'in/rho' and subsequently converts that to a chemical potential field file 'in/omega'. This field is then used as input to an ITERATE command, which converges the field. After convergence, the converged chemical potential field (in basis format) is written to 'out/omega' and the concentration field is written in RGRID format to 'out/rho_rgrid' for visualization. The output from this is redirected into 'logs/solution'.

If the *quickrun* execution has completed and the directory has not yet been cleaned (see the next section), then the *fullrun* execution pathway can be completed simply by executing the third command listed above.

Resetting the Example

Because of the large number of files produced in the 'in/', 'out/', and 'logs/' subdirectories, a script is provided to automatically delete all of these files to reset the example. After running an example, the files generated by it can be removed by entering

```
./clean
```

on the command line. This leaves the example directory empty for another run.