Using PWSCF and QMCPACK to perform total energy calculations of condensed systems*

Luke Shulenburger

July 17, 2014

Abstract

The goal of this lab will be to introduce you to the somewhat specialized problems involved in performing diffusion Monte Carlo calculations on condensed matter as opposed to the atoms and molecules that were the focus of earlier labs. Calculations will be performed on two different systems. Firstly, we will perform a series of calculations on BCC beryllium focusing on the necessary methodology to limit finite size effects. Secondly, we will perform calculations on graphene as an example of a system where qmcpacks ability to handle cases with mixed periodic and open boundary conditions is useful. This example will also focus on strategies to limit memory usage for such systems. All of the calculations performed in this lab will utilize the project suite tools that vastly simplify the process by automating the steps of generating trial wavefunctions and performing DMC calculations.

1 Preliminaries

For any DMC calculation, we must start with a trial wavefunction. As is typical for our calculations of condensed matter, we will produce this wavefunction using density functional theory. Specifically, we will use quantum espresso to generate a slater determinant of single particle orbitals. This is done as a three step process. First, we calculate the converged charge density by performing a DFT calculation with a fine grid of k-points to fully sample the Brilloiun zone. Next, a non-self consistent calculation is performed at the specific k-points needed for the supercell and twists needed in the DMC calculation (more on this later). Finally, a wavefunction is converted from the binary representation used by quantum espresso to the portable hdf5 representation used by qmcpack.

*Sandia National Laboratories is a multiprogram laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under Contract No. DE-AC04-94AL85000.

The choice of k-points necessary to generate the wavefunctions dependes on both the supercell chosen for the DMC calculation and by the supercell twist vectors needed. Recall that the wavefunction in a plane wave DFT calculation is written using Bloch's theorem as:

$$\Psi(\vec{r}) = e^{i\vec{k}\cdot\vec{r}}u(\vec{r}) \tag{1}$$

Where $\vec{k}$ is confined ot the first Brillouin zone of the cell chosen and $u(\vec{r})$ is periodic in this simulation cell. A plane wave DFT calculation stores the periodic part of the wavefunction as a linear combination of plane waves for each single particle orbital at all k-points selected. The symmetry of the system allows us to generate an arbitrary supercell of the primitive cell as follows: Consider the set of primitive lattice vectors, $\{\mathbf{a}_1^p, \mathbf{a}_2^p, \mathbf{a}_3^p\}$. We may write these vectors in a matrix, $\mathbf{L}_p$, whose rows are the primitive lattice vectors. Consider a non-singular matrix of integers, $\mathbf{S}$. A corresponding set of supercell lattice vectors, $\{\mathbf{a}_1^s, \mathbf{a}_2^s, \mathbf{a}_3^s\}$, can be constructed by the matrix product

$$\mathbf{a}_i^s = S_{ij}\mathbf{a}_j^p \tag{2}$$

If the primitive cell contains N_p atoms, the supercell will then contain $N_s = |\det(\mathbf{S})|N_p$ atoms.

Now, the wavefunciton at any point in this new supercell can be related to the wavefunction in the primitive cell by finding the linear combination of primitive lattice vectors that maps this point back to the primitive cell:

$$\vec{r}' = \vec{r} + x\mathbf{a}_1^p + y\mathbf{a}_2^p + z\mathbf{a}_3^p = \vec{r} + \vec{T} \tag{3}$$

where x, y, z are integers. Now the wavefunction in the supercell at point $\vec{r}'$ can be written in terms of the wavefunction in the primitive cell at $\vec{r}'$ as:

$$\Psi(\vec{r}) = \Psi(\vec{r}')e^{i\vec{T}\cdot\vec{k}} \tag{4}$$

where $\vec{k}$ is confined to the first Brillouin zone of the primitive cell. We have also chosen the supercell twist vector which places a constraint on the form of the wavefunction in the supercell. The combination of these two constraints allows us to identify family of N k-points in the primitive cell that satisfy the constraints. Thus for a given supercell tiling matrix and twist angle, we can write the wavefunction everywhere in the supercell by knowing the wavefunction a N k-points in the primitive cell. This means that the memory necesary to store the wavefunction in a supercell is only linear in the size of the supercell rather than the quadratic cost if symmetry were neglected.

2 Total energy of BCC beryllium

As was discussed in this mornings lectures when performing calculations of periodic solids with QMC, it is essential to work with a reasonable size supercell rather than the primitive cells that are common in mean field calculations. Specifically, all of the finite size correction schemes discussed in the morning require that the exchange-correlation hole be considerably

smaller than the periodic simulation cell. Additionally, finite size effects are lessened as the distance between the electrons in the cell and their periodic images increases, so it is advantageous to generate supercells that are as spherical as possible so as to maximize this distance. However, there is a competing consideration in that for calculating total energies we often want to be able to extrapolate the energy per particle to the thermodynamic limit by means of the following formula in 3 dimensions:

$$E_{\text{inf}} = C + E_N/N \tag{5}$$

This formula derived assuming the shape of the supercells is consistent (more specifically that the periodic distances scale uniformly with system size), meaning we will need to do a uniform tiling, ie, 2x2x2, 3x3x3 etc. As a 3x3x3 tiling is 27 times larger than the supercell and the practical limit of DMC is on the order of 200 atoms (depending on Z), sometimes it is advantagous to choose a less spherical supercell with fewer atoms rather than a more spherical one that is too expensive to tile.

In the case of a BCC crystal, it is possible to tile the one atom primitive cell to a cubic supercell by only doubling the number of electrons. This is the best possible combination of a small number of atoms that can be tiled and a regular box that maximizes the distance between periodic images. We will need to determine the tiling matrix S that generates this cubic supercell by solving the following equation for the coefficients of the S matrix:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \\ s_{31} & s_{32} & s_{33} \end{bmatrix} \cdot \begin{bmatrix} 0.5 & 0.5 & -0.5 \\ -0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 \end{bmatrix} \tag{6}$$

We will now use the project suite to generate the trial wavefunction for this BCC beryllium.

Fortunately, the project-suite will handle determination of the proper k-vectors given the tiling matrix. All that is needed is to place the tiling matrix in the Be-2at-setup.py file. Now the definition of the physical system is:

```
bcc_Be = generate_physical_system(
    lattice    = 'cubic',
    cell       = 'primitive',
    centering  = 'I',
    atoms      = 'Be',
    constants  = 3.490,
    units      = 'A',
    net_charge = 0,
    net_spin   = 0,
    Be         = 2,
    tiling     = [[a,b,c],[d,e,f],[g,h,i]],
    kgrid      = kgrid,
    kshift     = (.5,.5,.5)
    )
```

Where the tiling line should be replaced with the row major tiling matrix from above. This script file will now perform a converged DFT calculation to generate the charge density in a directory called bcc-beryllium/scf and perform a non self consistend DFT calculation to generate single particle orbitals in the direcotry bcc-beryllium/nscf. Fortunately, the project suite will calculate the required k-points needed to tile the wavefunction to the supercell, so all that is necessary is the granularity of the supercell twists and whether this grid is shifted from the origin. Once this is finished, it performs the conversion from pwscf's binary format to the hdf5 format used by qmcpack. Finally, it will optimize the coefficients of one-body and two-body jastrow factors in the supercell defined by the tiling matrix.

Run these calculations by executing the script Be-2at-setup.py. You will notice that such small calcuations as are required to generate the wavefunction of Be in a one atom cell are rather inefficent to run on a high performance computer such as vesta in terms of the time spent doing calculations versus time waiting on the scheduler and booting compute nodes. One of the benefits of the portable hdf format that is used by qmcpack is that you can generate data like wavefunctions on a local workstation or other convenient resource and only use high performance clusters for the more expensive QMC calculations.

In this case, the wavefunction is generated in the directory bcc-beryllium/nscf-2at_222/pwscf_output in a file called pwscf.pwscf.h5. It can be useful for debugging purposes to be able to verify the contents of this file are what you expect. For instance, you can use the tool h5ls to check the geometry of the cell where the dft calculations were performed, or number of k-points or electrons in the calculation. This is done with the command: h5ls -d pwscf.pwscf.h5/supercell or h5ls -d pwscf.pwscf.h5/electrons.

In the course of running Be-2at-setup.py, you will get an error when attempting to perform the vmc and wavefunction optimization calculations. This is due to the fact that the wavefunction has been generated supercell twists of the form (+/- 1/4, +/- 1/4, +/- 1/4). In the case that the supercell twist contains only 0 or 1/2, it is possible to operate entirely with real arithmetic. The executabe that has been indicated in Be-2at-setup.py has been compiled for this case. Note that where this is possible, the memory usage is a factor of two less than the general case and the calculations are somewhat faster. However, it is often necessary to perform calculations away from these special twist angles in order to reduce finite size effects. To fix this, delete the directory bcc-beryllium/opt-2at, change the line in near the top of Be-2at-setup.py from

```
qmcpack      = '/soft/applications/qmcpack/build_XL_real/bin/qmcapp'
```

to

```
qmcpack      = '/soft/applications/qmcpack/build_XL_complex/bin/qmcapp'
```

and rerun the script.

When the optimiztion calculation has finished, check that everything as proceeded correctly by looking at the output in the opt-2at directory. Firstly, you can grep the output file for Delta to see if the cost function has indeed been decreasing during the optimization. You should find something like:

```
OldCost: 4.8789147e-02 NewCost: 4.0695360e-02 Delta Cost:-8.0937871e-03
```

```
OldCost: 3.8507795e-02 NewCost: 3.8338486e-02 Delta Cost:-1.6930674e-04
OldCost: 4.1079105e-02 NewCost: 4.0898345e-02 Delta Cost:-1.8076319e-04
OldCost: 4.2681333e-02 NewCost: 4.2356598e-02 Delta Cost:-3.2473514e-04
OldCost: 3.9168577e-02 NewCost: 3.8552883e-02 Delta Cost:-6.1569350e-04
OldCost: 4.2176276e-02 NewCost: 4.2083371e-02 Delta Cost:-9.2903058e-05
OldCost: 4.3977361e-02 NewCost: 4.2865751e-02 Delta Cost:-1.11161830-03
OldCost: 4.1420944e-02 NewCost: 4.0779569e-02 Delta Cost:-6.4137501e-04
```

Which shows that the starting wavefunction was fairly good and that most of the optimizaiton occurred in the first step. Confirm this by using qmca to look at how the energy and variance changed over the course of the calculation with teh comand: qmca -q ev -e 10 *.scalar.dat executed in the opt-2at directory. You should get output like the following:

		LocalEnergy	Variance	ratio
opt	series 0	−2.159139 +/− 0.001897	0.047343 +/− 0.000758	0.0219
opt	series 1	−2.163752 +/− 0.001305	0.039389 +/− 0.000666	0.0182
opt	series 2	−2.160913 +/− 0.001347	0.040879 +/− 0.000682	0.0189
opt	series 3	−2.162043 +/− 0.001223	0.041183 +/− 0.001250	0.0190
opt	series 4	−2.162441 +/− 0.000865	0.039597 +/− 0.000342	0.0183
opt	series 5	−2.161287 +/− 0.000732	0.039954 +/− 0.000498	0.0185
opt	series 6	−2.163458 +/− 0.000973	0.044431 +/− 0.003583	0.0205
opt	series 7	−2.163495 +/− 0.001027	0.040783 +/− 0.000413	0.0189

Now that the optimization has completed successfully, we can perform dmc calculations. The first goal of the calculations will be to try to eliminate the one body finite size effects by twist averaging. The script Be-2at-qmc.py has the necessary input. Note on line 42 two twist grids are specified, (2,2,2) and (3,3,3). Change the tiling matrix in this input file as in Be-2at-qmc.py and start the calculations. Note that this workflow takes advantage of qmcpack's ability to group jobs. If you look in the directory dmc-2at_222 at the job submission script, (dmc.qsub.in) you will note that rather than operating on an xml input file, qmcapp is targeting a text file called dmc.in. This file is a simple text file that contains the names of the 8 xml input files needed for this job, one for each twist. When operated in this mode, qmcpack will use mpi groups to run multiple copies of itself within the same mpi context. This is often useful both in terms of organizing calculations and also for taking advantage of the large job sizes that computer centers often encourage.

The dmc calculations in this case are designed to complete in a few minutes. When they have finished running, first look at the scalar.dat files corresponding to the dmc calculations at the various twists in dmc-2at_222. Using a command like 'qmca -q ev -e 32 *.s001.scalar.dat' (with a suitably chosen number of blocks for the equilibration), you will see that the dmc energy in each calcuation is nearly identical within the statistical uncertainty of the calculations. In the case of a large supercell, this is often indicative of a situation where the Brilloiun zone is so small that the one body finite size effects are nearly converged without any twist averaging. In this case, however, this is because of the symmetry of the system. For this cubic supercell, all of the twist angles chosen in this shifted 2x2x2 grid are equivalent by symmetry. In the case where substantial resources are required to equilibrate

the dmc calculations, it can be beneficial to avoid repeating such twists and instead simply weight them properly. In this case however where the equilibration is inexpensive, there is no benefit to adding such complexity as the calculations can simply be averaged together and the result is equivalent to performing a single longer calcuation.

Using the command qmc -a -q ev -e 16 *.s001.scalar.dat, average the dmc energies in dmc-2at_222 and dmc-2at_333 to see whether the one body finite size effects are converged with a 3x3x3 grid of twists. As beryllium as a metal, the convergence is quite poor (0.025 Ha / Be or 0.7 eV / Be). If this were a production calculation it would be necessary to perform calculations on much larger grids of supercell twists to eliminate the one body finite size effects.

In this case there are several other calculations that would warrent a high priority. A script Be-16at-qmc.py has been provided where you can imput the appropriate tiling matrix for a 16 atom cell and perform calculations to estimate the two body finite size effects which will also be quite large in the 2 atom calculations. This script will take approximately 30 minutes to run to completion, so depending on interest, you can either run it, or also work to modify the scripts to address the other technical issues that would be necessary for a production calculation such as calculating the population bias or the timestep error in the dmc calculations.

Another useful exercise would be to attempt to validate this pseudopotential by calculating the ionization potential and electron affinity of the isolated atom and comparing to the experimental values: IP = 9.3227 eV , EA = 2.4 eV.

3 Handling a 2D system: graphene

In this section we will examine a calculation of an isolated sheet of graphene. As graphene is a two dimensional system, we will take advantage of qmcpack's ability to mix periodic and open boundary conditions to eliminate and spurious interaction of the sheet with its images in the z direction. Run the script graphene-setup.py which will generate the wavefunction and optimize one and two body jastrow factors. In the script, notice line 160: bconds = 'ppn' in the generate_qmcpack function which specifies this mix of open and periodic boundary conditions. As a consequence of this, the atoms will need to be kept away from this open boundary in the z direction as the electronic wavefunction will not be defined outside of the simulation box in this direction. For this reason, all of the atom positions in at the beginning of the file have z coordinates 7.5. At this point, run the script graphene-setup.py.

Aside from the change in boundary conditions, the main thing that distinguished this kind of calculation from the beryllium example above is the large amount of vacuum in the cell. While this is a very small calculation designed to run quickly in the tutorial, in general a more converged calculation would quickly become memory limited on an architecture like BG/Q. When the initial wavefunciton optimizaiton has completed to your satisfaction, run the scripts graphene-loop-buffer.py and graphene-loop-mesh.py. These examine within variational Monte Carlo two approaches to reducing the memory required to store the wavefunction. In graphene-loop-mesh.py, the spacing between the b-spline points is varied uniformly.

The mesh spacing is a prefactor to the linear spacing between the spline points, so the memory usage goes as the cube of the meshfactor. When you run the calculations, examine the .s000.scalar.dat files with qmca to determine the lowest possible mesh spacing that preserves both the vmc energy and the variance. Similarly, the script graphene-loop-buffer.py uses a feature which generates two spline tables for the wavefunction. One will have half of the mesh spacing requested in the input file and will be valid everywhere. The second one will only be defined in the smallest parallelpiped that contains all of the atoms in the simulation cell with minimum distance given by the buffer size. Again, see what the smallest possible buffer size is that preserves the vmc energy and variance.

Finally, edit the file graphene-final.py which will perform two DMC calculations. In the first, (qmc1) replace the following lines:

```
meshfactor    = xxx,
precision     = '---',
truncate      = False,
buffer        = 0.0,
```

using the values you have determined to perform the calculation with as small as possible of wavefunction. Note that we can also use single precision arithmetic to store the wavefunction by specifying precision='single'. When you run the script, compare the output of the two DMC calculations in terms of energy and variance. Also see if you can calculate the fraction of memory that you were able to save by using a meshfactor other than 1, a buffer table and single precision arithmetic.

4 Conclusion

Upon completion of this lab, you should be able to use the project suite to perform DMC calculations on periodic solids when provided with a pseudopotential. You should also be able to reduce the size of the wavefunction in a solid state calculation in cases where memory is a limiting factor.

5 Acknowledgment

This tutorial was created with support from Sandia National Laboratories.

Sandia National Laboratories is a multiprogram laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under Contract No. DE-AC04-94AL85000.