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

Luke Shulenburger

July 26, 2012

Abstract

The tutorial will guide you through the process of performing total energy calculations of solids using quantum Monte Carlo. This will include the full workflow from performing DFT calculations to set up the trial wavefunction, converging the b-spline representation of the wavefunction, optimizing jastrow factors and performing Diffusion Monte Carlo calculations.

1 Introduction

This tutorial session will focus on how to perform accurate QMC calculations on solids using pwscf for creating trial wavefunctions and QMCPACK for the QMC. The concepts and tools needed will be introduced in the context of calculating a pressure versus volume curve for beryllium in the body centered cubic (BCC) structure. Due to the limited computational resources available for this tutorial, many of these calculations will be under converged, but complete results will be provided in the course of this document.

Please note that this tutorial will be highly dependent on the setup-qmc.pl tool for generating pwscf and QMCPACK input files. This tool has been preconfigured and placed in your path. In general you will need edit the configuration file setup-qmc-conf.pm located in the utils directory of QMCPACK to specify the locations of several tools used throughout.

Also for this tutorial, a general shell script is provided that will include all of the necessary flags for setup-qmc.pl and is easily customizable for other machines and physical systems.

The general outline of the tutorial is follows. The necessary input DFT calculations for a QMC simulation will be discussed. Then a wavefunction will be generated from a DFT calculation for beryllium. Next, VMC will be used to optimize Jastrow factors for the many body wavefunction. Then the proper convergence of the b-spline representation of the DFT wavefunction will be examined. With that in hand, DMC calculations will be discussed

*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.

including the convergence of the time step. Finally, the finite size errors inherent in these calculations will be confronted.

2 Initial DFT Calculations

The first step in performing quantum Monte Carlo calculations is to perform a DFT calculation on the same system. This is necessary to provide the trial wavefunction that is necessary both for improving the efficiency of the QMC calculation (importance sampling) and for mitigating the sign problem. In this case we will be focusing on beryllium, so we will start with a pwscf input file as given below:

```
 &control
     calculation='scf'
     restart_mode='from_scratch',
     tstress = .true.
     prefix='Be',
     pseudo_dir = './',
     outdir='out'
     wf_collect=.true.
     disk_io='low'
 /
 &system
     ibrav=0, celldm(1) =4.75, nat=  1, ntyp= 1,
     nspin=1,
     degauss=0.001,
     smearing='mp',
     occupations='smearing',
     ecutwfc =120
     ecutrho =480
 /
 &electrons
     conv_thr =  1.0d-10
     mixing_beta = 0.7
 /
CELL_PARAMETERS
  0.5   0.5   0.5
 -0.5   0.5   0.5
 -0.5  -0.5   0.5
ATOMIC_SPECIES
  Be   9.01   Be.ncpp
ATOMIC_POSITIONS
  Be 0.00 0.00 0.00
K_POINTS AUTOMATIC
```

```
20 20 20 1 1 1
```

As the goal of the calculation is to produce an extremely accurate DFT wavefunction, ecutwfc and the k-points should be converged to a very high degree. For this purpose, it is often better to monitor convergence of the stresses (which depend linearly on the wavefunctions) than on the total energy which improves quadratically with the quality of the wavefunctions.

Copy the directory sluke/BeTest to each of your home directories. This input file is included in the BeTest directory and is called Be-scf.in. Create a directory for the output file (mkdir out) and then start a series of calculations using the provided runscript. This will execute the commands provided inline.

```
pw.x < Be-scf.in > Be-scf.out
```

2.1 Pseudopotentials

In the calculation on Be, note that the cutoff is higher than is typical in modern electronic structure codes. This is due to two factors. The first is that the treatment of nonlocal pseudopotentials in QMC presently requires the use of norm conserving pseudopotentials rather than the PAW or ultrasoft formalisms which are now in use. The second reason for the high cutoff is due to the particular construction of this pseudopotential. It was created using opium (http://opium.sourceforge.net) with the only constraint in mind that it accurately reproduces the all electron results within LDA. This is often a practical way to generate pseudopotentials for QMC as the increased plane wave cutoffs do not result in increased run time within QMC and the DFT calculation is many orders of magnitude faster than the QMC. The only exception to this rule is when there is difficulty fitting the wavefunction into the memory of the computer during the QMC. This will be discussed more thoroughly in section 3.

It may also be practical to use pseudopotentials available in various libraries distributed with DFT codes. The ppconvert tool which is a part of the qmcutils distribution can convert many of these to the xml format which QMCPACK understands. For the purpose of this tutorial, any norm conserving pseudopotential in UPF or NCPP format may be used for the DFT part of the QMC calculations.

3 Creating and testing wavefunctions for QMCPACK

The plane wave representation for the wavefunction used in pwscf is not a good choice for quantum Monte Carlo. This is because the largest computational effort involved in QMC is evaluating the trial wavefunction and related quantities when an electron is moved from $\vec{r}$ to $\vec{r}'$. With a plane wave basis, this requires evaluating all N of the basis functions for every move.

Rather than using these plane waves, QMCPACK calculations are typically performed using a b-spline representation of the wavefunction. Practically speaking, this involves ex-

tracting the plane waves from a pwscf calculation and then choosing a regular grid upon which to construct the spline representation. In this section you will construct the wavefunction from pwscf calculations and test that the splines are sufficiently dense within QMCPACK.

3.1 Extracting Wavefunctions at Particular k-points

There is a direct mapping from k-points in DFT calculations to non-periodic boundary conditions in quantum Monte Carlo (essential for twist averaging). For this reason and for others detailed in section 6.2.1, it is important to be able to extract the wavefunction at arbitrary k-points in the first Brillouin Zone of the system. setup-qmc.pl simplifies this process by creating input files for pwscf to get the wavefunction at any k-point using the charge density from the converged DFT calculation found in section 2. Following this pwscf calculation, the tool pw2qmcpack.x should be run to generate an hdf file which is then prepared for QMCPACK using wfconvert. Many of the quantities computed in this tutorial are sensitive to the finite size effects when performed in very small supercells. For this reason, we will work in an 8 atom supercell now and will test the errors systematically later.

QMCPACK will be able to tile the wavefunction from the primitive cell represented in the pwscf input file to this supercell using the Bloch theorem, however it will need wavefunctions at particular k-points to make this work. This process can be automated using setup-qmc.pl as follows

```
setup-qmc.pl --gettilemat --supercellsize 8 Be-scf.in
setup-qmc.pl --genwfn --tilemat 1 1 -1 -1 0 -2 -1 2 0 Be-scf.in
```

This will generate an input file for pwscf called Be-S8nscf.in. This input file should be identical to Be-scf.in except for a few critical points. Instead of a scf calculation, the calculation will now be nscf so as to avoid changing the charge density from the previously converged calculation. Secondly, the wavefunction will be collected by a single process using wf_collect = .true. This should allow the mpi version of quantum espresso to be used where available. All symmetry is also turned off in order to avoid removing any k-points. Finally, the last section of the file: K_POINTS is changed to include 8 specific k-points that will be used in the supercell wavefunction.

Run this calculation with the following command:

```
pw.x < Be-S8-nscf.in > Be-S8-nscf.out
```

The wavefunction can now be extracted to hdf format using the pw2qmcpack.x tool that is included with the modified version of quantum espresso available from: (qmc-tools). All that is needed for this is a simple input file that tells pw2qmcpack.x where the wavefunctions from the pwscf calculation are and whether or not to generate the b-splines directly. Invoke pw2qmcpack.x as follows:

```
pw2qmcpack.x < Be-S8-pw2x.in > Be-S8-pw2x.out
```

Now this is converted to the eshdf format used by QMCPACK with the following command

```
wfconvert --nospline --eshdf Be-S8.h5 out/Be.pwscf.h5 >& Be-S8-wfconvert.out
```

The nospline argument keeps the wavefunction in reciprocal space, which will be useful in convergence testing below, but can significantly increase the setup time in QMCPACK for large systems. wfconvert can also be invoked with the argument –factor xx which will generate the real space b-splines directory with a given mesh spacing.

4 Optimizing Jastrow Factors within QMC

The next step in performing QMC calculations of beryllium is to produce as accurate a many body wavefunction as possible. Although many forms have been proposed for many body wavefunctions of periodic systems, we will stick to the standard Slater-Jastrow form whereby the Slater determinant obtained from DFT is multiplied by correlation factors which explicitly account for the approach of electrons to other electrons and of electrons to the ions in the system. A very general form of these Jastrow factors which is used in QMCPACK is as follows

$$J(r - r') = e^{f(|r-r'|)} \tag{1}$$

where the function $f(x)$ is represented as a one dimensional spline.

It is not possible to optimize this function analytically, so we will optimize its form using variational Monte Carlo. There are several different formalisms for doing this optimization. Probably the simplest to understand conceptually is the variance minimization approach. This technique takes advantage of the zero variance principle whereby the variance of a VMC simulation with any many body eigenstate as the wavefunction will be zero. Knowing this, VMC simulations are performed and the parameters which influence the Jastrow factors (in this case spline values) are varied in order to reduce the variance of the simulation.

Recently, successful techniques have been devised which also take into account the variational nature of VMC and work to minimize the total energy calculated by varying the parameters in the wavefunction. We will use one such approach to optimize Jastrow factors for our aluminum example. Again, we will use setup-qmc.pl to generate a QMCPACK input file. Invoke that script with the following command:

```
setup-qmc.pl --optwvfcn --vmctimestep 1.0 --vmcblocks 512 \
  --vmcequiltime 10.0 --vmcdecorrtime 5.0 --tilemat 1 1 -1 -1 0 -2 -1 2 0 \
  --optsamples 16384 --walkers 64 --optloops 12 --onebodysplinepts 8 \
  --twobodysplinepts 8 --splfactor 1.0 --wvfcnfile Be-S8.h5 Be-scf.in
```

To run these calculations, it is necessary to have some understanding of how parallelism works in QMCPACK. QMCPACK takes advantage of distributed memory parallelism (such as between nodes of a supercomputer) via MPI and shared memory parallelism (such as within a node) using OpenMP. On modern supercomputers, this can result in large benefits because the communication is reduced linearly with the number of OpenMP threads used per node. In this case we will have logged directly into the nodes of the supercomputer and will run QMCPACK using only OpenMP parallelism. To do this, set the number of threads to 4 using the following command

```
export OMP_NUM_THREADS=4
```

Now QMCPACK should be invoked to find the optimal Jastrow factor. First move to the directory optimization and invoke QMCPACK as follows

```
qmcapp opt-Be-S8.xml >& opt-Be-S8.out &
```

Hybrid parallelism can also be used by invoking qmcapp via the standard mpi process on your supercomputer taking into account that the number of processes requested should not be the total number of CPU cores to use, but should likely correspond to either the number of processors on the nodes or to the number of nodes. As an example, to run QMCPACK on a computer with 4 nodes which each have 8 processors, commands similar to the following would be used:

```
export OMP_NUM_THREADS=8
mpirun -np 4 qmcapp opt-Be-S8.xml >& opt-Be-S8.out &
```

While QMCPACK is optimizing the wavefunction, take some time to examine the input file that was generated by setup-qmc.pl:

```
<?xml version=''1.0''?>
<simulation>
  <project id=''opt-Be-S8'' series=''1''>
    <application name=''qmcapp'' role=''molecu'' class=''serial'' version=''0.2''>
      Optimization of jastrows for Be-S8
    </application>
  </project>

  <random seed=''49154''/>
  <qmcsystem>
  <simulationcell>
    <parameter name=''lattice''>
      2.375   7.125   2.375
      2.375   2.375   -7.125
      -7.125   2.375   2.375
      </parameter>
      <parameter name=''bconds''>p p p</parameter>
      <parameter name=''LR_dim_cutoff''>15</parameter>
  </simulationcell>
  </qmcsystem>
  <particleset name=''i'' size=''8''>
    <group name=''Be''>
      <parameter name=''charge''>2.000000</parameter>
      <parameter name=''valence''>2.000000</parameter>
      <parameter name=''atomicnumber''>4.000000</parameter>
    </group>
    <attrib name=''position'' datatype=''posArray'' condition=''1''>
    0   0   0
    0   0.5   0.5
    0.5   0.25   0.25
    0.25   0.875   0.375
    0.5   0.75   0.75
    0.25   0.375   0.875
    0.75   0.625   0.125
    0.75   0.125   0.625
    </attrib>
    <attrib name=''ionid'' datatype=''stringArray''>
      Be  Be  Be  Be  Be  Be  Be  Be
    </attrib>
  </particleset>
```

```
  <particleset name=''e'' random=''yes'' randomsrc=''i''>
    <group name=''u'' size=''8''>
      <parameter name=''charge''>-1</parameter>
    </group>
    <group name=''d'' size=''8''>
      <parameter name=''charge''>-1</parameter>
    </group>
  </particleset>
  <wavefunction name=''psi0'' target=''e''>
    <determinantset type=''einspline'' href=''../Be-S8.h5'' tilematrix=''1 1 -1 -1 0 -2 -1 2 0'' twistnum=
      <basisset/>
      <slaterdeterminant>
        <determinant id=''updet'' size=''8'' ref=''updet''>
          <occupation mode=''ground'' spindataset=''0''>
          </occupation>
        </determinant>
        <determinant id=''downdet'' size=''8'' ref=''downdet''>
          <occupation mode=''ground'' spindataset=''0''>
          </occupation>
        </determinant>
      </slaterdeterminant>
    </determinantset>
    <jastrow name=''J2'' type=''Two-Body'' function=''Bspline'' print=''yes''>
      <correlation speciesA=''u'' speciesB=''u'' size=''8''>
        <coefficients id=''uu'' type=''Array''>  0.424691989726135  0.344977533525888  0.249005479000796
0.159694777099558  0.0909862591866385  0.0460450768991445  0.0206925589489831  0.00825573399272362
</coefficients>
      </correlation>
      <correlation speciesA=''u'' speciesB=''d'' size=''8''>
        <coefficients id=''ud'' type=''Array''>  0.599154574333419  0.460218238112388  0.316262493017312
0.194376144111787  0.106787090306476  0.0524055027544289  0.022954658656253  0.00896654256094429
</coefficients>
      </correlation>
    </jastrow>
    <jastrow name=''J1'' type=''One-Body'' function=''Bspline'' print=''yes'' source=''i''>
      <correlation elementType=''Be'' cusp=''0.0'' size=''8''>
        <coefficients id=''Be'' type=''Array''> 0  0  0  0  0  0  0  0  </coefficients>
      </correlation>
    </jastrow>
  </wavefunction>
  <hamiltonian name=''h0'' type=''generic'' target=''e''>
    <pairpot type=''pseudo'' name=''PseudoPot'' source=''i'' wavefunction=''psi0'' format=''xml''>
      <pseudo elementType=''Be'' href=''/scratch/users/sluke/BeTutorial/Be.xml''/>
    </pairpot>
    <constant name=''IonIon'' type=''coulomb'' source=''i'' target=''i''/>
    <pairpot name=''MPC'' type=''MPC'' source=''e'' target=''e'' ecut=''60.0'' physical=''false''/>
    <pairpot name=''ElecElec'' type=''coulomb'' source=''e'' target=''e'' physical=''true''/>
    <estimator name=''KEcorr'' type=''chiesa'' source=''e'' psi=''psi0''/>
  </hamiltonian>

<loop max=''12''>
  <qmc method=''cslinear'' move=''pbyp'' checkpoint=''-1'' gpu=''yes''>
    <parameter name=''blocks''>   512 </parameter>
    <parameter name=''warmupsteps''> 0 </parameter>
    <parameter name=''steps''>    1 </parameter>
    <parameter name=''timestep''>  1.0  </parameter>
    <parameter name=''walkers''>  16 </parameter>
    <parameter name=''samples''>  16384  </parameter>
    <parameter name=''minwalkers''>  0.5 </parameter>
    <parameter name=''maxWeight''>    1e9 </parameter>
    <parameter name=''useDrift''>  yes </parameter>
    <estimator name=''LocalEnergy'' hdf5=''no''/>
    <cost name=''energy''>                  0.0 </cost>
    <cost name=''unreweightedvariance''>    1.0 </cost>
    <cost name=''reweightedvariance''>      0.0 </cost>
```

```
    <parameter name=''MinMethod''>rescale</parameter>
    <parameter name=''GEVMethod''>mixed</parameter>
    <parameter name=''beta''>  0.0 </parameter>
    <parameter name=''exp0''> -16 </parameter>
    <parameter name=''nonlocalpp''>no</parameter>
    <parameter name=''useBuffer''>no</parameter>
    <parameter name=''bigchange''>9.0</parameter>
    <parameter name=''alloweddifference''> 1.0e-3 </parameter>
    <parameter name=''stepsize''>4.0e-1</parameter>
    <parameter name=''stabilizerscale''>  1.0 </parameter>
    <parameter name=''nstabilizers''> 3 </parameter>
    <parameter name=''max_its''> 1 </parameter>
  </qmc>
</loop>
<loop max=''3''>
  <qmc method=''cslinear'' move=''pbyp'' checkpoint=''-1'' gpu=''yes''>
    <parameter name=''blocks''>   512 </parameter>
    <parameter name=''warmupsteps''> 0 </parameter>
    <parameter name=''steps''>    1 </parameter>
    <parameter name=''timestep''>  1.0  </parameter>
    <parameter name=''walkers''>  16 </parameter>
    <parameter name=''samples''>  16384  </parameter>
    <parameter name=''minwalkers''>  0.5 </parameter>
    <parameter name=''maxWeight''>    1e9 </parameter>
    <parameter name=''useDrift''>  yes </parameter>
    <estimator name=''LocalEnergy'' hdf5=''no''/>
    <cost name=''energy''>                   0.8 </cost>
    <cost name=''unreweightedvariance''>     0.0 </cost>
    <cost name=''reweightedvariance''>       0.2 </cost>
    <parameter name=''MinMethod''>quartic</parameter>
    <parameter name=''GEVMethod''>mixed</parameter>
    <parameter name=''beta''>  0.0 </parameter>
    <parameter name=''exp0''> -16 </parameter>
    <parameter name=''nonlocalpp''>yes</parameter>
    <parameter name=''useBuffer''>yes</parameter>
    <parameter name=''bigchange''>9.0</parameter>
    <parameter name=''alloweddifference''> 1.0e-4 </parameter>
    <parameter name=''stepsize''>4.0e-1</parameter>
    <parameter name=''stabilizerscale''>  1.0 </parameter>
    <parameter name=''nstabilizers''> 3 </parameter>
    <parameter name=''max_its''> 1 </parameter>
  </qmc>
</loop>
</simulation>
```

The geometry of the calculation is read from the pwscf input file and the unit cell is placed in the lattice tag. The bconds tag can be changed so that slabs or wires can be handled natively without any spurious interactions from images, to do this, simply change p (periodic) for a given boundary condition to n (non-periodic). Following this, particlesets are specified giving information about the electrons and ions in the problem.

The determinantset controls how the wavefunction is represented in the problem. Here type="einspline" specifies that we are using a b-spline basis set and meshfactor="1.0" specifies a mesh spacing. The meshfactor is the spacing in terms of the density of the real space points corresponding to the plane waves used in the DFT calculation. A meshfactor of 0.5 will give the same number of real space points as plane waves in the cell. Other notable sections include the Jastrow factor (which in this case starts as being represented by a 4 point spline that is all 0), and the optimization block at the end of the file.

In general each QMCPACK input file will consist of some header information followed

by a particleset, a wavefunction, a Hamiltonian and then any number of qmc sections which give the actions for QMCPACK to perform. Here we use a loop construct to repeat the optimization several times because the optimization needs to proceed somewhat self-consistently. At this point, take a minute to familiarize yourself with the QMCPACK users guide, which will provide a much more thorough explanation of how the input file is structured and what each element means.

This optimization will take about 10 minutes to run. However, the calculation will continuously write to files named opt-Be-S8.s0??.scalar.dat, with one for each iteration of the VMC optimization. There is another script, OptProgress.pl which can be used to monitor the progress of the optimization. The syntax to invoke it is as follows:

```
OptProgress.pl opt-Be-S8.s001.scalar.dat
```

When the calculation is finished, this will produce plot similar to Fig.1

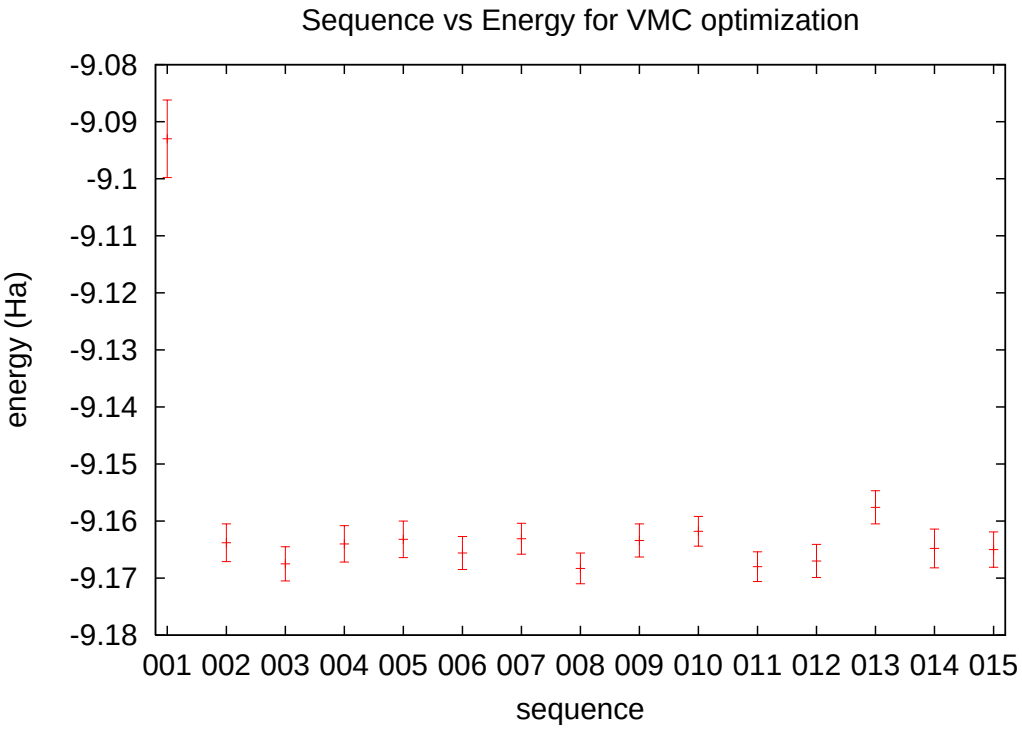

Figure 1: Total energy vs iteration number for optimization of Jastrow factors in Beryllium.

It is common for this optimization to proceed over several steps and there is no guarantee that the final optimization cycle will produce the best wavefunction (as measured by having the lowest energy and smallest variance). setup-qmc.pl will always analyze this directory to find the cycle that produced the best wavefunction and will include this in future calculations.

4.1 Testing Convergence of the Splines in QMCPACK

Now that the wavefunctions have been generated the proper density for the b-splines should be found. In the previous section we used a rather large mesh given by meshfactor="1.0" in the determinantset. This is typically larger than is necessary and as the memory needed to store the wavefunction goes as the cube of the meshfactor, this can cause calculations to be intractable.

In order to determine the smallest appropriate meshfactor, we will perform variational Monte Carlo calculations using the best Jastrow factor determined above. The quantities to monitor are the total energy which should decrease until saturating at large values of meshfactor, the variance which should decrease as meshfactor increases and the kinetic energy which depends only on the shape of the trial wavefunction including Jastrow factor.

To perform this convergence test, generate appropriate input files using setup-qmc.pl with the following command

```
setup-qmc.pl --splconv --withjas --vmctimestep 1.2 --vmcblocks 512 \
   --vmcequiltime 10.0 --vmcdecorrtime 5.0 --tilemat 1 1 -1 -1 0 -2 -1 2 0 \
   --numsamples 65536 --wvfcnfile Be-S8.h5 --walkers 256 --minfactor 0.4 \
   --maxfactor 1.0 --factorinc 0.05 Be-scf.in
```

This will generate input files with names like Be-S8-f0.40.xml in the directory bsplineConv. Run each of these calculations by changing to the directory bsplineConv and invoking qmcapp for each one of the input files in turn:

```
qmcapp Be-S8-f0.40.xml >& Be-S8-f0.40.out
qmcapp Be-S8-f0.50.xml >& Be-S8-f0.50.out
qmcapp Be-S8-f0.60.xml >& Be-S8-f0.60.out
qmcapp Be-S8-f0.70.xml >& Be-S8-f0.70.out
qmcapp Be-S8-f0.80.xml >& Be-S8-f0.80.out
qmcapp Be-S8-f0.90.xml >& Be-S8-f0.90.out
qmcapp Be-S8-f1.00.xml >& Be-S8-f1.00.out
```

When you have finished this (it should take several minutes) you can monitor the convergence of the results using the PlotBsplConv.pl utility distributed with QMCPACK:

```
PlotBsplConv.pl Be-S8-f0.40.s001.scalar.dat
```

Which will generate a plot which looks like Fig. 2. The convergence of the total energy can be somewhat misleading as the quadratic convergence of the energy with the wavefunction can mask differences in the wavefunction. The variance of the local energy is a much more reliable metric of the spline's convergence. It requires many fewer samples to converge and is a direct measure of how the spline factor will influence the cost of the DMC calculations to come. The number of steps required to obtain a particular statistical error in a DMC calculation goes linearly with the variance. In this case we see convergence with a meshfactor of roughly 0.7, which we will use in our future calculations.

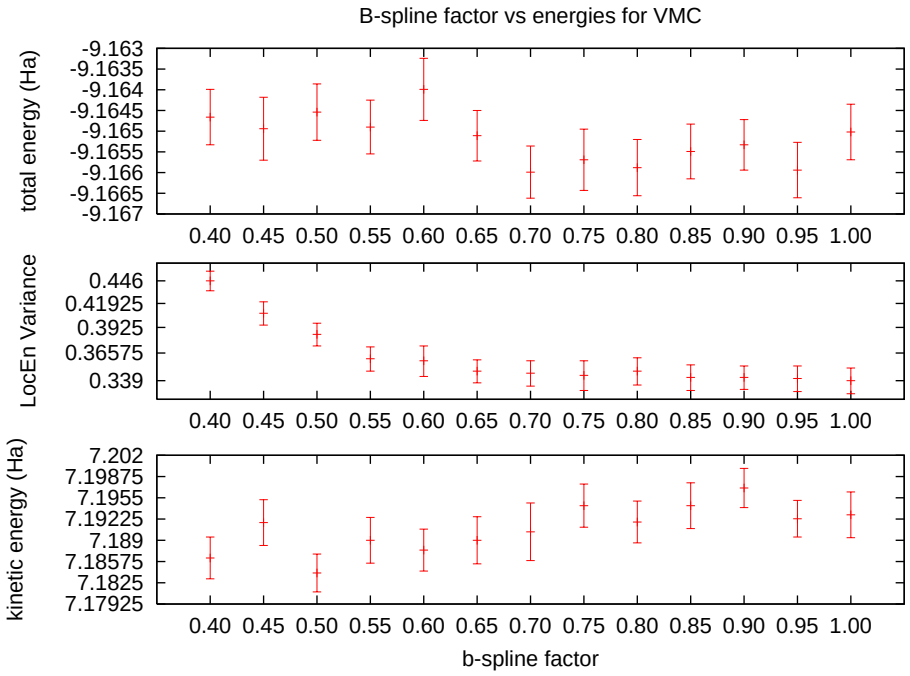

Figure 2: Total, kinetic energy and variance of beryllium as a function of b-spline spacing from VMC.

5 Performing DMC Calculations

Diffusion Monte Carlo (DMC) calculations will consume the vast majority of the CPU cycles in your QMC projects. This method differs from VMC in that it samples the observables you specify for the lowest energy wavefunction consistent with the many body nodes that you provide in your trial wavefunction.

In performing accurate DMC calculations, it is important to converge several technical parameters. The first of these is the average population of walkers in the calculation. In general this needs to be large so that fluctuations in the population do not unduly influence the results of the calculations. For relatively good trial wavefunctions like the one used here, 1000 walkers should be more than sufficient, however for wavefunctions with larger variance, like all electron wavefunctions of first row solids, this may need to be 10's of thousands.

The second technical parameter that will need to be converged is the size of the time step which is used in the DMC propagation. This controls the accuracy of the Green's function which is used to propagate the walkers. A rule of thumb is that the acceptance ratio upon moving the walkers should be around 99%. However, this should be precisely converged, a task which can again be accomplished by the combination of setup-qmc.pl and QMCPACK. Here is the input line for setup-qmc.pl

```
setup-qmc.pl --convdmctstep --vmcwalkers 64 --vmctimestep 1.2 \
   --vmcequiltime 12.0 --vmcdecorrtime 6.0 --tilemat 1 1 -1 -1 0 -2 -1 2 0 \
```

```
--targetpop 4096 --dmcruntime 102.5 --dmcequiltime 2.5 --dmcblocktime 0.5 \
--mindmctstep 0.005 --maxdmctstep 0.02 --dmctstepinc 0.005 --splfactor 0.70
 --wvfcnfile Be-S8.h5 Be-scf.in
```

The input file this produces will start with a short VMC calculation to produce a population of walkers more nearly distributed according to the trial wavefunction. After that a series of DMC calculations are done with successively shorter time steps. The first calculation (series 002) starts with a time step of 0.02, followed by time steps of 0.015, 0.01 and 0.005. In order to analyze the results of these calculations, it is necessary to understand how DMC calculations are analyzed. The PlotDMC.pl script can show both the trace of the local energy as well as the population at each time step. Look at the first 1000 steps of the first DMC calculation:

```
PlotDMC.pl Be-S8-dmcTsteps.s002.dmc.dat 1 1000
```

This should produce a plot like the one in figure 3 At the beginning of the local energy plot,

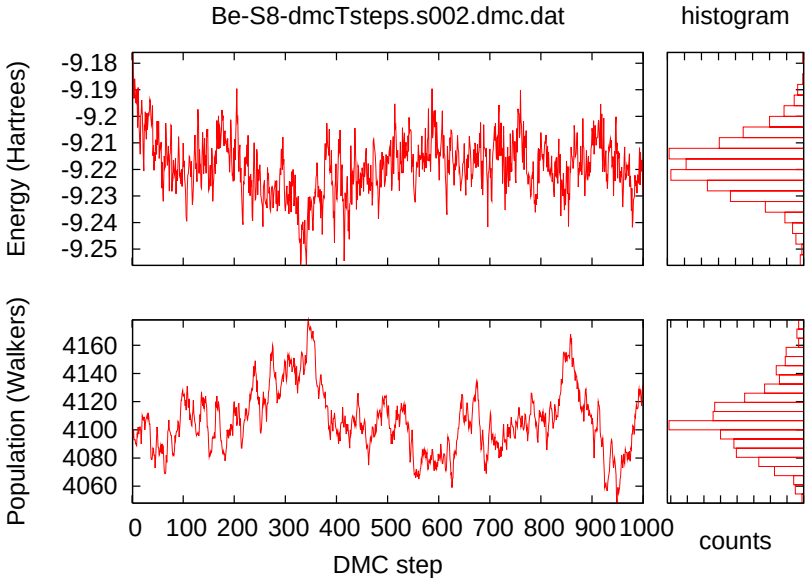

Figure 3: First 1000 steps of DMC calculation with time step of 0.02

there is a transient as the population of walkers equilibrates from sampling Ψ_T^2 to the mixed distribution $\Psi_T\Phi$. After this transient, the local energy sampled should be approximately Gaussian (note recent work by R. Hood which suggests it is not exactly Gaussian) and the population should also be Gaussian. Any large deviations from this suggest that the sampling may not be ergodic. Now look at the full calculation with the transient removed:

```
PlotDMC.pl Be-S8-dmcTsteps.s002.dmc.dat 200
```

This produces a figure like 4 The spikes in the population histogram notwithstanding, this

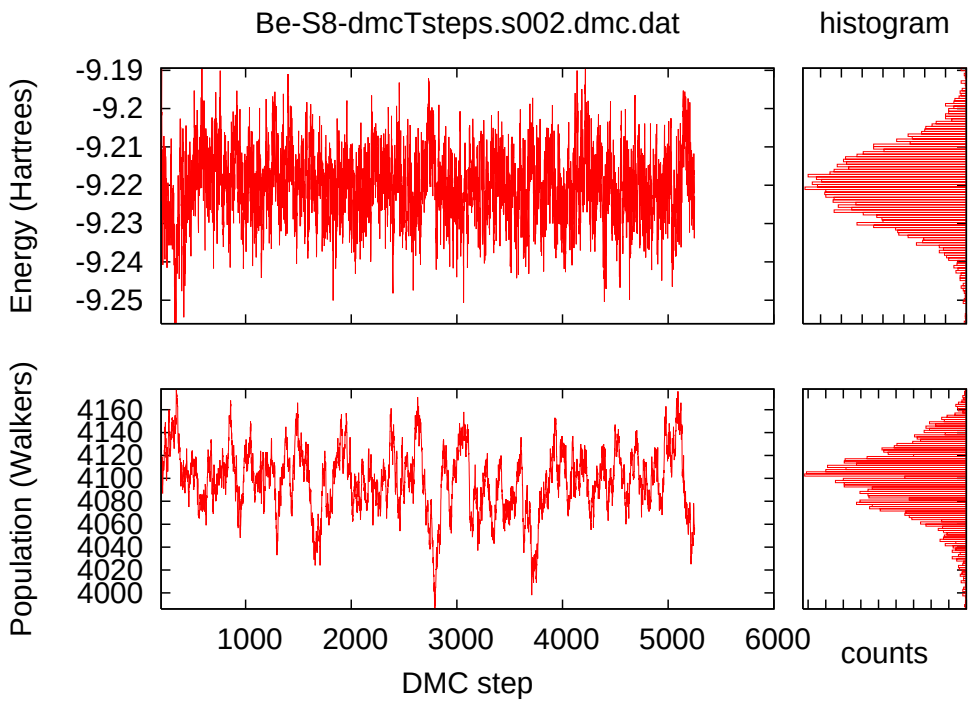

Figure 4: DMC calculation with time step of 0.02, first 200 steps removed to avoid transient

calculation appears to be producing reasonably good statistics. All of the various quantities which are reported in the scalar.dat files may now be examined using the energy.pl tool:

```
energy.pl Be-S8-dmcTsteps.s002.scalar.dat 50
```

Note that the first 50 blocks have been thrown out to avoid the transient (each block contains 25 steps so this is slightly more than in the plots above). This produces the output below.

```
LocalEnergy              =            -9.22005 +/-           0.00057
Variance                 =               0.339 +/-             0.011
LocalPotential           =            -16.4178 +/-           0.0036
Kinetic                  =              7.1978 +/-           0.0032
LocalECP                 =              4.2643 +/-           0.0021
NonLocalECP              =             -3.2580 +/-           0.0048
IonIon                   =      -12.25847000 +/-       0.00000000
ElecElec                 =             -5.1657 +/-           0.0010
MPC                      =             -4.9733 +/-           0.0012
KEcorr                   =      0.0643886418 +/-     0.0000000023
BlockWeight              =            102402 +/-               54
BlockCPU                 =           1.93430 +/-          0.00059
AcceptRatio              =         0.9966894 +/-        0.0000040
Efficiency               =         52939.882 +/-            0.000
```

```
------------------------------------------------------------
Corrected Energy       =           -8.96321 +/-           0.00057
```

All energies are given in Hartrees and the BlockCPU is given in seconds. Serial correlations are accounted for by using the blocking technique. This tool is used extensively by the other analysis tools.

With those preliminaries in hand, we will use the PlotTstepConv.pl tool to analyze the convergence of the DMC calculations with respect to the time step. Run the tool as follows:

```
PlotTstepConv.pl Be-S8-dmcTsteps.xml 50
```

Note that rather than actual data files, this tool directly analyzes the QMCPACK input file to determine the data to plot. The output of the tool should be similar to figure 5

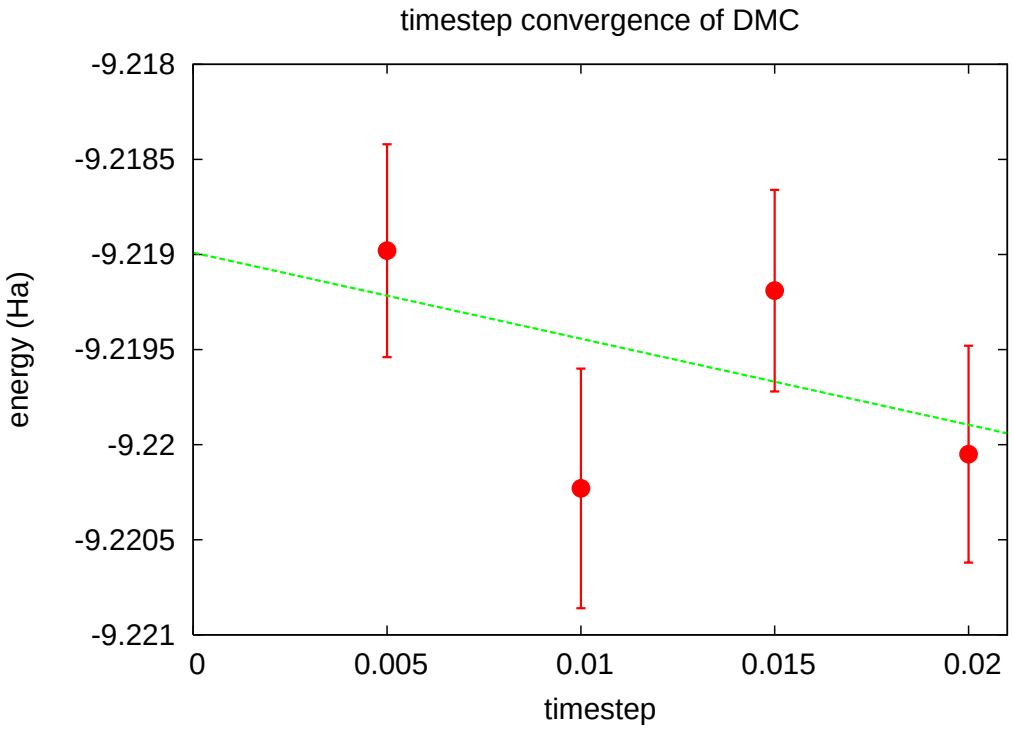

Figure 5: Total DMC energy as a function of time step for Be

The trend line that is drawn may be wildly inaccurate especially if large time steps are included. In this case, there appears to be a very weak dependence on the time step and a time step of 0.02 appears to be a good candidate for future DMC calculations.

6 Controlling Finite Size Effects

If one were to follow the above procedure and produce a curve of energy as a function of unit cell volume, the results would have little relation to reality. The reason for this is

the principle difference between performing QMC calculations on isolated systems and on condensed matter. Effects due to the size of the simulation cell in the QMC calculation lead to significant errors when comparing to physical systems. These effects can be grouped into two categories, one-body and two-body and strategies to minimize their influence on simulation results will be discussed in the next two sections.

6.1 One-Body Finite Size Effects and Twist Averaging

One-body finite size effects will be familiar to practitioners of Density Functional Theory. While Bloch's theorem allows for calculation with only a periodic part of the wavefunction, an integral over crystal momenta (over the first Brillouin Zone) is necessary to calculate observables of the infinite system. The calculations which have been performed in the preceding sections take place at the gamma point (that is $k = (0, 0, 0)$). Due to reasons of symmetry, the properties of the system at the gamma point are often an extremum of the integral over the first Brillouin Zone, exacerbating the problem of incomplete sampling of the integral.

In DFT calculations, the solution to this problem is to perform calculations at a fine grid of k-points and then to average those results. This is the purpose of the section:

```
K_POINTS AUTOMATIC
20 20 20 1 1 1
```

in Be-S8-scf.in which performs the DFT calculation at an 20x20x20 grid of k-points.

In QMC the solution is similar. Varying the k-points in the DFT calculation is equivalent to changing the boundary conditions for the many body wavefunction in the QMC calculation. This can be accomplished by generating a wavefunction from DFT that contains all of the k-points necessary to express the many-body wavefunction with the different boundary conditions in the QMC cell. Then the resulting QMC calculations will be performed with all of these different boundary conditions. Practically speaking, one needs to specify the boundary conditions for the QMC calculations using the –kgrid x x x and –kshift x x x options to setup-qmc.pl. The –kgrid option specifies the number of equally spaced boundary conditions in each direction and the –kshift option specifies an optional shift of this grid from being centered at the gamma point. On a theoretical note, DMC calculations performed away from the gamma point need to have their phase constrained by the trial wavefunction, not simply their nodes as is required at the gamma point.

When performing QMC calculations away from the gamma and L points, QMCPACK must work with complex wavefunctions, which results in roughly a factor of 2 penalty in the speed of the code. Often when combining twist averaging with supercells (as discussed in Sec. 6.2.1), a 2x2x2 grid of supercell twists is sufficient to converge the energy. However, particularly in the case of metals, it may be necessary to go beyond this small grid. There will be no change in the syntax of any of the input files in this case, but the complex aware version of QMCPACK (qmcpack_complex) must be used to run the resulting simulations.

The following commands will allow you to perform DMC calculations on a 2x2x2 grid of k-points for the 8 atom cell of beryllium we have been using above. Note that the DMC

timestep and meshfactor found above will not need to be changed when twist averaging. Additionally, jastrow factors tend to be very similar between the various twists, so it is sufficient to use the jastrow factor optimized at the gamma point for all of the subsequent twists.

```
setup-qmc.pl --genwfn --kgrid 2 2 2 --tilemat 2 -1 0 0 1 -2 1 1 1 Be-scf.in
mpirun -np 8 pw.x -npool 2 -inp Be-8twists-S8-nscf.in > Be-8twists-S8-nscf.out
pw2qmcpack.x < Be-8twists-S8-pw2x.in > Be-8twists-S8-pw2qmcpack.out
wfconv --nospline --eshdf Be-8twists-S8.h5 out/Be.pwscf.h5 >& Be-8twists-S8-wfc
rm out/Be.pwscf.h5
setup-qmc.pl --dmc --kgrid 2 2 2 --wvfcnfile Be-8twists-S8.h5 --splfactor 1.0 \
   --tilemat 2 -1 0 0 1 -2 1 1 1 --vmctimestep 1.0 --vmcblocks 512 --vmcequilti
   --vmcdecorrtime 5.0 --dmctstep 0.01 --dmcequiltime 2.5 --dmcruntime 15.00 \
   --dmcblocktime 0.1 --targetpop 4096 Be-scf.in
cd dmc-S8-8twists
mpirun -np 1 qmcapp Be-S8-dmc-tw0.xml >& Be-S8-dmc-tw0.out
mpirun -np 1 qmcapp Be-S8-dmc-tw1.xml >& Be-S8-dmc-tw1.out
mpirun -np 1 qmcapp Be-S8-dmc-tw2.xml >& Be-S8-dmc-tw2.out
mpirun -np 1 qmcapp Be-S8-dmc-tw3.xml >& Be-S8-dmc-tw3.out
mpirun -np 1 qmcapp Be-S8-dmc-tw4.xml >& Be-S8-dmc-tw4.out
mpirun -np 1 qmcapp Be-S8-dmc-tw5.xml >& Be-S8-dmc-tw5.out
mpirun -np 1 qmcapp Be-S8-dmc-tw6.xml >& Be-S8-dmc-tw6.out
mpirun -np 1 qmcapp Be-S8-dmc-tw7.xml >& Be-S8-dmc-tw7.out
cd ..
```

Once these calculations have completed, a twist averaged value of the quantities can be computed by using the TwistAvg.pl tool.

```
TwistAvg.pl Be-S8-dmc-tw0.s002.scalar.dat 8 50
```

This will perform the proper statistics to average together all of the twists. The computational cost of twist averaging is actually quite small. This small cost is due to the way in which the simulations with different twists are averaged together. The variance of these calculations is typically very close to each other, so to achieve a given overall error after the averaging, roughly the same number of samples are necessary as when running a calculation at a single k-point. The only exception to this is that all of the calculations must be equilibrated, so the cost of the initial VMC calculation and the warmup of the DMC calculation becomes overhead on the twist averaging.

Ideally, one would perform calculations with finer and finer grids of twists until the energy converges. Going to particularly large grids is often not necessary because of the use of supercells. Additionally, the error due to the finite sampling of the twists can be estimated from DFT calculations. The difference in energy between the unconverged QMC calculation and the converged one can be estimated by taking the difference between the energy of the DFT calculation used to create the wavefunction and a fully converged one. As non self consistent calculations in quantum espresso do not report their energy, setup-qmc.pl can generate an appropriate input file using the following command

```
mpirun -np 8 pw.x -npool 2 -inp Be-8twists-S8-pscf.in &> Be-8twists-S8-pscf.out
```

The input file this produces will be Be-S8-8twists-pscf.in. Following this methodology to produce finer grids of twists, you can produce a plot like Fig. 6 with the tool:

```
PlotTwistConv --supercellsize 8 Be-scf.in
```

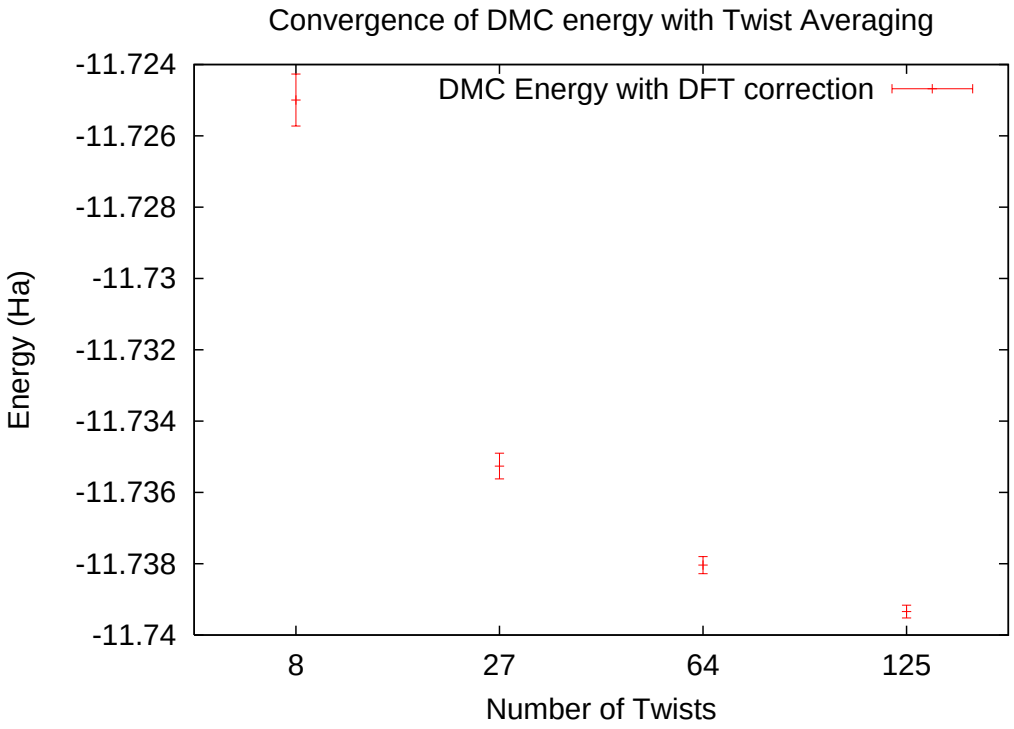

Figure 6: Total DMC energy vs number of twists for Be. Convergence is not achieved by 125 twists in the 8 atom supercell.

6.2 Two-Body Finite Size Effects

The other source of finite size effects comes from the spurious interactions of electrons in the simulation cell with their periodic images in neighboring cells. In DFT calculations these errors are not present because the exchange correlation functionals which deal with these interactions are all derived from calculations on the electron gas that are converged to the limit of infinitely large simulation cells. In this section we will introduce strategies for the mitigation of these two-body finite size effects.

6.2.1 Creating and Using Supercells

The simple solution to reducing the two-body finite size effects is to increase the size of the simulation cell. This can be done in two ways. First, the simulation cell specified in the

DFT calculation can be explicitly made into a supercell. The author generally avoids this approach because the size of the wavefunction increases as the cube of the multiple of the number of primitive cells. In practice the required memory will rapidly outstrip what is available on the nodes of your supercomputer and the calculations will become intractable.

The other approach to generating supercells is to use Bloch's theorem and the repeated zone scheme to generate the wavefunction only in the primitive cell. This recognizes that the wavefunction for each band is just a periodic function times a simple phase factor anywhere in space. Thus, by generating the wavefunction in the primitive cell for the k-points corresponding to these phase factors, the full supercell's wavefunction can be reconstructed by using a vastly smaller amount of data. The memory required for this approach scales only linearly with the number of copies of the primitive cell.

Supercells should also be chosen so as to minimize the distance between points in the supercell and their images. As an example, a long skinny supercell will have much larger finite size errors than a cubic one with the same volume. setup-qmc.pl handles the search for an optimal supercell by searching through all possible combinations with a given number of primitive cell copies and determining the radius of the largest sphere which fits inside these trial supercells. The supercell which fits the largest sphere is then chosen. All of the methods available to setup-qmc.pl take as an optional argument –supercellsize, and will then produce DFT or qmc simulations adapted to this supercell.

The choice remaining is how large of a supercell should be used. Supercells containing all numbers of copies of the primitive cell are not equal in terms of their computational efficiency. As an example, multiples of the primitive cell that can be made into a cube are advantageous because the number of electrons necessary to get a certain distance between points in the supercell and their images (thereby minimizing finite size effects) is relatively small. The tool getBestSupercell.pl can be used to search a range of possible supercell sizes and report on the best ones in terms of efficiency. It is invoked as follows:

```
getBestSupercell --min 8 --max 128 --maxentry 5 Be-scf.in
```

min and max control the range of supercells which should be searched and maxentry specifies the size of the search within each of these supercells. Due to the sub-optimal algorithm used by this tool, it is rather slow but fortunately only has to be run once for any given primitive cell geometry.

In the case of Be, the output of this command searching for supercells having between 8 to 128 copies of the primitive cell shows particularly advantageous supercells for the following multiples: 8, 28, 42, 54, and 128. A finite size scaling study for this system should do calculations on those supercells which searching for convergence.

Now to do calculations on a supercell which has 28 copies of the primitive cell, you can modify the lines in the runscript which contain supercellsizes and twistincs

```
supercellsizes=( 8 )
twistincs=( 2 )
```

Note that as above in the twist averaging example, it is not necessary to reconverge the dmc timestep and the meshfactor when changing supercell size.

6.2.2 Two-Body Finite Size Corrections

While the supercell approach is guaranteed to reduce the finite size effects, it is expensive. The cost of DMC calculations scales as the cube of the number of electrons. Given that we are often looking for per atom quantities, this reduces the actual cost to the square of the number of electrons. However, this still becomes extremely computationally demanding quite quickly.

For this reason, several two-body finite size correction schemes are available which attempt to estimate the two-body finite size errors and correct them. Typically we use two of these, the Model Periodic Coulomb interaction to correct the potential energy and the scheme of Chiesa, Ceperley and Martin to correct the kinetic energy. The necessary quantities are calculated automatically using the tools in this tutorial, with energy.pl for instance giving a line: Corrected Energy which includes both of these corrections.

While these corrections are not perfect for the smallest cell sizes, they do give a significantly better convergence of the total energy with cell size.

6.3 Putting it all Together

Although it will be far too computationally demanding for this tutorial, one should converge the parameters of interest with respect to the supercell size before performing detailed studies. In the case of Be, this would involve repeating the calculations as specified above but with different –supercellsize's. Additionally, multiple supercell twists should be included via the –kgrid option.

Once this has been done, the tool PlotFsConv.pl can be used to produce plots such as Fig. 7 using the command

```
PlotFsConv.pl --minsize 14 Be-scf.in
```

Note that the extrapolation of both the calculations with the 2 body correction and without go to the same value in the infinite system limit. If were not the case, it would suggest that there were difficulties (for example timestep convergence or insignificant twist averaging) and the finite size corrections should not be trusted.

7 Conclusion

This tutorial has covered using pwscf and QMCPACK to perform DMC calculations on BCC beryllium. The convergence of technical parameters including the b-spline mesh, dmc timestep, twist averaging and supercell size are specifically addressed. At this point, the reader should know how to perform converged DMC calculations on similar condensed systems.

8 Acknowledgment

This tutorial was created with support from Sandia National Laboratories.

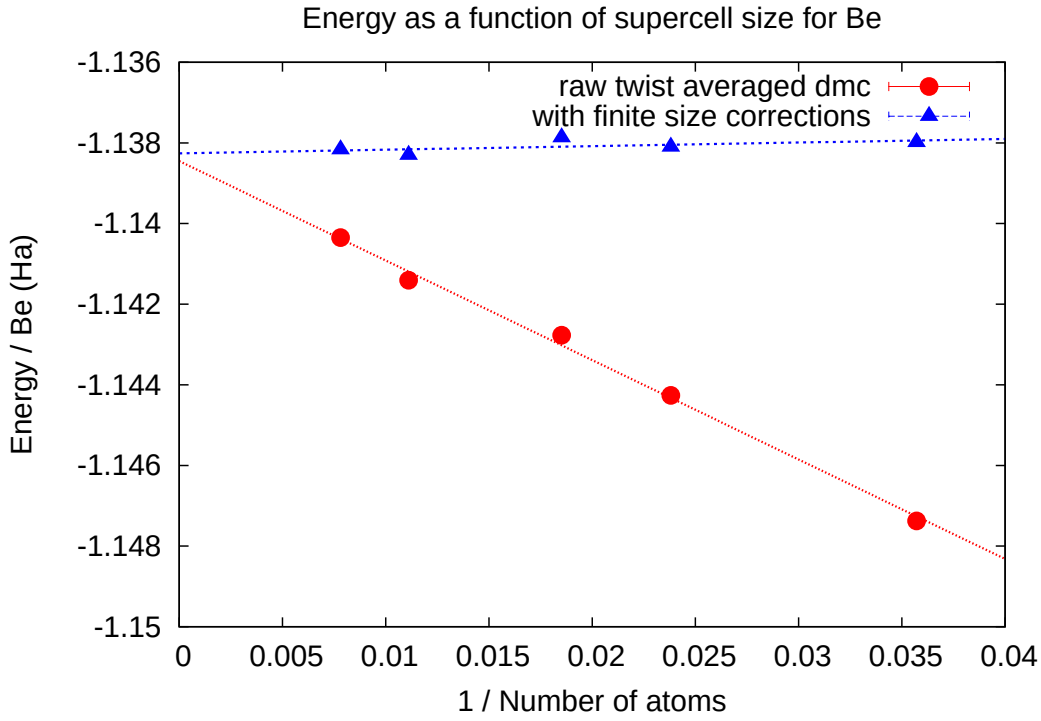

Figure 7: Total DMC energy vs supercell size. Convergence is already good for a supercell that contains 28 copies of the primitive cell.

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.