3D Electron Gas: A QMC Tutorial

Jeremy McMinis, Jeongnim Kim, and David Ceperley

July 22, 2012

1 Introduction

In this tutorial we will show you how to run a Quantum Monte Carlo calculation. We will be using a multi-purpose QMC code developed at Illinois, QMCPACK, to investigate a system, the 3D HEG, covered during the lectures. During the tutorial we will run VMC and DMC and analyze the results. The goal is to familiarize you with running a production level code. The lab will be purposefully kept short to allow interested students the opportunity to ask questions about QMC, the algorithms, wave functions, observables, and potentials coded in QMCPACK, and for developers, how to add things to the code.

2 Instructions

1. General lab instructions can be found at `http://mcc.physics.illinois.edu/doxy/html/a00005.html`.
2. First complete the general lab instructions, which include downloading the information for the HEG tutorial.
3. These instructions can be found on `http://QMCPACK.cmscc.org/tutorials` .
4. Go to your scratch directory and copy the tutorial materials over.
5. All runs have been performed and the data is in the directory structure, we'll talk about what the code is doing and how to analyze the results on the following pages.
6. You are free to modify the input files and resubmit. While we are waiting for your results to run, please ask questions!

3 HEG Calculation: The Code

QMCPACK is an open source continuum quantum Monte Carlo code. It has been scaled to run on the largest machines using both OpenMP and MPI. It can also be built and compiled for GPU machines using the CUDA compiler. It is built using cmake and several external libraries including Boost, Lapack/ACML, XML, FFTW, and hdf5. Einspline is also a suggested package. Pre-made tool chains exist for most large machines, where binaries are also available. More information and tutorials are available at http://QMCPACK.cmscc.org

4 HEG Calculation: The System

For this tutorial we will calculating the energy of the 3D paramagnetic (spin-balanced) electron gas for a smallish cell using a Slater-Jastrow wave function. In atomic units, the Hamiltonian for this system is,

$$\hat{H} = -\frac{1}{2}\sum_i \nabla^2 + \sum_{i<j}\frac{1}{r_{ij}} + C(r_s) \quad (1)$$

where the ∇^2 term is the usual kinetic energy operator, $\sum_{i<j}\frac{1}{r_{ij}}$ is the usual Coulomb potential, $r_s = (\frac{3\rho}{4\pi})^{1/3}$ is the Wigner-Seitz radius, the density is ρ, and $C(r_s)$ is an overall offset coming from the compensating background charge.

4.1 Setting up the System: Technical and Job Related Details

Let's start by looking at the Slater-Jastrow wavefunction calculation, `heg-SJ.xml`.

```
<simulation>
<project id="heg_SJ" series="0">
<application name="qmcapp">
Sample qmc run for Slater-Jastrow HEG.
</application>
</project>
<!-- set seed to -1 for random seed -->
<random seed="11"/>
<include href="heg.ptcl.xml"/>
<include href="heg.sj.wfs.xml"/>
<include href="heg.H.xml"/>
... DRIVER SECTIONS ...
</simulation>
```

1. `<simulation>`
   `...`
   `</simulation>`

   The structure of XML is hierarchical. The code knows to look in the `simulation` bracket to find the information for running the calculation.

2. `<project id="heg_SJ" series="0"/>`

   Here we name the project `id="heg_SJ"`. All the output files with data in them will begin with this prefix. The `series="0"` parameter allows us to start indexing blocks at some other number.

3. `<application name="qmcapp">`
   `Sample qmc run for Slater-Jastrow HEG.`
   `</application>`

   Here is a space to insert comments about the run.

4. `<random seed="11"/>`

   For debugging purposes it is possible to run with the same pseudo-random number seed. Setting it to `seed="-1"` generates a seed from the clock.

5. ```
   <include href="heg.ptcl.xml"/>
   <include href="heg.sj.wfs.xml"/>
   <include href="heg.H.xml"/>
   ```

   Here we can include other xml input files. This is useful when some parts of the input grow large (as the wavefunction and particle sets can).

6. `... DRIVER SECTIONS ...`

   The following sections are either included in the main xml file, or included as separate ones (as shown above). They typically consist of VMC, optimization, or DMC blocks.

4.2 Setting up the System: The Simulation Cell

First we'll define the cell the calculation lives in. This is located in the `heg.ptcl.xml` file.

```
<qmcsystem>
<simulationcell>
<parameter name="rs" condition="14">5</parameter>
<parameter name="bconds">p p p</parameter>
<parameter name="LR_dim_cutoff">6</parameter>
</simulationcell>
</qmcsystem>
```

Let's consider each of these lines separately to describe what they control.

1. ```
   <qmcsystem>
   ...
   </qmcsystem>
   ```

   The ptcl block must exist within the `qmcsystem` block either in the main file or, as here, in an included file.

2. ```
   <simulationcell>
   ...
   </simulationcell>
   ```

   The code knows to look in this section to find the information about the simulation cell we will proceed to define.

3. `<parameter name="rs" condition="14">5.0</parameter>`

   Here we define the Wigner-Seitz radius `rs` to be 5.0 and tell the cell that it should expect 14 electrons. This allows the code to size the cell correctly.

4. `<parameter name="bconds">p p p</parameter>`

   We are working in 3D here so we must define each dimension as periodic `p` or non-periodic `n`.

5. `<parameter name="LR_dim_cutoff">15</parameter>`

   Finally we define the long range cut-off. This parameter sets the k-space cut off for Ewald sums and other operators that live in momentum space. With r_c the real space cutoff (typically half the box side), the k-space cutoff $k_c = r_c^{-1} \times$`LR_dim_cutoff`. For `LR_dim_cutoff` too small we get errors in long range sums, while for too large `LR_dim_cutoff` our calculations are slow. The choice above, 15 is a reasonable one and need not be changed.
   ```

4.3 Setting up the System: The Particle Sets

Next we need to define our electronic species. This is also located in the `heg.ptcl.xml` file. It is not necessary to include it in the same block as the simulation cell block.

```
<particleset name="e" random="yes">
  <group name="u" size="7">
    <parameter name="charge">-1</parameter>
  </group>
  <group name="d" size="7">
    <parameter name="charge">-1</parameter>
  </group>
</particleset>
```

1. `<particleset name="e" random="yes">`
   `...`
   `</particleset>`

   Here we define a set of particles we'll call `e`, short for electrons. we indicate that their starting positions are random using the `random="yes"` tag.

2. `<group name="u" size="7">`
   `<parameter name="charge">-1</parameter>`
   `</group>`

   We name each group and define some characteristics for it including the number of particles and their charge. We name these two species `name="u"` and `name="d"` anticipating that we will use a spin-$\frac{1}{2}$ trial function. At this point the particles have no statistics.

4.4 Setting up the System: The Trial Wavefunction

We are using a trial wave function of the Slater-Jastrow form. This is a product state with a determinant of plane waves multiplied by a bosonic two body correlation factor. This input block is to be found in the `heg.sj.wfs.xml` file. Other wave functions are included along with the output they generate. Typically we start parameters out from all zeros and then optimize them to some value. Here we have included an optimized Jastrow factor.

```
<wavefunction name="psi0" target="e">
  <determinantset type="electron-gas" shell="1"/>
  <jastrow name="J2" type="Two-Body" function="Bspline">
    <correlation speciesA="u" speciesB="u" size="5">
    <coefficients id="uu" type="Array" optimize="yes">
1.082858193 0.6653279375 0.4358910287 0.2243616172 0.1102948764
    </coefficients>
    </correlation>
    <correlation speciesA="u" speciesB="d" size="5">
    <coefficients id="ud" type="Array" optimize="yes">
1.696171854 1.047722154 0.6275148566 0.3175982878 0.1446706214
    </coefficients>
    </correlation>
  </jastrow>
</wavefunction>
```

1. `<wavefunction name="psi0" target="e">`
   `...`
   `</wavefunction>`

   We name the trial wave function and assign a previously named particle set to it.

2. `<determinantset type="electron-gas" shell="1"/>`
   `</determinantset>`

   For a paramagnetic electron gas calculation we use a determinant of plane waves. The `"shell"=1` tag allows the code to determine how many plane wave states to build. This number is indexed the "c" way, starting from zero (e.g. shell:N_e; 0:1, 1:7, 2:19, 3:27, 4:33, etc.) It assumes that there are as many spin up particles as there are spin down particles if only one shell is given. Additional information is required for spin polarized calculations. Currently, only filled shell states are allowed for the simple Slater-Jastrow case.

3. `<jastrow name="J2" type="Two-Body" function="Bspline">`
   `...`
   `</jastrow>`

   Here is the block for the spin dependent B-Spline Jastrow. It is a two body term as indicated by `type="Two-Body"`. This provides correlation for like and not like spin electrons.

4. `<correlation speciesA="u" speciesB="d" size="5" >`
   `...`
   `</correlation>`

   The `speciesA="u"` and `speciesB="d"` tags identify which groups are being correlated. There are `size="6"` parameters for each `correlation` type which corresponds to `size`+3 knots on each B-spline (the extras ensure the cusp and boundary conditions).The default cusp and boundary conditions for our calculation are the same as the codes defaults, so they do not need to be changed.

5. `<coefficients id="uu" type="Array" optimize="yes">`
   `1.082858193 0.6653279375 0.4358910287 0.2243616172 0.1102948764`
   `</coefficients>`

   The `id="ud"` tag identifies the parameters by name and by setting `optimize="yes"` they are visible to the optimization algorithm. The default is that the `"uu"` parameters are identical to the `"dd"` ones, and the latter need not be specified in the input. The string of zeros are the starting parameters.

4.5 Setting up the System: The Hamiltonian

We keep the Hamiltonian in `heg.H.xml`. Again, it could be kept in the main input file if you prefer.

```
<hamiltonian name="h0" type="generic" target="e">
  <pairpot name="ElecElec" type="coulomb" source="e" target="e"/>
  <estimator type="gofr" name="gr" num_bin="200" rmax="9.712825"/>
</hamiltonian>
```

1. The kinetic energy term is included by default.
2. `<pairpot name="ElecElec" type="coulomb" source="e" target="e"/>`

   The `type="coulomb"` tag identifies the type of inter-particle potential. Both the target and source particles are identified, `source="e" target="e"`.

3. `<estimator type="gofr" name="gr" num_bin="200" rmax="9.712825"/>`

   Here we include an additional estimator for the pair correlation function, g(r). We must tell it the cutoff distance, `rmax` and the number of bins to use, `num_bin`.

5 HEG Calculation: The Algorithms

In the following sections we will outline the input file structure for the various algorithms we'll be utilizing to compute the energy of the HEG. Time does not permit detailed descriptions of the algorithms. For the curious or motivated several references are listed at the end of the tutorial for further consideration.

5.1 Variational Quantum Monte Carlo

Let's look in the input file called `heg-SJ-dmc.xml` . This input file includes a VMC and several DMC runs.

```
<qmc method="vmc" move="pbyp">
<parameter name="blocks"> 100  </parameter>
<parameter name="steps">  100  </parameter>
<parameter name="warmupsteps"> 100 </parameter>
<parameter name="tau">  5.0  </parameter>
<parameter name="samples">  800  </parameter>
</qmc>
```

1. `<qmc method="vmc" move="pbyp">`
   `...`
   `</qmc>`

   Here we define the algorithm that we are going to be using, `method="vmc"` and how particle moves are going to be made, `move="pbyp"`. It is also possible to make `move="walker"`moves in which all electrons are moved at the same time. Most of the time it is more efficient to move one particle at a time.

2. `<parameter name="blocks"> 100  </parameter>`
   `<parameter name="steps">  100  </parameter>`
   `<parameter name="warmupsteps"> 100 </parameter>`

   The total number of steps the walkers will be evolved though is `blocks`×`steps`+`warmupsteps`. Statistics will not be kept during the warm up.

3. `<parameter name="tau"> 5.0 </parameter>`

   This is the timestep used to evolve the walkers in imaginary time. We typically choose a time step where the accept rate is about 50%. The optimal chose is the one that maximizes the diffusion of the walkers through phase space, and thus minimizes the autocorrelation time. The total amount of imaginary time the system is evolved through will then be (`blocks`×`steps`+`warmupsteps`)× `tau`.

4. `<parameter name="samples">800</parameter>`

   `samples` are generated to use for the following step (usually DMC). It's the number of walkers we will use for the following algorithm.

5. `<estimator name="LocalEnergy" hdf5="no"/>`

   This line tells the code to write the output to a text file. It will be called [PROJECT NAME].s000.scalar.dat and contain all the observables in the Hamiltonian plus some additional information. If we want the output written to hdf5 then we simply change the `hdf5="no"` label. This is the default.

5.2 Diffusion Quantum Monte Carlo

```
<qmc method="dmc" move="pbyp">
  <parameter name="blocks">100</parameter>
  <parameter name="steps">100</parameter>
  <parameter name="tau">0.01</parameter>
  <parameter name="targetWalkers">800</parameter>
</qmc>
```

The DMC block structure is the same as the VMC one, only now we have the additional parameter `targetWalkers` which controls the size of the target population through the branching control loop. Branching is done differently for the first `warmupsteps` steps due to the energy transient.

There is also a conceptual difference between the VMC and DMC walkers. In VMC we sample a stream of *independent* random walkers. In DMC these random walks are correlated through branching/death process.

6 HEG Calculation: Running the Calculation

The process of running a calculation varies some from machine to machine. Typically, on a cluster, we submit jobs in a job script to a scheduler. The scheduler manages the large number of jobs submitted by all users and runs them according to some set of priorities. Here we present a typical submit script for Forge, `SJ.pbs` .

```
#!/bin/bash
#PBS -N heg_SJ
#PBS -j oe
#PBS -q nomss
#PBS -l walltime=00:05:00,nodes=1:ppn=4
#PBS -A gfi

module list
export MV2_ENABLE_AFFINITY=0

date
cd $PBS_O_WORKDIR

echo "nodefile="
cat $PBS_NODEFILE
echo "=end nodefile"

export NP=‘cat $PBS_NODEFILE | wc -l‘
export OMP_NUM_THREADS=4

export myexe=/uf/ac/jnkim/qmc/bin/qmcapp_real

export myinp=heg-SJ.xml
export myout=heg-SJ.log

mpirun_rsh -ssh -np ${NP} -hostfile ${PBS_NODEFILE} ${myexe} ${myinp} &> ${myout}

mkdir SJ
mv heg-SJ.log heg_SJ.* SJ
```

This submit script will run the input file named `myinp` using the QMCPACK executable and `myexe` and pipe the chatter into `myout` . After running it makes a folder and moves the output of the calculation into it. For more information on PBS options and other environment variables to set see the website of the machine you are running on.

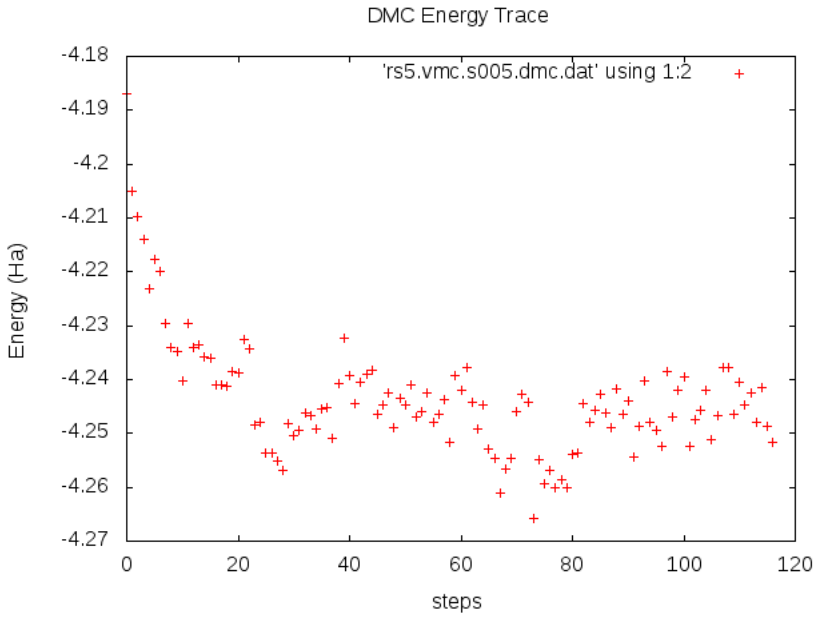

Figure 1: Trace of the energy during a DMC run. The transient is clearly visible on the left.

7 HEG Calculation: Data Analysis

Now let us analyze the results. If you have your own favorite data tool, feel free to use it. Otherwise you can use the `stats` program provided in the QMCPACK folder. It is a command line c code to compute averages, variances, and errors. Data correlations are taken care of by blocking. It's use is illustrated as follows,

```
stats FILENAME [START BLOCKING]
```

`START` is indexed like c, starting from 0. `BLOCKING` tells the code how many adjacent data points to average together before computing statistics. Both these parameters are optional and default to 0, and 1.

7.1 Files Generated

Let's look in the folder named `SJ-dmc` . This has a VMC and a few DMC runs.

The code generates a [`project id`].s[block number].scalar.dat file for each block and an additional [`project id`].s[block number].dmc.dat for each DMC block. Each file has a header that begins with a `#` and the column titles. Column 1 is always the block index and column 2 is always the energy. If you want to plot the trace of the first blocks data you would be able to do so in gnuplot as,

```
> plot 'rs5.tut.s000.scalar.dat' u 1:2
```

7.2 Thermalization

The DMC energy will have a transient period where the energy decays from the VMC energy. The period length depends on the quality of the trial wave function and the gap from the ground to first excited state. It is necessary to truncate it when averaging energies or other observables. An example transient is shown in Fig.1. When doing a time step extrapolation to zero time step it is important to take this into account.

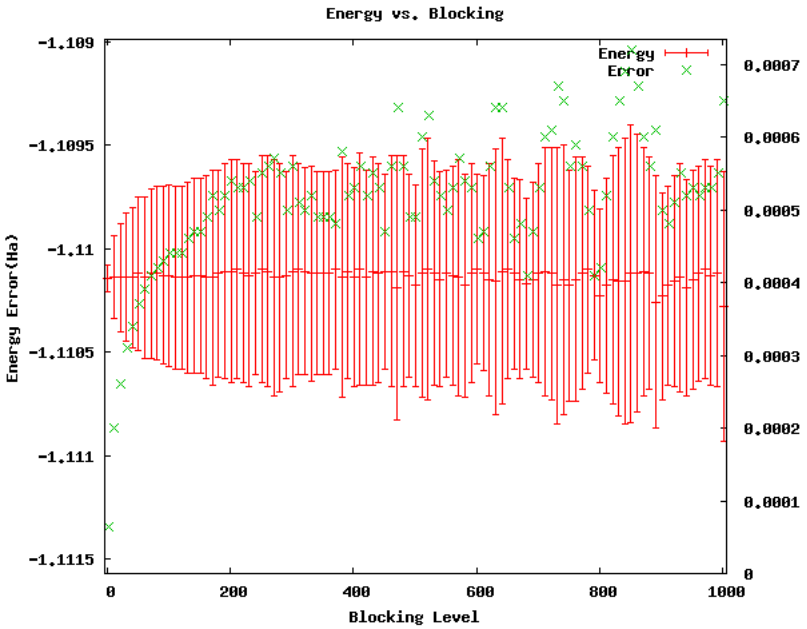

Figure 2: Blocking analysis of the energy during a DMC run.

7.3 Auto-correlation

The error bars computed using `stats` does not compute the autocorrelation time. It is necessary to perform a blocking analysis to determine when correlations have been eliminated from the data. You can automatically do a blocking analysis on the energy trace using the stats program provided as,

```
stats FILENAME [START BLOCKING NBLOCKS]
```

It should produce a .png file from a gnuplot script with the energy blocking analysis as illustrated in Fig.2. The error bars should grow to some value and then level off. The .p file has the gnuplot commands necessary to recreate the plots. If you omit the first two lines you can use gnuplot on your machine to view it.

7.4 Extrapolations

There are other extrapolations for the energy that should be performed for high quality results. We will address these later in the week. The results must be finite size, and finite time step extrapolated. You must also be sure you are using enough walkers that finite population size is not an issue. The issues are addressed in the literature listed at the end of the tutorial.

7.5 Pair Correlation Function

We computed the pair correlation function, g(r), by including it as an estimator in the Hamiltonian. The data for this observable is output in [`project id`].s[block number].stat.h5. A simple script is provided which takes the data from the h5 file, averages it, and plots it. You can run the plot using `processgr`[`project id`].s[block number].stat.h5 This produces the 3 .sum files which contain g(r) ready to be plot. The gnuplot command to plot them is at the top of the .sum file.

8 References

"Ground State of the Electron Gas by a Stochastic Method," Phys. Rev. Lett. 45, 566569 (1980)
"Quantum Monte Carlo simulations of solids," Rev. Mod. Phys. 73, 3383 (2001)
"Optimization of quantum Monte Carlo wave functions by energy minimization," J. Chem. Phys.Phys.

126, 084102 (2007)
"Strategies for improving the efficiency of quantum Monte Carlo calculations," Phys. Rev. E 83, 066706 (2011)
QMCPACK code suite, http://QMCPACK.cmscc.org

9 Running Optimization

QMCPACK uses the linearized method adapted to QMC by Umrigar and co-workers. This algorithm works by expanding the trial wave function in the basis of the derivatives of the trial wave function with respect to its parameters. After computing the overlap and Hamiltonian matrices during a VMC run, we solve a linear system to find the direction the best parameters lie in. We can choose to minimize the energy, variance, or some linear combination of them both. Following is an example code block for the optimization.

```
<loop max="2">
<qmc method="linear" move="pbyp" checkpoint="-1" gpu="no">
<!-- VMC parameters -->
  <parameter name="blocks">     100  </parameter>
  <parameter name="useDrift">  yes </parameter>
  <parameter name="warmupsteps"> 2 </parameter>
  <parameter name="steps"> 60 </parameter>
  <parameter name="timestep"> 5.0  </parameter>
<!-- VMC parameters -->
<!-- Optimization Parameters -->
  <parameter name="samples"> 24000 </parameter>
  <cost name="energy">                   0.0 </cost>
  <cost name="unreweightedvariance">     0.0 </cost>
  <cost name="reweightedvariance">       1.0 </cost>
  <parameter name="minwalkers">  0.0 </parameter>
<!-- Optimization Parameters -->
<!-- Optimization Parameters: Advanced -->
  <parameter name="GEVMethod">mixed</parameter>
  <parameter name="beta">  0.0 </parameter>
  <parameter name="minwalkers">  0.0 </parameter>
  <parameter name="bigchange">50.0</parameter>
  <parameter name="MinMethod">quartic</parameter>
  <parameter name="exp0">-16</parameter>
  <parameter name="alloweddifference"> 1.0e-4 </parameter>
  <parameter name="stepsize">  0.35 </parameter>
  <parameter name="nstabilizers"> 2 </parameter>
<!-- Optimization Parameters: Advanced -->
</qmc>
</loop>
```

1. `<loop max="2">`
   `...`
   `</loop>`

   This sets up the code to loop over the optimization routine `max="2"`times.

2. `<qmc method="linear" move="pbyp">`
   `...`
   `</qmc>`

   Again, we define the algorithm that we are going to be using, `method="linear",`and how particle moves are going to be made.

3. `<!-- VMC parameters -->`

   Comments are included in XML blocks through the use of `<!-- -->` brackets. The VMC parameters in this section are identical to the ones we previously discussed.

4. `<cost name="energy">                   0.0 </cost>`

```
<cost name="unreweightedvariance">     0.0 </cost>
<cost name="reweightedvariance">       1.0 </cost>
<parameter name="samples">     24000  </parameter>
<parameter name="minwalkers">  0.0 </parameter>
```

These parameters control the line minimization part of the optimization algorithm. Once the parameter directions are obtained from the eigenvalue equation, we perform a line minimization on `name="samples"` generated during the VMC run and minimize a cost function containing a linear combination of `energy`,`unreweightedvariance`, and `reweightedvariance`. By changing `minwalkers` we set the smallest percentage of walkers that are okay to optimize on as the parameters in the wave function change.

10 Further Work

Change the density, re-optimize the Jastrow, and compute the g(r). The density is changed in the heg.ptcl.xml file. The stable range for the paramagnetic phase, is $0 < r_s < 70$ or so. You can find an example set of opt blocks in opt.xml. You need to insert them into the trial wave function block you want to optimize (I suggest the Slater-Jastrow one).

1. How does g(r) change as your density changes?
2. How do the kinetic and potential energy components change?

10.1 Other Projects

Other ideas you can explore:

1. Increase the number of particles in the cell.
2. Change the spin polarization.
3. Change the K-point.
4. Change the potential.