Worksheet 1: Using localised orbitals in QMC calculations (1 hour)

Neil Drummond and Pablo López Ríos

2007 Summer School on Computational Materials Science
University of Illinois at Urbana-Champaign, USA
Friday 13th July, 2007

Scaling of QMC calculations with system size

The rate-determining step in most QMC calculations is the evaluation of the orbitals in the Slater wave function. If each of the N orbitals at the N electron positions is expanded in $\mathcal{O}(N)$ basis functions (e.g. plane waves) then the time taken to evaluate the Slater wave function scales as $\mathcal{O}(N^3)$. On the other hand, if a localised basis set such as a cubic spline representation is used then the the time taken to evaluate each orbital is independent of system size, and hence the time taken to evaluate the wave function scales as $\mathcal{O}(N^2)$. Finally, if the orbitals themselves are spatially localised and truncated then the number of nonzero orbitals to evaluate at each electron position is independent of system size, and so the time taken to compute the wave function is $\mathcal{O}(N)$. The use of localised bases and orbitals has allowed QMC calculations to be performed for systems with hundreds of atoms.

In this worksheet you will learn how to perform QMC calculations using localised orbitals represented by splines. This should demonstrate the importance of using a localised basis set, and draw your attention to some of the issues associated with the use of localised orbitals. You will also gain experience of using CASINO to perform VMC calculations. Detailed information about CASINO can be found in the CASINO manual (see `CASINO/manual/casino_manual.pdf`).

CASINO's input files and CASINOHELP

Go to the directory `Silicon_loc`. This contains the input files required to perform a VMC simulation of a 16-atom cell of silicon:

- DFT plane-wave orbitals (generated using the CASTEP code) are held in `pwfn.data`. This file also contains the atom species and positions.
- A silicon pseudopotential is held in `si_pp.data`.
- The parameters describing CASINO's Jastrow factor are held in `correlation.data`. All optimisable wave-function parameters go in this file.
- Finally, the input parameters required to drive the CASINO calculation are contained in the `input` file.

Have a look at the `input` file. Input parameters are specified in the form "keyword : value". To obtain information about a particular keyword, e.g. **interaction**, type *casinohelp interaction*. Try this for a few of the keywords in the `input` file. In fact most keywords can be omitted from the `input` file, in which case they are given default values. To see a complete list of keywords, type *casinohelp all*.

Plane-wave and blip orbitals

Running a VMC calculation with plane-wave orbitals

The `input` file is set up to run a VMC calculation using the plane-wave orbitals in `pwfn.data`. Here is a summary of the most relevant keywords in the `input` file (use CASINOHELP to obtain more information):

neu, ned Number of up- and down-spin electrons in the simulation cell. (32 of each in this case.)

atom_basis_type Representation of orbitals to be used, e.g. "plane-wave" or "blip". For the first calculation this should be set to "plane-wave".

periodic Is the system periodic? (Yes, in this case).

npcell Array of primitive cells making up the simulation cell[1]. In this case the entire 16-atom cell is treated as a single primitive cell.

runtype Type of calculation to perform. (VMC in this case.)

nequil Number of configuration moves to perform at the start of a VMC simulation to equilibrate the Metropolis algorithm. A few thousand moves is typical.

nmove Number of VMC energy data to calculate per "block" of moves. (At the end of each block CASINO checkpoints and writes out the mean energy for that block.)

nblock The total number of "blocks" of moves to perform.

corper The VMC energy is evaluated every **corper**th configuration move to minimise inefficiencies due to serial correlation. In fact, by choosing a large value of **corper**, serial correlation can be eliminated from the VMC data. The total number of actual configuration moves in a block[2] is **nmove**×**corper**. **corper** $= 4$ is typical for VMC work.

dtvmc CASINO uses a Gaussian transition-probability density for trial electron moves; **dtvmc** is its variance. If **opt_dtvmc** is set to 1 then **dtvmc** will be "optimised" so that about half of the proposed electron moves are accepted, which roughly maximises the efficiency of VMC.

Now run CASINO by doing one of the following:

- If you are running on a serial machine without a queueing system (such as your laptop) then type *runqmc*;
- If you are running on a parallel machine with a queueing system (such as `tungsten`) then type something like *runqmc nnodes 4 time 00:05* (which means "run CASINO on 4 processors with a time limit of 5 minutes"). If CASINO is run on 4 processors then an independent random walk is performed on each processor and the results are averaged before being written to `vmc.hist`. So, for a given `input` file, it will take just as long to run CASINO on 4 processors as on 1, but 4 times as much data will be gathered in the former case, and hence the statistical error bars will be smaller by a factor of 2.

When CASINO has finished running, three new files should have appeared: `out`, which contains a summary of the progress of the CASINO calculation, `vmc.hist`, which contains the raw data generated by CASINO, and `config.out`, which contains data required to continue the VMC calculation. It is usually a good idea to read through the `out` file, to check that the calculation has proceeded as expected.

Type *reblock* to analyse the data in `vmc.hist`. You will be asked which units you wish to use (up to you) and to enter a block length for reblocking the data. Examine the table of standard error in the mean energy against block length, and choose a block length such that the standard error has converged (e.g. a block length of 8). Take a note of the total energy with its reblocked error bar.[3] To see a plot of the reblocked error bar against the reblocking transformation, type *plot_reblock*. For a VMC run with **corper**=4, the data are largely free of serial correlation, and hence the error bar does not rise appreciably before reaching its plateau. Also take a note of the CPU time taken to perform the calculation, which can be found at the bottom of the `out` file.

Once you have finished looking at the output files, type *clearup* to delete them.

[1] Recall that e.g. a $2 \times 2 \times 2$ $\mathbf{k}$-vector mesh must be unfolded into a simulation cell consisting of $2 \times 2 \times 2$ primitive cells.

[2] In fact the total number of configuration moves per block is **nmove**×**corper**×**nvmcave** (use CASINOHELP if you're interested), but we are leaving **nvmcave** at its default value of 1.

[3] Note that the error bars on the energies reported at the end of each block in the `out` file do not take serial correlation into account; they are therefore underestimates, unless **corper** is sufficiently large that the energy data are uncorrelated.

Generating blip orbitals

It is very rare to use a plane-wave representation of the orbitals in QMC calculations as we have just done, because it is painfully slow for all but the simplest cases. Instead the plane-wave orbitals should be represented in terms of cubic spline functions on a real-space grid.

To generated a B-spline ("blip") representation of the orbitals in `pwfn.data`, type *blip*. You will be asked to enter the "grid multiplicity", which determines the fineness of the spline grid in real space[4]. Choose a multiplicity of 2, which is a fairly typical value. BLIP will then ask you whether you want to test the blip grid by random sampling or by evaluating the kinetic energy of each blip orbital numerically. If you have plenty of time then answer "Y" to both these questions; otherwise answer "Y" to the first and "N" to the second. The random sampling test estimates the overlap α of the blip and plane-wave orbitals by a Monte Carlo method; the overlap should be close to 1. The kinetic-energy test allows you to compare the kinetic energies of the blip and plane-wave orbitals, which should be very similar if the grid multiplicity is large enough.

When BLIP has finished, a `bwfn.data` file containing the blip orbitals should have appeared.

Running a VMC calculation with blip orbitals

To tell CASINO to use the blip orbitals in `bwfn.data`, use your favourite text editor to change the value of the **atom_basis_type** keyword in `input` from "plane-wave" to "blip". Then run CASINO, as you did for the plane-wave orbitals, and use REBLOCK to obtain the energies with their error bars. Do the energies obtained with the plane-wave and blip representations of the orbitals agree? How do the CPU times compare?

When you are convinced that representing plane-wave orbitals in a blip basis is a good idea, type *clearup*.

Varying the grid multiplicity (longer activity—optional)

Study the effect of varying the grid multiplicity by producing a graph of VMC energy against the multiplicity. In order for the energy differences to be statistically significant, you will need to increase the number of VMC iterations (**nmove** in the `input` file) or use several processors on a parallel computer. You should find that the VMC energy obtained with blip orbitals converges to the energy obtained with the plane-wave orbitals as the multiplicity gets large.

Can you work out what prevents one from using an arbitrarily large grid multiplicity in practice?

Localised orbitals

The calculations that you have performed so far have used the canonical orbitals generated by the plane-wave DFT code, which extend over the entire simulation cell; you have merely changed the way in which those orbitals are represented. Each time an electron is moved, every orbital must be evaluated at the electron's new position in order to compute the Slater determinant. If spatially localised orbitals were used, however, then only those orbitals in the vicinity of the electron's new position would have to be evaluated; the rest could be assumed to be zero. Spatially localised orbitals can be created by forming linear combinations of the original extended orbitals. The algorithm for constructing localised orbitals involves maximising the overlap of the orbitals with specified localisation regions[5].

Let's carry out a localisation transformation for the orbitals in `pwfn.data`. Copy `centres.dat` from the `Other_files` directory into `Silicon_loc`, and have a look at the file. The `centres.dat` file defines the regions in which the localised orbitals are constructed. The localisation regions may be either spherical or parallelepiped-shaped, and are centred on the points listed in `centres.dat`[6]. This file also specifies the number of localised orbitals to be contained in each region.

Type *localizer*. This will generate a file called `pwfn.data.localized` containing the localised orbitals represented in a plane-wave basis. Rename `pwfn.data` as `pwfn.data.extended` and copy `pwfn.data.localized`

[4] If the $\mathbf{G}$ vectors of the plane-wave calculation form an $n \times n \times n$ mesh then the grid of points in real space is an $mn \times mn \times mn$ mesh, where m is the grid multiplicity.

[5] At present localised orbitals can only be formed from orbitals at Γ; this is not a major limitation because calculations for large systems (e.g. large molecules or periodic systems with defects) are usually performed at Γ only anyway.

[6] In practice one normally uses parallelepiped-shaped localisation regions, because the portion of blip grid assigned to the localised orbital is parallelepiped-shaped, and one may as well use all of this grid.

to `pwfn.data`. Then type *blip*, and choose the multiplicity to be 2 as before. (Note that if a `centres.dat` file is present then BLIP will produce localised orbitals that are truncated to zero outside the localisation regions defined in `centres.dat`. If you want to produce a non-truncated blip representation of the plane wave orbitals in `pwfn.data` then you must remove `centres.dat`.)

Running a VMC calculation with localised orbitals

Make sure that **atom_basis_type** is set to "blip" in the `input` file and run a VMC calculation using RUNQMC. How do the energies compare with those that you obtained before?

Near the end of the `out` file the "hit fraction" is reported: this is the fraction of localised-orbital evaluations in which it has been necessary to compute the orbital value (i.e. where the orbital was nonzero). The smaller this fraction, the greater the speedup resulting from localisation.

Has the use of localised orbitals made a significant difference to the simulation time? Can you explain what you have found?

As usual, type *clearup* when you have finished.

Changing the size of the localisation regions (longer activity—optional)

Study the effect of changing the size of the localisation regions by varying the localisation "radii" in `centres.dat`[7], then using LOCALIZER to generate new orbitals, then using BLIP to generate a new `bwfn.data` file, before running CASINO. If you make the localisation regions small then the hit fraction will be low and calculations will be rapid, but the energy will become increasingly inaccurate. Keeping the radii all the same, plot the VMC energy and hit fraction against the cutoff radius. Do you think localised orbitals could be useful in this system?[8]

[7] For a parallelepiped-shaped localisation region, the faces of the parallelepiped are parallel to those of the simulation cell and tangential to a sphere whose radius is the localisation radius.

[8] "Useful" values for the localisation radius are typically 7–8 a.u., so localised orbitals only start to become useful in somewhat larger systems.