QMCPACK Tutorial: Molecular Calculations

Miguel A. Morales[*]

Lawrence Livermore National Laboratory, Livermore, California 94550, USA

(Dated: July 15, 2014)

Contents

I. Exercise #1: Basics 3

A. Generation of a HF wave-function with GAMESS 3

B. Optimize the wave-function 4

C. Time-step Study 6

D. Walker Population Study 7

II. Exercise #2: Wave-function options 9

A. Influence of Jastrow on VMC energy with HF wave-function 10

B. Generation of wave-functions from DFT using GAMESS 11

C. Optimization and DMC calculations with DFT wave-functions 12

D. Generation of a CISD wave-functions using GAMESS 13

E. Optimization of Multi-Determinant wave-function 13

III. Exercise #3: Orbital sets and Configurations in Multi-Determinant Wave-functions 15

A. GAMESS input 17

1. HF input 17

2. DFT calculations 18

3. Multi-Configuration Self-Consistent Field (MCSCF) 18

4. Configuration Interaction (CI) 19

5. GUGA: Unitary Group CI package 19

a. GUGA-MCSCF 19

[*] `moralessilva2@llnl.gov`

b. GUGA-CASCI 20
c. GUGA-Truncated CI 20
d. GUGA-SOCI 20
6. ECP 21

B. convert4qmc 22

C. Wave-function Optimization XML block 25

D. VMC and DMC XML block 27

E. Wave-function XML block 29

I. EXERCISE #1: BASICS

The purpose of this exercise is to show how to generate wave-functions for QMCPACK using GAMESS and to optimize the resulting wave-functions using VMC. This will be followed by a study of the time-step and walker population dependence of DMC energies. The exercise will be performed on a water molecule at the equilibrium geometry.

A. Generation of a HF wave-function with GAMESS

From the top directory, go to "GAMESS/DFT/HF". This directory contains an input file for a HF calculation of a water molecule using BFD ECPs and the corresponding cc-pVTZ basis set. The input file should be named: "H2O.HF.inp". Study the input file. If the student wishes, he can refer to section A for a more detailed description of the GAMESS input syntax. There will be a better time to do this soon, so we recommend that the student continues with the exercise at this point. After you are done, execute GAMESS with this input and store the standard output in a file named "H2O.HF.output". Finally, use *convert4qmc* to generate the QMCPACK particleset and wavefunction files. It is always useful to rename the files generated by *convert4qmc* to something meaningful, since by default they are called sample.Gaussian-G2.xml and sample.Gaussian-G2.ptcl.xml. In a standard computer (without cross-compilation), these tasks could be accomplished by the following commands.

```
cd ${TRAINING_TOP}/GAMESS/DFT/HF
rungms H2O.HF.inp > H2O.HF.out
convert4qmc -gamessAscii H2O.HF.output -add3BodyJ
mv sample.Gaussian-G2.xml H2O.HF.wfs.xml
mv sample.Gaussian-G2.ptcl.xml H2O.ptcl.xml
```

Due to the particular requirements of Vesta these executions can not be performed on the login nodes, they must be performed in the compute nodes. In order to accomplish this, we must make a submission script with the appropriate commands and submit it to the batch system. Two submission scripts have been provided, one for the GAMESS execution called submit_gamess.csh and one for the submission of convert4qmc, with a similar descriptive name. Study these input files. (In this and all other exercises, you will need to make all submission scripts executables with the command **chmod u+x script.csh**.) When you are ready, start by submitting the GAMESS execution to the batch system using "./script_gamess.csh" (This script calls the GAMESS run script, which itself calls qsub. Do not attempt to submit this specific script with qsub). The HF energy of the system is " -16.9600590022". To search for the energy in the output file quickly, you can use **grep"TOTAL ENERGY =" H2O.HF.output**. When the calculation completes, submit the execution of the converter using "qsub submit_convert.csh" (all QMCPACK execution scripts will be submitted with qsub). This is a good time to review section B, which contains a description on the use of the converter.

B. Optimize the wave-function

When the execution of the previous steps is completed, there should be 2 new files called H2O.HF.wfs.xml and H2O.ptcl.xml. Now we will use VMC to optimize the Jastrow parameters in the wave-function. From the top directory, go to "QMCPACK/SJ/First_Run/Optimize". Copy the xml files generated in the previous step to the current directory. This directory should already contain a basic QMCPACK input file for an optimization calculation (optm.xml) and a submission script (submit.csh). Open optm.xml with your favorite text editor and modify the name of the files that contain the wavefunction and particleset XML blocks. These files are included with the commands: <include href="ptcl.xml"/ > and <include href=wfs.xml"/ > (the particle set must be defined before the wave-function). The name of the files should now be H2O.ptcl.xml and H2O.HF.wfs.xml. Study both files and submit when you are ready. Notice that the location of the ECPs has been set for you, in your own calculations you have to make sure you obtain the ECPs from the appropriate libraries and convert them to QMCPACK format using ppconvert. This is a good time to study section C, which contains a review of the main parameters in the optimization XML block, while this calculation finishes. The previous steps can be accomplished by the following commands:

```
cd ${TRAINING_TOP}/QMCPACK/SJ/First_Run/Optimize
```

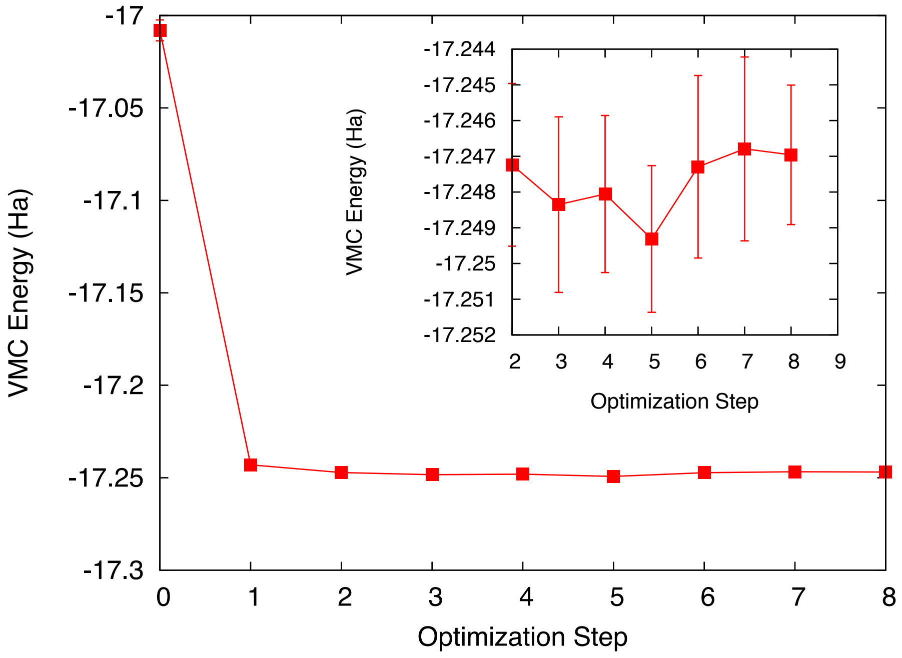


FIG. 1: VMC energy as a function of optimization step.

```
cp ${TRAINING_TOP}/GAMESS/DFT/HF/H2O.*.xml .
qsub ./submit.csh
```

Use the analysis tool **qmca** to analyze the results of the calculation. Obtain the VMC energy and variance for each step in the optimization and plot it using your favorite program. Remember that **qmca** has built-in functions to plot the analyzed data. The resulting energy as a function of optimization step should look qualitatively similar to figure 1 below.

The energy should decrease quickly as a function of the number of optimization steps. After 6-8 steps, the energy should be converged to $\sim 2 - 3mHa$. To improve convergence, we would need to increase the number of samples used during the optimization. You can check this for yourself on your free time. With optimized wave-functions we are in a position to perform VMC and DMC calculations. The modified wave-function files after each step are written in a file named ID.sNNN.opt.xml, where ID is the identifier of the calculation defined in the input file (this is defined in the project XML block with parameter “id”) and NNN is a series number which increases with every executable xml block in the input file.

C. Time-step Study

Now we will study the dependence of the DMC energy with time-step. From the top directory, go to "QMCPACK/SJ/First_Run/TimeStep". This folder contains a basic xml input file (vmc_dmc.xml) that performs a short VMC calculation and three DMC calculations with varying time-steps (0.1, 0.05, 0.01). Copy the particle set and the last optimization file from the previous folder (the file called H2O.sNNN.opt.xml with the largest value of NNN). Rename the optimized wave-function to any suitable name if you wish, for example H2O.HF.opt.xml, and change the name of the particle set and wave-function files in the input file. An optimized wave-function can be found in the reference files (same location) in case it is needed. Using the submission script of the previous exercise as a base, create a submission script for this step and submit the run. Set the number of nodes to 32 (2 places must be changed), the number of threads to 16 and leave the number of tasks at 1.

The main steps needed to perform this exercise are:

cd ${TRAINING_TOP}/QMCPACK/SJ/First_Run/Timestep

cp ${TRAINING_TOP}/QMCPACK/SJ/First_Run/Optimize/H2O.HF.s007.xml H2O.HF.opt.xml

cp ${TRAINING_TOP}/QMCPACK/SJ/First_Run/Optimize/H2O.ptcl.xml .

cp ${TRAINING_TOP}/QMCPACK/SJ/First_Run/Optimize/submit.csh .

vim vmc_dmc.xml

vim submit.csh

qsub ./submit.csh

While these runs complete, go to section D and review the basic VMC and DMC input blocks. Notice that in the current DMC blocks, as the time-step is decreased the number of blocks is also increased. Why is this?

When the simulations are finished, use **qmca** to analyze the output files and to plot the DMC energy as a function of time-step. Results should be qualitatively similar to those presented in figure 2, in this case we present more time-steps with well converged results to better illustrate the time-step dependence. In realistic calculations, the time-step must be chosen small enough so that the resulting error is below the desire accuracy. Alternatively,

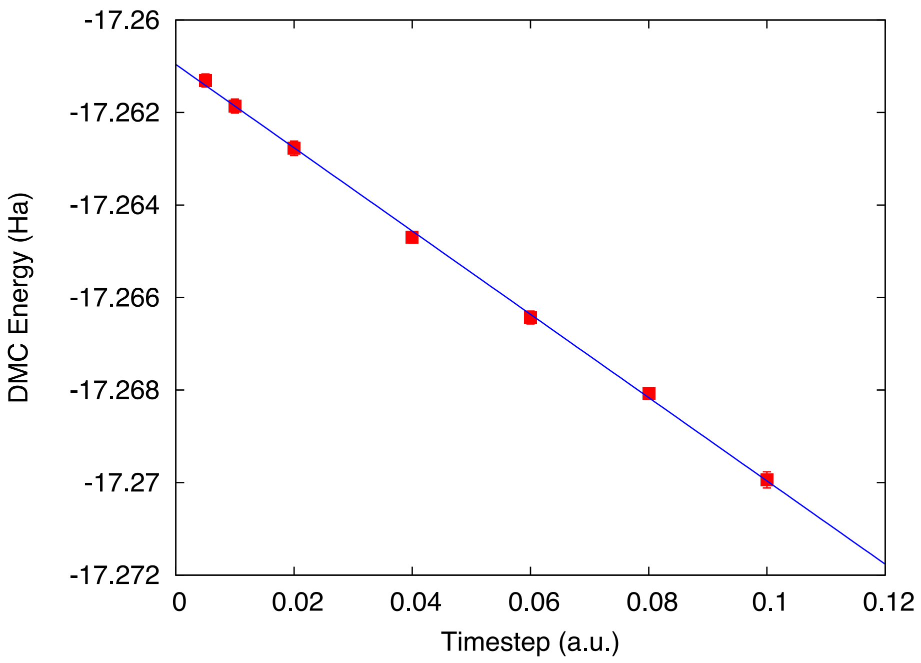


FIG. 2: DMC energy as a function of timestep.

various calculations can be performed and the results extrapolated to the zero time-step limit.

D. Walker Population Study

Now we will study the dependence of the DMC energy with the number of walkers in the simulation. Remember that, in principle, the DMC distribution is reached in the limit of an infinite number of walkers. In practice, the energy and most properties converge to high accuracy with ∼100-1000 walkers. The actual number of walkers needed in a calculation will depend on the accuracy of the VMC wave-function and on the complexity and size of the system. Also notice that using too many walkers is not a problem, at worse it will be inefficient since it will cost more computer time than necessary. In fact, this is the strategy used when running QMC calculations on large parallel computers since we can reduce the statistical error bars efficiently by running with large walker populations distributed across all processors.

From the top directory, go to "QMCPACK/SJ/First_Run/NumberWalkers". Copy the optimized wave-function and particle set files used in the previous calculations to the current folder, these are the ones generated on step 2 of this exercise. Optimized wave-functions files can also be found in "REFERENCE/QMCPACK/SJ/First_Run/Optimize". The directory contains a sample DMC input file and submission script. Make 3 directories named NWx, with x values 60,120,480 and copy the input file and submission script to each one. Go to "NW60", and, in the input file, change the name of the wave-function and particle set files (in this case they will be located one directory above, so use "../H2O.HF.opt.xml" for example), change "targetWalkers" to 60, change the number of steps to 100, the time-step to 0.04 and the number of blocks to 400. Notice that "targetWalkers" is one way to set the desired (average) number of walkers in a DMC calculation. Modify the submission script to use 2 nodes and 16 threads and submit the run. We recommend setting ∼2*(#threads) walkers per node (slightly smaller than this value), which is the reason why we use 2 nodes when we want 60 walkers in total.

Repeat the same procedure in the other folders by setting (targetWalkers=120, steps=100, timestep=0.04, blocks=200,nodes=4,threads=16) in NW120 and (targetWalkers=480, steps=100, timestep=0.04, blocks=100,nodes=16,threads=16) in NW480. When the simulations complete, use **qmca** to analyze and plot the energy as a function of the number of walkers in the calculation. As always, figure 3 shows representative results of the dependence of the energy on the number of walkers for a single water molecule. As shown, less than 240 walkers are needed to obtain an accuracy of 0.1 mha.

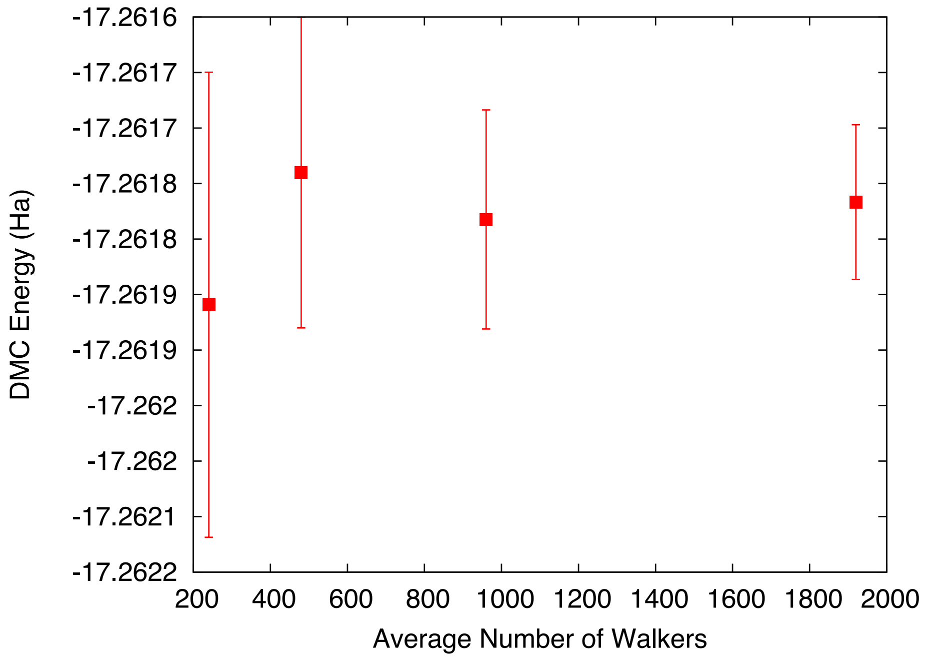


FIG. 3: DMC energy as a function of the average number of walkers.

II. EXERCISE #2: WAVE-FUNCTION OPTIONS

From this point on in the tutorial we assume familiarity with the basic parameters in the optimization, VMC and DMC XML input blocks of QMCPACK. In addition, we assume familiarity with the submission system. As a result, the folder structure will not contain any prepared input or submission files, the student will generate them using generic versions provided in "QMCPACK/Generic_Files" and "GAMESS/Generic_Files". In the case of QMCPACK sample files, you will find optm.xml, vmc_dmc.xml and submit.csh files. Some of the options in these files can be left unaltered, but many of them will need to be tailored to the particular calculation.

In this exercise we will study the dependence of the DMC energy on the choices made in the wave-function ansatz. In particular, we will study the influence/dependence of the VMC energy with the various terms in the Jastrow. We will also study the influence of the VMC and DMC energies on the single particle orbitals used to form the Slater determinant in single determinant wave-functions. For this we will use wave-functions generated with various exchange-correlation functionals in DFT. Finally, we will optimize a simple multi-determinant wave-function and study the dependence of the energy o the number of configurations used in the expansion. All of these exercises will be performed on the water molecule at equilibrium.

A. Influence of Jastrow on VMC energy with HF wave-function

In this section we will study the dependence of the VMC energy on the various Jastrow terms, e.g. one-body, two-body and three-body. From the top directory, go to “QMCPACK/SJ/Jastrow”. We will compare the single determinant VMC energy using a two-body Jastrow term, both one- and two-body terms and finally one-, two- and three-body terms. Since we are interested in the influence of the Jastrow, we will use the HF orbitals calculated in exercise # 1. Make three folders named 2J, 1-2J and 1-2-3J. For both 2J and 1-2J (we have already optimized a wave-function for the 1-2-3J case, so the steps will be slightly different in this case), copy the file optm_vmc.xml and the sample submission file from “QMCPACK/Generic_Files” . This input file performs both wave-function optimization and a VMC calculation. Copy the un-optimized HF wave-function and particle set files from “GAMESS/DFT/HF”, if you followed the instructions in exercise #1 these should be named H2O.HF.wfs.xml and H2O.ptcl.xml. Otherwise, you can obtained them from the REFERENCE files. Modify the file H2O.HF.wfs.xml to remove the appropriate jastrow blocks. For example, for a two-body Jastrow (only), you need to eliminate the jastrow blocks named <jastrow name=”J1” and <jastrow name=”J3”. In the case of 1-2J, remove only <jastrow name=”J3”. Recommended settings for the optimization run are: nodes=32, threads=16, blocks=250, samples=128000, time-step=0.5, 8 optimization loops, and in the VMC section we recommend walkers=16, blocks=1000, steps=1, substeps=100. Notice that samples should always be set to blocks*threads_per_node*nodes = 32*16*250=128000. Repeat the process in both 2J and 1-2J cases. For the 1-2-3J case, the wave-function has already been optimized in the previous exercise. Copy the optimized HF wave-function and the particle set from QMCPACK/SJ/First_Runs/Optimize. Copy the input file and submission script from any of the previous runs and remove the optimization block from the input, just leave the VMC step. In all three cases, modify the submission script and submit the run.

These simulations will take several minutes to complete. This is an excellent opportunity to go to section E and review the wavefunction XML block used by QMCPACK. When the

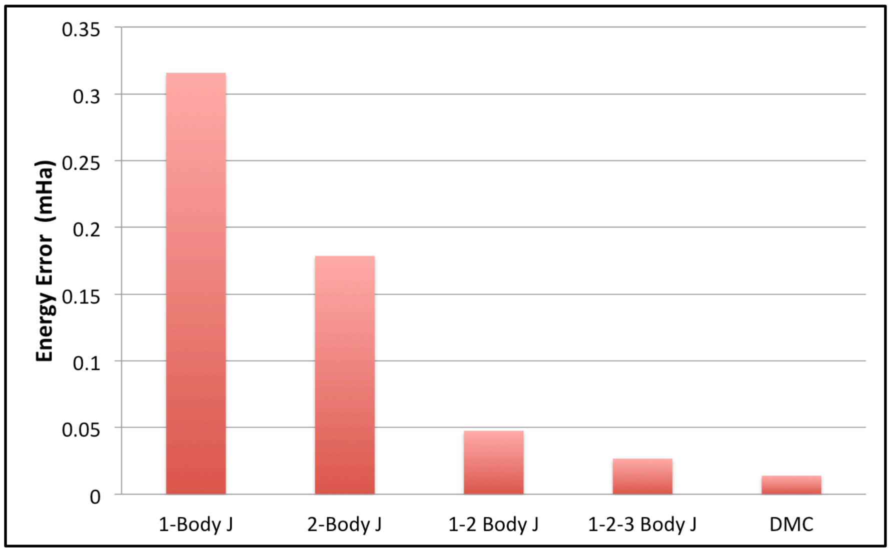


FIG. 4: VMC energy as a function of Jastrow type.

simulation are completed, use **qmca** to analyze the output files. Using your favorite plotting program (e.g. gnu plot), plot the energy and variance as a function of the Jastrow form. Figure 4 shows a typical result for this calculation. As can be seen, the VMC energy and variance depends strongly on the form of the Jastrow. Since the DMC error bar is directly related to the variance of the VMC energy, improving the Jastrow will always lead to a reduction in the DMC effort. In addition, systematic approximations (time-step, number of walkers, etc) are also reduced with improved wave-functions.

B. Generation of wave-functions from DFT using GAMESS

In this section we will use GAMESS to generate wave-functions for QMCPACK from DFT calculations. From the top folder, go to "GAMESS/DFT". In order to demonstrate the variation in DMC energies with the choice of DFT orbitals, we will choose the following set of exchange-correlation functionals (PBE, PBE0, BLYP, B3LYP). For each functional, make a directory using your preferred naming convention (e.g. the name of the functional). Go into each folder and copy a generic GAMESS input file for a ROHF calculation from "GAMESS/Generic_Files", a file named rohf.inp should exist. Rename the file with your preferred naming convention, we suggest using H2O.DFT.inp, where DFT is the name of the functional used in the calculation. At this point, this input file should be identical to the one used to generic the HF wave-function used in exercise #1. In order to perform a DFT calculation we only need to define "DFTTYP" in the $CONTRL ... $END section and set it to the desired functional type, for example "DFTTYP=PBE" for a PBE functional. This variable must be set to (PBE, PBE0, BLYP, B3LYP) to obtain the appropriate functional in GAMESS. For a complete list of implemented functionals, see the GAMESS input manual.

Modify the sample submission script found in "GAMESS/Generic_Files" and submit the job (independently on each folder for each DFT functional). Remember to include the execution of the converter in the submission script, a three-body Jastrow term should be added to the wave-function through the converter. After each calculation is done, we suggest renaming the particle set and wave-function files to something meaningful, for example H2O.ptcl.xml and H2O.DFT.wfs.xml, where DFT is the functional used. We will use the resulting wave-functions in the following exercise.

C. Optimization and DMC calculations with DFT wave-functions

In this section we will optimize the wave-function generated in the previous step and perform DMC calculations. From the top directory, go to "QMCPACK/SJ/SPOrbitals". The steps required to achieve this are identical to those used to optimize the wave-function with HF orbitals. Make individual folders for each calculation and obtain the necessary files to perform optimization, VMC and DMC calculations from "QMCPACK/Generic_Files". A file named optm_vmc_dmc.xml should exist that contains all three execution blocks. For each functional, make the appropriate modifications to the input files and copy the particle set and wave-function files from the appropriate directory in "GAMESS/DFT". We recommend the following settings: nodes=32, threads=16, (in optimization) blocks=250, samples=128000, timestep=0.5, 8 optimization loops, (in VMC) walkers=16, blocks=100, steps=1, substeps=100, (in DMC) blocks 400, targetWalkers=960, timestep=0.01. Submit the runs and analyze the results using **qmca** .

How do the energies compare against each other? How do they compare against DMC energies with HF orbitals?

D. Generation of a CISD wave-functions using GAMESS

In this section we will use GAMESS to generate a multi-determinant wave-function with CISD. In CISD, the Schrodinger equation is solved exactly in a basis of determinants including the HF determinant and all its single and double excitations.

Due to technical problems with GAMESS in the BGQ architecture of VESTA, we are unable to use CISD properly in GAMESS. For this reason, the output of the calculation is already provided in the directory.You'll see several input and output files named H2O.XXX.inp and H2O.XXX.out, where XXX is one of the following multi-determinant methods: CISD, CASSCF, CASCI, SOCI. There will be time in the next step to study the GAMESS input files and the description in section A. In the next exercise we will use the CISD output, in the next exercise we will use the remaining files. Since the output is already provided, the only thing ended is to use the converter to generate the appropriate QMCPACK files. Copy a submission script from GAMESS/Generic_Files and execute the converter for all the output files in the directory (with the exception of CASSCF, which is used to generate orbitals but it doesn't contain appropriate CI coefficients). In all cases we used PRTMO=.T. in the GUESS section, so you should read these orbitals from the output (-readInitialGuess 40). The highest occupied orbital in any determinant should be 34, so reading 40 orbitals is a safe choice. In this case, it is important to rename the xml files with meaningful names, for example H2O.CISD.wfs.xml. A threshold of 0.0075 is sufficient for the calculations in the training.

E. Optimization of Multi-Determinant wave-function

In this section we will optimize the wave-function generated in the previous step. There is no difference in the optimization steps if a single determinant and a multi-determinant wave-function. QMCPACK will recognize the presence of a multi-determinant wave-function and will automatically optimize the linear coefficients by default. Go to "QMCPACK/MSD/CISD" and make a folder called thr0.01. Copy the particle set and wave-function files created in the previous step to the current directory. With your favorite text editor, open the wave-function file H2O.CISD.wfs.xml. Look for the multideterminant XML block and change the "cutoff" parameter in detlist to 0.01. Then follow the same steps used in the subsection “Optimization and DMC calculations with DFT wave-functions” to optimize the wave-function. Similar to this case, design a QMCPACK input file that performs wave-funtion optimization followed by VMC and DMC calculations. Submit the calculation.

While this run is completed, go to section A and review the GAMESS input file description. When the run is completed, go to the previous directory and make a new folder named thr0.0075. Repeat the steps performed above to optimize the wave-function with a cutoff of 0.01, but use a cutoff of 0.0075 this time. This will increase the number of determinants used in the calculation. Instead of starting from an un-optimized wave-function, start from optimized wave-function from thr0.01. You will need to modify the file and change the cutoff in detlist to 0.0075 with a text editor. Repeat the optimization steps and submit the calculation.

When you are done, use **qmca** to analyze the results. Compare the energies at these two coefficient cutoffs with the energies obtained with DFT orbitals. Do to the time limitations of this tutorial it is not practical to optimize the wave-functions with a smaller cutoff, since this would require more samples and longer runs due to the larger number of optimizable parameters. Figure 5 shows the results of such exercise, the DMC energy as a function of the cutoff in the wave-function. As can be seen, a large improvement in the energy is obtained as the number of configurations is increased.

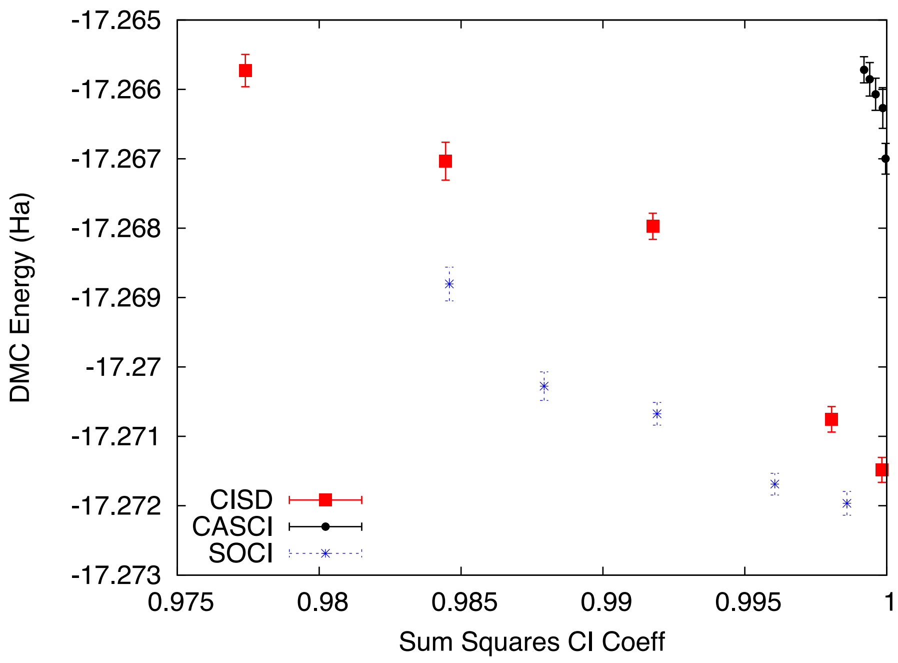


FIG. 5: DMC energy as a function of the sum of the square of CI coefficients from CISD.

III. EXERCISE #3: ORBITAL SETS AND CONFIGURATIONS IN MULTI-DETERMINANT WAVE-FUNCTIONS

In this exercise we will study the dependence of the DMC energy on the set of orbitals and the type of configurations included in a multi-determinant wave-function. Since the un-optimized wave-functions were generated in subsection "Generation of a CISD wave-functions using GAMESS" of exercise #2, we can skip this section and go straight to the wave-function optimization. Go to "QMCPACK/MSD" and make folders for the remaining wave-function types: CASCI and SOCI. The exercise has already been performed with a CISD wave-function in exercise #2.

A CASCI wave-function is produced from a CI calculation that includes all the determinants in a CAS calculation, in this case using the orbitals from a previous CASSCF calculation. In this case we used a CAS(8,8) active space, that includes all determinants generated by distributing 8 electrons in the lowest 8 orbitals. A SOCI calculation is similar to the CAS-CI calculation, but in addition to the determinants in the CAS it also includes all single and double excitations from all of them, leading to a much larger determinant set. Since we now have considerable experience optimizing wave-functions and calculating

DMC energies, we will eave it to the student to complete the remaining tasks on its own. If you need help, refer to previous exercises in the tutorial. Perform optimizations for both wave-functions using cutoffs in the CI expansion of 0.01 an 0.0075. If there is enough time left, try to optimize the wave-functions with a cutoff of 0.005. Analyze the results and plot the energy as a function of cutoff for all three cases, CISD, CAS-CI and SOCI.

Figure 5 shows the result of similar calculations using more samples and smaller cutoffs. The results should be similar to those produced in the tutorial. For reference, the exact energy of the water molecule with ECPs is approximately -17.276. From the results of the tutorial, how does the selection of determinants is related to the expected DMC energy? What about the choice in the set of orbitals?

Appendix A: GAMESS input

In this section we provide a brief description of the GAMESS input needed to produce trial wave-function for QMC calculations with QMCPACK. We assume basic familiarity with GAMESS input structure, in particular regarding the input of atomic coordinates and the definition of gaussian basis sets. This section will focus on the generation of the output files needed by the converter tool, *convert4qmc*. For a description of the converter, see B.

Only a subset of the methods available in GAMESS can be used to generate wave-functions for QMCPACK and we restrict our description here to these. For a complete description of all the options and methods available in GAMESS, please refer to the official documentation which could be found in "http://www.msg.ameslab.gov/gamess/documentation.html".

Currently, *convert4qmc* can process output for the following methods in GAMESS (in SCFTYP) : RHF, ROHF, and MCSCF. Both HF as well as DFT calculations (any DFT type) could be used in combination with RHF and ROHF calculations. For MCSCF and CI calculations, ALDET, ORMAS and GUGA drivers can be used (see below for details).

1. HF input

The following input will perform a restricted HF calculation on a closed-shell singlet (multiplicity=1). This will generate RHF orbitals for any molecular system defined in $DATA ... $END.

```
 $CONTRL SCFTYP=RHF RUNTYP=ENERGY MULT=1
ISPHER=1 EXETYP=RUN COORD=UNIQUE MAXIT=200 $END
$SYSTEM MEMORY=150000000 $END
$GUESS GUESS=HUCKEL $END
$SCF DIRSCF=.TRUE. $END
$DATA
...
Atomic Coordinates and basis set
...
$END
```

Main options:

1. SCFTYP: Type of SCF method, options: RHF, ROHF, MCSCF, UHF and NONE.
2. RUNTYP: Type of run. For QMCPACK wave-function generation this should always be ENERGY.
3. MULT: Multiplicity of the molecule.
4. ISPHER: Use spherical harmonics (1) or cartesian basis functions (-1).
5. COORD: Input structure for the atomic coordinates in $DATA.

2. DFT calculations

The main difference between the input for a RHF/ROHF calculation and a DFT calculation is the definition of the DFTTYP parameter. If this is set in the $CONTROL section, a DFT calculation will be performed with the appropriate functional. Notice that while the default values are usually adequate, DFT calculations have many options involving the integration grids and accuracy settings. Make sure you study the input manual to be aware of these. Refer to the input manual for a list of the implemented exchange-correlation functionals.

3. Multi-Configuration Self-Consistent Field (MCSCF)

MCSCF calculations are performed by setting SCFTYP=MCSCF in the $CONTROL section. If this option is set, a $MCSCF section must be added to the input file with the options for the calculation. An example section for the water molecule used in the tutorial is shown below.

```
$MCSCF CISTEP=GUGA MAXIT=1000 FULLNR=.TRUE.
ACURCY=1.0D-5 $END
```

The most important parameter is CISTEP, which defines the CI package used. The only options compatible with QMCPACK are: ALDET, GUGA, and ORMAS. Depending on the package used, additional input sections are needed.

4. Configuration Interaction (CI)

Configuration interaction (full CI, truncated CI, CAS-CI, etc) calculations are performed by setting SCFTYP=NONE and CITYP=GUGA,ALDET,ORMAS. Each one of this packages requires further input sections, which are typically slightly different to the input sections needed for MCSCF runs.

5. GUGA: Unitary Group CI package

The GUGA package is the only alternative if one wants CSFs with GAMESS. Below we provide a very brief description of input sections needed to perform MCSCF, CASCI, truncated CI and SOCI with this package. For a complete description of these methods and all the options available, please refer to the GAMESS input manual.

a. GUGA-MCSCF

The following input section performs a CASCI calculation, with a CAS that includes 8 electrons in 8 orbitals (4 DOC and 4 VAL), e.g. CAS(8,8). NMCC is the number of frozen orbitals (doubly occupied orbitals in all determinants), NDOC is the number of double occupied orbitals in the reference determinant, NVAL is the number of singly occupied orbitals in the reference (for spin polarized cases), and NVAL is the number of orbitals in the active space. Since FORS is set to .TRUE., all configurations in the active space will be included. ISTSYM defines the symmetry of the desired state.

```
 $MCSCF CISTEP=GUGA MAXIT=1000 FULLNR=.TRUE.
ACURCY=1.0D-5 $END
$DRT GROUP=C2v NMCC=0 NDOC=4 NALP=0 NVAL=4
ISTSYM=1 MXNINT= 500000 FORS=.TRUE. $END
```

b. GUGA-CASCI

The following input section performs a CASCI calculation, with a CAS that includes 8 electrons in 8 orbitals (4 DOC and 4 VAL), e.g. CAS(8,8). NFZC is the number of frozen orbitals (doubly occupied orbitals in all determinants). All other parameters are identical to those in the MCSCF input section.

```
 $CIDRT GROUP=C2v NFZC=0 NDOC=4 NALP=0 NVAL=4 NPRT=2
ISTSYM=1 FORS=.TRUE. MXNINT= 500000 $END
$GUGDIA PRTTOL=0.001 CVGTOL=1.0E-5 ITERMX=1000 $END
```

c. GUGA-Truncated CI

The following input sections will lead to a truncated CI calculation, in this particular case it will perform a CISD calculation since IEXCIT is set to 2. Other values in IEXCIT will lead to different CI truncations, for example IEXCIT=4 will lead to CISDTQ. Notice that only the lowest 30 orbitals will be included in the generation of the excited determinants in this case. For a full CISD calculation, NVAL should be set to the total number of virtual orbitals.

```
 $CIDRT GROUP=C2v NFZC=0 NDOC=4 NALP=0 NVAL=30 NPRT=2
ISTSYM=1 IEXCIT=2 MXNINT= 500000 $END
$GUGDIA PRTTOL=0.001 CVGTOL=1.0E-5 ITERMX=1000 $END
```

d. GUGA-SOCI

The following input section performs a SOCI calculation, with a CAS that includes 8 electrons in 8 orbitals (4 DOC and 4 VAL), e.g. CAS(8,8). Since SOCI is set to .TRUE., all single and double determinants from all determinants in the CAS(8,8) will be included.

```
 $CIDRT GROUP=C2v NFZC=0 NDOC=4 NALP=0 NVAL=4 NPRT=2
ISTSYM=1 SOCI=.TRUE. NEXT=30 MXNINT= 500000 $END
$GUGDIA PRTTOL=0.001 CVGTOL=1.0E-5 ITERMX=1000 $END
```

6. ECP

To use Effective Core Potentials (ECP) in GAMESS, you must define a {$ECP ... $END } block. There must be a definition of a potential for every atom in the system, including symmetry equivalent ones. In addition, they must appear in the particular order expected by GAMESS. Below is an example of an ECP input block for a single water molecule using BFD ECPs. To turn on the use of ECPs, the option "ECP=READ" must be added to the CONTROL input block.

```
 $ECP
O-QMC GEN 2 1
3
6.00000000 1 9.29793903
55.78763416 3 8.86492204
-38.81978498 2 8.62925665
1
38.41914135 2 8.71924452
H-QMC GEN 0 0
3
1.000000000000 1 25.000000000000
25.000000000000 3 10.821821902641
-8.228005709676 2 9.368618758833
H-QMC
$END
```

Appendix B: convert4qmc

To generate the particleset and wavefunction XML blocks required by QMCPACK in calculations with molecular systems, the converter *convert4qmc* must be used. The converter will read the standard output from the appropriate Quantum Chemistry calculation and will generate all the necessary input for QMCPACK. Below we describe the main options of the converter for GAMESS output. In general, there are 3 ways to use the converter depending on the type of calculation performed. The minimum syntax for each option is found below. For a description of the xml files produced by the converter, see section E.

1. For all single determinant calculations (HF and DFT with any DFTTYP):

   convert4qmc -gamessAscii single_det.out

   - single_det.out is the standard output generated by GAMESS.

2. *(This option is not recommended. Use option below to avoid mistakes.)*For multi-determinant calculations where the orbitals and configurations are read from different files (for example when using orbitals from a MCSCF run and configurations from a subsequent CI run):

   convert4qmc -gamessAscii orbitals_multidet.out -ci cicoeff_multidet.out

   - orbitals_multidet.out is the standard output from the calculation that generates the orbitals. cicoeff_multidet.out is the standard output from the calculation that calculates the CI expansion.

3. For multi-determinant calculations where the orbitals and configurations are read from the same file, using PRTMO=.T. in the GUESS input block:

   convert4qmc -gamessAscii multi_det.out -ci multi_det.out -readInitialGuess Norb

   - multi_det.out is the standard output from the calculation that calculates the CI expansion.

Options:

- **-gamessAscii file.out**: Standard output of GAMESS calculation. With the exception of determinant configurations and coefficients in multi-determinant calculations, everything else is read from this file including: atom coordinates, basis sets, single particle orbitals, ECPs, number of electrons, multiplicity, etc.
- **-ci file.out**: In multi-determinant calculations, determinant configurations and coefficients are read from this file. Notice that single particle orbitals are **NOT** read from this file. Recognized CI packages are: ALDET, GUGA and ORMAS. Output produced with the GUGA package **MUST** have the option "NPRT=2" in the CIDRT or DRT input blocks.
- **-threshold cutoff**: Cutoff in multi-determinant expansion. Only configurations with coefficients above this value are printed.
- **-zeroCI**: Sets to zero the CI coefficients of all determinants, with the exception of the first one.
- **-readInitialGuess Norb**: Reads **Norb** initial orbitals ("INITIAL GUESS ORBITALS") from GAMESS output. These are orbitals generated by the GUESS input block and printed with the option "PRTMO=.T.". Notice that this is useful only in combination with the option "GUESS=MOREAD" and in cases where the orbitals are not modified in the GAMESS calculation, e.g. CI runs. This is the recommended option in all CI calculations.
- **-NaturalOrbitals Norb**: Read **Norb** "NATURAL ORBITALS" from GAMESS output. The natural orbitals must exists in the output, otherwise the code aborts.
- **-add3BodyJ**: Adds three-body Jastrow terms (e-e-I) between electron pairs (both same spin and opposite spin terms) and all ion species in the system. The radial function is initialized to zero and the default cutoff is 10.0 bohr. The converter will add a one- and two-body Jastrow to the wavefunction block by default.

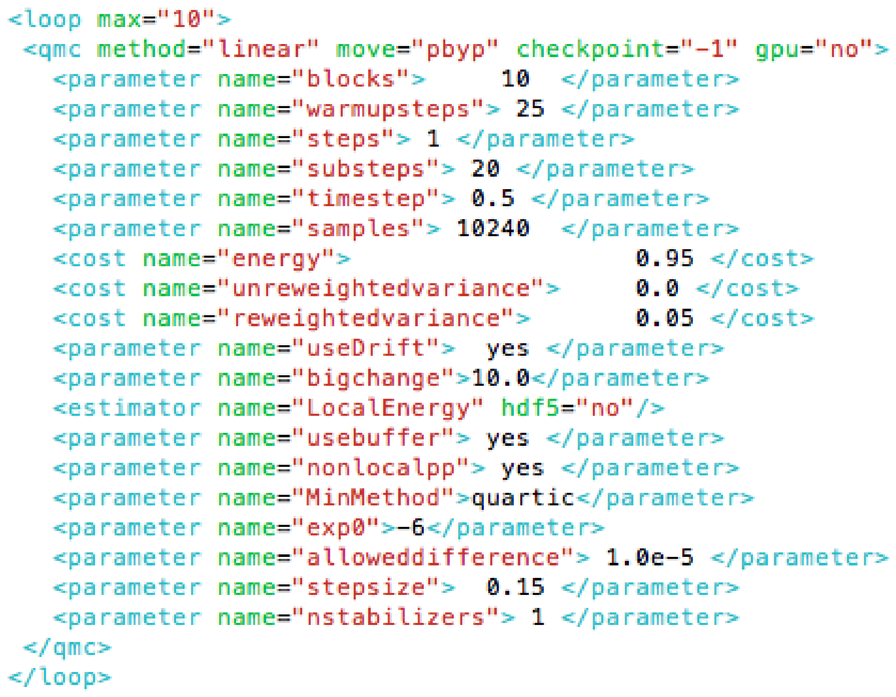
Useful notes:

- The type of single particle orbitals read by the converter depends on the type of calculation and on the options used. By default, when neither **-readInitialGuess** or

-NaturalOrbitals are used, the following orbitals are read in each case (notice that **-readInitialGuess** or **-NaturalOrbitals** are mutually exclusive):
-RHF and ROHF: "EIGENVECTORS"
-MCSCF: "MCSCF OPTIMIZED ORBITALS"
-GUGA, ALDET, ORMAS: Can not read orbitals without **-readInitialGuess** or **-NaturalOrbitals** options.

- The single particle orbitals and printed CI coefficients in MCSCF calculations are not consistent in GAMESS. The printed CI coefficients correspond to the next-to-last iteration, they are not recalculated with the final orbitals. So in order to get appropriate CI coefficients from MCSCF calculations, a subsequent CI (no SCF) calculation is needed to produce consistent orbitals. In principle, it is possible to read the orbitals from the MCSCF output and the CI coefficients and configurations from the output of the following CI calculations. This could lead to problems in principle, since GAMESS will rotate initial orbitals by default in order to obtain an initial guess consistent with the symmetry of the molecule. This last step is done by default and can change the orbitals reported in the MCSCF calculation before the CI is performed. In order to avoid this problem, it is highly recommended to use option #3 above to read all the information from the output of the CI calculation, this requires the use of "PRTMO=.T." in the GUESS input block. Since the orbitals are printed after any symmetry rotation, the resulting output will always be consistent.

Appendix C: Wave-function Optimization XML block



```
<loop max="10">
 <qmc method="linear" move="pbyp" checkpoint="-1" gpu="no">
   <parameter name="blocks">     10  </parameter>
   <parameter name="warmupsteps"> 25 </parameter>
   <parameter name="steps"> 1 </parameter>
   <parameter name="substeps"> 20 </parameter>
   <parameter name="timestep"> 0.5 </parameter>
   <parameter name="samples"> 10240  </parameter>
   <cost name="energy">                   0.95 </cost>
   <cost name="unreweightedvariance">     0.0 </cost>
   <cost name="reweightedvariance">       0.05 </cost>
   <parameter name="useDrift">  yes </parameter>
   <parameter name="bigchange">10.0</parameter>
   <estimator name="LocalEnergy" hdf5="no"/>
   <parameter name="usebuffer"> yes </parameter>
   <parameter name="nonlocalpp"> yes </parameter>
   <parameter name="MinMethod">quartic</parameter>
   <parameter name="exp0">-6</parameter>
   <parameter name="alloweddifference"> 1.0e-5 </parameter>
   <parameter name="stepsize">  0.15 </parameter>
   <parameter name="nstabilizers"> 1 </parameter>
 </qmc>
</loop>
```

FIG. 6: Sample XML optimization block.

Options:

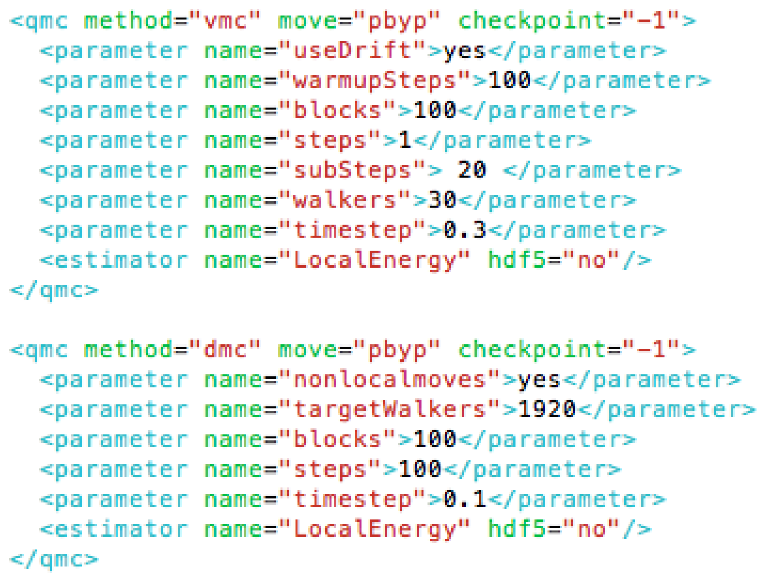
- bigchange: (default 50.0) largest parameter change allowed
- usebuffer: (default no) Save useful information during VMC
- nonlocalpp: (default no) Include non-local energy on 1-D min
- MinMethod: (default quartic) Method to calculate magnitude of parameter change quartic: fit quartic polynomial to 4 values of the cost function obtained using reweighting along chosen direction linemin: direct line minimization using reweighting rescale: no 1-D minimization. Uses Umrigars suggestions.
- stepsize: (default 0.25) step size in either quartic or linemin methods.
- alloweddifference: (default 1e-4) Allowed increased in energy

- exp0: (default -16.0) Initial value for stabilizer (shift to diagonal of H) Actual value of stabilizer is 10^{exp0}
- nstabilizers: (default 3) Number of stabilizers to try
- stabilizaterScale: (default 2.0) Increase in value of exp0 between iterations.
- max_its: (default 1) number of inner loops with same sample
- minwalkers: (default 0.3) minimum value allowed for the ratio of effective samples to actual number of walkers in a reweighting step. The optimization will stop if the effective number of walkers in any reweighting calculation drops below this value. Last set of acceptable parameters are kept.
- maxWeight: (defaul 1e6) Maximum weight allowed in reweighting. Any weight above this value will be reset to this value.

Recommendations:

- Set samples to equal to (#threads)*blocks.
- Set steps to 1. Use substeps to control correlation between samples.
- For cases where equilibration is slow, increase both substeps and warmupsteps.
- For hard cases (e.g. simultaneous optimization of long MSD and 3-Body J), set exp0 to 0 and do a single inner iteration (max_its=1) per sample of configurations.

Appendix D: VMC and DMC XML block



```
<qmc method="vmc" move="pbyp" checkpoint="-1">
  <parameter name="useDrift">yes</parameter>
  <parameter name="warmupSteps">100</parameter>
  <parameter name="blocks">100</parameter>
  <parameter name="steps">1</parameter>
  <parameter name="subSteps"> 20 </parameter>
  <parameter name="walkers">30</parameter>
  <parameter name="timestep">0.3</parameter>
  <estimator name="LocalEnergy" hdf5="no"/>
</qmc>

<qmc method="dmc" move="pbyp" checkpoint="-1">
  <parameter name="nonlocalmoves">yes</parameter>
  <parameter name="targetWalkers">1920</parameter>
  <parameter name="blocks">100</parameter>
  <parameter name="steps">100</parameter>
  <parameter name="timestep">0.1</parameter>
  <estimator name="LocalEnergy" hdf5="no"/>
</qmc>
```

FIG. 7: Sample XML blocks for VMC and DMC calculations.

General Options:

- move: (default ”walker”) Type of electron move. Options: ”pbyp” and ”walker”.
- : checkpoint: (default ”-1”) (If ¿ 0) Generate checkpoint files with given frequency. The calculations can be restarted/continued with the produced checkpoint files.
- useDrift: (default ”yes”) Defines the sampling mode. useDrift = ”yes” will use Langevin acceleration to sample the VMC and DMC distributions, while useDrift=”no” will use random displacements in a box.
- warmupSteps: (default 0) Number of steps warmup steps at the beginning of the calculation. No output is produced for these steps.
- blocks: (default 1) Number of blocks (outer loop).
- steps: (default 1) Number of steps per blocks (middle loop).
- sub steps: (default 1) Number of substeps per step (inner loop). During sub steps, the local energy is not evaluated in VMC calculations, which leads to faster execution.

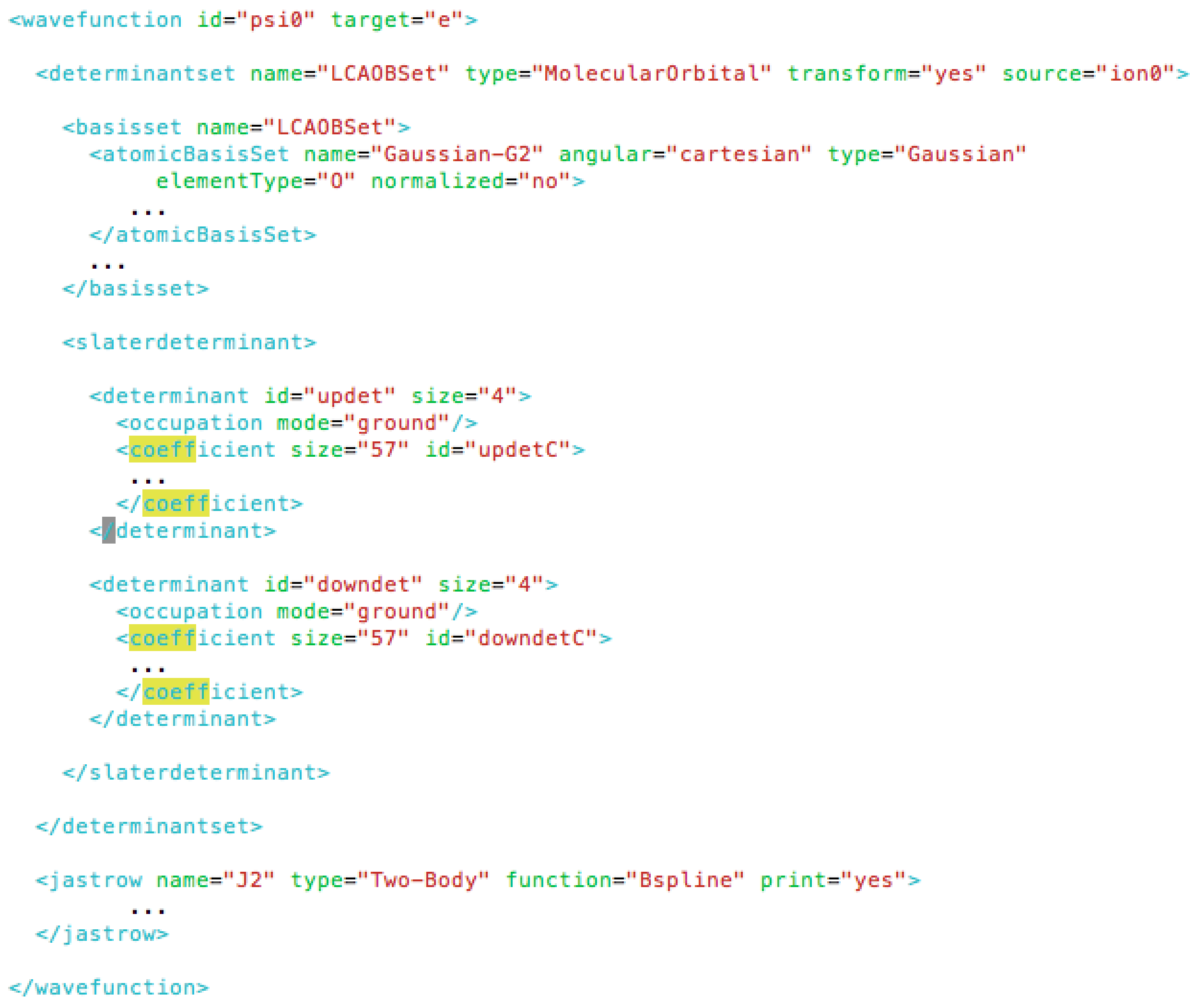
In VMC calculations, set sub steps to the average autocorrelation time of the desired quantity.

- time step: (default 0.1) Electronic time step in bohr.
- samples: (default 0) Number of walker configurations saved during the current calculation.
- walkers: (default #threads) In VMC, sets the number of walkers per node. The total number of walkers in the calculation will be equal to walkers*(# nodes).

Options unique to DMC:

- targetWalkers: (default #walkers from previous calculation, e.g. VMC.) Sets the target number of walkers. The actual population of walkers will fluctuate around this value. The walkers will be distributed across all the nodes in the calculation. On a given node, the walkers are split across all the threads in the system.
- nonlocalmoves: (default "no") Set to "yes" to turns on the use of Casula's T-moves.

Appendix E: Wave-function XML block

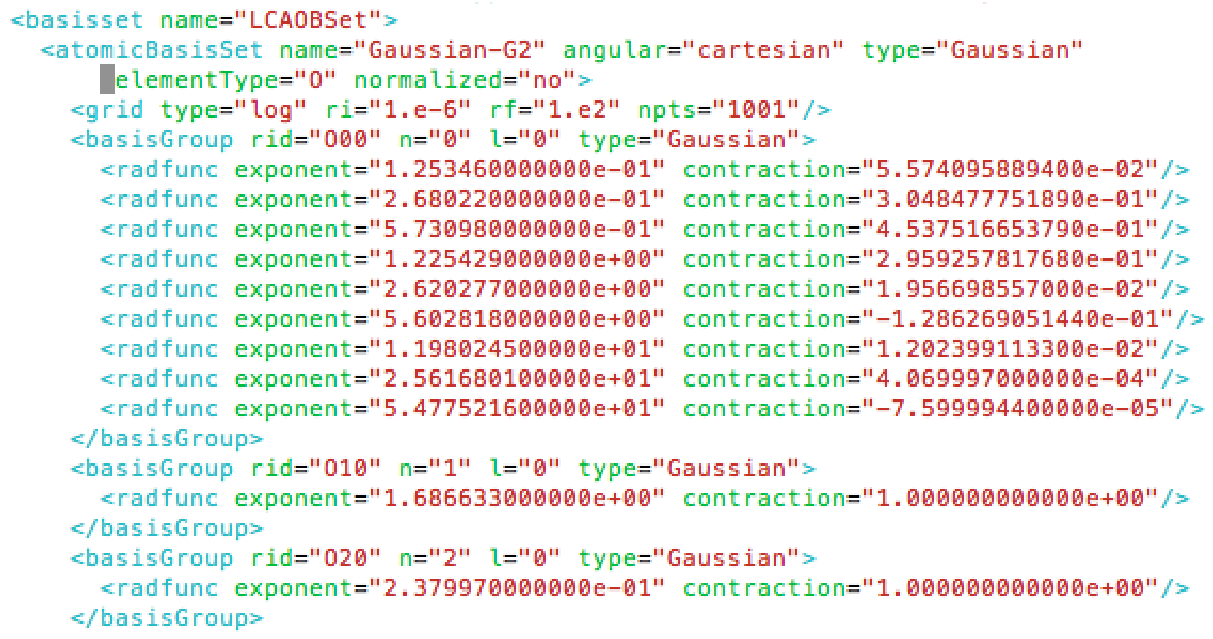


```
<wavefunction id="psi0" target="e">

  <determinantset name="LCAOBSet" type="MolecularOrbital" transform="yes" source="ion0">

    <basisset name="LCAOBSet">
      <atomicBasisSet name="Gaussian-G2" angular="cartesian" type="Gaussian"
           elementType="O" normalized="no">
         ...
      </atomicBasisSet>
      ...
    </basisset>

    <slaterdeterminant>

      <determinant id="updet" size="4">
        <occupation mode="ground"/>
        <coefficient size="57" id="updetC">
         ...
        </coefficient>
      </determinant>

      <determinant id="downdet" size="4">
        <occupation mode="ground"/>
        <coefficient size="57" id="downdetC">
         ...
        </coefficient>
      </determinant>

    </slaterdeterminant>

  </determinantset>

  <jastrow name="J2" type="Two-Body" function="Bspline" print="yes">
         ...
  </jastrow>

</wavefunction>
```

FIG. 8: Basic framework for a single determinant determinantset XML block.

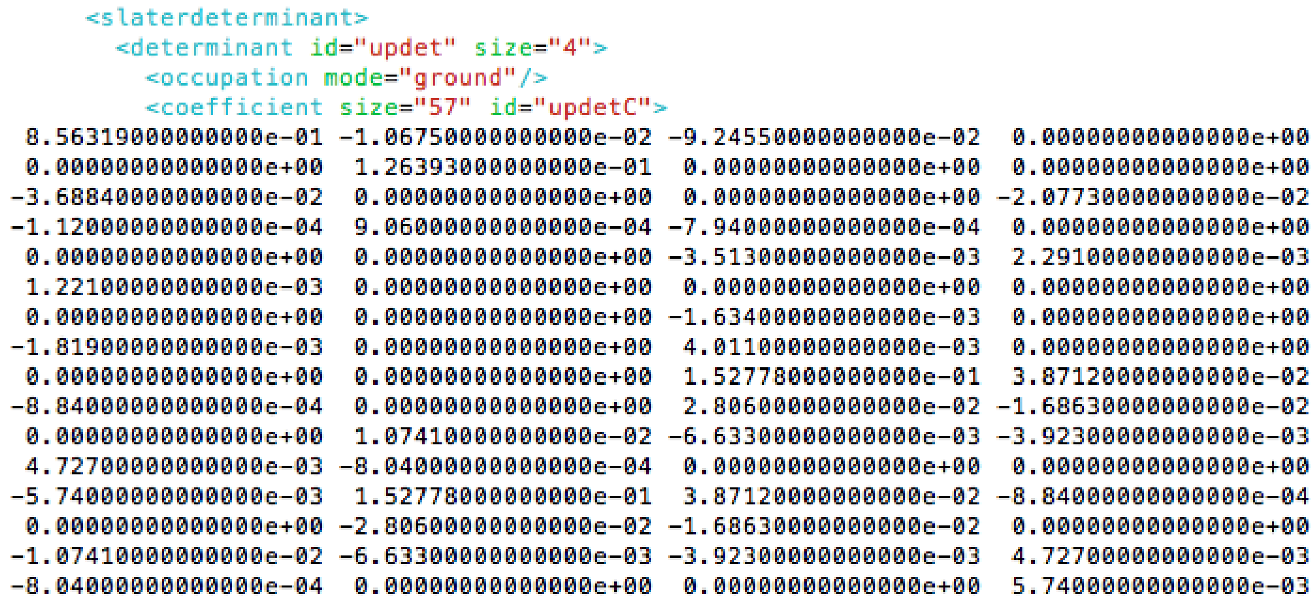
In this section we describe the basic format of a QMCPACK wavefunction XML block. Everything listed in this section is generated by the appropriate converter tools. Little to no modification is needed when performing standard QMC calculations. As a result, this section is meant mainly for illustration purposes. Only experts should attempt to modify these files (with very few exceptions like the cutoff of CI coefficients and the cutoff in Jastrow functions) since changes can lead to unexpected results.

A QMCPACK wavefunction XML block is a combination of a determinantset , which contains the anti-symmetric part of the wave-function, and one or more jastrow blocks. The syntax of the anti-symmetric block depends on whether the wave-function is a single



```
<basisset name="LCAOBSet">
  <atomicBasisSet name="Gaussian-G2" angular="cartesian" type="Gaussian"
      elementType="O" normalized="no">
    <grid type="log" ri="1.e-6" rf="1.e2" npts="1001"/>
    <basisGroup rid="O00" n="0" l="0" type="Gaussian">
      <radfunc exponent="1.253460000000e-01" contraction="5.574095889400e-02"/>
      <radfunc exponent="2.680220000000e-01" contraction="3.048477751890e-01"/>
      <radfunc exponent="5.730980000000e-01" contraction="4.537516653790e-01"/>
      <radfunc exponent="1.225429000000e+00" contraction="2.959257817680e-01"/>
      <radfunc exponent="2.620277000000e+00" contraction="1.956698557000e-02"/>
      <radfunc exponent="5.602818000000e+00" contraction="-1.286269051440e-01"/>
      <radfunc exponent="1.198024500000e+01" contraction="1.202399113300e-02"/>
      <radfunc exponent="2.561680100000e+01" contraction="4.069997000000e-04"/>
      <radfunc exponent="5.477521600000e+01" contraction="-7.599994400000e-05"/>
    </basisGroup>
    <basisGroup rid="O10" n="1" l="0" type="Gaussian">
      <radfunc exponent="1.686633000000e+00" contraction="1.000000000000e+00"/>
    </basisGroup>
    <basisGroup rid="O20" n="2" l="0" type="Gaussian">
      <radfunc exponent="2.379970000000e-01" contraction="1.000000000000e+00"/>
    </basisGroup>
```

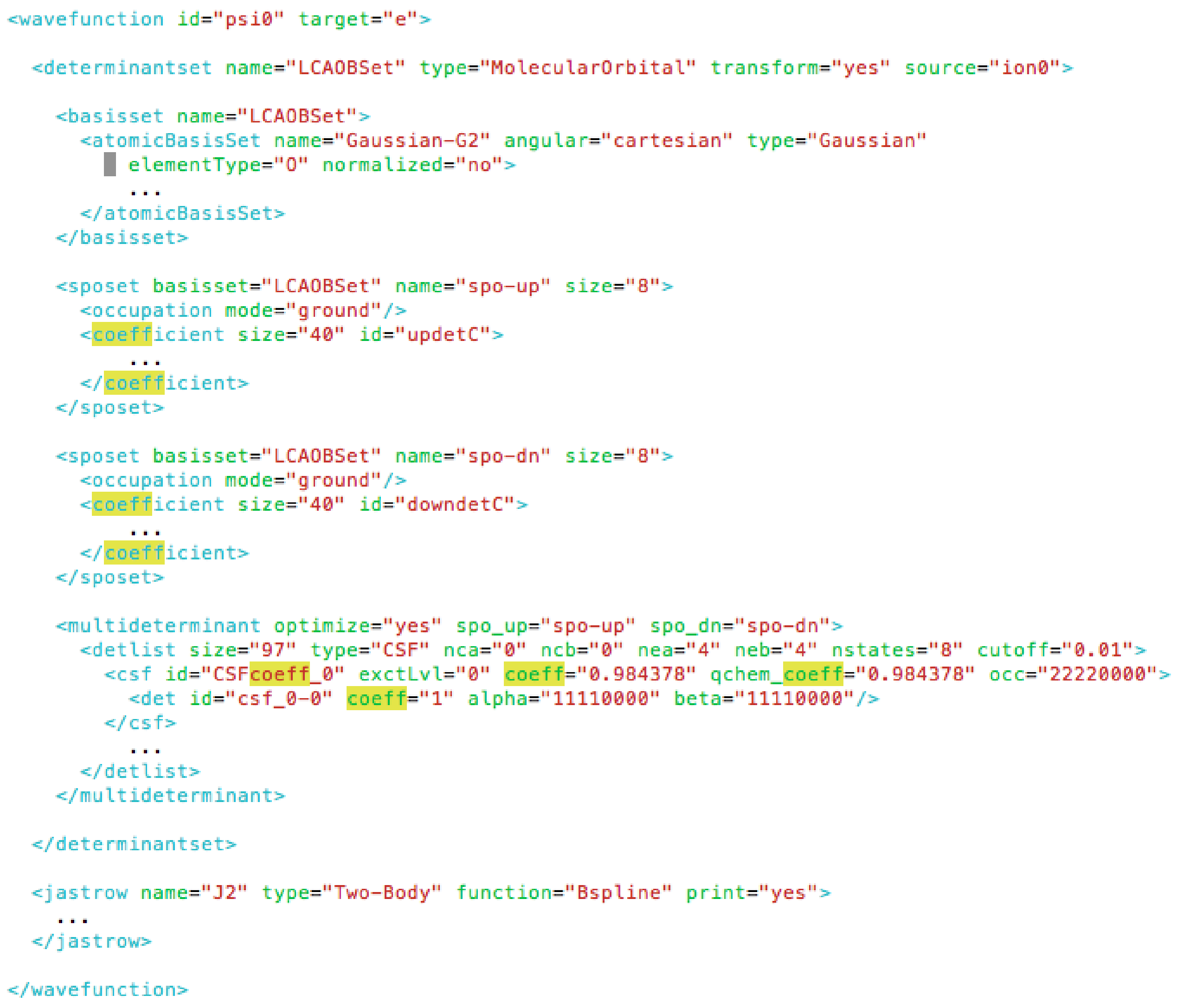
FIG. 9: Sample XML block for an atomic orbital basis set.



```
     <slaterdeterminant>
       <determinant id="updet" size="4">
         <occupation mode="ground"/>
         <coefficient size="57" id="updetC">
  8.56319000000000e-01 -1.06750000000000e-02 -9.24550000000000e-02  0.00000000000000e+00
  0.00000000000000e+00  1.26393000000000e-01  0.00000000000000e+00  0.00000000000000e+00
 -3.68840000000000e-02  0.00000000000000e+00  0.00000000000000e+00 -2.07730000000000e-02
 -1.12000000000000e-04  9.06000000000000e-04 -7.94000000000000e-04  0.00000000000000e+00
  0.00000000000000e+00  0.00000000000000e+00 -3.51300000000000e-03  2.29100000000000e-03
  1.22100000000000e-03  0.00000000000000e+00  0.00000000000000e+00  0.00000000000000e+00
  0.00000000000000e+00  0.00000000000000e+00 -1.63400000000000e-03  0.00000000000000e+00
 -1.81900000000000e-03  0.00000000000000e+00  4.01100000000000e-03  0.00000000000000e+00
  0.00000000000000e+00  0.00000000000000e+00  1.52778000000000e-01  3.87120000000000e-02
 -8.84000000000000e-04  0.00000000000000e+00  2.80600000000000e-02 -1.68630000000000e-02
  0.00000000000000e+00  1.07410000000000e-02 -6.63300000000000e-03 -3.92300000000000e-03
  4.72700000000000e-03 -8.04000000000000e-04  0.00000000000000e+00  0.00000000000000e+00
 -5.74000000000000e-03  1.52778000000000e-01  3.87120000000000e-02 -8.84000000000000e-04
  0.00000000000000e+00 -2.80600000000000e-02 -1.68630000000000e-02  0.00000000000000e+00
 -1.07410000000000e-02 -6.63300000000000e-03 -3.92300000000000e-03  4.72700000000000e-03
 -8.04000000000000e-04  0.00000000000000e+00  0.00000000000000e+00  5.74000000000000e-03
```

FIG. 10: Sample XML block for the single slater determinant case.

determinant or a multi-determinant expansion. Figure 8 shows the general structure of the single determinant case. The determinantset block is composed of a basisset block, which defines the atomic orbital basis set, and a slaterdeterminant block, which defines the single particle orbitals and occupation numbers of the Slater determinant. Figure 9 shows a section of a basisset block for an oxygen atom. The structure of this block is rigid and should not



```
<wavefunction id="psi0" target="e">

  <determinantset name="LCAOBSet" type="MolecularOrbital" transform="yes" source="ion0">

    <basisset name="LCAOBSet">
      <atomicBasisSet name="Gaussian-G2" angular="cartesian" type="Gaussian"
        elementType="O" normalized="no">
        ...
      </atomicBasisSet>
    </basisset>

    <sposet basisset="LCAOBSet" name="spo-up" size="8">
      <occupation mode="ground"/>
      <coefficient size="40" id="updetC">
        ...
      </coefficient>
    </sposet>

    <sposet basisset="LCAOBSet" name="spo-dn" size="8">
      <occupation mode="ground"/>
      <coefficient size="40" id="downdetC">
        ...
      </coefficient>
    </sposet>

    <multideterminant optimize="yes" spo_up="spo-up" spo_dn="spo-dn">
      <detlist size="97" type="CSF" nca="0" ncb="0" nea="4" neb="4" nstates="8" cutoff="0.01">
        <csf id="CSFcoeff_0" exctLvl="0" coeff="0.984378" qchem_coeff="0.984378" occ="22220000">
          <det id="csf_0-0" coeff="1" alpha="11110000" beta="11110000"/>
        </csf>
        ...
      </detlist>
    </multideterminant>

  </determinantset>

  <jastrow name="J2" type="Two-Body" function="Bspline" print="yes">
    ...
  </jastrow>

</wavefunction>
```

FIG. 11: Basic framework for a multi-determinant *determinantset* XML block.

be modified. Figure 10 shows a (piece of a) sample of a slaterdeterminant block. The slaterdeterminant block consists of 2 determinant blocks, one for each electron spin. The parameter "size" in the determinant block refers to the number of single particle orbitals present while the "size" parameter in the coefficient block refers to the number of atomic basis functions per single particle orbital.

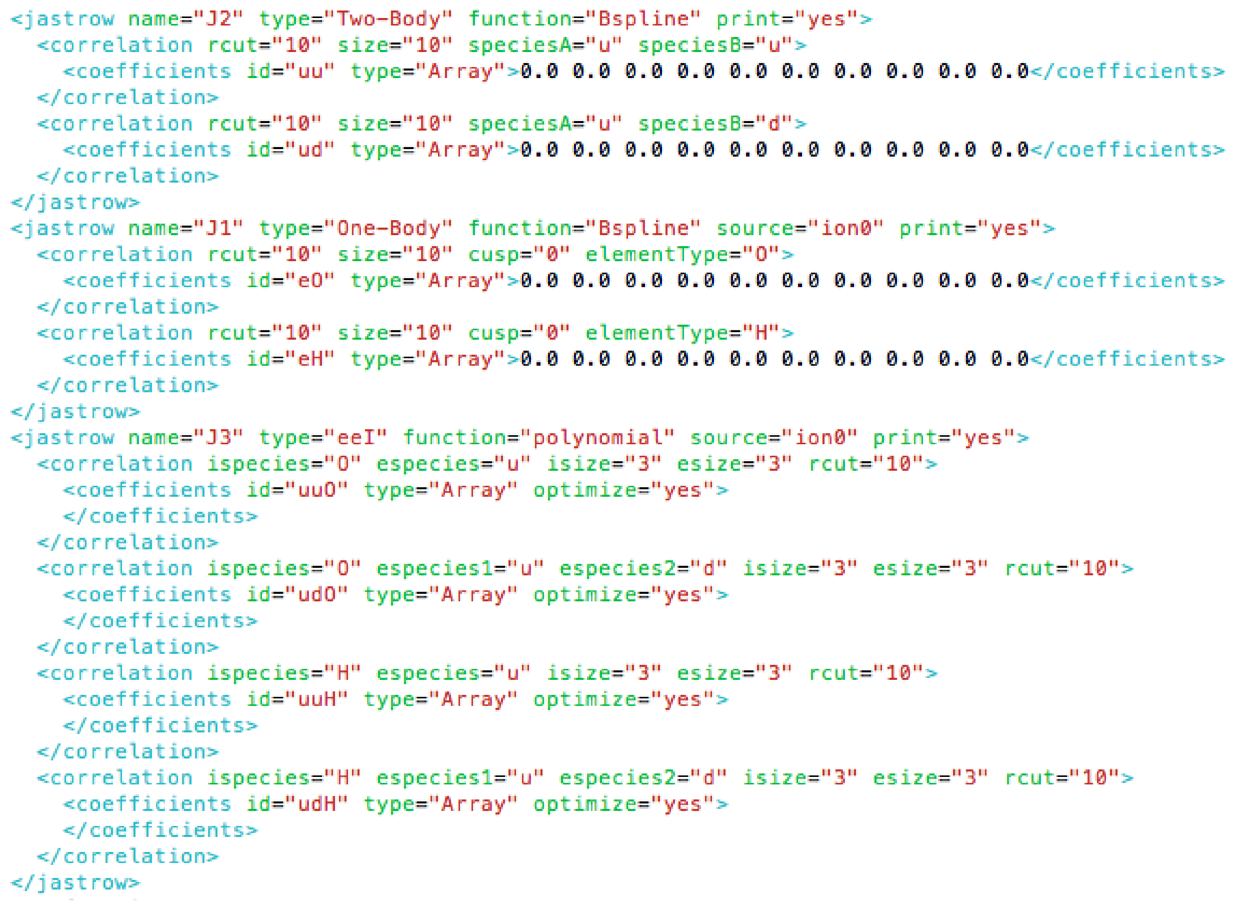
Figure 11 shows the general structure of the multi-determinant case. Similar to the single determinant case, the determinantset must contain a basisset block. This definition is identical to the one described above. In this case, the definition of the single particle orbitals must be done independently from the definition of the determinant configurations, the latter is done in the sposet block while the former is done on the multideterminant block. Notice that 2 sposet sets must be defined, one for each electron spin. The name of reach sposet set is required in the definition of the multideterminant block. The determinants are defined in terms of occupation numbers based on these orbitals.

There are various options in the multideterminant block that users should be aware of.

- cutoff: (*IMPORTANT!*) Only configurations with (absolute value) “qchem_coeff” larger than this value will be read by QMCPACK.
- optimize: Turn on/off the optimization of linear CI coefficients.
- coeff: (in csf) Current coefficient of given configuration. Gets updated during wavefunction optimization.
- qchem_coeff: (in csf) Original coefficient of given configuration from GAMESS calculation. This is used when applying a cutoff to the configurations read from the file. The cutoff is applied on this parameter and not on the optimized coefficient.
- nca and nab: number of core orbitals for up/down electrons. A core orbital is an orbital that is doubly occupied in all determinant configurations, not to be confused with core electrons. These are not explicitly listed on the definition of configurations.
- nea and neb: number of up/down active electrons (those being explicitly correlated).
- nstates: number of correlated orbitals
- size (in detlist): contains the number of configurations in the list.

The remaining part of the determinantset block is the definition of jastrow factor. Any number of these can be defined. Figure 12 shows a sample jastrow block including one-, two- and three-body terms. This is the standard block produced by *convert4qmc* with the option **-add3BodyJ** (this particular example is for a water molecule). Optimization of individual radial functions can be turned on/off using the “optimize” parameter. It can be added to any coefficients block, even though it is currently not present in the J1 and J2 blocks.

This training assumes basic familiarity with the UNIX operating system. In particular, we use simple scripts written in “csh”. In addition, we assume that the student has obtained



```
<jastrow name="J2" type="Two-Body" function="Bspline" print="yes">
  <correlation rcut="10" size="10" speciesA="u" speciesB="u">
    <coefficients id="uu" type="Array">0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</coefficients>
  </correlation>
  <correlation rcut="10" size="10" speciesA="u" speciesB="d">
    <coefficients id="ud" type="Array">0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</coefficients>
  </correlation>
</jastrow>
<jastrow name="J1" type="One-Body" function="Bspline" source="ion0" print="yes">
  <correlation rcut="10" size="10" cusp="0" elementType="O">
    <coefficients id="eO" type="Array">0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</coefficients>
  </correlation>
  <correlation rcut="10" size="10" cusp="0" elementType="H">
    <coefficients id="eH" type="Array">0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</coefficients>
  </correlation>
</jastrow>
<jastrow name="J3" type="eeI" function="polynomial" source="ion0" print="yes">
  <correlation ispecies="O" especies="u" isize="3" esize="3" rcut="10">
    <coefficients id="uuO" type="Array" optimize="yes">
    </coefficients>
  </correlation>
  <correlation ispecies="O" especies1="u" especies2="d" isize="3" esize="3" rcut="10">
    <coefficients id="udO" type="Array" optimize="yes">
    </coefficients>
  </correlation>
  <correlation ispecies="H" especies="u" isize="3" esize="3" rcut="10">
    <coefficients id="uuH" type="Array" optimize="yes">
    </coefficients>
  </correlation>
  <correlation ispecies="H" especies1="u" especies2="d" isize="3" esize="3" rcut="10">
    <coefficients id="udH" type="Array" optimize="yes">
    </coefficients>
  </correlation>
</jastrow>
```

FIG. 12: Sample Jastrow XML block.

all the necessary files and executables, and that the location of the training files are located at ${TRAINING_TOP}.

The goal of the training not only to familiarize the student with the execution and options in QMCPACK, but also to introduce him/her to important concepts in quantum Monte Carlo calculations and many-body electronic structure calculations.