Tutorial: Quantum Monte Carlo with QMC=Chem

In this tutorial, We will study the dissociation energy of N_2 using a Hartree-Fock (HF) trial wave function and a complete active space (CAS-SCF) trial wave function. The HF wave function for dissociated N_2 is computed within restricted open-shell HF with a spin multiplicity of 7. These wave functions were prepared using the GAMESS [1] program. The dissociation energy is evaluated by calculating the energy difference of N_2 at $R=4\,\text{Å}$ and at $R=1.1\,\text{Å}$, where R is the inter-atomic distance.

In your directory, you should have:

```
$ ls
1.1.CAS/     4.CAS/        job_1.1.CAS.160core  job_4.HF.160core
1.1.CAS.out  4.CAS.out        job_1.1.HF.160core
1.1.HF/      4.HF/        job_1.1.HF.1core
1.1.HF.out   4.HF.out        job_4.CAS.160core
```

- The **.out* files are the GAMESS output files
- The *job_** files are the files needed to submit a job on Curie
- The directories are the corresponding QMC=Chem EZFIO database files [2]

Running a VMC calculation

Single core run

To access the input data, run

```
$ qmcchem_edit.py 1.1.HF
```

This command will open a temporary file containing the different parameters of the simulation. Modify them as follows:

```
# Simulation
# --------------------

end_condition  = "wall_time > 300"
jastrow        = False
method         = "VMC"
nucl_fitcusp   = True
num_step       = 10000
sampling       = "Langevin"
time_step      = 0.2
title          = "HF, 1.1 angstroms"
walk_num       = 20
```

end_condition

Stopping condition of the run. 5 minutes is fine.

jastrow

If true, use a Jastrow factor to improve the trial wave function. In this tutorial, we will not use it.

method

VMC: Variational Monte Carlo.

nucl_fitcusp

Impose the correct electron-nucleus cusp at the nucleus to avoid the divergence of the energy at the nuclei. This doesn't change the energy but considerably reduces its variance.

num_step

Number of steps per block. This is usually adjusted such that the time spent to compute one block is not too small or not too large. Typically, for very short runs 20 seconds is OK, and for usual production runs, this parameters is adjusted to 10 minutes per block.

sampling

The Monte Carlo sampling algorithm. Langevin is the best for VMC.

time_step

Simulation time step. Using the Langevin algorithm, 0.2 is usually a good choice.

title

Title of the run. You can put whatever you want.

walk_num

Number of walkers (independent trajectories in VMC).

When you save the fiel and exit the text editor, the EZFIO database has been updated. You can now run the QMC calculation. First, run a single-core run:

```
$ ccc_msub -A <your_curie_account> 1.1.HF.1core.sub
```

In QMC=Chem, there is no output file. At any time, you can see what has been computed by running:

```
$ qmcchem_result.py -s 1.1.HF
```

The output of this command should look like this:

```
#                                 Summary
#----------------------------------------------------------------------
Number of blocks                   : 26
Number of blocks per core          : 26
Total CPU time                     : 0:04:51
CPU time / block                   : 11.192(79)
Acceptance rate                    : 0.92348(10)
#----------------------------------------------------------------------
e_loc                              : -108.9842(52)
Variance of e_loc                  : 25.75(30)
Min of e_loc                       : -351.467898466
Max of e_loc                       : 503.693209743
#----------------------------------------------------------------------
```

Number of blocks

Total number of blocks in the database.

Number of blocks per core

Each CPU core has individually made this number of blocks.

Total CPU time

Sum of the CPU times of all the cores.

CPU time / block

Average CPU time per block.

Acceptance rate

Average Metropolis acceptance rate.

e_loc

Average of the local energy.

Variance of e_loc

Variance (σ^2) of the local energy.

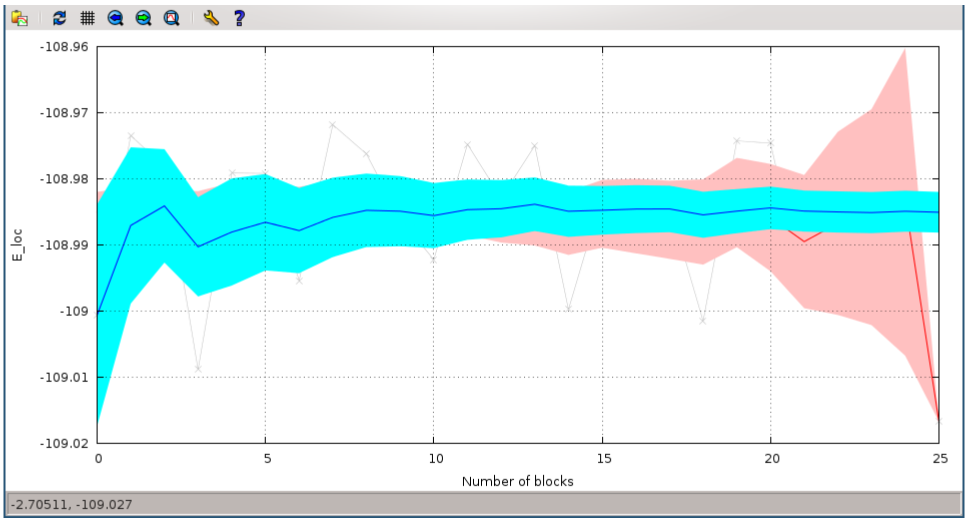
Min/Max of e_loc

Min or Max value of the local energy encountered in the simulation.

At the end of the run, check that the average of the local energy corresponds to the Hartree-Fock energy given by GAMESS (within the error bars).

You can plot the convergence of the local energy using:

```
$ qmcchem_result.py -p E_loc 1.1.HF
```



The blue curve is the convergence plot of the local energy by cumulating blocks from the first block to the last block. The red curve is the same convergence plot but using the blocks from the last one to the first one. If the calculation is converged, the blocks are independent between each other and the shape of the curve should not depend on the order in which the blocks are taken. If the blue and the red convergence plots are not compatible, the QMC run is not converged.

Multi-core run

Your first calculation has finished. If you want, you can add more blocks to the EZFIO database. To do this, run a calculation in parallel using

```
$ ccc_msub -A <your_curie_account> 1.1.HF.160core.sub
```

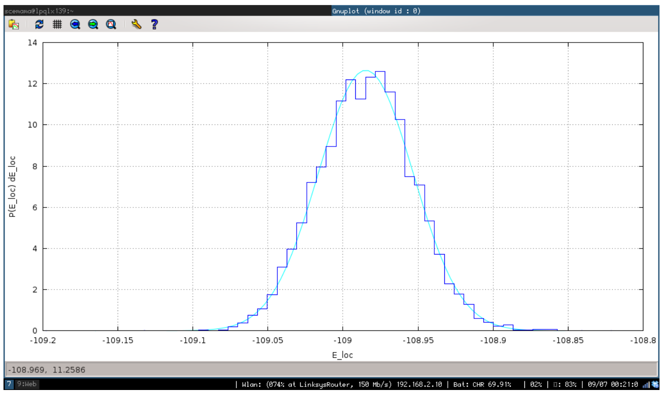
Check that the error bar is significantly reduced, and that the total CPU time is 160x larger:

```
$ qmcchem_result.py -s -c 1.1.HF
#                                 Summary
#-------------------------------------------------------------------------
Number of blocks                   : 3168
Number of blocks per core          : 27
Total CPU time                     : 9:49:16
CPU time / block                   : 11.292(19)
Acceptance rate                    : 0.923571(13)
#-------------------------------------------------------------------------
```

```
e_loc                                : -108.98489(56)
Variance of e_loc                    : 26.082(58)
Min of e_loc                         : -397.069319894
Max of e_loc                         : 4547.98196953
#----------------------------------------------------------------------
```

Now, you have enough blocks to verify that the blocks have a Gaussian distribution:

```
$ qmcchem_result.py -H E_loc 1.1.HF
```



Run VMC calculations for the 3 other trial wave functions and check that the energy corresponds to the energy given by GAMESS.

Running DMC calculations

For each EZFIO directory, modify the simulation parameters as follows:

```
$ qmcchem_edit.py 1.1.HF

method          = "DMC"
sampling        = "Brownian"
time_step       = 0.0001
title           = "1.1 HF DMC"
walk_num        = 40
```

method

Choose DMC to perform a Diffusion Monte Carlo calculation.

sampling

Choose the Brownian motion for DMC. Langevin is not adapted.

time_step

With the Brownian motion, this time step is sufficiently small to obtain a small time step error, and the Metropolis acceptance rate is close to 99.9%.

walk_num

We use a DMC algorithm with a fixed number of walkers with no population control bias. The counter part is that with a small number of walkers, additional fluctuations of the local energy are introduced. It is preferable to increase the number of walkers for the DMC calculation.

This set of parameters is fine for all the runs. As the effective time step is approximately 10 times less than in VMC, the total computational time to obtain an error bar comparable to the error bar obtained in VMC will be 10 times longer.

Run a first short DMC calculation with a small number of cores (typically one node), such that the walkers move from the VMC distribution to the DMC distribution. Clear the computed data by un-commenting *clear(blocks)*, since this calculation it is not well converged:

```
$ qmcchem_edit.py 1.1.HF

# Clear
# --------------------

# clear(all_blocks)
clear(blocks)
# clear(jastrow)
# clear(walkers)
```

Then, run a longer calculation on 10 nodes.

You should obtain these energies:

Nodes	R	<E> (DMC) (a.u)
HF	1.1	-109.4869(63)
	4.0	-109.1498(76)
CAS	1.1	-109.5094(66)
	4.0	-109.1360(61)

Dissociation energies:

W.F.	Delta E (a.u)
HF	0.1883
CAS	0.3246
DMC/HF	0.3371(98)
DMC/CAS	0.3734(90)
Exact	0.3632

Adding a new property

In this section, we will modify the sources of QMC=Chem to compute a new property. The 3D space is partitioned in two subspaces separated by the plane perpendicular to the N-N bond. We will compute the probability to find 1,2,3,4,...,14 electrons in one subspace. The corresponding local operator is implemented as an array P(elec_num). P(m) = 1.d0 where m is the number of electrons in the subspace, and P = 0.d0 eleswhere. The average of this operator will give the probability of finding 1,2,3,4,...,14 electrons in the subspace.

Adding the property to the sources

First, go into the QMC=Chem source directory:

```
$ cd ${QMCCHEM_PATH}/src
```

Create a new file, named *properties_cecam.irp.f* with the following content:

```
!==============================================================================!
! PROPERTIES
!==============================================================================!


BEGIN_PROVIDER [ double precision, proba_N2, (14) ]
 implicit none
 BEGIN_DOC
! Probability of finding N electrons on one N atom in N2
 END_DOC
 integer :: i, n

 n = 0
 proba_N2 = 0.d0
 do i=1,elec_num
   if (elec_coord(i,3) > 0.d0) then
      n += 1
   endif
 enddo
 if (n>0) then
   proba_N2(n) = 0.5d0
   proba_N2(elec_num-n) = 0.5d0
 endif


END_PROVIDER
```

Do not remove the 3 first commented lines: they are used by an embedded shell script to detect that what follows are properties to compute.

Then, build the program:

```
$ cd ${QMCCHEM_PATH}
$ make
```

Before runnning tests, we will have to restore the VMC parameters in our EZFIO databases.

Restoring the VMC configuration

QMC=Chem keeps track of all the modifications if the EZIO database:

```
$ qmcchem_log.py 1.1.HF
       |         Date         |               MD5                |
 -------------------------------------------------------------------------------
     1 |  2013-07-08 14:05:39 | 97395378eaa00b194de0536dbd172153 | Edit
     2 |  2013-07-08 14:05:42 | 97395378eaa00b194de0536dbd172153 | Generate new walkers
     3 |  2013-07-08 14:05:43 | 97395378eaa00b194de0536dbd172153 | Start run
     4 |  2013-07-08 14:06:08 | 97395378eaa00b194de0536dbd172153 | Stop run
     5 |  2013-07-08 14:06:28 | f622a3fc6e35fc3a75717d43e1b84de2 | Edit
     6 |  2013-07-08 14:06:36 | 9e8b5122372c7f6e7698a8a55861131b | Edit
     7 |  2013-07-08 14:06:43 | 9e8b5122372c7f6e7698a8a55861131b | Clear all_blocks
     8 |  2013-07-08 15:41:20 | 295d77618e1507bbd3152d6e33610ddb | Start run
     9 |  2013-07-08 15:43:28 | 295d77618e1507bbd3152d6e33610ddb | Stop run
    10 |  2013-07-09 11:38:40 | 93b358fb4ead5aa061b40c2807ea0e73 | Edit
    11 |  2013-07-09 11:38:59 | 93b358fb4ead5aa061b40c2807ea0e73 | Start run
    12 |  2013-07-09 11:44:07 | 93b358fb4ead5aa061b40c2807ea0e73 | Stop run
```

From this data, you can identify that the DMC run should be at step number 10, as the MD5 key has changed. To verify this, run:

```
$ qmcchem_log.py log 10 1.1.HF
Date        :  2013-07-09 11:38:40
MD5         :  93b358fb4ead5aa061b40c2807ea0e73
Description :  Edit

Wave function
=============

N_atoms         = 2
N_electrons     = 14 (7 alpha, 7 beta)
N_det           = 1
N_MOs           = 60
N_AOs           = 70
no Jastrow
nuclear cusp fitting

DMC
====

time_step       = 0.0001
sampling        = Brownian
N_steps         = 10000
N_walkers       = 40

Modified
========

simulation/http_server
simulation/time_step
simulation/sampling
electrons/elec_walk_num
simulation/method
simulation/title
```

You see that it is a DMC run, and that *simulation/method* has been modified from the previous step. This confirms it is the first DMC calculation. Now, you can check out the configuration of the VMC run just before this DMC run:

```
$ qmcchem_log.py checkout 9 1.1.HF

 Date        :  2013-07-09 23:22:29
 MD5         :  295d77618e1507bbd3152d6e33610ddb
 Description :  Checked out 9

 Wave function
 =============

 N_atoms         = 2
 N_electrons     = 14 (7 alpha, 7 beta)
 N_det           = 1
 N_MOs           = 60
 N_AOs           = 70
 no Jastrow
```

```
nuclear cusp fitting

VMC
====

time_step       = 0.2
sampling        = Langevin
N_steps         = 10000
N_walkers       = 20

Modified
========

electrons/elec_coord.gz
simulation/http_server
simulation/time_step
simulation/print_level
simulation/sampling
electrons/elec_walk_num
simulation/method
simulation/title
```

You can verify that this corresponds to the VMC configuration.

Running the code with the new property to sample

Now, when you run *qmcchem_edit.py*, a new item appears:

```
# Properties
# ----------

...
( ) e_ref_weight
( ) proba_n2
( ) voronoi_charges
...
```

Activate the *proba_n2* property by putting an *X* between the brackets:

```
(X) proba_n2
```

Then, submit VMC calculations for both the HF and the CAS-SCF trial wave functions at $R=1.1$ Å The results can be checked using:

```
$ qmcchem_result.py -t proba_n2 1.1.HF

#                                 proba_n2
#-----------------------------------------------------------------------
#                    Idx                            Average
    1 0.000000
    2 0.000000
    3 0.000156(15)
    4 0.01629(26)
    5 0.09477(52)
    6 0.23309(38)
    7 0.15568(52)
    8 0.23309(38)
    9 0.09477(52)
   10 0.01629(26)
   11 0.000156(15)
   12 0.000000
   13 0.000000
   14 0.000000
```

Now, check out the corresponding DMC calculations and sample the histograms. The DMC sampled quantities correspond to the mixed distribution $\Psi_t \Phi_{\mathrm{FN}}$. A first-order approximation to the properties computed with Φ^2_{FN} can by obtained by $\langle O \rangle_{\Phi^2_{\mathrm{FN}}} = 2\langle O \rangle_{\Psi_t \Phi_{\mathrm{FN}}} - \langle O \rangle_{\Psi_t^2}$

Here are the expected probabilities P(n):

n	HF	DMC(HF)	2DMC-VMC(HF)	CAS-SCF	DMC(CAS-SCF)	2DMC-VMC(CAS-SCF)
4	0.016	0.010	0.004	0.004	0.004	0.004
5	0.095	0.077	0.059	0.054	0.051	0.048
6	0.233	0.240	0.247	0.244	0.244	0.244
7	0.156	0.173	0.190	0.198	0.201	0.204
8	0.233	0.240	0.247	0.244	0.244	0.244
9	0.095	0.077	0.059	0.054	0.051	0.048
10	0.016	0.010	0.010	0.004	0.004	0.004

Going from the HF wave function to the CAS-SCF wave function tends to increase the weight of the neutral components (the probabilities of finding 7 electrons), which is expected. One can also remark that even with HF nodes, this is realized by the DMC algorithm. Using CAS-SCF nodes, the trial wave function has much better probabilities, and the DMC has less work to do. This shows that the the CAS-SCF nodes are much more physical than the HF nodes, and illustrates the difference observed in total energies when going from HF nodes to CAS-SCF nodes.

---

1 http://www.msg.ameslab.gov/gamess/

2 http://ezfio.sourceforge.net . EZFIO is the Easy Fortran I/O library generator written with IRPF90. The data is organized using the filesystem tree in plain text (eventually gzipped) files.