

USING PYSCF FOR MOLECULES AND SOLIDS

ANOUAR BENALI

COMPUTATIONAL SCIENCE DIVISION
ARGONNE NATIONAL LABORATORY
9700 S. CASS AVE. LEMONT IL, USA



QMCPACK



FUNDING: CENTER FOR PREDICTIVE SIMULATION OF FUNCTIONAL
MATERIALS, DOE BES COMPUTATIONAL MATERIALS SCIENCES PROGRAM

ACKNOWLEDGEMENTS

Supported by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, Materials Sciences and Engineering Division, as part of the Computational Materials Sciences Program.

CPSFM

Center for Predictive Simulation
of Functional Materials



- For full VMC and DMC background theory please refer to relevant literature or previous workshops


- Youtube:

https://www.youtube.com/channel/UCdca2X8NEbjX_oYv60vS4gA

- Read and practice full labs in QMCPACK Manual!!!

Variational Monte Carlo

Average of the
local energy by a
probability
distribution



Re-written in an importance sampled form in terms of the probability density

Energy

$$E_V[\Psi_T, \{a_i\}] = \frac{\langle \Psi_T | \hat{H} | \Psi_T \rangle}{\langle \Psi_T | \Psi_T \rangle} = \frac{\int d\vec{r} \Psi_T^* \hat{H} \Psi_T}{\int d\vec{r} \Psi_T^* \Psi_T} = \int d\vec{r} \pi(\vec{r}) E_L(\vec{r})$$

Variance

$$\sigma^2 = \frac{\langle \Psi_T | (\hat{H} - E_V)^2 | \Psi_T \rangle}{\langle \Psi_T | \Psi_T \rangle} = \langle \hat{H}^2 \rangle - E_V^2$$

**VMC
Distribution**

$$\pi(\vec{r}) = \frac{|\Psi_T(\vec{r})|^2}{\int d\vec{r}' |\Psi_T|^2}$$

**Local
Energy**

$$E_L(\vec{r}) = \frac{\langle \vec{r} | \hat{H} | \Psi_T \rangle}{\langle \vec{r} | \Psi_T \rangle}$$

- Variational Principle: Rigorous upper bound to the energy (= only for exact wfn)

$$E_V[\Psi_T] \geq E_{exact}$$

- Zero-Variance Principle: (Only!!!) Exact solution has zero variance

$$E_V \xrightarrow{\sigma^2 \rightarrow 0} E_{exact}$$

Variational calculations depend crucially on the form of trial wavefunction used. By selecting trial wavefunctions on physically motivated grounds, accurate wavefunctions may be obtained.

Diffusion Monte Carlo

Any initial state $|\psi\rangle$, that is not orthogonal to the ground state $|\phi_0\rangle$, will evolve to the ground state in the long time limit:

$$\lim_{\tau \rightarrow \infty} |\psi(\tau)\rangle = c_0 e^{-\epsilon_0 \tau} |\phi_0\rangle$$

In the DMC method the imaginary time evolution results in excited states decaying exponentially fast, whereas in the VMC method any excited state contributions remain and contribute to the VMC energy.

m

In position space: $\lim_{\tau \rightarrow \infty} |\psi(\mathbf{R}, \tau)\rangle = c_0 e^{-\epsilon_0 \tau} \phi_0(\mathbf{R})$

SKIPPING MANY DEVELOPMENTS

We introduce a guiding/trial function $\psi_G(\mathbf{R})$, which closely approximates the ground state.

$$f(\mathbf{R}, \tau) = \psi_G(\mathbf{R}) \psi(\mathbf{R}, \tau)$$

$$-\frac{\delta f(\mathbf{R}, \tau)}{\delta \tau} = \underbrace{\left[\sum_{i=1}^N -\frac{1}{2} \nabla^2_i f(\mathbf{R}, \tau) \right]}_{\text{diffusion}} - \underbrace{\nabla \cdot \left[\frac{\nabla \psi(\mathbf{R})}{\psi(\mathbf{R})} f(\mathbf{R}, \tau) \right]}_{\text{branching}} + \underbrace{(E_L(\mathbf{R}) - E_T) f(\mathbf{R}, \tau)}_{\text{drift}}$$

E_T is a trial energy introduced to maintain normalization of the projected solution at large τ

E_L is a local energy $E_L(\mathbf{R}) = \frac{\hat{H}\psi(\mathbf{R})_G}{\psi(\mathbf{R})_G}$

The fixed-node approximation

electron : fermion
(anti-symmetric wavefunction)

‘minus sign problem’

The fixed-node approximation: $f^{\text{FN}}(\mathbf{R}, \infty) = \Psi_{\text{T}}^{\text{FN}}(\mathbf{R}) \Phi_0^{\text{FN}}(\mathbf{R})$

If nodes of $\Psi_{\text{T}}(\mathbf{R})$ are exact $\Phi_0^{\text{FN}}(\mathbf{R}) = \Phi_0(\mathbf{R})$ 

$$\Psi_{\text{T}}(R) = J(R) \Psi_{\text{AS}}(R) = e^{J_1 + J_2 + \dots} \sum_k^M C_k D_k^{\uparrow}(\varphi) D_k^{\downarrow}(\varphi)$$

Anti-symmetric function (Pauli principle)

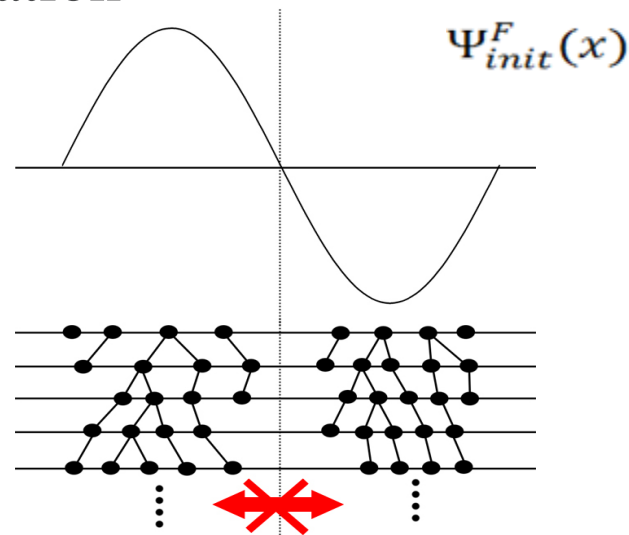
$$D_k^{\sigma} = \begin{vmatrix} \varphi_1(r_1) & \dots & \varphi_1(r_{N^{\sigma}}) \\ \vdots & \ddots & \vdots \\ \varphi_{N^{\sigma}}(r_1) & \dots & \varphi_{N^{\sigma}}(r_{N^{\sigma}}) \end{vmatrix}$$

Single-particle orbitals

$$\varphi_i = \sum_l^{l=N_b} C_l^i \Phi_l$$

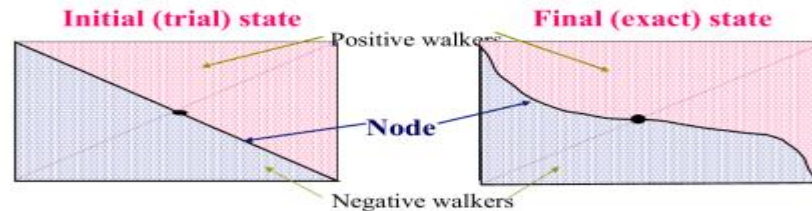
Basis sets: molecular orbitals,
plane-wave, grid-based orbitals...

$$\Phi_l$$



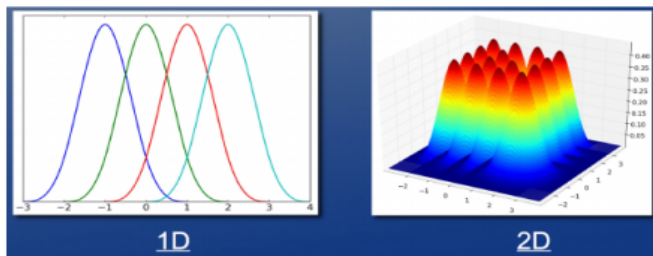
Model fermion problem: Particle in a box

Symmetric potential: $V(\mathbf{r}) = V(-\mathbf{r})$
Antisymmetric state: $f(\mathbf{r}) = -f(-\mathbf{r})$



Single particle orbitals (Splines)

- Plane wave representations are inefficient
- A real space basis is much more efficient
 - Extensive testing shows that 3D b-splines are a good choice for efficiency
 - Only 64 basis elements are nonzero at each point (versus thousands or more for plane waves)



- $B_{ijk}(x,y,z) = a_i(x)b_j(y)c_k(z)$
- Efficient routines exist to convert from a plane wave basis to b-splines with an arbitrary real space spacing using the FFT

$$\Psi_T = \exp(J) D^\dagger(\{\phi\}) D^\dagger(\{\phi\}),$$

$$D = \det[A] = \det \begin{bmatrix} \phi_1(\mathbf{r}_1) & \cdots & \phi_1(\mathbf{r}_N) \\ \vdots & \ddots & \vdots \\ \phi_N(\mathbf{r}_1) & \cdots & \phi_N(\mathbf{r}_N) \end{bmatrix},$$

$$\phi_n(x, y, z) = \sum_{i'=i-1}^{i+2} b_x^{i',3}(x) \sum_{j'=j-1}^{j+2} b_y^{j',3}(y) \sum_{k'=k-1}^{k+2} b_z^{k',3}(z) p_{i',j',k',n}.$$

LCAO with GTOs

$$\Psi_T = \exp(J) D^\dagger(\{\phi\}) D^\dagger(\{\phi\}),$$

$$D = \det[A] = \det \begin{vmatrix} \phi_1(\mathbf{r}_1) & \cdots & \phi_1(\mathbf{r}_N) \\ \vdots & \vdots & \vdots \\ \phi_N(\mathbf{r}_1) & \cdots & \phi_N(\mathbf{r}_N) \end{vmatrix},$$

QMCPACK implements linear combination of atomic orbitals (LCAO) and Gaussian basis sets in (non)periodic boundary conditions.

$$\phi(\mathbf{r}) = R_l(r) Y_{lm}(\theta, \phi)$$

where $Y_{lm}(\theta, \phi)$ is a spherical harmonic, l and m are the angular momentum and its z component, and r, θ, ϕ are spherical coordinates.

This is the Quantum Chemistry route.

Using Pyscf

- Modern open source Python code wrapping multiple fast and optimized C++ libraries.
- Handles multiple quantum chemistry (QC) methods with high efficiency for OBC systems
- Generates automatically HDF5 file readable by QMCPACK
- Large developer group with expertise in QC and programming.
- Downloadable via `> pip install pyscf`
- GitHub: <https://github.com/pyscf/pyscf>

Workflow

- 1- Running DFT/HF with Pyscf
- 2- generating input files for QMCPACK using convert4qmc
- 3- (if All electron) Applying CuspCorrection to the orbitals
- 4- Optimization of 1,2 and 3 body Jastrow
- 5- DMC run

Molecules

WORK CASE (H2O – ALL ELECTRONS)

- Detailed example is available in
> \$HOME/qmcpack_workshop_2019/day1_pyscf_molecules/

```
#!/usr/bin/env python
```

```
from pyscf import gto
from pyscf import scf, dft, df
```

```
mol = gto.Mole()
mol.verbose = 5
mol.atom = """
```

H	0.00000000	0.75720000	-0.46920000
H	0.00000000	-0.75720000	-0.46920000
O	0.00000000	0.00000000	0.11730000

```
"""
mol.unit='A'
mol.basis = 'cc-pvtz'
#mol.pseudo = 'bfd-vtz'
mol.spin=0 #Value of S where the spin multiplicity is 2S+1
mol.build()
```

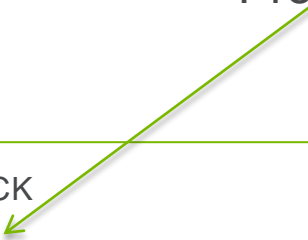
```
#Hartree Fock
#mf = scf.ROHF(mol)
```

```
#DFT
mf = dft.ROKS(mol)
mf.xc = 'b3lyp'
```

```
e_scf=mf.kernel()
```

```
#Section for QMCPACK
title="H2O_AE_DFT"
from PyscfToQmcpack import savetoqmcpack
savetoqmcpack(mol,mf,title=title)
```

Important:
Prefix!



BEFORE RUNNING!

```
#Section for QMCPACK  
title="H2O_AE_DFT"  
from PyscfToQmcpack import savetoqmcpack  
savetoqmcpack(mol,mf,title=title)
```

The system needs to know where PyscfToQmcpack.py file is located.

Provided with QMCPACK download in ~/qmcpack/src/QMCTools.

Make sure this path is part of your PYTHONPATH.

WORK CASE (H2O – ECP NCSU)

<https://pseudopotentiallibrary.org/>
(NWChem Format)

```
#!/usr/bin/env python
```

```
from pyscf import gto
from pyscf import scf, dft, df
```

```
mol = gto.Mole()
mol.verbose = 5
mol.atom = """
H      0.00000000    0.75720000   -0.46920000
H      0.00000000   -0.75720000   -0.46920000
O      0.00000000    0.00000000    0.11730000
"""
mol.unit='A'
#mol.basis = 'cc-pvtz'
#mol.pseudo = 'bfd-vtz'
mol.spin=0 #Value of S where the spin multiplicity is 2S+1
#mol.build()
```

```
mol.basis = {'H': gto.parse("""
H S
23.843185 0.00411490
10.212443 0.01046440
4.374164 0.02801110
1.873529 0.07588620
...
"""),
'O': gto.parse("""
O S
54.775216 -0.0012444
25.616801 0.0107330
11.980245 0.0018889
6.992317 -0.1742537
2.620277 0.0017622
1.225429 0.3161846
0.577797 0.4512023
0.268022 0.3121534
0.125346 0.0511167
O S
...
""})}
mol.ecp = {'O': gto.basis.parse_ecp("""
O nelec 2
O ul
1 12.30997 6.000000
3 14.76962 73.85984
2 13.71419 -47.87600
O S
2 13.65512 85.86406
"""),
'H': gto.basis.parse_ecp("""
H nelec 0
H ul
1 21.24359508259891 1.000000000000000
3 21.24359508259891 21.24359508259891
2 21.77696655044365 -10.85192405303825
H S
2 1.000000000000000 0.000000000000000
""})}
```

1) RUNNING PYSCF

```
> python H2O_AE_DFT.py | tee H2O_AE_DFT.out
```

Generates a HDF5 file containing all necessary information to run QMCPACK

2) CONVERT4QMC

- Check manual for complete description of all functionalities.

```
> convert4qmc -pyscf H2O_AE_DFT.h5 -production -addCusp
```

Generates 3 files:

- H2O_AE_DFT.structure.xml (geometry file)
- H2O_AE_DFT.wfj.xml (Wavefunction file)
- H2O_AE_DFT.qmc.in.xml (QMC blocks file)

3) CUSPCORRECTION

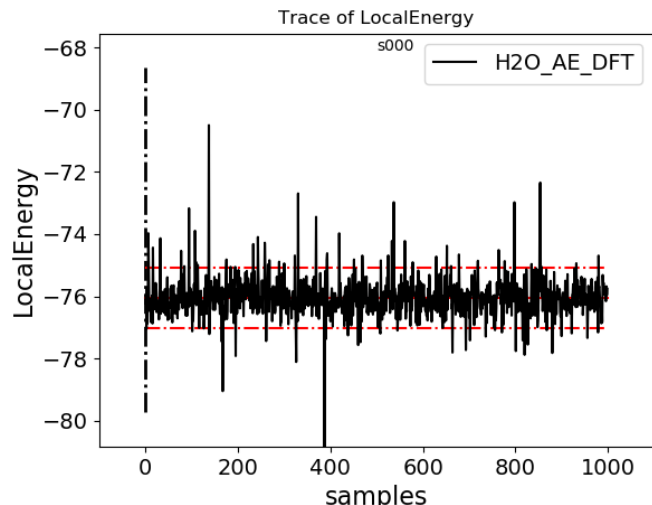
Wavefunction file

```
<?xml version="1.0"?>
<qmcsystem>
  <wavefunction name="psi0" target="e">
    <determinantset type="MolecularOrbital" name="LCAOBSset" source="ion0" transform="yes" cuspCorrection="yes" href="../H2O_AE_DFT.h5">
      <slaterdeterminant>
        <determinant id="updet" size="5" cuspInfo="../CuspCorrection/updet.cuspInfo.xml">
          <occupation mode="ground"/>
          <coefficient size="58" spindataset="0"/>
        </determinant>
        <determinant id="downdet" size="5" cuspInfo="../CuspCorrection/downdet.cuspInfo.xml">
          <occupation mode="ground"/>
          <coefficient size="58" spindataset="0"/>
        </determinant>
      </slaterdeterminant>
    </determinantset>
  </wavefunction>
</qmcsystem>
```

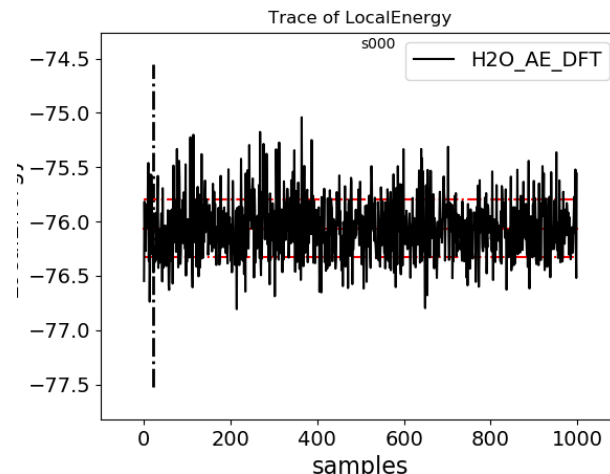
- > cd CuspCorrection
- > qmcpack cusp.xml > cusp.out

addCusp: CUSPCORRECTION

> When running all electron calculations, Orbitals are centered in the atom. This implies that during the VMC/DMC run, an electron can be moved too close to the nuclei. This would lead to a significantly large fluctuations (large variance)



	LocalEnergy	Variance	ratio
H2O_AE_DFT series 0	-76.046041 +/- 0.031209	94.281437 +/- 33.977140	1.2398



	LocalEnergy	Variance	ratio
H2O_AE_DFT series 0	-76.061034 +/- 0.008427	17.649611 +/- 0.510212	0.2320

WORK CASE (H2O – HF_AE)

\$HOME/qmcpack_workshop_2019/day1_pyscf_molecules/H2O/HF_AE/CuspCorrection

$$\Psi_T(R) = J(R)\Psi_{AS}(R) = e^{J_1+J_2+\dots} \sum_k^M C_k D_k^\uparrow(\varphi) D_k^\downarrow(\varphi)$$

With no Jastrow function, one should recover at VMC level, the Hartree Fock Energy.

$E_{\text{HF}} = -76.0571274202736$ Ha.

	LocalEnergy	Variance	ratio
VMC: H2O_AE_HF series 0	-76.061252 +/- 0.008513	17.414698 +/- 0.424133	0.2290

Note: To analyze the QMC outputs, use the “qmca” tool :
> qmca -q ev *.scalar.dat

USING ECP

Do not use addCusp. The converter will detect if ECP were used and will add the following to the Hamiltonian:

```
<?xml version="1.0"?>
<simulation>
  <project id="H2O_DFT_NCSU" series="0"/>
  <include href="../H2O_DFT_NCSU.structure.xml"/>
  <include href="H2O_DFT_NCSU.wfj.xml"/>
  <hamiltonian name="h0" type="generic" target="e">
    <pairpot name="ElecElec" type="coulomb" source="e" target="e" physical="true"/>
    <pairpot name="IonIon" type="coulomb" source="ion0" target="ion0"/>
    <pairpot name="PseudoPot" type="pseudo" source="ion0" wavefunction="psi0" format="xml">
      <pseudo elementType="H" href="../H.qmcpp.xml"/>
      <pseudo elementType="O" href="../O.qmcpp.xml"/>
    </pairpot>
  </hamiltonian>
```

By default, convert4qmc will name your ECP *.qmcpp.xml

Make sure the names and the locations match.

For NCSU ECP, you can download the XML format from:

<https://pseudopotentiallibrary.org/>

PRODUCTION FLAG

This flag will generate the most generic set of QMC blocks for production runs.

By default, it will assume:

- 1 VMC block expected to provide better samples (reduce equilibration)
- 15 Jastrow Optimization blocks
- 1 VMC long run to generate good samples
- 1 DMC block

These blocks need to be modified by hand depending on the system, hardware and accuracy you want to reach. For some system it might be a significant overkill, for others, it might be too expensive to run.

Check examples for more sensible blocks.

JASTROW OPTIMIZATION

- A detailed description of all the flags in the optimization block is described in the manual.

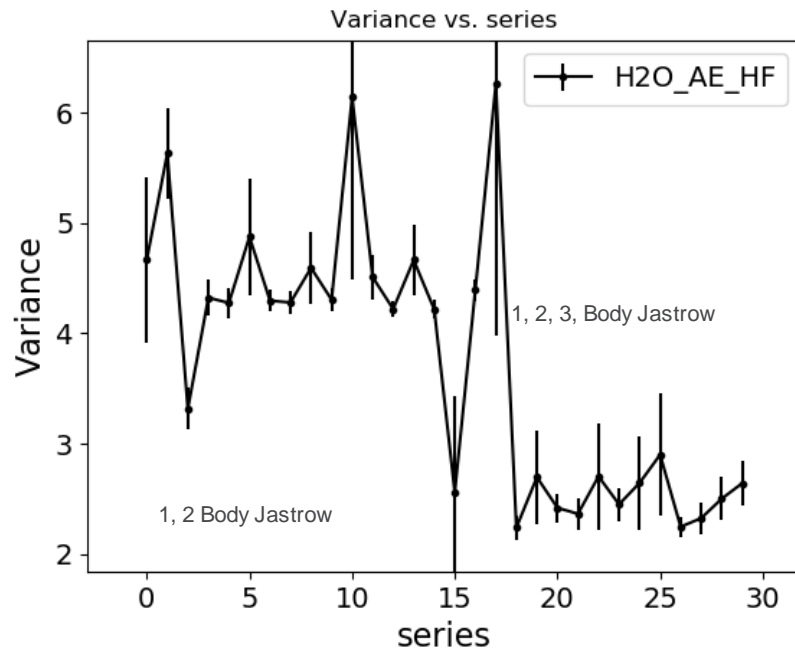
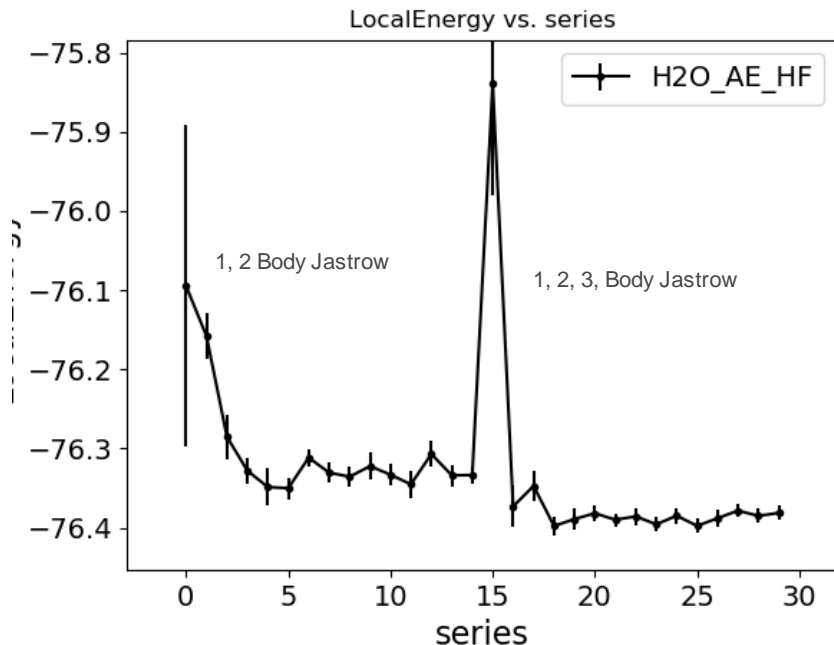
Optimization strategy:

- 1 and 2 body Jastrows only at first (with no 3 body).
- Aggressive for a small number of loops
- Add 3 body Jastrow after converging 1 and 2. Reoptimize all of them.

```
<loop max= 4 >
  <qmc method="linear" move="pbyp" checkpoint="-1">
    <estimator name="LocalEnergy" hdf5="no"/>
    <parameter name="blocks">40</parameter>
    <parameter name="warmupSteps">2</parameter>
    <parameter name="timestep">0.5</parameter>
    <parameter name="walkers">1</parameter>
    <parameter name="samples">80000</parameter>
    <parameter name="substeps">5</parameter>
    <parameter name="usedrift">no</parameter>
    <parameter name="MinMethod">OneShiftOnly</parameter>
    <parameter name="minwalkers">0.1</parameter>
  </qmc>
</loop>
```

```
<loop max="10">
  <qmc method="linear" move="pbyp" checkpoint="-1">
    <estimator name="LocalEnergy" hdf5="no"/>
    <parameter name="blocks">80</parameter>
    <parameter name="warmupSteps">5</parameter>
    <parameter name="timestep">0.5</parameter>
    <parameter name="walkers">1</parameter>
    <parameter name="samples">160000</parameter>
    <parameter name="substeps">5</parameter>
    <parameter name="usedrift">no</parameter>
    <parameter name="MinMethod">OneShiftOnly</parameter>
    <parameter name="minwalkers">0.5</parameter>
  </qmc>
</loop>
</simulation>
```

JASTROW OPTIMIZATION



!!!! New Jastrow parameters stored in file *.s0X.opt.xml

VMC energy at series N corresponds to parameters optimized at series N-1

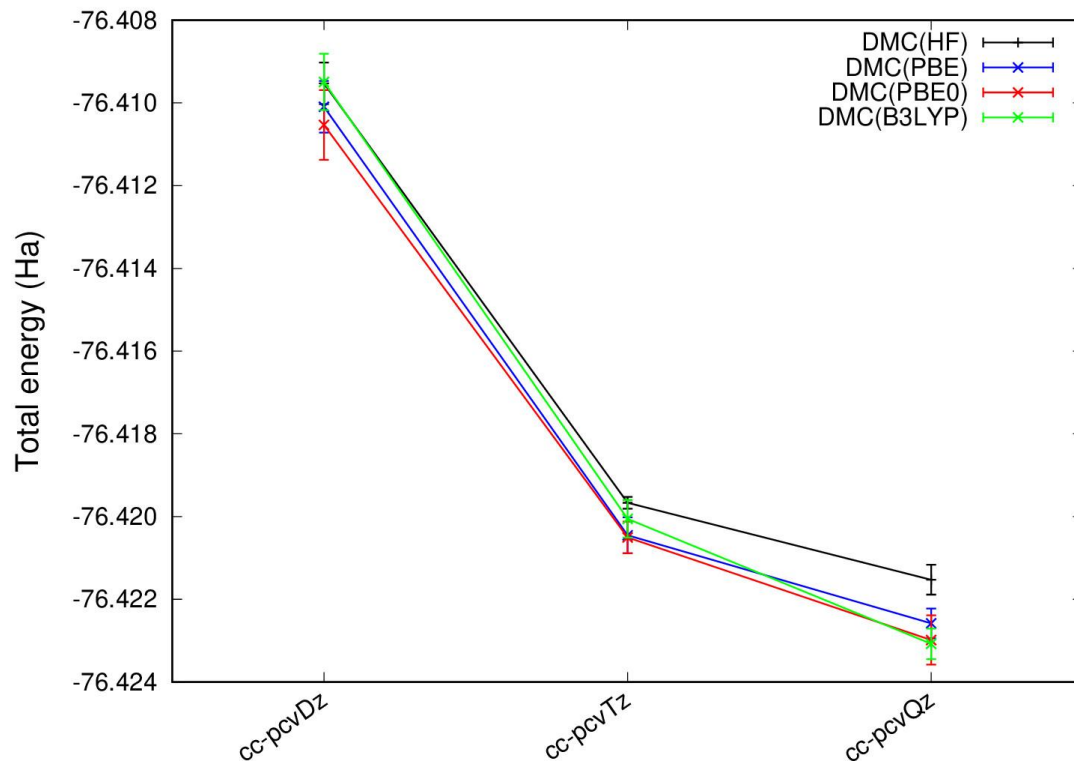
4) DMC

This is a typical DMC block where one needs to account for autocorrelation, time step correction etc..

```
<!-- DMC block -->
<qmc method="vmc" move="pbyp" checkpoint="-1">
  <estimator name="LocalEnergy" hdf5="no"/>
  <parameter name="walkers">1</parameter>
  <parameter name="samplesperthread">1</parameter>
  <parameter name="stepsbetweensamples">10</parameter>
  <parameter name="substeps">30</parameter>
  <parameter name="warmupSteps">100</parameter>
  <parameter name="blocks">200</parameter>
  <parameter name="timestep">0.1</parameter>
  <parameter name="usedrift">no</parameter>
</qmc>
<qmc method="dmc" move="pbyp" checkpoint="20">
  <estimator name="LocalEnergy" hdf5="no"/>
  <parameter name="targetwalkers">16000</parameter>
  <parameter name="reconfiguration">no</parameter>
  <parameter name="warmupSteps">100</parameter>
  <parameter name="timestep">0.001</parameter>
  <parameter name="steps">20</parameter>
  <parameter name="blocks">50</parameter>
  <parameter name="nonlocalmoves">yes</parameter>
</qmc>
</simulation>
```

Please follow examples in
\$HOME/qmcpack_workshop_2019/day1_pyscf_molecules/H2O
and README.rst for guidance.

ANALYSIS OF NODAL SURFACE AND BASISSET EFFECT



- Small dependence of DMC on the choice of trial Wavefunction
- 2mHa error on the basisset size starting cc-pcvTz.
- Still large error in energy even at large basisset (exp: -73.4368Ha)

Solids

- Please refer to literature or Lab 4 of the QMCPACK manual for a full description on how to run solids in QMC and use twist averaging theory.
- This section will only explain main differences between GTOs and splines and PBC and OBC using PYSCF.

WORK CASE (CARBON DIAMOND 2X1X1 SUPERCELL)

- Detailed example is available in
> \$HOME/qmcpack_workshop_2019/day1_pyscf_solids/

```
#!/usr/bin/env python
```

```
import numpy as np
from pyscf.pbc import gto, scf, dft
from pyscf.pbc import df
```

```
cell = gto.Cell()
cell.a      = ""
           3.37316115 3.37316115 0.000000
           0.000000 3.37316115 3.37316115
           3.37316115 0.000000 3.37316115
           ""
cell.dimension = 3
cell.basis     = 'bfd-vdz'
cell.ecp       = 'bfd'
cell.unit      = 'B'
cell.atom      = ""
              C  0.000000 0.00000000 0.00000000
              C  1.68658058 1.68658058 1.68658058
              ""
```

```
cell.drop_exponent = 0.1
```

```
cell.verbose = 5
cell.charge   = 0
cell.spin     = 0
cell.build()
```

```
sp_twist=[0.0, 0.0, 0.0]
```

```
kpts = [[ 0.0, 0.0, 0.0],
         [0.46567485, 0.46567485, -0.46567485]]
```

```
supcell=cell
```

```
mydf = df.FFTDF(supcell,kpts)
```

```
mydf.auxbasis = 'weigend'
```

```
mf = scf.KRHF(supcell,kpts).density_fit()
```

```
mf.exxdiv = 'ewald'
```

```
mf.with_df = mydf
```

```
e_scf=mf.kernel()
```

```
ener = open('e_scf','w')
```

```
ener.write('%s\n'% (e_scf))
```

```
print('e_scf',e_scf)
```

```
ener.close()
```

```
title="S2-twist1"
```

```
from PyscfToQmcpack import savetoqmcpack
```

```
savetoqmcpack(cell,mf,title=title,kpts=kpts,sp_twist=sp_twist)
```

Important:
Prefix!

WORK CASE (CARBON DIAMOND 2X1X1 SUPERCELL)

- `cell.drop_exponent = 0.1`

Some Basis functions in the BFD basis sets are too diffused, leading to ~2200 repeated images to restore periodicity while adding numerical instability and linear dependence.

It's better to remove some diffused function with exponent smaller than 0.1

WORK CASE (CARBON DIAMOND 2X1X1 SUPERCELL)

- `mydf = df.FFTDF(supcell,kpts)`

We use density fitting to generate the electronic integrals. While in the simple case, density fitting can speed up calculations significantly, they carry significant approximation that can generate large errors.

The characters of these PBC DF methods are summarized in the following table

Subject	FFTDF	AFTDF	GDF	MDF
Initialization	No	No	Slow	Slow
HF Coulomb matrix (J)	Fast	Slow	Fast	Moderate
HF exchange matrix (K)	Slow	Slow	Fast	Moderate
Building ERIs	Slow	Slow	Fast	Moderate
All-electron calculation	Huge error	Large error	Accurate	Most accurate
Low-dimension system	N/A	0D,1D,2D	0D,1D,2D	0D,1D,2D

WORK CASE (CARBON DIAMOND 2X1X1 SUPERCELL)

- `sp_twist=[0.0, 0.0, 0.0]`

```
kpts = [[ 0.0, 0.0, 0.0],  
[0.46567485, 0.46567485, -0.46567485]]
```

The goal is to run a supercell 2x1x1 at the QMC level. However, we chose to run it as a primitive cell and use a tiling of 2 0 0 0 1 0 0 0 1 that corresponds to 2 kpoints defined explicitly. (See Nexus talk by J. Krogel for how to specify the kpoints in the primitive cell).

This will lead to a non primitive cell at the QMC level.

TILING AND UNFOLDING ORBITALS

We performed the HF calculation in a primitive cell using Kpoints.

In Pyscf, when saving for qmcpack, tiling and expanding the orbitals to the supercell is performed. The orbitals will be unfolded with a phase at the SCF level, and so will the coordinates.

This can be seen in the structure.xml file:

```
<?xml version="1.0"?>
<qmcsystem>
  <simulationcell>
    <parameter name="lattice">
      6.746322300000000e+00  6.746322300000000e+00  0.000000000000000e+00
      0.000000000000000e+00  3.373161150000000e+00  3.373161150000000e+00
      3.373161150000000e+00  0.000000000000000e+00  3.373161150000000e+00
    </parameter>
    <parameter name="bconds">p p p</parameter>
    <parameter name="LR_dim_cutoff">15</parameter>
  </simulationcell>
  <particleset name="ion0" size="4">
    <group name="C">
      <parameter name="charge">4</parameter>
      <parameter name="valence">4</parameter>
      <parameter name="atomicnumber">6</parameter>
    </group>
    <attrib name="position" datatype="posArray">
      0.000000000000e+00  0.000000000000e+00  0.000000000000e+00
      1.6865805800e+00  1.6865805800e+00  1.6865805800e+00
      3.3731611500e+00  3.3731611500e+00  0.000000000000e+00
      5.0597417300e+00  5.0597417300e+00  1.6865805800e+00
    </attrib>
    <attrib name="ionid" datatype="stringArray">
      C C C C
    </attrib>
  </particleset>
  <particleset name="e" random="yes" randomsrc="ion0">
    <group name="u" size="8">
      <parameter name="charge">-1</parameter>
    </group>
    <group name="d" size="8">
      <parameter name="charge">-1</parameter>
    </group>
  </particleset>
</qmcsystem>
```

TILING AND UNFOLDING ORBITALS

We performed the HF calculation in a primitive cell using Kpoints.

The evaluation of GTOs within PBC at the QMC level implies that:

The orbitals are evaluated at a distance r in the unit cell (2x1x1 cell in this case) and then the contributions of the periodic images are added by evaluating the orbital at a distance $r + T$ where T is a translation of the cell lattice vector.

In the current implementation, the number of periodic images is an input parameter named PBCimages, which takes three integers corresponding to the number of periodic images along the supercell axes (X, Y and Z axes for a cubic cell). By default these parameters are set to PBCimages= 5 5 5 but they require manual convergence checks.

```
<?xml version="1.0"?>
<qmcsystem>
  <wavefunction name="psi0" target="e">
    <determinantset type="MolecularOrbital" name="LCAOBS" source="ion0" transform="yes"
twist="0 0 0" href="C_Diamond-211.h5" PBCimages="5 5 5">
      <slaterdeterminant>
        <determinant id="updet" size="8">
          <occupation mode="ground"/>
          <coefficient size="52" spindataset="0"/>
        </determinant>
        <determinant id="downdet" size="8">
          <occupation mode="ground"/>
          <coefficient size="52" spindataset="0"/>
        </determinant>
      </slaterdeterminant>
    </determinantset>
  </wavefunction>
</qmcsystem>
```


SUPERTWIST

We performed the HF calculation in a primitive cell using Kpoints. The number of kpoints is decided by the tiling, but the coordinates of the kpoints depends on the super twist used (in this case equivalent to a Gamma point in a 2x1x1 supercell).

In QMC, from the previous point, the evaluation of the translated orbitals will have to accommodate a phase factor computation similar to the unfolding of the kpoint orbitals from the primitive cell to the supercell.

This phase is defined as $\exp(ik \cdot g)$ where k is the super twist, and g are the PBC periodic images. When the supertwist is 0 0 0 the the phase is 1, leading to a wavefunction with real coefficients.

However, when the phase is different than 1 (or -1) the wavefunction is complex. It becomes necessary to run the complex version of QMCPACK.

While QMCPACK can handle complex wavefunction from pyscf, the phase computation is in development and should be released soon.

1) RUNNING PYSCF

```
> python diamondC_2x1x1.py | tee diamondC_2x1x1.out
```

Generates a S2-twist1.h5 file containing all necessary information to run QMCPACK

2) CONVERT4QMC

- Check manual for complete description of all functionalities.

```
> convert4qmc -pyscf S2-twist1.h5 -production
```

Generates 3 files:

- S2-twist1.structure.xml (geometry file)
- S2-twist1.wfj.xml (Wavefunction file)
- S2-twist1.qmc.in.xml (QMC blocks file)

3) VMC

\$HOME/qmcpack_workshop_2019/day1_pyscf_solids/diamondC_2x1x1-Gaussian_pp_kpts/VMC

$$\Psi_T(R) = J(R)\Psi_{AS}(R) = e^{J_1+J_2+\dots} \sum_k^M C_k D_k^\uparrow(\varphi) D_k^\downarrow(\varphi)$$

With no Jastrow function, one should recover at VMC level, the Hartree Fock Energy.

$E_{\text{HF}} = -21.20673553792076$ Ha.

	LocalEnergy	Variance	ratio
VMC series 0	-21.204825 +/- 0.011473	4.322113 +/- 0.121838	0.2038

In this specific case, the HF energy was given by pyscf in Ha/unit cell.

Make sure to be consistent.

4) DMC

\$HOME/qmcpack_workshop_2019/day1_pyscf_solids/diamondC_2x1x1-Gaussian_pp_kpts/DMC

This is a typical DMC block where one needs to account for autocorrelation, time step correction etc..

```
</simulation>
<qmc method="vmc" move="pbyp" checkpoint="-1">
  <estimator name="LocalEnergy" hdf5="no"/>
  <parameter name="walkers">1</parameter>
  <parameter name="samplesperthread">1</parameter>
  <parameter name="stepsbetweensamples">10</parameter>
  <parameter name="substeps">30</parameter>
  <parameter name="warmupSteps">100</parameter>
  <parameter name="blocks">200</parameter>
  <parameter name="timestep">0.1</parameter>
  <parameter name="usedrift">no</parameter>
</qmc>
<qmc method="dmc" move="pbyp" checkpoint="20">
  <estimator name="LocalEnergy" hdf5="no"/>
  <parameter name="targetwalkers">16000</parameter>
  <parameter name="reconfiguration">no</parameter>
  <parameter name="warmupSteps">100</parameter>
  <parameter name="timestep">0.001</parameter>
  <parameter name="steps">20</parameter>
  <parameter name="blocks">50</parameter>
  <parameter name="nonlocalmoves">yes</parameter>
</qmc>
</simulation>
```

Please follow examples in
\$HOME/qmcpack_workshop_2019/day1_pyscf_solids
and README.rst for guidance.

$E_{\text{HF}} = -21.20673553792076 \text{ Ha}$

	LocalEnergy	Variance	ratio	
VMC series 0	-21.672800 +/- 0.028277	1.330476 +/- 0.034129	0.0614	
DMC series 1	-21.912048 +/- 0.011781	1.299570 +/- 0.005588	0.0593	

Significant decrease in
energy with DMC

SUMMARY

- Pyscf is fully interfaced with QMCPACK allowing to use a wide range of basissets and nodal surfaces with both molecules and solids.
- Shifting kpoints away from supertwist Gamma leads to a complex wavefunction and the application of a phase factor to the orbital. This final phase factor is still in development and will be ready at the next release of the code.
- DMC shows a small dependence of the energy to the size of the basisset and the functional (nodal surface). This last point is system dependent and not general
- Using GTOs with PBC increases significantly the computation time but decreases significantly the memory footprint.

QUESTION?!

Variational Monte Carlo

The variational principle may be derived by expanding a normalized trial wavefunction, Ψ_T , in terms of the exact normalized eigenstates of the Hamiltonian,

$$\psi_T = \sum_{i=0}^{\infty} c_i \psi_i \quad , \quad \text{where the expansion coefficients, } c_i, \text{ are normalised} \quad \sum_{i=0}^{\infty} |c_i|^2 = 1 \quad .$$

The expectation of the many-body Hamiltonian, \hat{H} , may be evaluated

$$\begin{aligned} \langle \psi_T | \hat{H} | \psi_T \rangle &= \langle \sum_i c_i \psi_i | \hat{H} | \sum_j c_j \psi_j \rangle \\ &= \sum_i \sum_j c_i^* c_j \langle \psi_i | \hat{H} | \psi_j \rangle \\ &= \sum_i |c_i|^2 \epsilon_i \quad , \quad \text{where} \quad \epsilon_i = \langle \psi_i | \hat{H} | \psi_i \rangle \quad . \end{aligned}$$

The expectation value of a trial wavefunction with the Hamiltonian must therefore be greater than or equal to the true ground state energy.

Variational calculations depend crucially on the form of trial wavefunction used. By selecting trial wavefunctions on physically motivated grounds, accurate wavefunctions may be obtained.

Diffusion Monte Carlo

(Projector Monte Carlo)

Imaginary time Schrödinger equation ($\tau = it$)

$$\frac{\partial |\psi\rangle}{\partial \tau} = -\hat{H}|\psi\rangle,$$

The state $|\psi\rangle$ is expanded in eigenstates of the Hamiltonian $|\psi\rangle = \sum_{i=0}^{\infty} c_i |\phi_i\rangle$, where $\hat{H}|\phi_i\rangle = \epsilon_i |\phi_i\rangle$

A formal solution is

$$|\psi(\tau_1 + \delta\tau)\rangle = e^{-\hat{H}\delta\tau} |\psi(\tau_1)\rangle$$

where the state $|\psi\rangle$ evolves from imaginary time τ_1 to a later time $\delta\tau$.

If the initial state is expanded in energy ordered eigenstates,

$$|\psi(\delta\tau)\rangle = \sum_{i=0}^{\infty} c_i e^{-\epsilon_i \delta\tau} |\phi_i\rangle$$

Any initial state $|\psi\rangle$, that is not orthogonal to the ground state $|\phi_0\rangle$, will evolve to the ground state in the long time limit:

m

$$\lim_{\tau \rightarrow \infty} |\psi(\tau)\rangle = c_0 e^{-\epsilon_0 \tau} |\phi_0\rangle$$

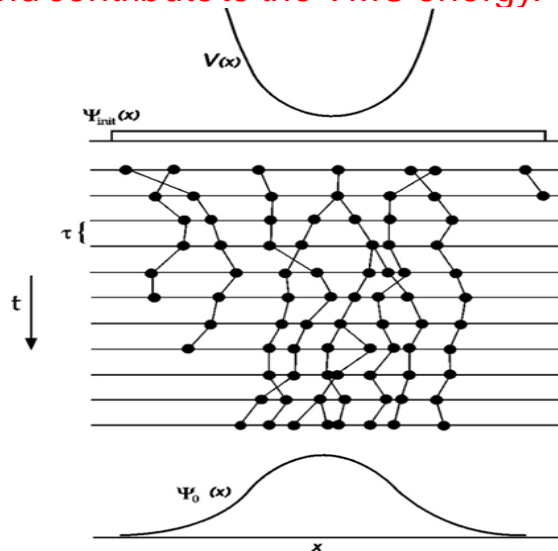
In the DMC method the imaginary time evolution results in excited states decaying exponentially fast, whereas in the VMC method any excited state contributions remain and contribute to the VMC energy.

In position space: $\lim_{\tau \rightarrow \infty} |\psi(\mathbf{R}, \tau)\rangle = c_0 e^{-\epsilon_0 \tau} \phi_0(\mathbf{R})$

$$-\frac{\delta\psi(\mathbf{R}, \tau)}{\delta\tau} = \left[\sum_{i=1}^N -\frac{1}{2} \nabla^2_i \psi(\mathbf{R}, \tau) \right] + \boxed{(V(\mathbf{R}) - E_T) \psi(\mathbf{R}, \tau)}$$

Density of diffusing particles

Branching term:
potential-dependent increase or
decrease in the particle density.



W. M. C. Foulkes *et al.*, Rev. Mod. Phys. 73, 33 (2001).

Diffusion Monte Carlo

(Projector Monte Carlo)

$$-\frac{\delta\psi(\mathbf{R}, \tau)}{\delta\tau} = \left[\sum_{i=1}^N -\frac{1}{2} \nabla_i^2 \psi(\mathbf{R}, \tau) \right] + (V(\mathbf{R}) - E_T)\psi(\mathbf{R}, \tau)$$

Can be solved with MC method but leads to very inefficient algorithm.

The potential $V(\mathbf{R})$ is unbounded in coulombic systems and hence the rate term, $(V(\mathbf{R}) - E_T)$, can diverge. Large fluctuations in the particle density leads to impractically large statistical errors.

Linear scaling for bosons (ground state can be made real)

Exponential scaling for fermions

Does not account for the anti fermionic nature of electrons; Exchange of any 2 electrons imposes a change of sign in the wavefunction.

Important sampling

(trial wavefunction)

We introduce a guiding/trial function $\psi_G(\mathbf{R})$, which closely approximates the ground state.

$$f(\mathbf{R}, \tau) = \psi_G(\mathbf{R})\psi(\mathbf{R}, \tau)$$

Previous equation becomes

$$-\frac{\delta f(\mathbf{R}, \tau)}{\delta \tau} = \underbrace{\left[\sum_{i=1}^N -\frac{1}{2} \nabla_i^2 f(\mathbf{R}, \tau) \right]}_{\text{diffusion}} - \underbrace{\nabla \cdot \left[\frac{\nabla \psi(\mathbf{R})}{\psi(\mathbf{R})} f(\mathbf{R}, \tau) \right]}_{\text{branching}} + \underbrace{(E_L(\mathbf{R}) - E_T)f(\mathbf{R}, \tau)}_{\text{drift}}$$

E_T is a trial energy introduced to maintain normalization of the projected solution at large τ

E_L is a local energy $E_L(\mathbf{R}) = \frac{\hat{H}\psi(\mathbf{R})_G}{\psi(\mathbf{R})_G}$

QUESTION?!

