QMCPACK Lab

Table of Contents

List of Tables

List of Examples

Part I. VMC and optimization

Table of Contents

Chapter 1. Applications to molecular systems: Li2

Table of Contents

1. How to run QMCPACK

On Tungsen cluster, change directory to `scratch-global`

```
cd scratch-global
tar xf ~/Li2.tar
cd Li2
```

The tar file can be downloaded with **wget**

```
wget http://cms.mcc.uiuc.edu/qmcpack/qmcss07/input/li2.tar
tar xf li2.tar
```

We will start an interactive session using 1 node (2 processors) as

```
bsub -Is -n1 -W 1:00 tcsh
```

Once your session starts, you can execute QMCPACK as

```
qmcapp input-xml > out
```

Note that your session will be terminated after an hour. Restart an interactive session with the **bsub**. During the lab hours, use the interactive queue to execute small jobs.

1.1. To run it on the laptop:

```
$QMCPATH/qmcapp input-xml > out
```

Here, `$QMCPATH` is the path where QMCPACK is built, e.g., *topdir*/build/bin.

2. QMC optimized Slater-type orbitals

2.1. Running VMC
2.2. More on the many-body wavefunction

The first QMCPACK application is to perform VMC calculation of Li_2 using an optimized wavefunction from Umrigar *et al.*, J. Chem. Physics **99**, 2865 (1993). The main input file is Li2.vmc.xml.

2.1. Running VMC

The main input file Li2.vmc.xml has instructions on what problem to solve and how to run QMCPACK.

Example 1.1. Main input file to run a VMC

```
<?xml version="1.0"?>❶
<simulation>❷
  <project id="Li2" series="0"/>❸
  <application name="qmcapp" role="molecu" class="serial" version="0.2">❹
    QMC of Li2 molecule using Slater-type Orbitals by Umrigar et al, see
    JCP 99, 2865 (1993).  The spin configuration is (3,3).
  </application>
  <random parallel="true" seed="-1"/>
  <!-- include a file which defines the wavefunctions -->
  <include href="Li2.STO.unr.xml"/>❺
  <hamiltonian name="h0" type="generic" target="e">❻
    <pairpot name="ElecElec" type="coulomb" source="e" target="e"/>
    <pairpot name="Coulomb" type="coulomb" source="i" target="e"/>
    <pairpot name="IonIon" type="coulomb" source="i" target="i"/>
  </hamiltonian>
  <qmc method="vmc" target="e" move="walker">❼
    <parameter name="blocks">10</parameter>
    <parameter name="steps">10</parameter>
    <parameter name="timeStep">0.05</parameter>
    <parameter name="walkers">100</parameter>
    <parameter name="useDrift">yes</parameter>
  </qmc>
  <qmc method="vmc" target="e" move="walker">❽
    <parameter name="blocks">500</parameter>
    <parameter name="steps">20</parameter>
    <parameter name="timestep">0.05</parameter>
  </qmc>
</simulation>❾
```

❶ DO NOT MODIFY. First line of a valid xml file.

❷ DO NOT MODIFY. Starting a qmc simulation. See the matching ending at the end.

❸ Define a project.

❹ Note on a series of runs.

❺ Include an external file which defines qmcsystem consisting of electronic and ionic system and a trial wavefunction. See Li2.STO.unr.xml.

❻ Define a many-body hamiltonian.

❼ Perform a short vmc.

❼ Perform a long vmc.

2.2. More on the many-body wavefunction

A single particle orbital of this example is expressed as a linear combination of spherical (atomic) orbitals

$\varphi_i = \Sigma_I \Sigma_l \Sigma_m C^i_{lm;I} u_{lm} (r-R_I) Y_{lm} (\theta,\varphi)$

Here, the radial function *u* is a Slater-type orbital

$u(r)=N(\zeta,p) r^p e^{\zeta r}$

with an normalization factor *N* and a power *p*. The full description of the trial wavefunction is shown in Example 1.2

Example 1.2. Wavefunction definition for Li$_2$

```
<wavefunction name="psi0" target="e">
  <determinantset type="MO" key="STO" transform="no" source="i"> ❶
    <basisset> ❷
      <atomicBasisSet type="STO" elementType="Li" normalized="no" expandYlm="no"> ❸
        <basisGroup rid="R0" n="1" l="0" m="0" type="Slater"> ❹
          <radfunc exponent="3.579103" node="1"/> ❺
        </basisGroup>
        <basisGroup rid="R1" n="1" l="0" m="0" type="Slater">
          <radfunc exponent="2.338523" node="1"/>
        </basisGroup>
        <basisGroup rid="R2" n="2" l="0" m="0" type="Slater">
          <radfunc exponent="0.707563" node="2"/>
        </basisGroup>
        <basisGroup rid="R3" n="2" l="1" m="0" type="Slater">
          <radfunc exponent="0.532615" node="2"/>
        </basisGroup>
      </atomicBasisSet>
    </basisset>
    <slaterdeterminant> ❻
      <determinant id="updet" spin="1" size="3"> ❼
        <coefficient id="updetC" type="Array">❽
          0.606630 1.0 0.0 0.061592  0.606630  1.0 0.0 -0.061592
          0.603086 1.0 0.0 2.946e-3 -0.603086 -1.0 0.0  2.946e-3
          0.104957 0.0 1.0 0.305729  0.104957  0.0 1.0 -0.305729
        </coefficient>
      </determinant>
      <determinant id="downdet" spin="-1" size="3"> ❾
        <coefficient id="downdetC" type="Array">
          0.606630 1.0 0.0 0.061592  0.606630  1.0 0.0 -0.061592
          0.603086 1.0 0.0 2.946e-3 -0.603086 -1.0 0.0  2.946e-3
          0.104957 0.0 1.0 0.305729  0.104957  0.0 1.0 -0.305729
        </coefficient>
      </determinant>
    </slaterdeterminant>
  </determinantset>
</wavefunction>
```

❶

Start of `determinantset` $$\psi(\{\mathbf{R}\}) = \sum_{\mathrm{d}} \mathrm{C_d} \mathrm{D}^{\uparrow}_{\mathrm{d}} \mathrm{D}^{\downarrow}_{\mathrm{d}}$$

❷

Start a generic basis set for the single-particle orbitals. In this example, the basis set consists of two identical sets of atomic orbitals for each Li ion.

❸

Start an atomic basis set for "Li".

❹

A `basisGroup` represents a basis function $\phi_{nlm}(\mathbf{r}) = u^n_{lm}(r) Y_{lm}(\theta,\phi)$ for a given angular momentum configurations, *l* and *m*. The additional index *n* is used to distinguish different basis sets of the same angular momenta. The radial part of the basis set u_{lm} can have one or more radial functors $f_k(r)$ as $$u_{lm}(r) = \sum_k c_k N_k f_k(r)$$

❺

Define a radial functor $f_k(r)$.

❻

Define a pair of Dirac determinants $D^{\uparrow} D^{\downarrow}$ This trial wavefunction has only one Slater Determinant. However, multiple `slaterdeterminant`s can be used [See multi determinant example]. The default value $C_d=1$ is omitted.

❼

Define a Dirac determinant for the spin-up component.

❽

Define a matrix of N_{up} by N_{basis} for N_{up} single-particle orbitals (linearly independent orbitals). They are generally provided by QMCPACK utility programs which extract the solutions from external QC/DFT packages. For this example, the table V are used to generate `coefficient`.

❾

Define a Dirac determinant for the spin-down component. This node is identical to the up component.

The total basis set space will be created by expanding it according to the order of the particles in an ionic system as listed in the Table 1.1.

Table 1.1. Relations of `determinant/coefficient` to the basis set

Li 1				Li 2			
R1(0,0,0)	**R2(0,0,0)**	**R3(1,0,0)**	**R4(1,1,0)**	**R1(0,0,0)**	**R2(0,0,0)**	**R3(1,0,0)**	**R4(1,1,0)**
0.606630	1.0	0.0	0.061592	0.606630	1.0	0.0	-0.061592
0.603086	1.0	0.0	2.946e-3	-0.603086	-1.0	0.0	2.946e-3
0.104957	0.0	1.0	0.305729	0.104957	0.0	1.0	-0.305729

3. Adding a Pade Two-Body correlation function

Starting with the QMC optimized wave function in Section 2, we are going to add correlation functions (aka Jastrow functions) to improve the variational energy and variance.

Open Li2.vmc.xml and add a line after <include/> as

```
<wavefunction name="psi0" target="e">
  <jastrow name="Jee" type="Two-Body" spin="yes" function="pade">  ❶
    <parameter id="juu_b" name="B">0.821683</parameter> ❷
  </jastrow>
</wavefunction>
```

❶

A two-body Jatrow function

$$J_2(\{\mathbf{R}\}) = \exp\left(-\sum_{i,j} f(r_{ij})\right)$$

is added to the trial wavefunction. The attributes `jastrow/@type='Two-Body'` and `jastrow/@function='pade'` are used to uniquely determine what object is used to represent the two-body Jastrow. `determinantset` is identitcal to the previous example.

❷

Define a parameter B for the Pade functor $f(r) = \frac{Ar}{1+Br}$ Here, the coefficient *A* is ommitted, since it is set by the cusp conditions. An important attribute is `parameter/@id='juu_b'` which is used later by optimization engines in Example 1.3.

and save it. Or, download the Li2.j2pade.vmc.xml.

4. How to optimize a wavefunction

Li2.j2pade.opt.xml uses **optimize** QMC action shown in Example 1.3 to obtain *juu_b* which minimizes either variance or energy or their combination using a simple gradient-based optimization method. More on the optimizations will be discussed in other section. The most important element is `optimize` with the parameters whose IDs are registered by the `wavefunction` section. This example has only one variable with ID="juu_b" for the parameter *B* of a Pade functor.

Example 1.3. QMC section for optimization of Pade parameter `juu_b`

```
<qmc method="optimize" move="pbyp">
  <parameter name="blocks">500</parameter>
  <parameter name="steps">20</parameter>
  <parameter name="timestep">0.5</parameter>
  <parameter name="walkers">100</parameter>
  <optimize> juu_b</optimize>
  <cost name="energy"> 0.0 </cost>
  <cost name="variance"> 1.0 </cost>
  <cost name="difference"> 1.0 </cost>
  <parameter name="minWalkers"> 0.4 </parameter>
  <parameter name="useWeight"> no </parameter>
  <optimizer method="cg">
    <parameter name="max_steps">20</parameter>
    <parameter name="tolerance"> 1e-6 </parameter>
    <parameter name="stepsize"> 0.1 </parameter>
    <parameter name="friction"> 5 </parameter>
    <parameter name="epsilon"> 1e-4 </parameter>
  </optimizer>
  <parameter name="power"> 2 </parameter>
  <parameter name="correlation"> 0.01 </parameter>
</qmc>
```

5. Experiments and Results

Change the parameters of VMC simulations and compare the results with dataspork.

- move
- timeStep
- useDrift

6. Advanced topics

6.1. Using a Two-Body correlation function of Wagner-Mitas functor

This example shows an alternative correlation function to the previous section using the functors introduced by Wagner-Mitas (J. Chem. Phys. **126**,034105 (2007)). The difference lies in the functional form of *f(r)* radial function for the Two-Body Jastrow. The Wagner-Mitas radial function is a linear combination of "Wagner-Mitas" functors g_{WM} as where each WM functor is given as $g_{WM} = \frac{1 - z(r/r_c)}{1 + Bz(r/r_c)}$ The scaling function is *z(x) = x*x*(6-8*x+3*x*x)* for a given cutoff radius r_c. Download the Li2.STO.3.xml and run it as

```
$QMCPATH/qmcapp Li2.STO.3.xml
```

Example 1.4. Two-Body Jastrow function with WM functors

```
<wavefunction name="psi0" target="e">
  <jastrow name="J2" type="Two-Body" function="any" print="yes"> ❶
    <basisset>
      <atomicBasisSet type="WM" elementType="e" normalized="yes" expandYlm="no"> ❷
      <grid type="linear" ri="0" rf="10" npts="101"/> ❸
        <basisGroup rid="R0" n="1" l="0" m="0" type="WM"> ❹
          <radfunc id="ee0" exponent="1.55606692e+00" contraction="4.42863468e-01"/> ❺
          <radfunc id="ee1" exponent="2.92608972e+00" contraction="-1.19855434e+00"/>
          <radfunc id="ee2" exponent="5.16813113e+00" contraction="9.39917398e-01"/>
          <radfunc id="ee3" exponent="2.25725654e+00" contraction="-0.5" type="cusp"/> ❻
        </basisGroup>
      </atomicBasisSet>
    </basisset>
  <jastrow>
  <determinantset type="MO" key="STO" transform="no" source="i">
   ....
  </determinantset>
</wavefunction>
```

❶

Define a Two-Body Jastow using generic radial functions as indicated by `jastrow/@function='any'`. This node contains a `basisset` whose definition was discussed for the molecular basis set.

❷

The Two-Body Jatrow function for the electronic system as indicated by `wavefunction/@target='e'` can have only one `atomicBasisSet` for the electrons independently of the spin states.

❸

Define a cutoff radius `grid/@rf` of the `basisGroup/radfunc` elements. Internally, the radial part of each `basisGroup` is represented by a cubic spline function on a linear grid as indicated by `grid/@method='linear'`. `grid/@npts` sets the number of grid points (knots) for the spline functions. The first grid point `grid/@ri` is zero.

❹

Start of a radial function *f(r)* whose components are given as the child `radfunc` elements.

❺

Define the C_k (contracton) and B_k (exponent) for a WM functor. Each WM functor is identified by unique names `radfunc/@id` which are again utilized by the optimization procedure. The type of each `radfunc` is set by the parent node `basisGroup/@type='WM'`.

❻

The WM function does not satisfy the cusp condition, although by using many WM functions, one can mimic the cusp condition very closely. In order to amend this problem, a radial functor which imposes the cusp condition can be added. `radfunc/@contraction='-0.5'` is fixed by the e-e cusp condition.

The attribute names `contraction` and `exponent` are originated from the contracted Gaussian-type or Slater-type bais set. `contraction` denotes the contraction factor of individual radial functors and `exponent` is a scaling parameter of radial functors, e.g., ζ of a Slater-type orbital.

In the same spirit, the parameter *B* of $g_{WM} = \frac{1-z(r/r_c)}{1+Bz(r/r_c)}$ is set by `exponent`.

Using a Two-Body Jastrow of WM functors introduces more variables that can be varied by the optimization engines. Both `contraction` and `exponent` can be optimized as shown in Example 1.5. `contraction` is named as '`radfunc/@id`'_C, while `exponent` as '`radfunc/@id`'_E.

The last `radfunc/@id='ee3'` corrects the cusp condition of the Two-Body Jastrow function which cannot be enforced by the contraction of WM functors.

Example 1.5. Listing optimizable variables given by `radfunc`

```
<qmc method="optimize" move="pbyp" completed="no">
  ....
  <optimize>ee0_C ee1_C ee2_C ee0_E ee1_E ee2_E ee3_E</optimize>
  ...
</qmc>
```

6.2. Adding Three-Body correlation function

This example introduces Three-Body Jastrow function. The most general Three-Body Jastrow function implemented in QMCPACK is Geminal Three-Body function by Casula and Sorella (J. Chem. Phys. **121**, 7110 (2004)).

$$J_3 = \exp\left(-\sum_{i,j}\sum_{a}\sum_{b}\sum_{l,m}\sum_{l',m'}\lambda_{a,l,m;b,l',m'}\phi_{l,m}(\mathbf{r}_i-\mathbf{R}_a)\phi_{l',m'}(\mathbf{r}_j-\mathbf{R}_b)\right)$$

Here, *i* and *j* denote the electron indices, *a* and *b* the ionic indices. For this example, we will use a special case of Geminal Three-Body function where the ionic indicies are identical, i.e., with $l\ (l')=0$ components.

Example 1.6. Three-Body Jastrow function for Li_2

```
<wavefunction name="psi0" target="e">
  <jastrow name="J3" type="Three-Body" function="any" transform="yes" print="yes" source="i"> ❶
    <basisset> ❷
      <atomicBasisSet type="WM" elementType="Li" normalized="yes" expandYlm="no">
        <grid type="linear" ri="0" rf="2" npts="101"/>
        <basisGroup rid="R0" n="1" l="0" m="0" type="WM">
          <radfunc id="j3Li0" exponent="4.55664" contraction="1.0"/>
        </basisGroup>
        <basisGroup rid="R1" n="1" l="0" m="0" type="WM">
          <radfunc id="j3Li1" exponent="1.51161" contraction="1.0"/>
        </basisGroup>
        <basisGroup rid="R2" n="1" l="0" m="0" type="WM">
          <radfunc id="j3Li2" exponent="0.5" contraction="1.0"/>
        </basisGroup>
      </atomicBasisSet>
    </basisset>
    <coefficient type="lambda" id="j3c" offset="0" diagonal="yes" sameBlocksForGroup="yes"> ❸
       <lambda i="0" j="0" c="-1.73649239e-01"/> ❹
       <lambda i="0" j="1" c="-2.49253960e+00"/>
       <lambda i="0" j="2" c="3.16682365e+00"/>
       <lambda i="1" j="1" c="3.67818042e-01"/>
       <lambda i="1" j="2" c="-1.09726145e+00"/>
       <lambda i="2" j="2" c="2.54268472e-01"/>
    </coefficient>
  </jastrow>
   ....
</wavefunction>
```

❶

Add a Three-Body Jastrow function. The attributes are

type

Type of Jastrow function, Three-Body

function

Type of radial functors. **any** indicates any pre-defined functors can be used.

transform

With `yes`, use a spline function. For `function='any'`, transform is always `yes`.

source

The name or id of `particleset` which provide ionic centers.

print

With `yes`, the radial functors are printed out in a file. This is ignored for MPI jobs.

❷

Define a "Molecular Basis Set" similarly to the single-particle orbitals in a Molecular Basis Set.

❸

Start of the λΛ coefficients.

type

Data type. `type='lambda'` lists the coefficient one by one.

id

ID of the coefficients. A set of names for the coefficients is generated for optimization as '`coefficient/@id`'_*i*_*j* where *i*=`lambda/@i` and *j*=`lambda/@j`.

offset

Offset for the indices `lambda/@i` and `lambda/@j`.

diagonal

With `yes`, an optimized version of Three-Body Jastrow using Diagonal Block Sparse Matrix method is used. If `diagonal='no'`, general Three-Body method is used. The performance penalty of use of the general Three-Body method is insignificant for Li_2 but can be measurable for large-scale systems.

sameBlocksForGroup

With `yes`, the coefficient blocks for the particles are set to be identical if they belong to the same species (group). If `sameBlocksForGroup='no'`, all the coefficients have to be provided explicitly. Otherwise, only the first block for the first Li atom will be assign. The second block will be set to zero.

❹

Define a coefficient $\lambda_{i,j}$ where *i*(*j*) are composite indices for the ionic index *a*(*b*) and angular momenta $l(l^{'})$ and $m(m^{'})$.

Finally, `optimize` will have more variables to optimize:

```
<optimize> j3c_0_0 j3c_1_1 j3c_2_2 j3c_0_1 j3c_0_2 j3c_1_2</optimize>
```

Chapter 2. Applications to solids: Bulk Carbon

Table of Contents

On Tungsen cluster, `BulkCarbon.tar` can be found on your home directory. Follow these steps:

```
cd scratch-global
tar xf ~/BulkCarbon.tar
cd BulkCarbon
```

The tar file can be downloaded with **wget**

```
wget http://cms.mcc.uiuc.edu/qmcpack/qmcss07/input/BulkCarbon.tar
```

We will submit batch jobs for this exercise, since each run can take 15 minutes to hours. The files ending with **batch** are batch scripts and should be used as

```
bsub < batch-script
```

1. Introduction to bulk calculations

We will calculate the total energy of bulk carbon using Slater-Jastrow wavefunction whose one-particle orbitals are the solution of DFT calculation in a plane-wave basis set. Being a crystalline system, we need to specify a supercell for the electrons and ionic systems. Although it is possible to create an input file from scratch, this is seldom necessary since we use standard DFT or Quantum Chemistry packages to obtain one-particle orbitals and build a trial wavefunction with it. Everything defined within `qmcsystem` is automatically generated by a conversion tool **pw2qmcpack.x** a post-processing tool for Quantum Espresso http://www.pwscf.org/.

Another important difference of this exercise from Li2 is that we use pseudopotentials for Carbon atoms instead of a Coulomb potential.

Example 2.1. How to define a many-body Hamiltonian with pseudopotentials.

```
<hamiltonian name="h0" type="generic" target="e">
  <pairpot name="ElecElec" type="coulomb" source="e" target="e"/>
  <pairpot name="IonIon" type="coulomb" source="ion0" target="ion0"/>
  <pairpot name="PseudoPot" type="pseudo" source="ion0" wavefunction="psi0" format="xml"> ❶
    <pseudo elementType="C" href="C.ldaopt.xml"/>
  </pairpot>
</hamiltonian>
```

❶

Define a `pairpot` of `pseudo` (pseudopotentials). Since non-local pseudopotential evaluations involve calculating ratio, it is important to tell which `wavefunction` is used. Several formats are supported. Here, we will use C.ldaopt.xml.

For periodic systems, we have to treat long-range interactions and long-range correlation with care. We will use Ewald breakup method by V. Natoli and D. M. Ceperley, J. Comput. Phys. **117**, 171-178 (1995) and K. P. Esler (note is available upon request).

2. Using DFT orbitals in a plane-wave basis set

The main input file is C.pw.xml. A single particle orbital of this example is expressed as a linear combination of plane-wave basis functions:

$$\phi_i(\mathbf{r}) = \sum C_i^{\mathbf{k}} e^{i\mathbf{k}\cdot\mathbf{r}}$$

Here, the coefficients

$$C_i^{\mathbf{k}}$$

are obtained by solving a KS equation within the DFT. Below is the xml section to define a Slater determinant wavefunction.

Example 2.2. How to use single-particle orbitals in a plane-wave basis set.

```
  <wavefunction name="psi0" target="e">
    <determinantset type="PW" href="c8.pwscf.h5" version="0.10">❶
      <basisset ecut="2.500000000000000E+001">❷
        <grid dir="0" npts="32" closed="no"/>
        <grid dir="1" npts="32" closed="no"/>
        <grid dir="2" npts="32" closed="no"/>
      </basisset>
      <h5tag name="twistIndex">❸
        0
      </h5tag>
      <h5tag name="twistAngle">❹
        0.000000000000000    0.000000000000000    0.000000000000000
      </h5tag>
      <slaterdeterminant>❺
        <determinant id="updet" size="16">
        <occupation mode="ground" spindataset="0">❻
          </occupation>
        </determinant>
        <determinant id="downdet" size="16">
          <occupation mode="ground" spindataset="0">
          </occupation>
        </determinant>
      </slaterdeterminant>
    </determinantset>
    </wavefunction>
```

❶

type

Tell QMCPACK that the single-particle orbitals are in a plane-wave basis set so that a specialized class can be used for the calculations.

href

Name of hdf5 file which contains large binary data. This file contains all the eignvectors at the k-points used by DFT calculations.

version

Version of the external hdf5 file so that we can handle future changes.

❷

Define parameters which determine the plane-wave basis set. The conversion tool writes the FFT grid for the given energy cutoff.

❸

Select the index of the twist angle of this wavefunction.

❹

The k-point value in a reduced unit. Here, using Gamma point (0,0,0).

❺

Define a Determinant set which has two <determinant/> for up and down electrons.

❻

Use only the 16 lowest eigenstates.

3. Using DFT orbitals on a grid

A single particle orbital of this example is now tabulated on a regular grid. We use a spline method to evaluate the orbitals and their graidents and laplacians. Friday lecture, Order(N) methods in QMC, by Dario Alfe will discuss the details on the real-space wavefunctions. The orbitals in this example are identical to those in the previous section on the grid points and we will interpolate the values between the grid points. The main input file is C.bs.xml.

Example 2.3. How to use single-particle orbitals on a regular grid.

```
  <wavefunction name="psi0" target="e">
    <determinantset type="bspline" href="c8.pwscf.h5" version="0.10">❶
      <basisset ecut="2.500000000000000E+001">❷
        <grid dir="0" npts="32" closed="no"/>
        <grid dir="1" npts="32" closed="no"/>
        <grid dir="2" npts="32" closed="no"/>
      </basisset>
      <h5tag name="twistIndex">❸
        0
      </h5tag>
      <h5tag name="twistAngle">❹
        0.000000000000000    0.000000000000000    0.000000000000000
      </h5tag>
      <h5tag name="eigenstates">❺
        eigenstates_32_32_32
      </h5tag>
      <slaterdeterminant>❻
        <determinant id="updet" size="16">
        <occupation mode="ground" spindataset="0">❼
          </occupation>
      </determinant>
      <determinant id="downdet" size="16">
        <occupation mode="ground" spindataset="0">
        </occupation>
      </determinant>
    </slaterdeterminant>
  </determinantset>
  </wavefunction>
```

❶

type

Tell QMCPACK that the single-particle orbitals are on a regular grid so that a specialized class can be used for the calculations.

href

Name of hdf5 file which contains large binary data. The same file is used to store both the plane-wave and real-space wavefunctions.

version

Version of the external hdf5 file so that we can handle future changes.

❷

Define the real-space grid.

❸

Select the index of the twist angle of this wavefunction. Identical to the plane-wave basis set.

❹

The k-point value in a reduced unit. Here, using Gamma point (0,0,0). Identical to the plane-wave basis set.

❺

The name of hdf5 group name which has the real-space wavefunctions. The conversion tool uses the grid information and create a group.

❻

Define a Determinant set which has two <determinant/> for up and down electrons. Identical to the plane-wave basis set.

❼

Use only the 16 lowest eigenstates.

4. Adding RPA Jastrow Function

The main input file is [C.xml]() which has `jastrow/@function='rpa'`. Note that nothing else, e.g., `parameter`s, is specified.

```
<include href="c8.pwscf.xml"/>
 <wavefunction name="psi0" target="e">
   <jastrow name="Jee" type="Two-Body" function="rpa"/>
 </wavefunction>
```

For RPA Jastrow, all the parameters will be determined internally: R_s (density) and the radial function of the short-range part and the **k**-dependent long-range part.

5. Optimizing One- and Three-Body Jastrow parameters

We will further improve the variational trial wavefunction by adding One- and Three-Body Jastrow functions. The main input file is [C.123.xml](). The key section is

```
<qmc method="optimize">
  <optimize> c0_C c1_C c2_C j3c_0_0 j3c_0_1 j3c_0_2 j3c_1_1 j3c_1_2 j3c_2_2</optimize>
</qmc>
```

which lists the names of variational paremeters to be varied to optimize a combination of energy and variance.