

SMR 1595 - 6

**Joint DEMOCRITOS - ICTP School on
CONTINUUM QUANTUM MONTE CARLO METHODS
12 - 23 January 2004**

**COMPUTER LABORATORY SESSION
VARIATIONAL MONTE CARLO AND
VARIANCE OPTIMIZATION EXAMPLES**

Friedemann SCHAUTZ and Claudia FILIPPI
Instituut-Lorentz, Universiteit Leiden, The Netherlands

These are preliminary lecture notes, intended only for distribution to participants.

Variational Monte Carlo and Variance Optimization Examples

Friedemann Schautz and Claudia Filippi

January 12, 2004

1 Programs and input

There are two programs, `vmc` and `fit` inside the `bin` directory which are used to perform a Variational Monte Carlo (VMC) calculation and a corresponding variance minimization, respectively.

A combined input is used for both steps, defining the system (geometry of the molecule), the wave function and operational parameters (number of steps etc.). With beryllium (`beryllium.inp`) as an example one would execute the commands

```
../../bin/vmc < beryllium.inp > beryllium.vmc.out
```

```
../../bin/fit < beryllium.inp > beryllium.fit.out
```

for VMC and optimization. The VMC run produces the following files:

<code>energies.dat</code>	energies along the run suitable for plotting
<code>mc_configs_new</code>	random walker positions for optimization
<code>mc_configs_start</code>	initial random walker position
<code>restart_vmc</code>	restart information for continuing a run
<code>input.log</code>	echo of settings
<code>output.log</code>	additional information

Variance minimization is done on the set of configurations found in the file `mc_configs`, so copy `mc_configs_new` to `mc_configs` if you want to use the newly sampled points in the optimization. After running `fit` there will be a file called `parameters.new` containing the new set of wave function parameters.

Instead of pasting the new parameters into the input file one can also put them in a separate file and load it from the input file, see `be_load_parameters.inp` for an example.

The input file `be_commented.inp` contains a description of all input variables and flags, and is therefore a good starting point to explore the VMC/FIT programs.

2 Form of the wave function

The trial wave function Ψ_T that is used in all examples has the following structure:

$$\Psi_T(R) = J(R) \sum_i d_i D_i^\uparrow(R) D_i^\downarrow(R) \quad (1)$$

with the simple Jastrow factor

$$J = \prod_{ij,\uparrow\downarrow} \exp \left\{ \frac{1}{2} \frac{r_{ij}}{1 + b r_{ij}} \right\} \times \prod_{ij,\uparrow\uparrow} \exp \left\{ \frac{1}{4} \frac{r_{ij}}{1 + b r_{ij}} \right\}. \quad (2)$$

The Slater determinants D_i are built from single particle functions ('molecular orbitals') which are linear combinations of Slater functions ('atomic orbitals'). In a typical ground state wave function the Hartree-Fock determinant is the dominant contribution, possibly followed by some low excitations accounting for so-called near degeneracies (see for example the Beryllium atom below).

$$D_{\text{HF}} = \begin{vmatrix} \phi_1(\mathbf{x}_1) & \dots & \phi_1(\mathbf{x}_N) \\ \vdots & & \vdots \\ \phi_N(\mathbf{x}_1) & \dots & \phi_N(\mathbf{x}_N) \end{vmatrix} \quad (3)$$

$$\phi_i(\mathbf{r}) = \sum_{\alpha k} c_{k\alpha}^i N_{k\alpha} r_\alpha^{n_{k\alpha}-1} e^{-\zeta_{k\alpha} r_\alpha} Y_{l_{k\alpha} m_{k\alpha}}(\hat{\mathbf{r}}_\alpha) \quad (4)$$

Such a wave function has the following variational parameters: Parameter b of the Jastrow factor, coefficients of atomic orbitals in molecular orbitals $c_{k\alpha}^i$, basis function exponents $\zeta_{k\alpha}$ and coefficients in front of determinants d_i . These parameters can be varied together or some are kept fixed to improve convergence at the cost of a worse wave function.

3 Input example: Beryllium atom ground state

3.1 About the input format

There are two kinds of things in the input : Variable lists start with

```
&listname
```

followed by pairs variable-name value and keywords (sometimes followed by arguments) usually start a block of input (a matrix for instance) ended by the word 'end'. Outside such blocks comments (started by '#') and blank lines can appear anywhere. the keyword 'load filename' inserts the contents of the file filename at the given point. The keyword `fit_input` separates the VMC input from the FIT input.

3.2 Variational Monte Carlo related input

Title and random number generator seed

```
&general title 'Be atom ground state'
&general seed 1837465927472523
```

There are two versions of the Metropolis Algorithm, `imetro 1` or `imetro 2`. Type 1 is the simple algorithm in a box with wave function gradient and has the following parameters: step size for moves `delta`, cut-off parameter for gradient (0..1) `fbias`. Note: `fbias=0` yields simple box algorithm. Type 2 is the drift-diffusion algorithm with step size `tau`.

```
&vmc imetro 1 delta .7 fbias 1.0
&vmc imetro 2 tau 0.1
```

The next variables control how many steps are computed and which output is generated.

```
# write energies each n blocks to file energies.dat
&vmc nwrite 10
```

```
# Parameters of the VMC run
#  nstep      : number of Monte Carlo steps per block
#  nblkeq     : number of blocks in equilibration phase
#  nblk       : number of blocks for sampling averages
#  nconf_new  : number of configurations written for variance optimization
```

```
&blocking_vmc nstep 100 nblk 10000 nblkeq 1 nconf_new 2000
```

The next part of the input defines the molecule (here a single atom). An 'atom type' is a charge with a basis set, an 'atom' is an 'atom type' put at a specific point in space.

```
# Number of atom types, number of atoms
&atoms nctype 1 natom 1
```

```
# Nuclear charge for each atom type
znuc
4.
end
```

```
# place atom type i at location : x y z Type
# coordinates in Bohr !
geometry
0. 0. 0.      1
end
```

Now the trial wave function of the general form specified above is defined. This consists of the following specifications

- (Slater-type) atomic basis functions
- Orbitals (linear combinations of the basis functions)
- Slater determinants where the orbitals become populated with electrons
- Determinant part multiplied by Jastrow factor

```

# Number of slater basis functions (one line per atom type)
# 1s  2s 2px 2py 2pz  3s 3px 3py 3pz n3dzr n3dx2 n3dxy n3dxz n3dyz
basis
  2   1  1  1  1    0  0  0  0  0    0  0  0  0
end

# exponents of these basis functions
exponents 6
  4.70474785  3.09432724  1.08044852  1.05484882  1.05484882  1.05484882
end

# 'linear combination of atomic orbitals'
# rows correspond to orbitals , columns to basis functions
lcao      5      6
  0.660225207  1. -0.259057109  0.  0.  0.
  0.0974117356  0.  1.          0.  0.  0.
  0.            0.  0.          1.  0.  0.
  0.            0.  0.          0.  1.  0.
  0.            0.  0.          0.  0.  1.
end

# determinants: orbitals filled with electrons.
# first line : coefficients of determinants in the wavefunction
# remaining lines: occupation of orbitals (orbital index according to lcao)
# spin-up electrons first, than spin-down electrons
# (nelec= total number of electrons, nup= spin-up electrons, this defines
# implicitly the spin state )

&electrons nelec 4 nup 2

determinants      4
  1. -0.105206319 -0.105206319 -0.105206319
  1  2  1  2
  1  3  1  3
  1  4  1  4
  1  5  1  5
end

```

Note: spherical symmetry requires that the 3 $1s(2)2p(2)$ determinants ($1s(2)2px(2)$, $1s(2)2py(2)$ and $1s(2)2pz(2)$) must have the same coefficient.

The simple Jastrow factor used here has only one free parameter b , since the first parameter is fixed to 0.5 by the electron-electron cusp-condition.

```
jastrow_parameter
  0.5  1.16853828
end
```

This is all for VMC, the rest controls variance minimization.

3.3 Variance minimization input

```
fit_input

# ndata : number of configurations to use (<= nconf_new from above)
# eguess: trial energy
# ipr   : printing level
&fit ndata 2000 eguess -14.670  ipr 1
```

The nucleus-electron cusp conditions can be enforced either exactly (`icusp 1`) or by using a penalty function (`icusp 2`). In the latter case the function to be minimized becomes the sum of the variance of the local energy and the cusp penalty which becomes zero if the cusp conditions are satisfied. The weight parameter `cuspw` controls the relative importance of the two parts. If `icusp` is set to 0 the cusp conditions are not enforced at all.

```
&fit icusp 1 cuspw 1.
```

Now one needs to specify how many and which parameters should be optimized. The number of parameters in the optimization will in general be less than the number of parameters in the wave function for two reasons: First, some parameters are eliminated due to symmetry requirements on the wave function (see `equivparms`) below, the cusp-condition (if `icusp=1`), and redundancy of orbital similarity transformations (see `pivot` below). Second, one might want to 'freeze' certain parameters, e.g. basis function exponents to enhance convergence of the optimization. (In 'production' calculations on larger systems it is customary to optimize the parameters of the jastrow factor only).

```
fitparms
1 4 1 1  orbital coefficients, exponents, determinant coefficients, jastrow parameters
1 3      pairs (orbital,basisfunction) corresponding to matrix 'lcao' above
1 2 3 4  exponents to be optimized
2        coefficients in front of which determinants should be optimized
end
```

The next two lists are used to impose the cusp-condition correctly. The information could be obtained automatically from the basis and orbital specification above, but we have to help the program a bit, so we repeat ...

```
# list of orbital angular momenta
lorbital
0 0 1 1 1
```

```
end
```

```
# list of main quantum number and angular momentum of basis functions
nlbasis
1 0 1 0 2 0 2 1 2 1 2 1
end
```

The `pivot` list defines how orbitals can mix: One integer for each orbital, orbitals with the same (and non zero) absolute value of this integer belong to the same 'group' and will be mixed. A negative value means that the orbital can contribute to other orbitals in the group but not vice versa. For the wave function used here, we can always add a contribution from the 2s orbital to the 1s orbital since whenever 2s is occupied 1s is occupied as well (1s is always occupied). But the contrary is not true, since there are determinants where 2s is empty. The 2p orbitals all have different symmetry and should therefore not be allowed to mix.

```
pivot
1 -1 0 0 0
end
```

Finally, there is the specification which parameters are equivalent and should not be optimized separately, for `lcao`, exponents and determinants. These are lists of pairs saying that the first object gets its value from the second one. For orbitals this are pairs of pairs: $ijkl$ fixes the coefficient of basis function j in orbital i to the value of basis function l in orbital k . Negative numbers fix a quantity to be minus the other one.

```
equivparms 0 2 2 # number of pairs for lcao, exponents and determinants
            # orbitals
5 4 6 4 # exponents
3 2 4 2 # determinant coefficients
end
```

In order to automatize the optimization iteration one needs to remove the sections `exponents`, `lcao`, `determinants`, `jastrow_parameter` from the input and puts them into a file, say `parameters.dat` which is included using `load parameters.dat` and replaced by the corresponding output of the optimization procedure (the file `parameters.new`).

3.4 Diffusion Monte Carlo input

For a diffusion Monte Carlo calculation basically the same input as for the corresponding VMC calculation applies. In addition the following parameters need to be specified:

```
&dmc tau 0.02 etrial -14.6
&blocking_dmc nblk 10 nstep 200 nmove 5 nwalkers 10 nblkeq 2
```

Here `tau` is the imaginary time step, `etrial` the DMC trial energy, `nblk` and `nstep` are the number of blocks and 'steps' per block. In contrast to the VMC calculation each step consists of `nmove` passes after which branching is

performed. Therefore the total number of passes is equal to `nblk * nstep * nmove`. The initial number of random walkers is given by `nwalkers`.

The DMC part is provided as a third executable (`bin/dmc`).

4 Acknowledgment

The VMC and variance minimization code is a simplified version of the code **CHAMP** written by Cyrus Umrigar and Claudia Filippi (with contributions of Friedemann Schautz). For variance minimization the QuenchAnneal-library of Peter Nightingale and Cyrus Umrigar is used. This is a modified Leveberg-Marquart algorithm allowing uphill moves. The diffusion Monte Carlo addition was provided by Saverio Moroni.

The tutorial code provided is limited to all-electron calculations of molecules using Slater-type basis functions and the simple Jastrow factor described above. Also only a simple VMC algorithm is included.

The complete **CHAMP** code featuring non-local pseudopotentials, more sophisticated Jastrow factors and VMC algorithms as well as periodic boundary conditions can be obtained on a collaboration basis with one of the authors only.