minichem

Version 1.0

User Manual

Alexander Oleynichenko

January 4, 2019

Contents

1 General considerations

`minichem` is a tiny quantum chemistry program written in educational purposes. It is based mainly on two textbooks:

- [1] A. Szabo, N. Ostlund, "Modern Quantim Chemistry";
- [2] T. Helgaker, P. Jorgensen, J. Olsen, "Molecular Electronic-Structure Theory".

The source code is written in the C99 programming language. Currently `minichem` is oriented on the Unix-like operation systems.

Features:

- single-point energy calculations;
- restricted (RHF) and unrestricted (UHF) Hartree-Fock theories;
- DIIS convergence technique;
- OpenMP parallelization;
- Available elements: all periodic table (but only non-relativistic treatment is available!);
- cartesian basis sets, arbitrary angular momentum of basis functions.

The source code is available on Github:
`https://github.com/aoleynichenko/minichem`

Authos is looking forward to any comments and questions!
`alexvoleynichenko@gmail.com`.

2 Compilation and testing

Required libraries:

- `libc` – the C standard library;
- `MPI` – Message Passing Interface (higher-level routines for the distributed-memory communication environment);

- `OpenMP` – Open Multi-Processing (API for shared memory multiprocessing programming);
- `BLAS` and `LAPACK` – numerical linear algebra libraries.

Tools needed to compile the source code:

- `CMake`
- `make`
- C compiler (GNU or Intel are recommended)

How to compile:

```
$ mkdir build && cd build
$ cmake ..
$ make [-jN]
```

Test set can be found in the `test` folder. The testing system requires the Python2 programming language environment to be installed on your system. How to test:

```
$ cd test
$ python test.py
```

3 Getting started

How to run `minichem`:

```
$ minichem.x <input-file.inp>
```

Executable `minichem.x` sends the output to the stdout, hence it is more convenient to redirect it to file:

```
$ minichem.x <input-file.inp> | tee <output-file.out>
```

The input language design resembles the NWChem input files language [3]. As well as NWChem, `minichem` works as an interpreter: it executes input files written in a very simple scripting language. The `minichem` scripting language largely coincides with the NWChem language; wherever possible, the names of directives and sections have been chosen in that way that the input file can be interpreted by both quantum-chemical programs without some extra edits.

`minichem` input files consist of simple (top-level) directives and sections (compound directives). Directives are single-line commands; sections combine sets of directives. Usually sections specify control parameters for the separate modules

(for example, convergence parameters of the SCF procedure). `minichem` reads the input file sequentially, line by line, and changes its internal variables according to the given values of the parameters specified in it. Once `minichem` reaches the `task` directive, the execution of the input file stops and calculation begins. Several `task` directives are allowed in one input file; between them the user can change any parameters, thus organizing a cascade of calculations.

The `minichem` language also supports single-line comments beginning with #. Language is case insensitive.

Let us consider the RHF/STO-3G calculation of a benzene molecule. Some explanations to all the directives encountered are given in the comments. Detailed descriptions of all possible directives and sections are given in Section 4. A few other examples can be found in Section 5.

```
# C6H6 single-point energy
#
# RHF molecular orbitals will be written to the molden-format
# file c6h6.mos

# name for the task
start C6H6

# allowed memory usage
memory 10 mb

# print input file before executing it
echo

# number of OpenMP threads
nproc 8

# geometry (cartesian)
# default: charge = 0
geometry
  C    0.000    1.396    0.000
  C    1.209    0.698    0.000
  C    1.209   -0.698    0.000
  C    0.000   -1.396    0.000
  C   -1.209   -0.698    0.000
  C   -1.209    0.698    0.000
  H    0.000    2.479    0.000
  H    2.147    1.240    0.000
  H    2.147   -1.240    0.000
  H    0.000   -2.479    0.000
  H   -2.147   -1.240    0.000
  H   -2.147    1.240    0.000
end
```

```
# basis set specification
# (keyword SPHERICAL -- only for compatibility with NWChem)
basis "ao basis" SPHERICAL
H    S
      3.42525091           0.15432897
      0.62391373           0.53532814
      0.16885540           0.44463454
C    S
     71.6168370            0.15432897
     13.0450960            0.53532814
      3.5305122            0.44463454
C    S
      2.9412494           -0.09996723
      0.6834831            0.39951283
      0.2222899            0.70011547
C    P
      2.9412494            0.15591627
      0.6834831            0.60768372
      0.2222899            0.39195739
end

# options for the SCF module
# by default: singlet
scf
  print "overlap"   # print AO overlap integrals
  diis 5           # enable DIIS, max subspace dim = 5
  maxiter 20       # max number of SCF iterations
end

# export calculated data (molecular orbitals)
out
  molden   # to the MOLDEN .mos format
end

# do RHF calculation
task scf
```

4 Input files format

4.1 start

Syntax:

```
start <string name>
```

The start directive specifies a short task identifier (no whitespaces are allowed). This identifier is used in the names of some temporary and output files.

4.2 echo

Syntax:

```
echo
```

If the `echo` directive is specified in the input file, `minichem` redirects the contents of the input file to standard output. It is recommended always to use this directive.

4.3 memory

Syntax:

```
memory <integer> <string units>
# <units>: one of b (bytes), kb, mb, mw (megawords), gb
```

The maximum amount of RAM that can be allocated and used by the program.

4.4 nproc

Syntax:

```
nproc <integer>
```

Number of OpenMP threads (for parallel execution).

4.5 task

Syntax:

```
task <string theory>
```

The directive specifies which quantum chemical method is to be used to solve an electronic problem.

Since currently only the HF method is implemented, the `task` directive can be called only with the `scf` argument:

```
task scf
```

4.6 geometry

Syntax:

```
geometry [units <string units default angstroms>]
  [charge <integer default 0>]
  [mult <integer default 1>]
  <string elem> <real x y z>
  . . .
end
```

The `geometry` compound directive specifies the cartesian coordinates of the atoms, distance units (boron or angstrom), total system charge spin multiplicity. Given the charge and multiplicity values, the program can automatically choose which version of the Hartree-Fock method shoukd be used.

4.7 charge

Syntax:

```
charge <integer>
```

The directive sets the total charge of the system (a.u.). Added for compatibility with NWChem.

4.8 basis

Syntax:

```
basis [<string name default "ao basis">] [spherical|cartesian]
  <string elem> <string shell_L>
    <real exponent> <real list_of_coefficients>
    . . .
  . . .
end
```

Gaussian basis set definition. Currently only cartesian basis sets are implemented (the `cartesian` keyword), so the `spherical` keyword is useful only for compatibility with NWChem.

The coefficients of the contracted basis functions are arranged in columns; the first column contains the exponents of primitive gaussian functions.

The angular momentum (`<string shell_L>`) of the block of basic functions is denoted, as usual, by the letters `S`, `P`, `D` ... The `SP` notation (one compressed function of type s, the second is of type p) is also allowed. Basis functions with arbitrary L are supported.

4.9 scf

Syntax:

```
scf
  [rhf | uhf]
  [singlet | doublet | triplet | quartet | quintet]
  [guess (core | eht)]
  [direct | nodirect]
  [maxiter <integer max_no_of_iterations default 50>]
  [diis [<integer subspace_dim default 5>] | nodiis]
  [print <string what>]
  [noprint <string what>]
end
```

The `scf` compound directive is used to configure the SCF procedure. It may contain one or more directives from the following list:

`guess` MO initial guess. Possible values:

- `core` – bare nuclei guess: all two-electron (electron-repulsion) integrals are neglected. Works well only for small systems with few electrons;
- `eht` – (default) extended Hückel theory [4] (uses the Wolfsberg-Helmholtz formula [5]).

`rhf/uhf` Hartee-Fock theory type (`rhf` – restricted, `uhf` – unrestricted).

`singlet/doublet/triplet/quartet/quintet` spin multiplicity.

`maxiter` max number of SCF iterations. Default value: 50.

`diis` enable DIIS convergence technique [6, 7]. The optional argument of the directive is the maximum dimension of the iterative subspace that is used for extrapolation (default value: 5). DIIS can be recommended in almost all cases and is therefore included by default.

`nodiis` disable DIIS.

`direct` enable direct SCF method (two-electron AO integrals are recomputed on each iteration without storage on disk). Direct SCF can work an order of magnitude slower than the conventional one. May be useful in case of lack of disk space.

`nodirect` conventinal SCF – read precomputed two-electron AO integrals from disk (default).

`print` what additional information should be printed (but not printed by default). The type of the argument is a string in double quotes. Possible arguments of the `print` directive are listed below:

`"final vectors analysis"`	MO expansion coefficients
`"overlap"`	AO overlap integrals
`"kinetic"`	AO kinetic energy integrals
`"potential"`	AO nuclear-attraction integrals
`"eri"`	AO two-electron (repulsion) integrals

`noprint` disables printing of additional information. Arguments are the same as for `print` (see above).

4.10 prop

Syntax:

```
prop (quadrupole)
  [center ((origin || com || coc || point <real x y z>) default origin)]
end
```

Properties calculations. Currently available properties are electric dipole and quadrupole moments.

As with any multipole moment, if a lower-order moment, monopole or dipole in this case, is non-zero, then the value of the quadrupole moment depends on the choice of the coordinate origin. Another properties can depend on coordinate origin too. The user also has the option `center` to choose the center of multipole expansion.

`center` Coordinate origin for properties calculations. Allowed variants are `origin` (point (0,0,0)), `com` (center of mass), `coc` (center of charge), `point` (arbitrary point given by th user). Default value – `origin`.

4.11 out

Syntax:

```
out
  [molden]
end
```

Export data for further processing by other programs. May contain one or more directives from the following list:

`molden` export information about the basis set and molecular orbitals to the MOLDEN format (`.mos`) [8, 9, 10]. At the moment, is implemented only for the RHF method.

5 Typical calculations

Li atom, UHF/STO-3G:

```
start Li
echo

geometry units atomic
  Li 0 0 0
end

basis "ao basis" SPHERICAL
Li    S
     16.1195750              0.15432897
      2.9362007              0.53532814
      0.7946505              0.44463454
Li    SP
      0.6362897             -0.09996723              0.15591627
      0.1478601              0.39951283              0.60768372
      0.0480887              0.70011547              0.39195739
end

scf
  uhf
  diis
  doublet
end

task scf
```

CH_3 radical, UHF/STO-3G

```
start CH3
echo

geometry
C     0.08745162    -0.08744725    -0.08742186
H     0.52503428     0.78909888    -0.52508604
H    -0.78884450    -0.52530342    -0.52536318
H     0.52530257    -0.52529218     0.78892712
end

# STO-3G
basis "ao basis" SPHERICAL
H    S
      3.42525091              0.15432897
      0.62391373              0.53532814
     0.16885540              0.44463454
```

```
C    S
     71.6168370              0.15432897
     13.0450960              0.53532814
      3.5305122              0.44463454
C    S
      2.9412494             -0.09996723
      0.6834831              0.39951283
      0.2222899              0.70011547
C    P
      2.9412494              0.15591627
      0.6834831              0.60768372
      0.2222899              0.39195739
end

scf
  uhf
  doublet
end

task scf
```

References

[1] A. Szabo and N.S. Ostlund. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*. Dover Books on Chemistry. Dover Publications, 1996.

[2] T. Helgaker, P. Jorgensen, and J. Olsen. *Molecular Electronic-Structure Theory*. Wiley, 2008.

[3] M. Valiev, E.J. Bylaska, N. Govind, K. Kowalski, T.P. Straatsma, H.J.J. van Dam, D. Wang, J. Nieplocha, E. Apra, T.L. Windus, and W.A. de Jong. NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations. *Comput. Phys. Commun.*, 181:1477, 2010.

[4] R. Hoffmann. An Extended Hückel Theory. I. Hydrocarbons. *J. Chem. Phys.*, 39(6):1397 – 1412, 1963.

[5] M. Wolfsberg and L. Helmholz. The Spectra and Electronic Structure of the Tetrahedral Ions MnO_4^-, CrO_4^{--}, and ClO_4^-. *J. Chem. Phys.*, 20(5):837, 1952.

[6] P. Pulay. Convergence acceleration of iterative sequences. the case of scf iteration. *Chem. Phys. Lett.*, 73(2):393, 1980.

[7] P. Pulay. Improved SCF convergence acceleration. *J. Comput. Chem.*, 3(4):556.

[8] G. Schaftenaar and J.H. Noordik. Molden: a pre- and post-processing program for molecular and electronic structures. *J. Comput. Aided Mol. Des.*, 14:123, 2000.

[9] G. Schaftenaar, E. Vlieg, and G. Vriend. Molden 2.0: quantum chemistry meets proteins. *J. Comput. Aided Mol. Des.*, 31:789, 2017.

[10] The Molden Format. `http://www.cmbi.ru.nl/molden/molden_format.html`.