MADNESS Molecular electronic structure calculations

Last Modification: 7/7/2016

This file is part of MADNESS.

Copyright (C) 2007, 2010 Oak Ridge National Laboratory

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or(at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

For more information please contact:

Robert J. Harrison
Oak Ridge National Laboratory
One Bethel Valley Road
P.O. Box 2008, MS-6367
Oak Ridge, TN 37831

email: harrisonrj@ornl.gov
tel: 865-241-3937
fax: 865-572-0680

Table of Contents

1 Overview **1**

1.1 Capabilities 1

1.2 Current status 1

2 Configuring and building **3**

3 Execution **5**

3.0.1 Environment variables 6

3.0.2 PBS script for LI-red 7

3.0.3 PBS script for SeaWulf 7

3.1 Input structure 9

3.2 Reproducibility 9

4 Geometry specification **11**

5 Ground state molecular DFT and HF **13**

5.1 XC — DFT exchange correlation 14

5.2 Restarting 14

5.3 Controlling convergence and accuracy 15

5.4 Geometry optimization 16

5.5 Properties 16

5.6 Plotting 17

5.7 Parallel execution 18

Chapter 1

Overview

1.1 Capabilities

1.2 Current status

The software is still in what might best described as pre-production quality. Most things mostly work, but there are enough quirks that unfortunately it is far from being a black box like most Gaussian codes.

Using the code to do actual science is the most effective mechanism for us to find and fix issues, as well as to prioritize implementation effort. Please report problems, issues and functionality requests to the MADNESS issue tracker at XXXXXXXXXXXXXXX.

Chapter 2

Configuring and building

This is just for LI-red for the purposes of the summer school. MADNESS should build out of the box on most Linux and Apple Macintosh boxes. Intel MKL is now free so you should alway use that (download from Intel), and there is an open source version of Intel TBB (Linux distributions should have that in their package managers). If you are building to run on a virtual machine you should configure with the never spin option.

1. Copy the modules from section 3.0.2 below into your `.bashrc` file. You will be using the Intel compiler, MPI, MKL and TBB which is the recommended stack when running on a cluster.
2. Clone MADNESS from `https://github.com/m-a-d-n-e-s-s/madness`. I (RJH) used revision `a1b4bae8acf57ae363f24cb8a7ea9e7229a0d95f` for testing.
3. `cd madness`
4. `sh autogen.sh` — only need to do this after initial checkout
5. `autoreconf` — only need to do this if a configure script has changed
6. Configure with this command

   ```
   ./configure --disable-shared \
        CC=mpiicc CXX=mpiicpc +\verb+ MPICC=mpiicc MPICXX=mpiicpc
   ```

   If you have installed LIBXC you can specify it with option

   ```
   --with-libxc=full-path-to-libxc+.
   ```

7. `make -j 10 libraries` — using just 10 processes on LI-red is to be social. Also, if building on your own machine note that some of the files can use over 2GB of memory to compile, so if you have limited memory you may need to restrict the number of processes. If compiling on `cn-mem` I always use `make -j libraries` which uses maximum parallelism (since `cn-mem` has 72 cores and 3 TB of memory.

8. `cd src/apps/moldft`

9. `make moldft`

10. `MAD_NUM_THREADS=10 ./moldft` — run a quick test (the `input` file by default does LDA water which is quick).

Chapter 3

Execution

To run `moldft` with a single, multi-threaded process simply invoke the name of the executable with the name of the input file on the command line. If the input file is ommited, it defaults to a file named `input`. E.g.,

```
./moldft inputfilename
```

Recommendations:

- Use Intel TBB if it is available since it is more scalable than the old task pool.
- By default, MADNESS will create one thread for every core in the computer, but even with TBB it may be more effective to use fewer threads (see the environment variable `MAD_NUM_THREADS`). E.g.,

  ```
  MAD_NUM_THREADS=10 ./moldft inputfile
  ```

- Oversubscribing the cores can lead to very poor performance due to cache contention and locks/mutexes being held by de-scheduled threads. Oversubscription can happen due to other processes (of yourself or other users) running at the same time, or from use of hyperthreading (which on Intel processors makes one phyiscal core appear to the system as two cores).
- Many threads seem to interact badly with how we are using MPI. Thus, it can help both robustness and performance to use multiple MPI processes per node with fewer threads per processes. As a side benefit you can bind the threads associated with each process to execute on the same socket, which improves thread scaling. See the SeaWulf batch script below for an example.

To run in parallel using MPI to create multiple, multi-threaded processes use the command `mpirun` (or on some systems `mpiexec`) — this is very system dependent so you may need additional options to specify where and how many processes are created (see below for use on SeaWulf and LI-red). E.g.,

```
export MAD_NUM_THREADS=10
mpirun -np 12 ./moldft inputfilename
```

3.0.1 Environment variables

`MAD_NUM_THREADS` — Sets the total number of threads to be used by MADNESS. When running with just one process all threads are devoted to computation (1 main thread with the remainder in the thread pool). When running with multiple MPI processes, one of the threads is devoted to communication. If you have 8 or more cores per node, it is recommended to leave at least one free for use by the O/S and MPI.

`MRA_DATA_DIR` — Full path to the directory containing the MRA data files (twoscale and autocorrelation coefficients; Gauss-Legendre quadrature weights). This is usually not needed unless you have moved the build/installation directory.

`MAD_BUFFER_SIZE` — Sets the buffer size (in bytes) used by the active messages (default is 1.5MBytes). Never needed by moldft?

`MAD_RECV_BUFFERS` — Sets the number of receive buffers used by the communication thread (default is 128 and a minimum of 32 is enforced). If you are experiencing hangs when running with MPI, making this number a bit larger (e.g., 256) can sometimes help. Too many buffers can cause performance problems.

`MAD_SEND_BUFFERS` — Sets the number of outstanding asynchronous send buffers (default 128 and a minimum of 32 is enforced). If you are experiencing hangs when running with MPI try making this smaller (to throttle the volume communication).

`MAD_NSSEND` — Every `MAD_NSSEND` messages, MADNESS requests a receipt acknowledgment from the receiver in an attempt to throttle the volume of communication. The default is the number of send buffers. Set to the value `1` to force receipt every message to be acknowledged. This makes things a bit slower but can workaround MPI hangs.

3.0.2 PBS script for LI-red

This example makes a 2 hour job running in the short queue on 8 nodes with one process per node and job name `moldft`. Each process will use 20 threads.

```
#!/bin/bash
#PBS -l nodes=8:ppn=1,walltime=02:00:00
#PBS -q short
#PBS -N moldft

export EXE=/home/rjh/madness/src/apps/moldft/moldft
export INPUT=benzenehf.in
export OUTPUT=benzenehf.out

export I_MPI_FABRICS=shm:ofa
export MAD_NUM_THREADS=20

cd $PBS_O_WORKDIR

mpirun -ppn 1 $EXE $INPUT >& $OUTPUT
```

The modules I (RJH) used for testing were (put the following in your `.bashrc`)

```
module load shared
module load torque/5.1.0
module load maui/3.3.1
module load gcc/4.9.2
module load intel/compiler/64/16.0.2/2016.2.181
module load intel/mkl/64/11.3.2/2016.2.181
module load intel-mpi/64/5.1.2/5.1.2.150.XXX
module load intel/tbb/64/4.4.3/2016.2.181
module load intel/vtune/2016/2016.1.1.434111
```

3.0.3 PBS script for SeaWulf

On SeaWulf the PBS option `ppn` is presently not consistent with that on LI-red — this will change soon. In the meantime the script is a little more complicated in order to get just two MADNESS processes per node with threads able to use all cores.

ASIDE: SeaWulf has 28 cores per node and a 40 Gbit/s InfiniBand network whereas LI-red has 24 cores per node and a 56 Gbit/s network. Somehow these factors (or others?) combine to make MADNESS execution on SeaWulf with 1

MPI process that uses 26 MADNESS threads/node not reliable. Hence, the job below uses two MPI processes per node each with 11 MADNESS threads.

```
#!/bin/bash
#PBS -l nodes=12:ppn=28,walltime=10:00:00
#PBS -q default
#PBS -N geomopt

# Probably best to put these in your .bashrc so don't need here
module load intel/compiler/64/16.0/2016.1.056.XXX
module load intel/mkl/64/11.3/2016.1.056.XXX
module load intel-mpi/64/5.1.2/5.1.2.150.XXX
module load intel/tbb/64/4.4.3/2016.2.181
export INTEL_LICENSE_FILE=28518@129.49.83.234

# Intel MPI should use shared-memory and OFED for communication
export I_MPI_FABRICS=shm:ofa

# We will create two processes per node --- bind to separate sockets
export I_MPI_PIN_DOMAIN=socket

# Should be set elsewhere but may not be
export I_MPI_HYDRA_BOOTSTRAP=rsh

# 14 cores per socket = Linux+MPI+unknown+madness
export MAD_NUM_THREADS=11

# Change to the directory from which the job was submitted
cd $PBS_O_WORKDIR

# Presently SeaWulf is configured so that to get a dedicated node we
# need to set ppn=28 above which gives us 28 processes per node, but we
# want two.  So need to make our own host file.
sort < $PBS_NODEFILE | uniq > $$.hosts
let NPROC=$PBS_NUM_NODES*2

# Paths to files
MOLDFT=/gpfs/home/rharrison/madness/src/apps/moldft/moldft
INPUTFILE=input
OUTPUTFILE=output

# Redirect output to a file so that can see it while job is running

mpirun -f $$.hosts -ppn 2 -n $NPROC $MOLDFT $INPUTFILE  >& $OUTPUTFILE
```

3.1 Input structure

Presently the input *must* contain both a `dft` and a `geometry` input block. The file is scanned for the first such block (so you can have multiple blocks in a file — just the first is used). Within these blocks, lines beginning with `#` are read as comments. Outside these blocks, data is ignored.

3.2 Reproducibility

For small molecules, the calculations should usually be fully reprodicible from run to run or with different numbers of threads or processes. However, for larger molecules, the different rounding error arising from different order of parallel execution can cause different execution paths. So results may vary in the last digit or so, and sometimes more especially if additional/fewer iterations are made. However, results at convergence should *always* agree to the that expected from the truncation and convergence thresholds, and the number of iterations between runs should only differ by about one. If you are seeing worse behavior than this, then please send the input and output to the MADNESS issue tracker.

Chapter 4

Geometry specification

```
geometry
    units atomic or angstrom (default atomic)
    eprec precision (default 1e-4)
    tag x y z
    pstag x y z charge
end
```

The units and energy precision (`eprec`) must presently be specified *before* coordinates of atoms are specified.

`units` — E.g., `units angstrom` — The user input units which can be `atomic` or `angstrom` (default is `atomic`). Beware — presently, the output file always contains atomic units.

`eprec` — E.g., `eprec 1e-5` — The energy error per atom in atomic units due to smoothing the nuclear potential (default is `1e-4`). The default is adequate for most chemical purposes (giving energy differences to at least `1e-5` atomic units per atom and geometries accurate to about `1e-3` atomic units) unless you are doing very high accuracy benchmark studies. Gradients may get noisy if you make `eprec` too small, though we don't have much experience with this yet.

`tag x y z` — E.g., `Be 0.3 -0.1 3.1` — Specifies an atom using its atomic symbol (case insensitive) and Cartesian coordinates.

`bqtag x y z charge` — E.g., `Bq9 20.0 20.0 20.0 -10.0` — The tag must begin with `Bq` (case insensitive). It is used to place a charge at an arbitrary location (e.g., to simulate an external field).

`pstag x y z charge` — E.g., `psBe 0.3 -0.1 3.1` — The tag must begin with `ps` (case insensitive). It is used to put a pseudopotential on selected atoms (*I think ???????*).

Chapter 5

Ground state molecular DFT and HF

```
dft
    directives/keywords
end
```

`nopen value` — E.g., `nopen 3` — The number of unpaired spin orbitals, $n_\alpha - n_\beta$ (default `0`)

`unrestricted` — Selects a spin-unrestricted calculation (default is spin restricted)

`xc value` — E.g., `xc HF` — Selects the exchange correlation potential (default is `LDA`). See XC section for more details.

`aobasis value` — E.g., `aobasis sto-3g` — Sets the atomic orbital basis used for the initial guess. Options are `sto-3g` (down to Iodine) or `6-31g` (down to Zinc, default).

`charge value` — E.g., `charge -1.0` — Total charge (default `0`) on the molecule. Atomic units.

`nvalpha value` — E.g., `nvalpha 2` — The number of alpha spin virtual orbitals to solve for (default `0`) — is this working now?

`nvbeta value` — E.g., `nvbeta 2` — The number of beta spin virtual orbitals to solve for (default `0`) — is this working now?

`no_orient` — Do not reorient/translate the molecule to orientation/center.

`core_type value` — E.g., `What is available?` Selects the pseudopotential to be used on all atoms (can also do mixed all-electron/pseudopotential calculation). Not heavily tested and unoptimized. (default is all electron).

`psp_calc` — Perform pseusopotential calculation on all atoms. Not heavily tested and unoptimized. (default is all-electron)

`L value` — E.g., `L 50` — Sets the computational box size to $[-L, L]^3$ atomic units (mostly for testing). Default is to find cube that contains all nuclei and to add 50 atomic units.

5.1 XC — DFT exchange correlation

Without LIBXC, the code just provides either Hartree-Fock (`xc HF`) or local (spin) density approximation (`xc LDA`, the default).

With LIBXC, in addition to HF and LDA (default) there are wide varity of GGA and hybrid functionals available — the ones that have been tested (to some extent) have been provided with simple input formats

- Becke-Perdew (91?) — `xc bp`
- Becke-Perdew 86 — `xc bp86`
- PBE — `xc pbe`
- PBE-0 — `xc pbe0`
- B3LYP — `xc b3lyp`

There is also a more general input format in which you can provide a list of functional names (using LIBXC's naming scheme) and the coefficient by which to scale it. For instance, to get PBE-0 you could specify `xc GGA_X_PBE .75 GGA_C_PBE 1. HF_X .25`.

We have not yet implemented the near linear-scaling algorithm for HF exchange, which as a consequence is fairly slow and may require a lot of memory (there is low-memory algorithm in the code but I don't think it is yet available as an input option).

5.2 Restarting

At completion of an HF or DFT calculation, the molecular orbitals are saved in the files `restartdata.*` (with one file per I/O server process). The projection of the orbtials onto the `sto-3G` AO basis set is saved into the file `restartaodata`.

`restart` — Restart from numerical orbitals from a previous calculation (default is no)

`restartao` — Restart from projection of orbitals onto AO basis set from a previous calculation (default is no unless doing geometry optimization). If a restart file is not found, or the file contains incompatible data then the default atomic guess is used.

`save value` — E.g., `save false` — Boolean flag to save (or not) orbitals at completion (default is true).

5.3 Controlling convergence and accuracy

The default convergence test is on both the 2-norm of change in density per atom (separately for each spin) between iterations and the residual error in each wave function.

```
  converged = (da < dconv * molecule.natom()) &&
              (db < dconv * molecule.natom()) &&
              (conv_only_dens || (max_residual < 5.0 * dconv))
```

`dconv value` — E.g., `dconv 1e-5` — SCF convergence criterion (default 1e-4 atomic units). Suggest decreasing this to `1e-5` for geometry optimization or property calculations.

`canon` — Solves for canonical orbitals or eigenfunctions (default is localized orbitals except for atoms and diatomics).

`pm` — Selects use of the Pipek-Mezy localized orbitals (default).

`boys` — Selects use of the Boys localized orbitals.

`maxrotn value` — E.g., `maxrotn 0.1` — Used to restrict maximum rotation of orbitals (default `0.25`)

`maxiter value` — E.g., `maxiter 20` — The maximum number of iterations (default is `20`)

`maxsub value` — E.g., `maxsub 5` — The size of the iterative subspace (default is `5`). Sometimes it helps to make this larger.

`protocol valuelist` — E.g., `protocol 1e-4 1e-6 1e-8` — Sets the solution protocol. The default is ‘`1e-4 1e-6` which means solve first using a truncation threshold of `1e-4` (using $k = 6$) and with a threshold of `1e-6` (using $k = 8$).

`orbitalshift` — E.g., `orbitalshift 0.1` — Shifts the occupied orbitals down in energy by the given amount (default `0`). Is this working?

`k value` — E.g., `k 8` — Sets the wavelet order to a fixed value (mostly only used for testing)

`convonlydens` — Just test on the change in the density for convergence.

5.4 Geometry optimization

By default geometry optimization is performed using the BFGS Hessian update algorithm. The convergence test is on all of the 2-norm of the gradient, the change in the energy between iterations, and the maximum change in Cartesian coordinates (all in atomic units). The

For geometry optimization it is recommended to select `dconv 1e-5` to obtain more accurate gradients.

`gopt` — Requests optimization of the geometry

`gtol value` — E.g., `gtol 1e-4` — Sets the convergence threshold for the 2-norm of the gradient (default `1e-3`).

`gtest value` — E.g., `gtest 1e-4` — Sets the convergence threshold for the maximum change in Cartesian coordinates (default `1e-3` atomic units).

`gval value` — E.g., `gval 1e-6` — Sets the available precision in the energy (default is `1e-5` atomic units).

`gprec value` — E.g., `gtest 1e-6` — Sets the available precision in the gradient (default is `1e-5` atomic units).

`gmaxiter value` — E.g., `gmaxiter 100` — Sets the maximum number of geometry optimization iterations (default is 20).

`algopt value` — E.g., `algopt SR1` — Selects the quasi-Newton update method (default is). Options are `BFGS` (default) or `SR1` (not heavily tested). Case sensitive.

5.5 Properties

`derivatives` — Compute the derivates (default is false).

`dipole` — Compute the molecular dipole moment (default is false — why?).

`response` — TBD

`response_freq` — TBD

`response_axis` — TBD

`rconv` — TBD

`efield` — TBD

`efield_axis x y z` — TBD

`print_dipole_matels` — TBD

5.6 Plotting

Plots are generated to OpenDX files. In the `moldft` source directory are two useful files

- `vizit.net` — An OpenDX visual program that displays a molecule (from file `molecule.dx`) along with positive+negative isosurfaces (with adjustable value)for a scalar field read from a file.
- `moldx.py` — A Python program you can run with your *output* file as standard input to produce a `molecule.dx` file. It is important to use your output file since `moldft` will (by default) translate and rotate the molecular coordinates.

`plotmos lo hi` — E.g., `plotmos 10 12` — Plots the molecular orbitals in the given inclusive range (default is none). Orbitals are numbered from zero. Seems like this needs extending to accomodate unrestricted calculations.

`plotdens` — Plots the total electronic charge density and, if spin unrestricted, the spin density (default is off).

`plotcoul` — Plots the total (electronic + nuclear) electrostatic potential (default is off).

`plotnpt value` — E.g., `plotnpt 501` — Sets the number of plots used per dimension in the cube of points (default 101).

`plotcell xlo xhi ylo yhi zlo zhi` — E.g., `plotcell -10 10 -15 15 -10 5` — Sets the cell (in atomic units) used for plotting (default is the entire simulation cell).

5.7 Parallel execution

`loadbal vnucfac parts` — E.g., `loadbal 12 2` — Adjusts data/loadbalance when running in parallel with MPI. `vnucfac` (default 12) is extra weight associated with nuclear potential and `parts` (default 2) is the number of partitions (or subtrees) per node. SCF

`nio value` — E.g., `nio 10` — The number of MPI processes to use as I/O servers (default is 1)