Hybrid Many-Body Interaction QM/MM Code User Manual

Greg Beran, Kaushik Nanda, and Ali Sebetci.
Department of Chemistry, University of California, Riverside.
Last modified: September 2009

Contents

1 Introduction

HMBI is great! Write a real introduction here.

2 Compilation

2.1 Compilation instructions

First, obtain the source code from CVS.

```
% cvs co hmbi
```

This will create a directory 'hmbi' at your current path. This directory is refered to as $HMBI for the remainder of these instructions. HMBI has serial and parallel versions. The parallel version uses MPI, and the execution scripts are designed to use LAM/MPI.

To compile:

- Change to the $HMBI/src directory.
- To make the serial version ("hmbi.serial"), type:

  ```
  % make serial
  ```

  (or simply type `make`).
- To make both the serial and parallel versions ("hmbi.parallel"), type:

  ```
  % make all
  ```
- To make everything and install it:

  ```
  % make install
  ```

  Installing copies the binaries to the $HMBI/bin directory and creates a useful softlink in that directory.

To remove all compiled files for a fresh compilation, change to the $HMBI/src directory and type:

```
% make clean
```

Be sure to add the directory $HMBI/bin to your $PATH to enable you to run it.

Finally, you should modify $HMBI/bin/hmbi.run to set the EXE_PATH, MPIHOME, and LAMHOME variables. The script is set up to use Sun Grid Engine for parallel jobs. Alternatively, a "hosts.txt" file listing the name of each host can be used. This file can be modified for other batch systems like PBS.

2.2 Software structure

The structure of the code is as follows:

main.C	- the overall driver for the program
params.C	- a class to store the HMBI job parameters
cluster.C	- a class defining the full cluster, handles most of the real work.
dimer.C	- a class defining a dimer, which consists of 2 monomers
monomer.C	- a class defining a monomer, which consists of atoms
atom.C	- a class defining an atom type
multipole.C	- a class to store multipole expansions.
polarizability.C	- a class to store polarizabilities
dlf_interface.C	- interface to the DL-FIND geometry optimization package
opt.C	- a class handling geometry optimization (currently deactived, use DL-FIND)
vector.C	- a vector class
matrix.C	- a matrix class

3 Running HMBI

3.1 Serial version

First, prepare your input file (see further instructions below for information on input file format). Given an input file 'job.in' and a desired output file 'job.out', run the job by typing:

```
% hmbi job.in > job.out
```

3.2 Parallel version

After creating the input file, create a script file to submit the job to your cluster scheduler. On our cluster (which uses the Sun Grid Engine), the script looks something like:

```
-----------------------------------------
#!/bin/sh

#$ -cwd
#$ -pe lammpi 24

hmbi ice16.in > ice16.out
-----------------------------------------
```

This will run the job using the SGE parallel environment "lammpi" using 24 processors. You should contact your system administrator to identify the appropriate parallel environment on your own cluster. The $HMBI/bin/hmbi.run script will collect the information about how many processors to use and pass it to the HMBI code.

4 Input format

The HMBI input consists of a series of sections, which can be found in any order. A '$<keyword>' marks the start of each section, and '$end' marks the end of each. The necessary sections are:

1. **$comment :** Job title information (optional)
2. **$molecule :** The molecule specification
3. **$hmbi :** HMBI keywords
4. **$qchem :** Q-Chem keywords
5. Either **$tinker** (Tinker keywords) or **$orient** (Orient keywords) or **$qchem2** (secondary Q-Chem keywords)

Optional sections include:

6. **$unit_cell :** Periodic boundary condition specification
7. **$embedding_charges :** Specification of embedding charges (only works with Tinker MM, not Orient.

A brief description of each section follows.

4.1 Job Title/Comments: $comment

Free format; it can include any comments desired by the user.

4.2 Molecule Specification: $molecule

The format for cluster specification depends on the many-body treatment type, because some MM types require atom connetivity information. The molecular input specification requires only the minimum information required to define the system for each type of job. **Atomic positions are always in units of Angstroms.**

Regardless of the job type, the first line of the $molecule section contains the overall charge and spin state of the cluster.

```
<charge> <spin>
```

The beginning of each monomer is then indicated by a "`--`" line, followed by the monomer specification. For example, a neutral cluster with an overall singlet spin state would look like:

```
$molecule
0 1
--
<monomer 1>
--
<monomer 2>
--
etc.
$end
```

The details of specifying monomers for each particular job type are discussed below.

Note on spin states: As discussed below, the electronic spin state for each monomer must be specified. The algorithms for determining the spin state of dimers formed from these monomers are somewhat crude. Two singlet monomers form a singlet dimer. Likewise, a singlet monomer and a doublet monomer combine to form a doublet. But if two doublets combine, for example, one can obtain either a singlet or a triplet. In such cases, the code assigns it to either an overall singlet (if there is an even number of electrons) or doublet (if there is an odd number of electrons).

4.2.1 Monomer specification for the *ab initio* Force Field (AIFF) many-body model

For AIFF many-body treatments, the first line of each monomer specification contains:

```
<charge> <spin>  <ionization potential in a.u.>
```

The ionization potential (in hartrees) is used in the asymptotic DFT correction to improve the polarizability predictions. Experimental or theoretical ionization potentials can be used. The results are moderately sensitive to the particular value of the potential.

The next lines contain the monomer geometry in standard Cartesian XYZ format. Each line corresponds to the position of one atom:

```
<atomic symbol> <x> <y> <z>
```

A sample AIFF-type geometry specification for the $(H_2O)_3$ is given below.

```
$comment
  Sample AIFF-type geometry specification for water trimer.
$end

$molecule
0 1
--
0 1 0.4638
O     -1.201363      0.936340      0.045523
H     -1.875414      1.146057     -0.611433
H     -1.109807     -0.037296      0.018825
--
0 1 0.4638
O     -0.044745     -1.610729     -0.008393
H      0.043174     -2.205957      0.745669
H      0.754975     -1.048497      0.017871
--
0 1 0.4638
O      1.585266      0.661860      0.055793
H      2.142455      1.072502     -0.615606
H      0.711544      1.090261     -0.036259
$end
```

Note: for the AIFF generation, a set of local coordinates will be defined for each monomer. These are defined as: The first atom of each monomer is the origin of the local coordinate system of that monomer. The positive z-axis of the local coordinate system is defined as the axis from the first atom to the second atom. xz-plane of the local coordinate system is the plane of the first three atoms of the monomer. For diatomic monomers, a dummy third atom may be added.

4.2.2 Monomer specification for Quantum or EE-PA many-body models

Jobs in which a lower-level quantum mechanical method (such as Hartree-Fock) or the electrostatically embedded pairwise-additive approximation is used use a molecular specification utilize a nearly identical molecular specification to the AIFF many-body jobs. The only difference is that the ionization potential is omitted on the charge/spin line. For example, one water monomer would be specified as:

```
--
0 1
O     -1.201363      0.936340      0.045523
H     -1.875414      1.146057     -0.611433
H     -1.109807     -0.037296      0.018825
```

4.2.3 Monomer specification for `Tinker` force fields (such as Amoeba)

The `Tinker` software package requires atomic connectivity information in addition to the atomic coordinates. The first line of the specification defines the charge and spin. Subesequent lines define the atoms as:

```
    <counter> <atomic symbol> <x> <y> <z> <atom_type>
<connectivity>
```

The `counter` is a simple counter for the atoms whose number starts at 1 on each monomer. Fragment boundaries should not disrupt chemical bonds. As always, the `xyz` Cartesian coordinates are in Angstroms. The `atom_type` is the molecular mechanics atom type, and it corresponds to the atom type in the chosen force field. Force-field specification is described in Section 4.5. and `connectivity` is a list of up to 6 integers denoting the other atoms to which this atom is bonded. A sample specification for the water trimer is shown below.

```
$comment
  Sample Tinker-type geometry specification for water trimer.
  The atom number corresponds to the Amoeba force field.
$end

$molecule
0 1
--
0 1 0.4638
1  O     -1.201363      0.936340      0.045523  22  2  3
2  H     -1.875414      1.146057     -0.611433  23  1
3  H     -1.109807     -0.037296      0.018825  23  1
--
0 1 0.4638
1  O     -0.044745     -1.610729     -0.008393  22  2  3
2  H      0.043174     -2.205957      0.745669  23  1
3  H      0.754975     -1.048497      0.017871  23  1
--
0 1 0.4638
1  O      1.585266      0.661860      0.055793  22  2  3
2  H      2.142455      1.072502     -0.615606  23  1
3  H      0.711544      1.090261     -0.036259  23  1
$end
```

4.3 HMBI Keyword Specification: $hmbi

A series of keywords of the format "`<parameter> = <value>`". See Chapter 5 for more details on the specific keywords.

4.4 Q-Chem Keyword Specification: $qchem and/or $qchem2

A series of Q-Chem keywords in standard Q-Chem format. This section becomes the $rem section in Q-Chem inputs. Caution, not all keywords work well. For example, frozen core orbital keywords are messy, since the number of frozen core orbitals depends on whether we are looking at a monomer or dimer.

4.5 Tinker Keyword Specification: $tinker

A series of Tinker keywords in standard Tinker format. This section becomes the *.key file for Tinker. If this section is present, there should not be a $aiff section.

4.6 AIFF Keyword Specification: $aiff

A series of keywords to be used in the CamCasp/Dalton and Orient program packages. If this section is present, there should not be a $tinker section.

Keyword: CamCaspHome
Values: The home directory of the CamCasp program package.
Default: None
Example: /home/software/camcasp-5.2.00

Keyword: OrientBasisSet
Values: Sadlej, cc-pVXZ, or aug-cc-pVXZ (where X = D, T, or Q).
Default: Sadlej
Notes: This is the basis set used to compute the AIFF multipole moments and polarizabilities.

Keyword: DampingFactor
Values: Floating point number
Default: None
Notes: This prevents the "polarization catastrophe" at short ranges when computing self-consistent induction energies. Typical values range roughly 1.5-2.0. For water, 1.45 appears to work well. This parameter should be determined empirically by benchmarking against a set of fully quantum mechanical calculations.

A sample AIFF input variable section:

```
$aiff
CamCaspHome = /home/software/camcasp-5.2.00
OrientBasisSet = sadlej
DampingFactor = 1.45
$end
```

4.7 Periodic Boundary Conditions Specification: $unit_cell

There are two options for specifying the unit cell.

The default uses the lengths of each side of the unit cell (a, b, & c) and the angles between them (alpha, beta, and gamma), where alpha is the angle between axes b & c, beta is the angle between a & c, and gamma is the angle between a & b. Axis lengths are in Angstroms, and angles in degrees.

For example, here is the specification for formamide crystal, which has (a,b,c) = (3.5432, 8.9512, 6.9741), alpha = gamma = 90 degrees, and beta = 101.051 degrees.

```
$unit_cell
3.5432 8.9512 6.9741
90.0 101.051 90.0
$end
```

The other format, which is activated by setting the $hmbi keyword `READ_LATTICE_VECTORS = TRUE`, looks for a list of three lattice vectors (as row vectors) that define the unit cell. For example, if the three vectors are (7.569, 0, 0), (0, 5.366, 0), and (-4.745, 0, 8.552), you would input:

```
$unit_cell
 7.569 0.000 0.000
 0.000 5.366 0.000
-4.756 0.000 8.552
$end
```

4.8 $embedding_charges

Charges are listed, in a.u., by fragment, in the same order as in the $molecule section, with "–" separating each fragment. For example, in a system with two water molecules, you might have:

```
$embedding_charges
--
-0.7781
0.3891
0.3891
--
-0.7781
0.3891
0.3891
$end
```

4.9 Sample Input File

Here is a sample input file on a water trimer.

```
$comment
Water Trimer
$end

$molecule
0 1
--
0 1
1  O       -0.571270    2.322980   -0.003174  22  2  3
2  H       -0.541626    2.994645    0.657473  23  1
3  H        0.319714    1.986524   -0.080413  23  1
```

```
--
0 1
1  O        2.038411    1.261129   -0.102281  22  2  3
2  H        2.545757    1.390343   -0.886606  23  1
3  H        2.008500    0.316696    0.040400  23  1
--
0 1
1  O        1.831197   -1.536089    0.156570  22  2  3
2  H        2.127985   -1.971116    0.938635  23  1
3  H        0.922213   -1.803155    0.031130  23  1
$end

$hmbi
jobtype = energy
path_qm = rimp2
path_mm = amoeba
mm_code = tinker
periodic = false
iprint = 0
local_2_body = true
cutoff1 = 7.1
cutoff0 = 8.1
$end

$qchem
exchange = hf
basis = aug-cc-pvdz
aux_basis = rimp2-aug-cc-pvdz
correlation = rimp2
purecart = 11111
thresh = 14
scf_convergence = 8
mem_static = 200
mem_total = 2000
symmetry = false
$end

$tinker
# Force Field Selection
PARAMETERS        /home/software/tinker/params/amoeba.prm
# Precision
DIGITS            8
$end
```

5 HMBI Keywords

The keywords in the $hmbi section can be entered in any order, and with any capitalization or spacing. The only requirement is that they are listed in the format "parameter = value". An exception is for the `QM_PATH` and `MM_PATH` keywords, for which the listed paths are case sensitive and must adhere to standard Unix/Linux rules.

The section begins with $hmbi, and ends with $end. Blank lines in the section are allowed, but comment lines are not. A list of keywords follows:

Keyword: `JOBTYPE`
Values: `SP` (or `SINGLEPOINT` or `ENERGY`) = Do an HMBI energy calculation [default]
`FORCE` = Do an HMBI gradient calculation
`OPT` = Optimize the geometry
`HESSIAN` (or `FREQ` or `FREQUENCY`) = Do an HMBI frequency calculation. Not yet implemented.
`EXPAND` = Take the input geometry, and scale the distance of each molecule from the center of mass. A tool for developing potential energy surfaces that expand/shrink the cluster intermolecular spacings. Requires the keyword `EXPANSION_FACTOR` as well.
Notes: The `EXPAND` jobtype creates two new files, a cartesian xyz file, new_geom.xyz, that can be visualized with Molden or other software, and a new input file, which is a direct copy of the original except it has the new geometry.

Keyword: QM_PATH or MM_PATH
Values: any valid Unix-type path, pointing to the QM or MM jobs (case-sensitive).
Notes: The path should be local (expressed in terms of the directory from which the program is being executed).

Keyword: MM_CODE
Values: TINKER, ORIENT
Notes: Tells the software which type of MM output to read.

Keyword: IPRINT
Values: 0 or any positive integer
Default: 0
Notes: Higher values print out more output

Keyword: PERIODIC
Values: True or False
Default: False
Notes: If True, must specify the $unit_cell section.

Keyword: READ_LATTICE_VECTORS
Values: True or False
Default: False
Notes: If False, the code looks for axis lengths (a,b,c) and angles (alpha, beta, gamma) to define unit cell. If True, the code looks for three lattice vectors, v1, v2, & v3, each of the form (xi,yi,zi). See the manual section regarding the $unit_cell section for more details.

Keyword: COUNTERPOISE

Values: True or False
Default: False
Notes: If True, it invokes Q-Chem's Jobtype = BSSE and extracts Counterpoise corrected energies. Warning: Q-Chem's BSSE jobtype is not fully compatible with all Q-Chem methods (e.g. dual-basis SCF).

Keyword: EMBEDDING_CHARGES
Values: True or False
Default: False
Notes: If True, it uses embedding charges to lessen the importance of many-body terms. The input file section $embedding_charges must be used in conjuction with this keyowrd.

Keyword: LOCAL_2_BODY
Values: True of False
Default: False
Notes: Turns on smooth local truncation of pairwise interactions. Truncated 2-body terms are treated at the MM level. Set in combination with CUTOFF1 and CUTOFF0.

Keyword: CUTOFF1
Values: Any positive real number
Default: False
Notes: Cutoff (in Angstroms) below which all 2-body interactions are treated at the QM level.

Keyword: CUTOFF0
Values: Any positive real number
Default: CUTOFF1 + 1.0
Notes: Cutoff (in Angstroms) above which all 2-body interactions are treated at the MM level. Between CUTOFF1 and CUTOFF0, QM 2-body interactions are smoothly damped.

Keyword: NEGLECT_MANY_BODY
Values: True of False
Default: False Notes: Turns off MM many-body terms. Purely for debugging or testing.

Keyword: EXPANSION_FACTOR
Values: Any positive real number
Default: 1.0
Notes: Scale factor for expanding or contracting the cluster geometry, in coordination with JOBTYPE = EXPAND. Values > 1 expand, and Values < 1 shrink it.

Keyword: MAX_OPT_CYCLES
Values: Positive integer
Default: 100
Notes: Maximum number of cycles to use in geometry optimizations. Default is 100 at present.