Hybrid Many-Body Interaction QM/MM Code User Manual

Greg Beran, Kaushik Nanda, Ali Sebetci, and Yonaton Heit.
Department of Chemistry, University of California, Riverside.
Last modified: March 2016

Contents

1 Introduction **2**

2 Compilation **2**
2.1 Compilation instructions . . . 2
2.2 Software structure . . . 3

3 Running HMBI **3**
3.1 Serial version . . . 3
3.2 Parallel version . . . 4

4 Input format **4**
4.1 Job Title/Comments: $comment . . . 6
4.2 Molecule Specification: $molecule . . . 6
4.2.1 Monomer specification for the *ab initio* Force Field (AIFF) many-body model 6
4.2.2 Monomer specification for Quantum or EE-PA many-body models . . . 8
4.2.3 Monomer specification for `Tinker` force fields (such as Amoeba) . . . 9
4.3 HMBI Keyword Specification: $hmbi . . . 10
4.4 Q-Chem Keyword Specification: $qchem and/or $qchem2 . . . 10
4.5 Molpro Keyword Specification: $molpro and $molpro_inst . . . 10
4.5.1 Molpro CBS limit . . . 10
4.5.2 Molpro CCSD(T) Correction . . . 12
4.6 Tinker Keyword Specification: $tinker . . . 12
4.7 AIFF Keyword Specification: $aiff . . . 12
4.8 Periodic Boundary Conditions Specification: $unit_cell . . . 13
4.9 $embedding_charges . . . 13
4.10 Quantum Espresso . . . 14
4.10.1 Quantum Espresso optimizations . . . 14
4.10.2 Quantum Espresso frequency calculations . . . 19
4.11 Sample Input File . . . 20

5 HMBI Keywords **21**
5.1 General . . . 21
5.2 AIFF . . . 25
5.3 Optimization And Forces . . . 28
5.4 Frequency . . . 29
5.5 Lattice Dynamics . . . 30
5.6 Quasi-Harmonic Approximation . . . 31
5.7 Symmetry . . . 31
5.8 Quantum Espresso . . . 32

6 Lattice Dynamics **33**
6.1 Introduction . . . 33
6.2 Usage . . . 33
6.3 Print Thermodynamic Properties with Temperature . . . 34

7 Quasi-Harmonic Approximation **34**
7.1 Introduction . . . 34
7.2 Derive Grüneisen Parameters and Reference Frequencies . . . 35
7.2.1 Determine the structure, frequencies, and Grüneisen parameters at the same time. . . . 35
7.2.2 Determine Grüneisen Parameters Separately from Frequencies . . . 35
7.2.3 Incorporating lattice relaxation into the Quasiharmonic Approximation . . . 37
7.3 Use Grüneisen parameters to Thermal Expand . . . 39

8 The "Tiered" Quasi-Harmonic Approximation **39**
8.1 Introduction . . . 39
8.2 Calculating the Electronic Internal Energy Curve (U_{el}) . . . 40
8.2.1 Obtaining the Reference geometry . . . 40
8.2.2 Generating the E(V) curve . . . 40
8.2.3 Optional: Performing Single-point Energy Corrections . . . 41
8.2.4 Fitting the E(V) curve in MATLAB . . . 42
8.3 Calculating the Helmholtz Free Energy Curve ($F_{vib}(T)$) . . . 42
8.4 Calculating the Gibbs Free Energy Curve Neglecting Pressure ($G(T)$) . . . 42
8.5 Calculating the Gibbs Free Energy Curve Including Pressure ($G(T, P)$) . . . 49

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 referred 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
supercell.C	- a class defining the supercell for lattice dynamics
quasiharmonic.C	- a class used for the quasi-harmonic approximation
dlf_interface.C	- interface to the DL-FIND geometry optimization package
opt.C	- a class handling geometry optimization (currently deactivated, 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
#$ -V
#$ -pe mpi 24

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

This will run the job using the SGE parallel environment "mpi" 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.

Alternatively running the command:

```
hmbi ice16.in 24 > ice16.out
```

will override the number of parallel processors requested in the queue script and instead enforce the amount of processors specified after the input filename. BE VERY CAREFUL WITH THIS!

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. **$hmbi :** HMBI keywords
4. Section(s) for one of the following qm methods:
   - **$qchem :** (Q-Chem keywords)
   - molpro has few sections
     (a) **$molpro:** (Molpro basis set keywords)
     (b) **$molpro_inst** (Molpro method keywords)

     Optional molpro sections include:

(c) For the CBS limit extrapolation

i. **$molpro_inst_HF** (Molpro Hartree-Fock method keywords for the CBS limit extrapolation)

ii. **$molpro_CBS:** (Molpro higher basis set keywords for the CBS limit extrapolation)

(d) For the CCSD(T) correction:

i. **$molpro_ccsdt_basis** (Molpro basis used for the CCSD(T) correction)

ii. **$molpro_ccsdt_mp2** (Molpro MP2 method used for CCSD(T) correction)

iii. **$molpro_ccsdt_inst** (Molpro CCSD(T) method used for the CCSD(T) correction)

- **$qe_species :** (Quantum Espresso atom types)
- **$qe_supercell :** (Quantum Espresso supercell grid)
- **$kpoints :** (Quantum Espresso kpoint grid)

5. Either section(s) for one of the following mm methods:
   - **$tinker** (Tinker keywords)
   - AIFF section:
     (a) **$aiff** (Orient keywords)
     (b) **$damping_factors_aiff**(Giving damping factors.)
   - **$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.)

8. **$reciprocal_space_points :** Define the k-points in a Monkhorst-Pack scheme for lattice dynamics.

9. **$thermal_parameters :** Specification the range of temperatures which thermodynamic properties are printed when performing lattice dynamics.

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 connectivity 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.> <monomer string>
```

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 monomer string is a unique identifier for each monomer. This identifier is used to determine the damping factor for each monomer pair as defined by $damping_factors_aiff

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 H2O
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 H2O
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 H2O
O      1.585266      0.661860      0.055793
H      2.142455      1.072502     -0.615606
H      0.711544      1.090261     -0.036259
$end

$damping_factors_aiff
H2O
H2O H2O 1.45
$end
```

The $damping_factors_aiff section is use to define the damping function between every pair of monomers. This damping function is used to avoid the polarization catastrophe at short ranges when computing self-consistent induction energies. The first line in this section includes every monomer type "monomer string". The next lines, the strings contain every combination of monomer pairs followed by the damping function for that pair.

```
<monomer string 1>  <monomer string 2> <damping function>
```

It is important that the number of monomer types is stated in the N_MONOMER_TYPES keyword in the $aiff section and that this number match the number of strings in the first line of the $damping_factors_aiff. Below is an example of a system with two different monomer types which allow for different damping functions for monomer pairs.
Note: for the AIFF generation, a set of local coordinates will be defined for each symmetrical unique monomer using their center of mass coordinates.

```
$comment
  Sample AIFF-type geometry specification for the phase IV
  ammonium nitrate unit cell
$end

$molecule
0 1
--
 1 1 0.8906 NH4
 N     4.309    1.359    4.529
 H     3.473    1.359    4.004
 H     4.309    0.550    5.102
 H     5.145    1.359    4.004
 H     4.309    2.169    5.102
--
-1 1 0.1447 NO3
 N     1.436    1.359    2.504
 O     2.494    1.359    1.894
 O     1.436    1.359    3.770
 O     0.378    1.359    1.894
--
-1 1 0.1447 NO3
 N     4.309    4.079    2.438
 O     5.367    4.079    3.048
 O     4.309    4.079    1.172
 O     3.251    4.079    3.048
 1 1 0.8906 NH4
 N     1.436    4.079    0.413
 H     2.272    4.079    0.938
 H     0.600    4.079    0.938
 H     1.436    3.269   -0.160
 H     1.436    4.888   -0.160
$end

$damping_factors_aiff
NH4 NO3
NH4 NH4 2.50
NO3 NO3 3.80
NH4 NO3 1.29
$end
```

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. Subsequent 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 4.6. 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 Molpro Keyword Specification: $molpro and $molpro_inst

A series of Molpro keywords in standard Molpro format. The basis set portion is placed into the $molpro section and the method (Hartree-Fock/MP2) is place into the $molpro_inst section in the same format as an Molpro input file.

A sample Molpro input variable sections:

```
$molpro
basis={
set,orbital
default,avtz
set,jkfit
default,avtz/jkfit
set,mp2fit
default,avtz/mp2fit
}
$end

$molpro_inst
{df-hf,basis=jkfit;}
{df-mp2,basis=mp2fit;}
$end
```

4.5.1 Molpro CBS limit

HMBI can do complete basis set limit extrapolation using Molpro for energy, forces, optimizations, and hessian calculations. This extrapolation used two basis sets of different ζ-levels to extrapolate energy to the complete basis set (Eq 1-3). Typically triple and quadrapole ζ-levels are used. Set

the keyword **CBS** to true to do a CBS limit calculation.

$$E_{HF}^{\infty} = E_{HF}^{large} + \frac{E_{HF}^{Y} - E_{HF}^{X}}{e^{1.54(y-x)} - 1} \tag{1}$$

$$E_{corr}^{\infty} = \frac{Y^3 E_{corr}^{Y} - X^3 E_{corr}^{X}}{Y^3 - X^3} \tag{2}$$

$$E_{MP2}^{\infty} = E_{HF}^{\infty} + E_{corr}^{\infty} \tag{3}$$

X = small basis ζ-level
Y = large basis ζ-level

The lower basis set keywords are placed into the $molpro section. The higher basis set keywords are placed to $molpro_CBS section. The Hartree-Fock and MP2 keywords are separated into $molpro_inst_HF and $molpro_inst respectively. All sections are in the same format as an Molpro input file. Note that the ζ-level of the two basis sets must match the **CBS_BASIS1** (smaller basis set) and the **CBS_BASIS2** (larger basis set) keywords in the $HMBI section.

```
$molpro
basis={
set,orbital
default,avtz
set,jkfit
default,avtz/jkfit
set,mp2fit
default,avtz/mp2fit
}
$end

$molpro_CBS
basis={
set,orbital
default,avqz
set,jkfit
default,avqz/jkfit
set,mp2fit
default,avqz/mp2fit
}
$end

$molpro_inst_HF
{df-hf,basis=jkfit;}
$end

$molpro_inst
{df-mp2,basis=mp2fit;}
$end
```

4.5.2 Molpro CCSD(T) Correction

HMBI can perform the CCSD(T) correction using Molpro for energy, forces, and optimizations calculations. While Hessian calculations using the CCSD(T) correction is implemented, numeral Hessians needed to computed using numerical gradients in Molpro2012. This Hessians have been found to be too computationally noisy for two-body Hessians. This correction takes the energy difference of MP2 and CCSD(T) in a small basis set (typically double ζ) and adds it to the energy in a large basis set. (EQ 4 and 5).

Set the keyword **CCSDT_CORRECTION** in the $HMBI section to true in order to do this correction.

$$\Delta^{CCSD(T)} = E^{small}_{CCSD(T)} + E^{small}_{MP2} \tag{4}$$

$$E^{large}_{CCSD(T)} \approx E^{large}_{MP2} + \Delta^{CCSD(T)} \tag{5}$$

In addition to the molpro sections or the sections used for the CBS molpro sections, three additional sections are used to perform the CCSD(T) correction. The keywords for the basis used for the CCSD(T) correction are placed into the $molpro_ccsdt_basis section. The keywords for the MP2 method are placed into $molpro_ccsdt_mp2 section. The keywords for the CCSD(T) method are placed into the $molpro_ccsdt_inst.

A sample input for a Molpro CCSD(T) correction is below. These do not include the sections necessary for a molpro job or a CBS limit extrapolation molpro job. While the Molpro and CBS Molpro example sections used density-fitting, Molpro2012 does not have density-fitting is not available for CCSD(T).

```
$molpro_ccsdt_basis
basis=avdz
$end

$molpro_ccsdt_mp2
hf
mp2
$end

$molpro_ccsdt_inst
ccsd(t)
$end
```

4.6 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.

4.7 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: N_MONOMER_TYPES
Values: positive integers
Default: 1
Notes: Number of monomer types.

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.

A sample AIFF input variable section:

```
$aiff
OrientBasisSet = sadlej
N_MONOMER_TYPES = 1
$end
```

4.8 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. This format does not have force and optimization implemented 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
```

4.9 $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
```

4.10 Quantum Espresso

Quantum Espresso (QE) employs periodic boundary conditions during its calculations which makes it incompatible with the energy breakdown HMBI typically employs. While it is still possible to use QE in HMBI, the typical computational savings you would see from running only monomer and dimer jobs in parallel will be absent as you will always be running the full crystal in each calculation.

4.10.1 Quantum Espresso optimizations

It is recommended that most optimizations be performed using the native QE optimizer (since it will be a little faster). Below is an example input file for an optimization performed in Quantum Espresso:

```
&CONTROL
  calculation = 'vc-relax',
  restart_mode = 'from_scratch',
  prefix = 'fullQE',
  disk_io = 'none',
  verbosity = 'high',
  etot_conv_thr = 2.0D-6,
  forc_conv_thr = 6.0D-4,
  outdir="./",
  nstep = 500,
/
&SYSTEM
  ibrav = 0,
  nat = 56,
  ntyp = 3,
  ecutwfc = 50.000000,
  ecutrho = 500.000000,
  vdw_corr = 'XDM',
  xdm_a1 = 0.6512,
  xdm_a2 = 1.4633,
/
&ELECTRONS
  electron_maxstep = 1500,
  conv_thr = 1.D-8,
  scf_must_converge = .TRUE.,
  mixing_beta = 0.5D0,
```

```
/
&IONS
/
&CELL
press = 0.D0,
/
ATOMIC_SPECIES
 H 1.00800 H.b86bpbe.UPF
 C 12.0100 C.b86bpbe.UPF
 O 16.0000 O.b86bpbe.UPF

ATOMIC_POSITIONS crystal
O     0.530162    0.840920    0.186651
H     0.459108    0.900130    0.258761
O     0.846839    0.492406    0.357912
H     0.882573    0.428621    0.486211
C     0.513499    0.710360    0.560818
H     0.425686    0.766789    0.610428
C     0.565131    0.606111    0.711257
H     0.517185    0.582054    0.879677
C     0.676984    0.531267    0.650798
H     0.715686    0.449865    0.770535
C     0.736955    0.562029    0.432573
C     0.687374    0.666318    0.279014
H     0.734430    0.688188    0.108769
C     0.575891    0.740155    0.344294
O     0.469838    0.159080    0.686651
H     0.540892    0.099870    0.758761
O     0.153161    0.507594    0.857912
H     0.117427    0.571379    0.986211
C     0.486501    0.289640    1.060818
H     0.574314    0.233211    1.110428
C     0.434869    0.393889    1.211257
H     0.482815    0.417946    1.379676
C     0.323016    0.468733    1.150798
H     0.284314    0.550134    1.270535
C     0.263045    0.437971    0.932573
C     0.312626    0.333681    0.779014
H     0.265570    0.311812    0.608769
C     0.424109    0.259845    0.844294
O     0.969838    0.340920    0.686651
H     1.040892    0.400130    0.758761
O     0.653161   -0.007594    0.857912
H     0.617427   -0.071379    0.986211
C     0.986501    0.210360    1.060818
H     1.074314    0.266789    1.110428
C     0.934869    0.106111    1.211257
H     0.982815    0.082054    1.379676
```

```
C     0.823016    0.031267    1.150798
H     0.784314   -0.050134    1.270535
C     0.763045    0.062029    0.932573
C     0.812626    0.166318    0.779014
H     0.765570    0.188188    0.608769
C     0.924109    0.240155    0.844294
O     0.030162    0.659080    0.186651
H    -0.040892    0.599870    0.258761
O     0.346839    1.007594    0.357912
H     0.382573    1.071379    0.486211
C     0.013499    0.789640    0.560818
H    -0.074314    0.733211    0.610428
C     0.065131    0.893889    0.711257
H     0.017185    0.917946    0.879677
C     0.176984    0.968733    0.650798
H     0.215686    1.050134    0.770535
C     0.236955    0.937971    0.432573
C     0.187374    0.833681    0.279014
H     0.234430    0.811812    0.108769
C     0.075891    0.759845    0.344294

K_POINTS automatic
1 1 3 1 1 1

CELL_PARAMETERS angstrom
10.372435976000   0.000000000000   0.000000000000
-0.000000000001   9.324338554000   0.000000000000
-0.000000000001   -0.000000000001   5.582421438000
```

For troubleshooting problems with the QE optimizations please refer to the following Quantum Espresso websites:

```
https://www.quantum-espresso.org/
https://www.quantum-espresso.org/Doc/INPUT_PW.html
```

If you would prefer to use the HMBI optimizer then below is an example of how to perform the same optimization in HMBI.

```
$comment
Resorcinol form Alpha (RESORA03) optimized using b86bpbeXDM
$end

$hmbi
jobtype = opt
neglect_many_body = true
qm_path = qe
mm_path = amoeba
qm_code = qe
mm_code = qchem
```

```
periodic = false
full_qm_only = true
space_symmetry = true
lattice_symmetry = true
qe_basis = 50
qe_basis_mult = 10
disp_type = xdm-b86bpbe
disp_correction = true
max_opt_cycles = 500
pressure = 0
$end

$molecule
0 1
--
0 1
O       5.499071    7.841023    1.041966
H       4.762065    8.393120    1.444515
O       8.783780    4.591362    1.998015
H       9.154428    3.996607    2.714232
C       5.326232    6.623639    3.130721
H       4.415402    7.149804    3.407667
C       5.861780    5.651584    3.970536
H       5.364468    5.427265    4.910725
C       7.021970    4.953713    3.633028
H       7.423407    4.194698    4.301451
C       7.644017    5.240549    2.414806
C       7.129745    6.212979    1.557574
H       7.617828    6.416897    0.607196
C       5.973388    6.901455    1.921996
--
0 1
O       4.873364    1.483316    3.833177
H       5.610371    0.931219    4.235726
O       1.588656    4.732976    4.789226
H       1.218008    5.327732    5.505443
C       5.046204    2.700700    5.921931
H       5.957034    2.174535    6.198878
C       4.510656    3.672755    6.761747
H       5.007968    3.897074    7.701935
C       3.350466    4.370626    6.424239
H       2.949029    5.129640    7.092662
C       2.728419    4.083789    5.206016
C       3.242691    3.111359    4.348784
H       2.754608    2.907441    3.398406
C       4.399048    2.422883    4.713207
--
0 1
```

```
O      10.059582    3.178854    3.833177
H      10.796589    3.730950    4.235726
O       6.774874   -0.070807    4.789226
H       6.404226   -0.665562    5.505443
C      10.232422    1.961470    5.921931
H      11.143252    2.487634    6.198878
C       9.696874    0.989415    6.761747
H      10.194186    0.765096    7.701935
C       8.536684    0.291544    6.424239
H       8.135247   -0.467471    7.092662
C       7.914637    0.578380    5.206016
C       8.428909    1.550810    4.348784
H       7.940826    1.754728    3.398406
C       9.585266    2.239286    4.713207
--
0 1
O       0.312854    6.145485    1.041966
H      -0.424153    5.593388    1.444515
O       3.597562    9.395146    1.998015
H       3.968210    9.989901    2.714232
C       0.140014    7.362869    3.130721
H      -0.770816    6.836704    3.407667
C       0.675562    8.334924    3.970536
H       0.178250    8.559243    4.910725
C       1.835752    9.032795    3.633028
H       2.237189    9.791809    4.301451
C       2.457799    8.745959    2.414806
C       1.943527    7.773528    1.557574
H       2.431610    7.569610    0.607196
C       0.787170    7.085052    1.921996
$end

$unit_cell
10.372435976 9.3243385540000006 5.5824214379999999
90.0 90.0 90.0
$end

$qe_species
 H 1.00800 H.b86bpbe.UPF
 C 12.0100 C.b86bpbe.UPF
 O 16.0000 O.b86bpbe.UPF
$end

$kpoints
1 1 3 1 1 1
$end
```

The following commands in the $hmbi section are essential for using a Quantum Espresso job:

```
qm_code = qe
neglect_many_body = true
periodic = false
full_qm_only = true
```

Optional commands that you may find useful:

```
qe_basis = 50
qe_basis_mult = 10
disp_type = xdm-b86bpbe
disp_correction = true
```

Note: Pay extra attention to which type of dispersion you are applying. If the dispersion type is XDM then you must also specify which exchange functional you are using otherwise HMBI will assume it is PBE.

Other sections you must initialize:

```
$qe_species
 H 1.00800 H.b86bpbe.UPF
 C 12.0100 C.b86bpbe.UPF
 O 16.0000 O.b86bpbe.UPF
$end

$kpoints
1 1 3 1 1 1
$end
```

The $qe_species section must contain the pseudopotential filename for each atom type you wish to use. The $kpoints section contains the kpoint grid to be used.

4.10.2 Quantum Espresso frequency calculations

Currently the only established method for calculating frequencies when using QE in HMBi is to use Phonopy. The use of this external program ensures that the dispersion correction you employ in the optimizations will also be used in the calculations of the frequencies. To calculate frequencies using QE/Phonopy set the following in the $hmbi section:

```
jobtype = freq
qm_code = qe
neglect_many_body = true
periodic = false
full_qm_only = true
```

HMBI will automatically generate the files necessary for running a Phonopy calculation and run these files. Since Phonopy generates the frequencies via finite difference this can take some time to calculate. Be sure to also set the $qe_species and $kpoints section! If you wish to employ lattice dynamics then also set the following section:

```
$qe_supercell
2 2 2
$end
```

This will generate a supercell of dimension 2x2x2.

4.11 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:

5.1 General

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.
`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.
Has not been tested with SYMMETRY for `EXPANSION_FACTOR`.
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
Values: Any valid Unix-type path, pointing to the QM jobs (case-sensitive).
Default: qm
Notes: The path should be local (expressed in terms of the directory from which the program is being executed).

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

Keyword: HESSIAN_FILES_PATH
Values: Any valid Unix-type path, pointing to the Hessian jobs (case-sensitive).
Default: hessian_path
Notes: The path should be local (expressed in terms of the directory from which the program is being executed).

Keyword: QM_CODE
Values: QCHEM, MOLPRO, G09, DALTON, QE, ORCA
Default: QCHEM
Notes: Tells the software with type of QM ouput to read.

Keyword: MM_CODE
Values: TINKER, AIFF, QCHEM, EE-PA, ORIENT, CHELPG, HIRSHFELD, CRYSTAL09
Default: AIFF
Notes: Tells the software which type of MM output to read.

Keyword: IPRINT or PRINT_LEVEL
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 and Q-chem is used for the QM, 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 conjunction with this keyword.

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 or False
Default: False Notes: Turns off MM many-body terms. Purely for debugging or testing.

Keyword: BUILD_FORCE_FIELD_ONLY
Values: True or False
Default: False
Notes: Turns off QM terms. Purely for debugging or testing.

Keyword: FULL_QM_ONLY
Values: True or False
Default: False
Notes: Runs this software using only the QM method on the crystal. Only implemented for Quantum Espresso

Keyword: FULL_MM_ONLY
Values: True or False
Default: False
Notes: Runs this software using MM method on the crystal. Only tested for TINKER

Keyword: ANALYZE_ONLY
Value: True or False
Default: False
Notes: All QM and MM jobs have be run. Determine final value for the job.

Keyword: CREATE_JOBS_ONLY
Value: True or False
Default: False
Notes: Create input jobs but do not run them.

Keyword: OLD_DIMER_SYMM or OLDDIMERSYMM
Value: True or False
Default: False

Notes: Use the old symmetry labels for the dimers.

Keyword: PRESSURE
Value: Any real number
Notes: Pressure in GPa
Only printed when PRESSURE is specified

Keyword: TEMPERATURE
Value: 0 or any positive real number
Default: 298
Notes: Can only perform on a Frequency or quasi-harmonic calculation.
In Kelvin.

Keyword: CBS
Value: True or False
Default: False
Notes: Complete basis set limit calculation. Can be used for energy, force, optimization, or hessian calculations. Molpro QM only.
Must include Keyword CBS_BASIS1 and CBS_BASIS2 keywords and the $molpro_CBS section in input file.

Keyword: CBS_BASIS1
Value: DZ, TZ, or QZ
Default: TZ
Notes: The smaller basis set in a complete basis set limit calculation.

Keyword: CBS_BASIS2
Value: DZ, TZ, or QZ
Default: QZ
Notes: The larger basis set in a complete basis set limit calculation.

Keyword: CCSDT_CORRECTION
Value: True or False
Default: False
Notes: CCSD(T) correction for energy, force, optimization, or hessian calculations. Molpro QM only.
Must include $molpro_ccsdt_inst and $molpro_ccsdt_mp2 sections in input file.

Keyword: CHANGE_VOLUME
Values: All real number
Default: 0.0
Notes: Expand or contract (negative to contract) volume of the unit cell isotropically. If SPACE_SYMMETRY is true, symmetry will be preserved.

Keyword: CHANGE_A
Values: All real number
Default: 0.0
Notes: Increase or decrease lattice parameter a. If SPACE_SYMMETRY is true, symmetry will be

preserved.

Keyword: CHANGE_B
Values: All real number
Default: 0.0
Notes: Increase or decrease lattice parameter c. If SPACE_SYMMETRY is true, symmetry will be preserved.

Keyword: CHANGE_C
Values: All real number
Default: 0.0
Notes: Increase or decrease lattice parameter c. If SPACE_SYMMETRY is true, symmetry will be preserved.

Keyword: CHANGE_ALPHA
Values: All real number
Default: 0.0
Notes: Increase or decrease lattice parameter alpha. If SPACE_SYMMETRY is true, symmetry will be preserved.

Keyword: CHANGE_BETA
Values: All real number
Default: 0.0
Notes: Increase or decrease lattice parameter beta. If SPACE_SYMMETRY is true, symmetry will be preserved.

Keyword: CHANGE_GAMMA
Values: All real number
Default: 0.0
Notes: Increase or decrease lattice parameter gamma. If SPACE_SYMMETRY is true, symmetry will be preserved.

Keyword: MEMORY
Values: Any positive integer
Default: 200
Notes: Memory (in megawords) used for a QM job. Molpro only

Keyword: COUPLE_GRADIENTS
Values: True or False
Default: False
Notes: Couples the gradients of the lattice parameters and the atoms.
Currently only implemented for Quantum Espresso.

5.2 AIFF

Keyword: DO_AIFF_ELECTROSTATICS
Values: True or False

Default: True
Notes: Include electrostatics in long-range two-body and many-body calculation.

Keyword: DO_AIFF_INDUCTION
Values: True or False
Default: True
Notes: Include induction in electrostatic calculation.
DO_AIFF_ELECTROSTATICS must be set to true.

Keyword: DO_AIFF_2BODY_DISPERSION
Values: True or False
Default: True
Notes: Do long-range 2-body dispersion

Keyword: DO_AIFF_3BODY_DISPERSION
Values: True or False
Default: True
Notes: Do long-range 3-body dispersion

Keyword: EWALD_TINFOIL_BOUNDARY
Values: True or False
Default: True
Notes: ???

Keyword: POLARIZATION_CUTOFF
Values: positive real numbers
Default: 15.0
Notes: ???

Keyword: TWO_BODY_DISPERSION_CUTOFF
Values: positive real numbers
Default: 20.0
Notes: ???

Keyword: THREE_BODY_DISPERSION_CUTOFF
Values: positive real numbers
Default: 10.0
Notes: ???

Keyword: CORRECT_MP2_DISPERSION
Values: True or False
Default: false
Notes: ???

Keyword: MAX_POLARIZATION_CYCLES
Values: positive integers
Default: 100
Notes: ???

Keyword: INDUCTION_CONVERGENCE
Values: positive integers
Default: 5
Notes: ???

Keyword: INDUCTION_GRADCONVERGENCE
Values: positive integers
Default: 5
Notes: ???

Keyword: INDUCTION_ITER_SCALING
Values: ???
Default: 1.0
Notes: ???

Keyword: PRECONVERGE_UNIT_CELL_MOMENTS
Values: True or False
Default: False
Notes: ???

Keyword: EWALD_KAPPA
Values: ???
Default: -1
Notes: ???

Keyword: EWALD_ACCURACY
Values: ???
Default: 15
Notes: ???

Keyword: RECIP_CUTOFFX
Values: ???
Default: -1
Notes: ???

Keyword: RECIP_CUTOFFY
Values: ???
Default: -1
Notes: ???

Keyword: RECIP_CUTOFFZ
Values: ???
Default: -1
Notes: ???

Keyword: DIREC_CUTOFFX
Values: ???

Default: -1
Notes: ???

Keyword: DIREC_CUTOFFY
Values: ???
Default: -1
Notes: ???

Keyword: DIREC_CUTOFFZ
Values: ???
Default: -1
Notes: ???

5.3 Optimization And Forces

Keyword: OPTIMIZER
Values: DLFIND, KNITRO, CONJUGATEGRADIENT or CG, STEEPESTDESCENT or SD or STEEPESTDESCENT, LBFGS or LBFGS
Default: DLFIND
Notes: Needed only when JOB_TYPE ŌPT. If you wish to use only the local optimizer then set CG, SD, or LBFGS. Recommend using DLFIND.

Keyword: MAX_OPT_CYCLES
Values: Positive integer
Default: 150
Notes: Maximum number of cycles to use in geometry optimizations.

Keyword: DO_FREQ_AFTER_OPT
Values: True or False
Default: False
Notes: Perform frequency calculation once optimization is complete.

Keyword: FREEZE_UNITCELLPARAMS
Values: True or False
Default: False
Notes: Freezes lattice parameters during optimizations.
Optimizes unit cell nuclear coordinates only.

Keyword: FREEZE_ATOM_COORDINATES
Values: True or False
Default: False
Notes: Freezes atom coordinates during optimizations.
Optimizes unit cell parameters only.

Keyword: UNFREEZE_LATTICE_PARAMS_AFTER
Values: 0 or positive integers

Default: 0
Notes: Freezes lattice parameters until a given number of optimization steps.

Keyword: FDIFF_GRAD
Values: True or False
Default: False
Notes: Determines gradient using finite difference. For debugging and testing only.

Keyword: STEPSIZE
Values: Postitive doubles from 0.1 - 1.0
Default: 0.65
Notes: Needed only when JOB_TYPE = OPT and using the local optimizer. Determines step length when doing line-search on a rejected step.

5.4 Frequency

Keyword: DEUTERATED
Values: True or False
Default: False
Notes: Hydrogen atoms are given the mass of deuterium atoms when determining vibrational frequencies.

Keyword: MOLECULE
Values: True or False
Default: False
Notes: Frequency of a molecule instead of for the Crystal. Necessary only for Quantum Espresso

Keyword: NO_FREQ_DISP or NO_FREQ_DISPLACEMENT
Values: True or False
Default: False
Notes: Does not displpace molecule if negative frequencies exist

Keyword: ARE_FORCES_AVAILABLE
Values: True or False
Default: False
Notes: The necessary force calculations were already performed. This is often the done doing the final step of an optimization.
This keyword set to true if ANALYZE_ONLY is true.

Keyword: ARE_HESSIANS_AVAILABLE
Values: True or False
Default: False
Notes: The necessary hessian calculations were already performed. This keyword set to true if ANALYZE_ONLY is true.

Keyword: SINGLE_FILE_MONOMER_HESS
Values: True or False

Default: False
Notes: QM monomer Hessians calculated in single job. If false, monomer Hessians are determined by finite difference in multiple jobs.
Molpro only. Q-chem always calculates for monomer hessian with a single job.

Keyword: FDIFF_HESS
Values: True or False
Default: False
Notes: Determine Frequency using finite difference. For debugging and testing only.
Incompatible with SPACE_GROUP equal to true.

5.5 Lattice Dynamics

Keyword: SUPERCELL_JOB
Values: True or False
Default: False
Notes: Perform supercell Hessian calculations to do lattice dynamics.
Must include $reciprocal_space_points section in input.

Keyword: SUPERCELL_SIZE_A
Values: positive integer
Default: 3
Notes: Length the unit cell is extended along lattice length a.

Keyword: SUPERCELL_SIZE_B
Values: positive integer
Default: 3
Notes: Length the unit cell is extended along lattice length b.

Keyword: SUPERCELL_SIZE_B
Values: positive integer
Default: 3
Notes: Length the unit cell is extended along lattice length c.

Keyword: SUPERCELL_ANALYZE_ONLY
Values: True or False
Default: False
Notes: Supercell Hessian calculation has already be performed.

Keyword: THERMAL_FOR_MULTIPLE_TEMPERATURES
Values: True or False
Default: False
Notes: Print thermodynamic properties for a range of temperature.
The range is in $thermal_parameters section. If this section is not included. The range is 0 to 298 K with intervals of 1 K

5.6 Quasi-Harmonic Approximation

Keyword: QUASIHARMONIC
Values: True or False
Default: False
Notes: Performs the quasi-harmonic approximation

Keyword: ARE_QHA_AVAILABLE
Values: True or False
Default: False
Notes: If false, frequencies for the reference cell and the grüneisen is determined. The frequencies are printed in the .freq file.
If true, all frequencies necessary are in the .freq file.

Keyword: SAVE_QHA
Values: True or False
Default: True [RECOMMENDED]
Notes: Place the jobs for each quasi-harmonic step in a separated directory.

Keyword: NUMBER_OF_QUASIHARMONIC_STEPS
Values: 0, 1, 2
Default: 0
Notes: If software stops in the when determining the frequencies for the Grüneisen parameters, restart at a given step. 0 to start at the reference volume. 1 to start at the plus volume. 2 to start at the minus volume.

Keyword: READ_QUASIHARMONIC_GEOMETRIES
Values: True or False
Default: False
Notes: The plus and minus structures are found in PlusFreq.in and MinusFreq.in respectively instead of simply expanding and compressing the reference cell.

5.7 Symmetry

Keyword: SYMMETRY
Values: True or False
Default: True
Notes: Allows for symmetry implementation.
If false all symmetry options will be set to false.

Keyword: SPACE_SYMMETRY
Values: True or False
Default : True when PERIODIC is true, otherwise false. Always false when SYMMETRY is false.
Notes: Uses space group symmetry to reduce the number of dimer calculations and degrees of freedom for optimization.
Not implemented for system using one atom monomer units.
For crystals consisting of monomers that belong to high symmetry point group,

this option has only been tested when the lattice angles are 90°
See doi:10.1002/jcc.23737.

Keyword: LATTICE_SYMMETRY
Values: True or False
Default: False, Always true when SPACE_SYMMETRY is true. Always false when SYMMETRY is false
Notes: Fixes lattice angles that are 90 or 120 degrees.
Lattice constants that are the same value are kept the same during optimization

Keyword: PRINT_SYMMETRY_INFO
Values: True or False
Default: False
Notes: Prints the number of symmetrical monomers and dimers.
Prints the symmetry operators which maps atoms to atoms in asymmetric unit.

Keyword: MONOMER_SYMMETRY
Values: True or False
Default: False
Notes: Reduces number of QM and TINKER monomers with space group symmetry.

Keyword:AIFF_SYMMETRY
Values: True or False
Default: True
Notes: Reduce the number of monomers used by AIFF with space group symmetry.

Keyword: SYMMETRY_TOLERANCE
Values: Positive integers
Default: True
Notes: Number of digits after the decimal of the symmetry tolerance.

5.8 Quantum Espresso

Keyword: DISP_CORRECTION or DISPERSION_CORRECTION
Values: True or False
Default: False
Notes: Enables dispersion correction for the Quantum Espresso job.

Keyword: DISP_TYPE or DISPERSION_TYPE
Values: D2, XDMPW86PBE, XDMB86BPBE, XDMBLYP, XDMREVPBE, XDMPBESOL, XDM-PBE or XDM
Default: XDM
Notes: Sets appropriate parameters in the Quantum Espresso job for the given dispersion type.

Keyword: FORCE_XC
Values: Any valid exchange type from Quantum Espresso.
Default: none

Notes: Enforces the given exchange potential. Be very careful with this parameter!

Keyword: QE_BASIS
Values: Positive Integers
Default: none
Notes: Defines the kinetic energy cutoff for the Quantum Espresso jobs.

Keyword: QE_BASIS_MULTIPLIER or QE_BASIS_MULT
Values: Positive Integers
Default: none
Notes: Defines the kinetic energy for charge density and potential cutoff for the Quantum Espresso jobs.

6 Lattice Dynamics

6.1 Introduction

HMBI has the ability to do lattice dynamics. Lattice dynamics allows the use to determine the vibrational modes at various **k**-points. The phonons at a particular **k**-point is found from the diagonalization of the mass-weighted supercell dynamical matrix,

$$\hat{D}_{\alpha,\beta}(l,l',\mathbf{k}) = \frac{1}{\sqrt{M_l M_{l'}}} \sum_{\kappa} \frac{\partial V}{\partial R_\alpha(0)\partial R_\beta(\kappa)} \exp\left(-2\pi i \mathbf{k}\cdot\delta\mathbf{R}_{l,l'}(0,\kappa)\right) \tag{6}$$

where $\frac{\partial V}{\partial R_\alpha(0)\partial R_\beta(\kappa)}$ is the elements of the supercell Hessian between the coordinate $\alpha(0)$ of atom l inside the central unit cell and coordinate $\beta(\kappa)$ of atom l' outside the central unit cell. $\delta\mathbf{R}_{l,l'}(0,\kappa)$ is the distance between these two atoms.

An advantage of fragment-based methods such as HMBI over other electronic structure methods such as periodic DFT or MP2 is that lattice dynamics supercell Hessian requires little additional computational cost compared to the standard unit cell Hessian. All the necessary QM terms are already determined in the Hessian for the unit cell. The only additional term is an supercell Hessian determined by the MM method.

6.2 Usage

In order perform lattice dynamics in HMBI your $HMBI section must include the following keyword "**SUPERCELL_JOB** = true". By default the size of the supercell is 3 × 3 × 3. The size of the supercell changed with the keywords **SUPERCELL_SIZE_A**, **SUPERCELL_SIZE_B**, and **SUPERCELL_SIZE_C**. If the supercell hessian has already been calculation, set **SUPERCELL_ANALYZE_ONLY** to true. In addition, the $reciprocal_space_points is used to determine the Monkhorst-Pack **k**-point grid. By default 3 × 3× 3 Monkhorst-Pack **k**-point grid.

An example of the $reciprocal_space_points:

```
$reciprocal_space_points
3 3 3
$end
```

6.3 Print Thermodynamic Properties with Temperature

HMBI uses the vibrational phonons determined from the lattice dynamic calculation to determine vibrational energy, isochoric heat capacity, and entropy (among others). The temperature in Kelvin is selected using the **TEMPERATURE** keyword in the $HMBI section. The default value is 298. If the user would like to view these properties over a range of temperatures, set the keyword **THERMAL_FOR_MULTIPLE_TEMPERATURE** to true and include a $thermal_parameters section.

The $thermal_parameters section contains one line and includes the minimum and maximum temperature as well as the temperature intervals in the follow format.

```
<Min Temp>  <Max Temp> <Interval>
```

Here is an example $thermal_parameters section with the default values.

```
$thermal_parameters
0 298 1
$end
```

7 Quasi-Harmonic Approximation

7.1 Introduction

The quasi-harmonic approximate is used to determine how a particular vibrational mode of a crystal changes with volume using Grüneisen parameters γ_i.

$$\gamma_i = \frac{\partial \omega_i}{\partial V} \tag{7}$$

Integrating Eq 7 gives,

$$f_i = f_i^{ref} \left(\frac{V}{V_{ref}} \right)^{-\gamma_i} \tag{8}$$

Using this approximation, the frequency any given unit cell can be determined which are used to calculate the Helmholtz vibration free energy.

$$F_{vib}(T) = N_A \sum_i \left(\frac{\hbar f_i}{2} + k_b T \ln \left[1 - exp\left(-\frac{\hbar f_i}{k_b T} \right) \right] \right) \tag{9}$$

The Helmholtz vibration free energy is included into the overall energy. Since the Helmholtz vibration free energy is dependent on temperature, the quasi-harmonic approximate can be used to determine how periodic crystals expand with temperature. The quasi-harmonic approximation is implemented specifically for temperature depend optimizations for periodic crystals but it can be applied to single point energy calculations. Before one can used the quasi-harmonic approximation, one must first obtain the Grüneisen Parameters and Reference Frequencies for EQ 8. Getting the Grüneisen Parameters and Reference Frequencies is discussed in the next section.

One thing to note is that the Quasi-harmonic approximation is completely compatible with lattice dynamics and it is recommended that lattice dynamics are used. See Chapter 6.

7.2 Derive Grüneisen Parameters and Reference Frequencies

In order to use the quasi-harmonic approximation as defined by HMBI, three sets of frequencies are need. The first set are the frequencies of a reference cell. The other two are the frequencies of the reference cell expanded (referred to as the PlusFreq cell) and compressed(referred to as the MinusFreq cell) usually by 10 $\mathring{A}^3$. There are two methods to derive Grüneisen parameters and reference frequencies which are in Sections 7.2.1 and 7.2.2. Note that when determining the Grüneisen parameters the what the keyword **JOBTYPE** is set to does not matter.

7.2.1 Determine the structure, frequencies, and Grüneisen parameters at the same time.

This easiest of the two methods to set up but the slowest to run. The optimization and frequency calculation for each of the three structures are done one at a time. In your $HMBI section set the following keywords:

```
QUASIHARMONIC = true
READ_QUASIHARMONIC_GEOMETRIES = false
ARE_QHA_AVAILABLE = false
```

This method expands and contrasts the reference cell by 10 $\mathring{A}^3$. Note that when determining Grüneisen parameters the keyword **JOBTYPE** is set to does not matter.

If your job is interrupted, you can restart it at certain points. Replace your input with the structure in "reference.in". This input is either the structure of your reference cell (what your input file optimized to) or the last step in the optimization of your reference cell. Then use the keyword **NUMBER_OF_QUASIHARMONIC_STEPS** to set which structure (reference, PlusFreq, or MinusFreq) to start with. If your reference structure did not optimize or did not finish calculating the frequencies, set **NUMBER_OF_QUASIHARMONIC_STEPS** to 0. If the frequencies for the "reference" cell are determined but not the frequencies for the "PlusFreq" cell, set **NUMBER_OF_QUASIHARMONIC_STEPS** to 1. If both the frequencies for the "reference" and "PlusFreq" cells are determined but not the frequencies for the "MinusFreq" cell, Set the keyword **NUMBER_OF_QUASIHARMONIC_STEPS** to 2. To know which step you are on, check the .freq file. If the "ReferenceFrequencies" string is in the .freq file, the reference frequencies have been determined. If the "PlusVolume" string is in the .freq file, the PlusFreq frequencies have been determined. If the "MinusVolume" string is in this file, the MinusFreq frequencies are known.

The frequencies derived from each cell is in the .freq file and must be in the same directory when utilizing the Grüneisen parameters.

7.2.2 Determine Grüneisen Parameters Separately from Frequencies

This method is quicker but has a lot of little steps that must be taken to implement. Here, the all structures and frequencies are determined separately and then combined to create the .freq file. While the frequencies for the Grüneisen parameters would already be obtain for the separate structure before creating the .freq file, the final job to create the .freq file orders and matches the frequencies base on vector mode overlap to the reference structures modes rather than be frequency size.

Do the following steps.

1. Optimize your structure. The optimized structure is your reference structure. Determine the frequency of this structure.

2. Set two sets of input file from the optimized reference structure. These are your PlusFreq and MinusFreq structures. For two structure $HMBI, include the keywords below. Note that the keyword **CHANGE_VOLUME** determines the finite difference step and as defined be is 10 $Å^3$. Feel free to change as see fit.

   ```
   For PlusFreq:

    FREEZE_UNITCELLPARAMS = true
    CHANGE_VOLUME = 10

   For MinusFreq:

    FREEZE_UNITCELLPARAMS = true
    CHANGE_VOLUME = -10
   ```

3. Optimize these structures and determine the frequencies. If you optimize and determine the frequencies using different inputs (the keyword **DO_FREQ_AFTER_OPT** allows you to do so in the same job), remember to remove the **CHANGE_VOLUME** keyword for your frequency calculation.

4. Set up your input file to obtain your .freq file. Use the keywords below in you $HMBI section. **Do not run yet.** Note that when determining the Grüneisen parameters what **JOBTYPE** is set to does not matter.

   ```
   QUASIHARMONIC = true
   READ_QUASIHARMONIC_GEOMETRIES = true
   ANALYZE_ONLY = true
   ARE_QHA_AVAILABLE = false
   ```

5. Create two additional input files named PlusFreq.in and MinusFreq.in. Place the PlusFreq geometry into PlusFreq.in and the MinusFreq geometry into MinusFreq geometry. Only the $molecule and $unit_cell sections of PlusFreq.in and MinusFreq.in are read.

6. Place the following jobs into the following directions. The QM_Path, MM_Path, and HESSIAN_FILES_PATH are defined in you $HMBI file with default values being “qm”, “mm”, and “hessian_files” respectively.

   (a) Jobs for the reference structure.

   - Place the reference QM force calculations into <base path>/<QM_PATH>/reference/
   - Place the reference MM force calculations into <base path>/<MM_PATH>/reference/
   - Place the reference QM Hessian calculations into <base path>/<HESSIAN_FILES_PATH>/qm/reference/

- Place the reference MM Hessian calculation into <base path>/<HESSIAN_FILES_PATH>/mm/reference/

(b) Jobs for the PlusFreq structure.

- Place the PlusFreq QM force calculations into <base path>/<QM_PATH>/PlusFreq/
- Place the PlusFreq MM force calculations into <base path>/<MM_PATH>/PlusFreq/
- Place the PlusFreq QM Hessian calculations into <base path>/<HESSIAN_FILES_PATH>/qm/PlusFreq/
- Place the PlusFreq MM Hessian calculation into <base path>/<HESSIAN_FILES_PATH>/mm/PlusFreq/

(c) Jobs for the MinusFreq structure.

- Place the MinusFreq QM force calculations into <base path>/<QM_PATH>/MinusFreq/
- Place the MinusFreq MM force calculations into <base path>/<MM_PATH>/MinusFreq/
- Place the MinusFreq QM Hessian calculations into <base path>/<HESSIAN_FILES_PATH>/qm/MinusFreq/
- Place the MinusFreq MM Hessian calculation into <base path>/<HESSIAN_FILES_PATH>/mm/MinusFreq/

7. Run job to obtain the .freq file.

7.2.3 Incorporating lattice relaxation into the Quasiharmonic Approximation

While the majority of the method remains the same as the previous section, this method will allow the lattice parameters of the plus and minus structure to relax by applying external pressure. This can be important for crystals that do not have symmetry in their lattice parameters (in particular triclinic crystals).

Do the following steps.

1. Optimize your structure. The optimized structure is your reference structure. Determine the frequency of this structure.

2. Now you need to apply external pressure to obtain the structures of your Plus and Minus structure. Ideally you should only have to run a few jobs to obtain structures that are about $\pm$ 10 Å^3. Unfortunately this is a somewhat blind process as you do not know what volume you will get from each pressure until you optimize your structure. Note that the naming logic of these structures is opposite to that of the fixed volume approach (ie. To obtain the expanded-volume structure (PlusFreq) you must release the external pressure). Also note that Pressure in HMBI is in units of GPa.

```
For PlusFreq:


 PRESSURE = -1
```

```
For MinusFreq:

 PRESSURE = 1
```

3. Optimize these structures and determine the frequencies. Once you have optimized two structures that are around 10 $Å^3$ from your reference volume, you need to determine the frequencies using different inputs. Remember to remove the **CHANGE_VOLUME** keyword for your frequency calculation.

4. Using the optimized reference geometry, set up your input file to obtain your .freq file. Use the keywords below in your $HMBI section. **Do not run yet.** Note that when determining the Grüneisen parameters what **JOBTYPE** is set to does not matter.

```
QUASIHARMONIC = true
READ_QUASIHARMONIC_GEOMETRIES = true
ANALYZE_ONLY = true
ARE_QHA_AVAILABLE = false
```

5. Create two additional input files named PlusFreq.in and MinusFreq.in. Place the PlusFreq geometry into PlusFreq.in and the MinusFreq geometry into MinusFreq geometry. Also update their respective $unit_cell sections. Only the $molecule and $unit_cell sections of PlusFreq.in and MinusFreq.in are read.

6. Place the following jobs into the following directions. The QM_Path, MM_Path, and HESSIAN_FILES_PATH are defined in you $HMBI file with default values being “qm”, “mm”, and “hessian_files” respectively.

   (a) Jobs for the reference structure.
   - Place the reference QM force calculations into <base path>/<QM_PATH>/reference/
   - Place the reference MM force calculations into <base path>/<MM_PATH>/reference/
   - Place the reference QM Hessian calculations into <base path>/<HESSIAN_FILES_PATH>/qm/reference/
   - Place the reference MM Hessian calculation into <base path>/<HESSIAN_FILES_PATH>/mm/reference/

   (b) Jobs for the PlusFreq structure.
   - Place the PlusFreq QM force calculations into <base path>/<QM_PATH>/PlusFreq/
   - Place the PlusFreq MM force calculations into <base path>/<MM_PATH>/PlusFreq/
   - Place the PlusFreq QM Hessian calculations into <base path>/<HESSIAN_FILES_PATH>/qm/PlusFreq/
   - Place the PlusFreq MM Hessian calculation into <base path>/<HESSIAN_FILES_PATH>/mm/PlusFreq/

   (c) Jobs for the MinusFreq structure.

- Place the MinusFreq QM force calculations into <base path>/<QM_PATH>/MinusFreq/
- Place the MinusFreq MM force calculations into <base path>/<MM_PATH>/MinusFreq/
- Place the MinusFreq QM Hessian calculations into <base path>/<HESSIAN_FILES_PATH>/qm/MinusFreq/
- Place the MinusFreq MM Hessian calculation into <base path>/<HESSIAN_FILES_PATH>/mm/MinusFreq/

7. Run job to obtain the .freq file.

7.3 Use Grüneisen parameters to Thermal Expand

Once the .freq is create, Grüneisen parameters can be obtain. The vibrational frequencies at any given volume can be obtained and the crystal can be optimized as function temperature and pressure. Define temperature (in Kelvin) with the keyword **TEMPERATURE** and optionally the pressure (in GPa) with the **PRESSURE**. The default temperature is 298 K. While Quasi-harmonic approximation was designed with temperature optimization in mind, it can be utilized for single point energy calculations. If **JOBTYPE** is set to "HESSIAN", it will be changed to "SP". To use the Grüneisen parameters, .freq file must be in the same directory and the following keywords must be in the $HMBI section of your input file.

```
QUASIHARMONIC = true
ARE_QHA_AVAILABLE = true
```

If lattice dynamics was used compute frequencies, the $reciprocal_space_points section must be included in the input file with the same Monkhorst-Pack **k**-point grid.

8 The "Tiered" Quasi-Harmonic Approximation

8.1 Introduction

This method attempts to address some of the problems associated with using the current Quasi-Harmonic Approximation (QHA). Most importantly we try to address the computational cost of using the QHA. While previously you would run the full QHA (opt, freq, and thermal expansion) at the desired level of theory (such as CCSD(T)/CBS) this can quickly become computationally prohibitive. Instead we have found that it is acceptable to perform optimizations and frequency calculations using a cheaper theory (eg. DFT) and single-point energy correct up to your desired level of theory.

Ultimately this method breaks up the calculation of the Gibbs Free Energy into multiple parts. From statistical thermodynamics, the Gibbs free energy combines the electronic internal energy U_{el}, the Helmholtz vibrational free energy F_{vib}, and a pressure-volume (PV) contribution.

$$G(T,P) = U_{el} + F_{vib}(T) + PV \tag{10}$$

In crystals at ambient pressure, the PV term contributes negligibly.

Typically the electronic internal energy contribution comes from HMBI:

$$U_{el}^{HMBI} = E_{1-body}^{QM} + E_{SR\ 2-body}^{QM} + E_{LR\ 2-body}^{MM} + E_{many\ body}^{MM} \tag{11}$$

The Helmholtz vibrational free energy is computed from standard harmonic oscillator vibrational partition functions as,

$$F_{vib}(T) = N_a \sum_i \left(\frac{\hbar\omega_i}{2} + k_b T \ln \left[1 - \exp\left(-\frac{\hbar\omega_i}{k_b T} \right) \right] \right) \tag{12}$$

where N_a is Avogadro's number, $\hbar$ is Plank's constant, k_b is the Boltzmann constant, and ω_i is the vibrational frequency of mode i. The first term corresponds to the zero-point vibrational contribution, while the second gives the thermal vibrational contribution.

The main difference between the previous QHA and the new method is that each of these contributions are now broken up into a series of curves which are then combined using a MATLAB script. By doing this we allow the electronic internal energy curve to become any desired level of theory which can then be combined with the frequencies which are generated using a lower level of theory.

The following sections will outline how to go about performing these calculations, problems to look out for, and ultimately what you should expect to see as a result of your efforts. While this currently requires a lot of supervision we hope in the future to make this method user-friendly.

8.2 Calculating the Electronic Internal Energy Curve (U_{el})

8.2.1 Obtaining the Reference geometry

First you must optimize the geometry of your structure. This can be done with any level of method but we recommend optimizing using Quantum Espresso and the B86bPBE-XDM pseudopotential. Prepare the input file and optimize the geometry. This optimized structure will be referred to as the reference structure from now on.

8.2.2 Generating the E(V) curve

Now we need to get an idea of how the electronic internal energy changes as a function of volume. There are two ways to accomplish this 1) Isotropically expand and contract the reference unit cell volume and perform a fixed-cell optimization on each struture. or 2) Apply external pressure to the reference structure and allow the structure to optimize. We recommend using the latter method as this will allow the lattice parameters to relax naturally. If you are using Quantum Espresso for the quantum calculation then for both methods it is recommended that the optimization be carried out using the Quantum Espresso optimizer instead of HMBI's.

Isotropic E(V) curve

For any desired number of points, make a copy of your reference geometry into a separate file add the following command to your $hmbi section:

```
FREEZE_UNITCELLPARAMS = true
CHANGE_VOLUME = 10
```

Then optimize the geometries.

Anisotropic E(V) curve

While this method is a little more hit and miss than the Isotropic method it can be essential for crystals whose unit cell lacks high symmetry. To accomplish this we apply external pressure to our reference structure and optimize. For HMBI this is as simple as adding the following command to the $hmbi section:

```
PRESSURE = 1
```

Note that pressure in HMBI is in units of GPa and in Quantum Espresso pressure is in units of Kbar. With this method the HMBI Final Energy will include the pressure contribution so you will need to subtract the PV term from the HMBI energy (or if using the HMBI optimizer then simply grep for the last HMBI Electronic Energy).

Since both methods are fairly repetative scripting is particularily useful at this step. The example script below will take the reference geometry, create the necessary files, alter the pressure, and submit the files to the queue:

```
#!/bin/bash

#Pressure in Kbar for QE
#These are the pressures I want to iterate over
for i in ‘seq 2 2 10‘ ‘seq -2 -2 -10‘
do
  echo pV${i}
  mkdir pV${i}
  cd pV${i}
  cp ../reference.in resorcinol.in
  sed -i "s/this/${i}/g" resorcinol.in
  cp ../r-mp2.sh r-pV${i}resAlpha.sh
  qsub r-pV${i}resAlpha.sh
  cd ../
done
```

When performing these calculations it is ideal to have at least 10 points on your E(V) curves. You also want to ensure you have a well defined energy well in addition to points out to at least 20-30 Å^3 away from your reference volume in both directions. For the calculation of the $F_{vib}(T)$ you will need two structures about $\pm$ 10 Å^3 away from your reference volume. It is recommended that you continue generating points until you achieve structures within 2-3 Å^3 of this target.

8.2.3 Optional: Performing Single-point Energy Corrections

After obtaining an E(V) plot at the cheaper theory you can then single-point energy correct up to your desired level of theory. What this involves is taking your volumes and geometries of each point along the E(V) curve, setting up a new HMBI job for the desired theory, and running the jobs with the following command set in the $hmbi section:

```
JOBTYPE = ENERGY
```

This can cause your energy well to shift since the potential energy surface has changed. In order to ensure that you have sampled the bottom of the new energy well sufficiently it is recommended you perform the SPE corrections immediately after you finish optimizing each geometry in the cheaper method. If you find that your original set of geometries does not sufficiently describe the new energy well then continue generating structures at the cheaper level of theory until you have a good E(V) curve for your desired level of theory.

8.2.4 Fitting the E(V) curve in MATLAB

Once you have obtained the E(V) plot at the desired energy level you will need to place the data in a file format that MATLAB can readily read in. An example of this is below:

```
#Volume (Ang^3) Energy (kJ/mol)
832.126858      -4012152.23356034
688.318397      -4012252.07190479
564.18181       -4012327.45258067
551.820153      -4012330.03371393
540.173259      -4012330.87271897
532.795677      -4012330.56600123
528.378883      -4012329.94051682
516.739809      -4012327.12134208
500.872309      -4012318.65886698
476.592477      -4012294.49410104
466.927207      -4012280.23177907
443.653594      -4012231.99856419
431.328664      -4012196.81229346
407.377253      -4012104.67778039
```

Note: MATLAB is sensitive to tabs so you'll want to ensure that only one tab or space exists between your data points. Name your file in a way that you and (hopefully) others can understand 3 months from now (Ex: e−el−b86bpbeXDM.dat).

The $E(V)$ curve will then be read into MATLAB and fitted to a Murnaghan equation of state,

$$E(V) = E_0 + \frac{B_0 V}{B_0'}\left[\frac{(V_0/V)^{B_0'}}{B_0' - 1} + 1\right] - \frac{B_0 V_0}{B_0' - 1} \tag{13}$$

where E_0, V_0, B_0, and B_0' are the fit parameters. E_0 gives the electronic energy at the minimum, V_0 is the molar volume at the minimum energy, B_0 is the bulk modulus, and B_0' is the first derivative of the bulk modulus with respect to pressure. This fit allows the MATLAB script to both interpolate and extrapolate over a range of volumes which it will generate.

8.3 Calculating the Helmholtz Free Energy Curve ($F_{vib}(T)$)

This section is mostly unchanged from the previous QHA. To generate the $F_{vib}(T)$ curves you must first generate your Grüneisen parameters. To do this, you must calculate the frequencies of your reference structure. You must also calculate the frequencies for two of the structures that were optimized for use in your electronic internal energy curve (preferably ones that are around 10 Å^3 away from your reference volume). After that follow the procedure outlined in Section 7.2.3 to generate the Grüneisen parameters. In particular you will be using the .freq file that is generated. Note that you only need to do this for the cheaper level of theory. Or put another way: even if the energy well has shifted you do not need to generate 3 new sets of frequencies for geometries that better-describe the new energy well (the Grüneisen parameters should take care of this).

8.4 Calculating the Gibbs Free Energy Curve Neglecting Pressure ($G(T)$)

The following is the MATLAB script which will generate the Gibbs Free Energy plots while taking your E(V) plot and the .freq file you generated. There are a few lines you will need to change to

get this working best for your system of interest:

```
test = fopen('resorcinolAlpha.freq'); <--- Will need to become whatever your .freq name is

e_el = importdata('e-el-b86bpbeXDM.dat'); <--- Will need to become whatever
     you labeled your E(V) plot

v_min = Energy_fit(4)-100;  <--- If you wish to change the minimum volume
 generated (units of Ang^3)

v_max = Energy_fit(4)+100;  <--- If you wish to change the maximum volume
 generated (units of Ang^3)

t0=0:10:100;   <--- Make your temperature range whatever you want
    (units of K)
t1=125:25:425;
Temp=cat(2,t0,t1);
.OR.
for temp = 0:25:300
```

Important files that will be generated include:

murnaghanEqFit.txt	- Keeps a log of the fitted Murnaghan EOS parameters - Note that this was only fit against the E(V) curve
freeEnergy.txt	- Keeps a record of the Gibbs Free Energy at a given volume and Temperature
fvibEnergy.txt	- Keeps a record of the Helmholtz Free Energy at a given volume and Temperature.
final_results.txt	- a summary of all the important values at different temperatures

Remember this script assumes that the pressure is 0 GPa.

```
%Script to compute Free Energy at given Temp and Volumes
%Self-generates Grueneisen parameters and Fvib calcs
clear
test = fopen('resorcinolAlpha.freq');
%test = fopen(freq_name);
ref_found = false;
plus_found = false;
minus_found = false;
ref_freqs = {};
plus_freqs = {};
minus_freqs = {};
x_old = 0;

%constants needed for the calculation
h= 6.62607363e-34; %Js
Na= 6.0221367e23; %1/mol
kb= 1.3806488e-23; %J/K
R= kb*Na;  %J/(mol K)
```

```
c= 2.99792458e8; %m/s
count_structures = 0;
%
%Step 0: Extract frequency data from file
%
tline = fgetl(test);
while ~feof(test)
    %disp(tline);
    if contains(tline,'Frequencies')
        blah = strsplit(tline);
        ref_vol = str2double(blah(2));
        ref_found = true;
        tline = fgetl(test);
        x_old = 0;
        count_structures = count_structures + 1;
    elseif contains(tline,'PlusVolume')
        blah = strsplit(tline);
        plus_vol = str2double(blah(2));
        plus_found = true;
        tline = fgetl(test);
        x_old = 0;
        count_structures = count_structures + 1;
    elseif contains(tline,'MinusVolume')
        blah = strsplit(tline);
        minus_vol = str2double(blah(2));
        minus_found = true;
        tline = fgetl(test);
        x_old = 0;
        count_structures = count_structures + 1;
    end

    x = str2double(tline);

    if (ref_found) && (plus_found) && (minus_found)
        minus_freqs = [minus_freqs x];
    elseif (ref_found) && (plus_found) && (~minus_found)
        plus_freqs = [plus_freqs x];
    elseif (ref_found) && (~plus_found) && (~minus_found)
        ref_freqs = [ref_freqs x];
    end

    if(abs(x_old-x) > 2000) && (x_old~=0)
        %fprintf('x: %f\tx_old: %f\n',x,x_old);
        count_structures = count_structures + 1;
    end

    if(x ~= 0)
        x_old = x;
```

```
    end

    tline = fgetl(test);
end

count_structures = count_structures / 3;
fprintf('count_structures: %i\n',count_structures);

%convert from a cell array to a regular array
ref_freqs = cell2mat(ref_freqs);
plus_freqs = cell2mat(plus_freqs);
minus_freqs = cell2mat(minus_freqs);

%get rid of NaN in the array
ref_freqs(isnan(ref_freqs)) = [];
plus_freqs(isnan(plus_freqs)) = [];
minus_freqs(isnan(minus_freqs)) = [];
fclose(test);

%
%Step 1: Get electronic energy data and fit it.
%
e_el = importdata('e-el-b86bpbeXDM.dat');
Volumes = e_el.data(:,1)';
Energies = e_el.data(:,2)';
[minEnergy,pos] = min(Energies);
minVolume = Volumes(pos);

F = @(a,x) a(1) + (a(2) * x)/a(3) .* (((a(4)./x).^(a(3)))/(a(3) - 1) + 1 )
 - ((a(2) * a(4) )/(a(3) - 1));

% Set initial guess parameters and perform least squares fit on the E(V)
% data.
a0 = [minEnergy, 5.0, 8.0, minVolume];
Energy_fit = lsqcurvefit(F,a0,Volumes,Energies);
fprintf('E(V) Fit Parameters:\n');
fprintf('   A0 = %.5f\n',Energy_fit(1));
fprintf('   V0 = %.3f\n',Energy_fit(4));
fprintf('   B0 = %.3f\n',Energy_fit(2));
fprintf('   B0prime = %.3f\n\n',Energy_fit(3));
Emin = Energy_fit(1);


%Keep a log of the Fit Parameters
name = sprintf( 'murnaghanEqFit.txt');
fitEq = fopen(name,'w');
fprintf(fitEq,'E(V) Fit Parameters:\n');
fprintf(fitEq,'   A0 = %.5f\n',Energy_fit(1));
```

```
fprintf(fitEq,'   V0 = %.3f\n',Energy_fit(4));
fprintf(fitEq,'   B0 = %.3f\n',Energy_fit(2));
fprintf(fitEq,'   B0prime = %.3f\n\n',Energy_fit(3));
fclose(fitEq);

%
%Step 2: Begin generating Fvib data
%

%First calculate the grueneisen parameters
grun = -1.*((log(plus_freqs./minus_freqs))./(log(plus_vol./minus_vol)));
grun(isnan(grun)) = 0;

%Volume to extrapolate from
v_min = Energy_fit(4)-100;
v_max = Energy_fit(4)+100;
vol_array = [v_min:v_max];
results = fopen('final_results.txt','w');
fprintf(results,'#Temperature (K)     Optimal Volume (Ang^3)    Free Energy (kJ/mol)
    Entropy (kJ/mol)    Enthalpy (kJ/mol)    Heat Capacity (J/(mol K))
Internal Energy (kJ/mol)\n');

t0=0:10:100;
t1=125:25:425;
Temp=cat(2,t0,t1);
%Loop over a set of temperatures
for temp = Temp;
%for temp = 0:25:25
  %Reset Fvib values to an empty array
  fprintf('\n\nTemp = %f\n',temp);
  fvib = {};
  hvib = {};
  svib = {};
  cv = {};
  %loop over a set of volumes
  for vol = vol_array

    %calculate new frequency
    %units: m/s * (cm/m) * (1/cm) = 1/s = Hz
    w =  ( c * 100 ) .* ref_freqs .* (( vol / ref_vol ).^( -1 .* grun ));
    %Calculate zero point energy contribution
    zpe = (sum(w) * h) / 2;

    %If relevant calculate other term
    if temp ~= 0
     %units: J/K * K = J
     h_vib_term2 = (h .* w) ./ ( exp( ( h .* w )./( kb * temp )) - 1);
     h_vib_term2 (~isfinite(h_vib_term2))=0;
```

```
     h_vib_term2 = sum(h_vib_term2);
     s_vib = ((h .* w) ./(temp .* ( exp( ( h .* w ).( kb * temp )) - 1)))
- (( kb ) .* log( 1 - exp( ( -1 * h .* w ).( kb * temp ))));
     s_vib(~isfinite(s_vib))=0;
     s_vib= sum(s_vib);
     Cv = (Na/(count_structures*kb)).*( ( ((h .* w) ./(temp .*
( exp( ( h .* w ).( kb * temp )) - 1))).^2).* exp( (h .* w )./
( kb * temp )));
     Cv(~isfinite(Cv))=0;
     Cv = sum(Cv);

    else
      entropy_enthalpy = 0.0;
      s_vib = 0;
      h_vib_term2 = 0;
      Cv = 0;
    end

    %Sum to create Fvib
    %units: (/mol) * (J + J) * (kJ / J) = kJ/mol
    s_vib =  Na * (s_vib) / (1000 * count_structures);
    h_vib =  Na * (zpe + h_vib_term2) / (1000 * count_structures);
    f_vib = h_vib - (temp * s_vib);
    fvib = [fvib f_vib];
    hvib = [hvib h_vib];
    svib = [svib s_vib];
    cv = [cv Cv];

  end
  %Call to calculate the gibbs free energy
  fvib = cell2mat(fvib);
  hvib = cell2mat(hvib);
  svib = cell2mat(svib);
  cv = cell2mat(cv);

  %
  % Step 3: Fit Fvib(V) data for current temperature
  %

  fvib_spline = spline(vol_array,fvib);
  spline_fvib_func = @(v) ppval(fvib_spline,v);

  hvib_spline = spline(vol_array,hvib);
  spline_hvib_func = @(v) ppval(hvib_spline,v);

  svib_spline = spline(vol_array,svib);
  spline_svib_func = @(v) ppval(svib_spline,v);
```

```
cv_spline = spline(vol_array,cv);
spline_cv_func = @(v) ppval(cv_spline,v);

%
% Step 4: Define Helmholtz free energy function by combining our E(V) fit
% and the f_vib data.
%

Free_energy = @(v) F(Energy_fit,v) + spline_fvib_func(v);
%find the minimum of the Gibb's Free Energy well
[Vmin, Fmin, info, output] = fminbnd(Free_energy, v_min, v_max);

% Print out the optimal volume and free energy
fprintf('Optimal Volume and Free Energy:\n%f Ang^3\t%f kJ/mol\n',Vmin,Fmin);
Smin = spline_svib_func(Vmin);
Hmin = spline_hvib_func(Vmin);
CVmin = spline_cv_func(Vmin);
fprintf('Optimal Entropy, Enthalpy and Cv:\n%f J/mol\t%f kJ/mol\t%f
J/(mol K)\n',Smin*1000,Hmin,CVmin);

% Finally add the F(V) result to the plot.
%plot(Volumes,Free_energy(Volumes) - Fmin);
%scatter(vol_array,fvib);
%hold on; % keep plot active while we add curves
%plot(vol_array,f_vib_func(vol_array));
%plot(vol_array,spline_fvib_func(vol_array));
%legend('Fvib data', 'Fvib lin fit', 'Fvib spline');
%legend('E(V) data', 'Murnaghan E(V)', 'Free Energy');
%hold off; % done with plot

%
%Step 5: Store results in 'logical' files
%

name = sprintf( 'freeEnergy%i.txt', temp );
fileID = fopen(name,'w');
fprintf(fileID,'#Volume (Ang^3)   Free Energy (kJ/mol)\n');
for i = vol_array
  fprintf(fileID,'%f %f\n',i,Free_energy(i));
  %fprintf('%f %f\n',i,Free_energy(i));
end

fclose(fileID);

name = sprintf( 'fvibEnergy%i.txt', temp );
fileID = fopen(name,'w');
fprintf(fileID,'#Volume (Ang^3)   Helmholtz Free Energy (kJ/mol)\n');
for i = vol_array
```

```
    fprintf(fileID,'%f %f\n',i,spline_fvib_func(i));
  end
  fclose(fileID);

  fprintf(results,"%f   %f   %f   %f   %f   %f     %f\n",temp,Vmin,
Fmin,Smin,Hmin,CVmin,F(Energy_fit,Vmin));

end

fclose(results);

disp('Made it to the end')
```

8.5 Calculating the Gibbs Free Energy Curve Including Pressure ($G(T, P)$)

This file just adds one additional for loop to go over the Pressures (in units of GPa).

```
%Script to compute Free Energy at given Temp and Volumes
%Self-generates Grueneisen parameters and Fvib calcs
clear

test = fopen('resorcinolAlpha.freq');
ref_found = false;
plus_found = false;
minus_found = false;
ref_freqs = {};
plus_freqs = {};
minus_freqs = {};
x_old = 0;

%constants needed for the calculation
h= 6.62607363e-34; %Js
Na= 6.0221367e23; %1/mol
kb= 1.3806488e-23; %J/K
R= kb*Na;  %J/(mol K)
c= 2.99792458e8; %m/s
count_structures = 0;
%
%Step 0: Extract frequency data from file
%
tline = fgetl(test);
while ~feof(test)
    %disp(tline);
    if contains(tline,'Frequencies')
        blah = strsplit(tline);
        ref_vol = str2double(blah(2));
        ref_found = true;
        tline = fgetl(test);
```

```
        x_old = 0;
        count_structures = count_structures + 1;
    elseif contains(tline,'PlusVolume')
        blah = strsplit(tline);
        plus_vol = str2double(blah(2));
        plus_found = true;
        tline = fgetl(test);
        x_old = 0;
        count_structures = count_structures + 1;
    elseif contains(tline,'MinusVolume')
        blah = strsplit(tline);
        minus_vol = str2double(blah(2));
        minus_found = true;
        tline = fgetl(test);
        x_old = 0;
        count_structures = count_structures + 1;
    end

    x = str2double(tline);

    if (ref_found) && (plus_found) && (minus_found)
        minus_freqs = [minus_freqs x];
    elseif (ref_found) && (plus_found) && (~minus_found)
        plus_freqs = [plus_freqs x];
    elseif (ref_found) && (~plus_found) && (~minus_found)
        ref_freqs = [ref_freqs x];
    end

    if(abs(x_old-x) > 2000) && (x_old~=0)
        %fprintf('x: %f\tx_old: %f\n',x,x_old);
        count_structures = count_structures + 1;
    end

    if(x ~= 0)
        x_old = x;
    end

    tline = fgetl(test);
end

count_structures = count_structures / 3;
fprintf('count_structures: %i\n',count_structures);

%convert from a cell array to a regular array
ref_freqs = cell2mat(ref_freqs);
plus_freqs = cell2mat(plus_freqs);
minus_freqs = cell2mat(minus_freqs);
```

```
%get rid of NaN in the array
ref_freqs(isnan(ref_freqs)) = [];
plus_freqs(isnan(plus_freqs)) = [];
minus_freqs(isnan(minus_freqs)) = [];
fclose(test);

%
%Step 1: Get electronic energy data and fit it.
%
e_el = importdata('e-el-dft-alpha.dat');
Volumes = e_el.data(:,1)';
Energies = e_el.data(:,2)';
[minEnergy,pos] = min(Energies);
minVolume = Volumes(pos);

F = @(a,x) a(1) + (a(2) * x)/a(3) .* (((a(4)./x).^(a(3)))/(a(3) - 1) + 1 )
 - ((a(2) * a(4) )/(a(3) - 1));

% Set initial guess parameters and perform least squares fit on the E(V)
% data.
a0 = [minEnergy, 5.0, 8.0, minVolume];
Energy_fit = lsqcurvefit(F,a0,Volumes,Energies);
fprintf('E(V) Fit Parameters:\n');
fprintf('   A0 = %.5f\n',Energy_fit(1));
fprintf('   V0 = %.3f\n',Energy_fit(4));
fprintf('   B0 = %.3f\n',Energy_fit(2));
fprintf('   B0prime = %.3f\n\n',Energy_fit(3));
Emin = Energy_fit(1);

name = sprintf( 'murnaghanEqFit.txt');
fitEq = fopen(name,'w');
fprintf(fitEq,'E(V) Fit Parameters:\n');
fprintf(fitEq,'   A0 = %.5f\n',Energy_fit(1));
fprintf(fitEq,'   V0 = %.3f\n',Energy_fit(4));
fprintf(fitEq,'   B0 = %.3f\n',Energy_fit(2));
fprintf(fitEq,'   B0prime = %.3f\n\n',Energy_fit(3));
fclose(fitEq);

%
%Step 2: Begin generating Fvib data
%

%First calculate the grueneisen parameters
grun = -1.*((log(plus_freqs./minus_freqs))./(log(plus_vol./minus_vol)));
grun(isnan(grun)) = 0;

%Volume to extrapolate from
v_min = Energy_fit(4)-100;
```

```
v_max = Energy_fit(4)+100;
vol_array = [v_min:v_max];

p0=0:0.01:1.0;
p1=1.5:0.5:5;
p2=6:1:20;
Press=cat(2,p0,p1);
Press=cat(2,Press,p2);

%Pressure in GPa
for press = Press;
  holdName = sprintf('final_resultsP%1.2f.txt', press);
  results = fopen(holdName,'w');
  fprintf(results,'#Temperature (K)     Optimal Volume (Ang^3)
 Free Energy (kJ/mol)    Entropy (kJ/mol)    Enthalpy (kJ/mol)
Heat Capacity (J/(mol K))    Internal Energy (kJ/mol)\n');

  t0=0:10:100;
  t1=125:25:425;
  Temp=cat(2,t0,t1);
  %Loop over a set of temperatures
  for temp = Temp;
    %Reset Fvib values to an empty array
    fprintf('\n\nTemp = %f\n',temp);
    fvib = {};
    hvib = {};
    svib = {};
    cv = {};
    %loop over a set of volumes
    for vol = vol_array

      %calculate new frequency
      %units: m/s * (cm/m) * (1/cm) = 1/s = Hz
      w =  ( c * 100 ) .* ref_freqs .* (( vol / ref_vol ).^( -1 .* grun ));
      %Calculate zero point energy contribution
      zpe = (sum(w) * h) / 2;

      %If relevant calculate other term
      if temp ~= 0
       %units: J/K * K = J
       h_vib_term2 = (h .* w) ./ ( exp( ( h .* w )./( kb * temp )) - 1);
       h_vib_term2 (~isfinite(h_vib_term2))=0;
       h_vib_term2 = sum(h_vib_term2);
       s_vib = ((h .* w) ./(temp .* ( exp( ( h .* w )./( kb * temp )) - 1)))
 - (( kb ) .* log( 1 - exp( ( -1 * h .* w )./( kb * temp ))));
       s_vib(~isfinite(s_vib))=0;
       s_vib= sum(s_vib);
       Cv = (Na/(count_structures*kb)).*( ( ((h .* w) ./(temp .*
```

```
( exp( ( h .* w )./( kb * temp )) - 1))).^2).* exp( (h .* w )./( kb * temp )));
       Cv(~isfinite(Cv))=0;
       Cv = sum(Cv);
       %fprintf('entr_enth: %f kJ/mol\n',entropy_enthalpy);
      else
        entropy_enthalpy = 0.0;
        s_vib = 0;
        h_vib_term2 = 0;
        Cv = 0;
      end

      %Sum to create Fvib
      %units: (/mol) * (J + J) * (kJ / J) = kJ/mol
      s_vib =  Na * (s_vib) / (1000 * count_structures);
      h_vib =  Na * (zpe + h_vib_term2) / (1000 * count_structures);
      f_vib = h_vib - (temp * s_vib);
      fvib = [fvib f_vib];
      hvib = [hvib h_vib];
      svib = [svib s_vib];
      cv = [cv Cv];
    end

    %Call to calculate the gibbs free energy
    fvib = cell2mat(fvib);
    hvib = cell2mat(hvib);
    svib = cell2mat(svib);
    cv = cell2mat(cv);

    fvib_spline = spline(vol_array,fvib);
    spline_fvib_func = @(v) ppval(fvib_spline,v);

    hvib_spline = spline(vol_array,hvib);
    spline_hvib_func = @(v) ppval(hvib_spline,v);

    svib_spline = spline(vol_array,svib);
    spline_svib_func = @(v) ppval(svib_spline,v);

    cv_spline = spline(vol_array,cv);
    spline_cv_func = @(v) ppval(cv_spline,v);

    PVterm = @(v) press * v * (1.0e-24) * Na;

    Free_energy = @(v) F(Energy_fit,v) + spline_fvib_func(v) + PVterm(v);

    %Now search over all available volumes for the optimal Gibbs' value
    [Vmin, Fmin, info, output] = fminbnd(Free_energy, v_min, v_max);

    % Print out the optimal volume and free energy
```

```
    fprintf('Optimal Volume and Free Energy:\n%f Ang^3\t%f kJ/mol\n',
Vmin,Fmin);
    Smin = spline_svib_func(Vmin);
    Hmin = spline_hvib_func(Vmin);
    CVmin = spline_cv_func(Vmin);
    fprintf('Optimal Entropy, Enthalpy and Cv:\n%f J/mol\t%f kJ/mol\t%f
 J/(mol K)\n',Smin*1000,Hmin,CVmin);
    fprintf('PVterm :\t%f kJ/mol\n',PVterm(Vmin));

    % Finally add the F(V) result to the plot.
    %plot(Volumes,Free_energy(Volumes) - Fmin);
    %scatter(vol_array,fvib);
    %hold on; % keep plot active while we add curves
    %plot(vol_array,f_vib_func(vol_array));
    %plot(vol_array,spline_fvib_func(vol_array));
    %legend('Fvib data', 'Fvib lin fit', 'Fvib spline');
    %legend('E(V) data', 'Murnaghan E(V)', 'Free Energy');
    %hold off; % done with plot

    %
    %Step 5: Store results in 'logical' files
    %

    name = sprintf( 'freeEnergyT%iP%1.2f.txt', temp, press );
    fileID = fopen(name,'w');
    fprintf(fileID,'#Volume (Ang^3)   Free Energy (kJ/mol)\n');
    for i = vol_array
      fprintf(fileID,'%f %f\n',i,Free_energy(i));
      %fprintf('%f %f\n',i,Free_energy(i));
    end

    fclose(fileID);

    name = sprintf( 'fvibEnergyT%iP%1.2f.txt', temp, press );
    fileID = fopen(name,'w');
    fprintf(fileID,'#Volume (Ang^3)   Helmholtz Free Energy (kJ/mol)\n');
    for i = vol_array
      fprintf(fileID,'%f %f\n',i,spline_fvib_func(i));
    end
    fclose(fileID);

    fprintf(results,"%f   %f   %f   %f   %f   %f     %f
\n",temp,Vmin,Fmin,Smin,Hmin,CVmin,F(Energy_fit,Vmin));

  end

  fclose(results);
end
```