MuSiC Documentation

Written by

Shaji Chempath, Louis A. Clark, Tina Düren, Amit Gupta, Martin J. Sanborn, and Lev Sarkisov

Contents

Chapter 1

Overview of Music

Music stands for **Mu**ltipurpose **Si**mulation **C**ode. It is designed, from the ground up, as an object-oriented molecular simulations code. Its object oriented designed hopes to achieve two goals: (1) simplify the process of adding new features to the code through the use of new objects and (2) make the code easier to understand and, thus, more accessible and useful to the simulation community.
In general, We expect two types of people to download and use this code: 1) Those who want to quickly run some simulations, already available with Music 2) Those who want to develop their own simulation code, but would like to start from an existing code/platform.
Development of music was initiated by Amit, Louis and Marty in somewhere around late 1990s. Other developers (jeff, yi, shaji, lev and tina) joined the group later and made contributions to different parts of the code
The current version of the code (music-3-0) can do molecular dynamics and Monte Carlo simulations in different ensembles. See Possibilities(Section **??**) for complete list of possible simulations.
this code can be freely downloaded and used. it is written in Fortran 90 This code has been tested only on the following systems/F90-compilers :

- Compaq compiler for Unix
- Compaq compiler for Linux
- NAG compiler for Linux
- Sun compiler

Chapter 2

Basics

2.1 What you need before starting

In order to use Music, you must have a Fortran 90 compiler. It should compile, with little trouble, on a variety of different compilers and architectures. The code was originally built using the Compaq Fortran compiler for Tru64r, but has successfully compiled using the Compaq Fortran compiler for Alpha Linux, Sun Microsystems Fortran Compiler for SPARC, and NAG F95 for Intel/Linux. Other compilers may work as well, but have not been tested. Memory requirements vary depending upon the simulation type, and may exceed 100 MB of physical RAM. The source code, documentation, and examples require about 4 MB of space, and the generated executable is usually about 1.2 MB, but a typical simulation will need about 20-50 MB of space during the run. Since the code is still under development, these requirements may change.

2.2 To obtain the code

2.2.1 Over the internet - a stable version

The code is available for license to academic and commercial parties. Please contact Randall Snurr at snurr@northwestern.edu. It can be downloaded from a password protected site.

2.2.2 For snurr-group users - current version under testing

The revisions to the code are maintained using **cvs**. CVS keeps track of revisions made to each file along with its log messages. To get the code we need to **checkout** the code from the CVS repository which is stored in `/home/software/CVSROOT`. The directory structure of the repository is as follows. Main directory is **music**. Subdirectories are **music/src, music/docs, music/utils, music/tests, music/ctrlfiles, music/drivers**.

- set the `CVSROOT` environment variable to `/home/software/CVSROOT` using

  ```
  setenv CVSROOT /home/software/CVSROOT
  ```

- Then to checkout the whole repository use :

  ```
  cvs checkout music
  ```

  This will create a directory called "music". To create a directory with a different name e.g. "Mymusic", use

  ```
  cvs checkout -d MyMusic music
  ```

- To checkout only one subdirectory (say music/src) with the new name "mysrc" use

  ```
  cvs checkout -d mysrc music/src
  ```

- to get help on CVS use

  ```
  cvs checkout -d myhelp help
  more myhelp/general
  ```

2.3 Directory Structure

Now you have a copy of music on your computer. We will assume that you named the directory as `music`. A number of directories will be created inside this top-level directory. See Table **??** for descriptions.

Directory Name	Description
`music/src`	Contains the Music source code (`*.F90`, `*.F`) and the `makemake` script
`music/drivers`	Contains the drivers (`*.F90`) for various simulation techniques, like MD, GCMC,HMC,
`music/docs`	Contains a copy of this documentation in LaTeX format.
`music/tests`	Contains standard tests used to check changes to the Music, and provides examples for
`music/ctrlfiles`	Contains sample control files for each of the different simulation drivers in `music/drive`

2.4 To create an executable

NOTE: We will assume that you are working in a Unix environment, or have access to a Unix-type shell.
NOTE: We will assume that you are using GNU make. Under some systems when a vendor-specific make is installed, you may find GNU make available as ***gmake*** *or* ***gnumake****.*
The instructions for Music are explicit so that anyone, either with a good deal of experience or a relatively new user, can compile and run the code.

There are several pre-packaged drivers for Music that can perform typical molecular simulation techniques. These drivers are listed in Table **??**.
*NOTE: All the files with code end with extension *.F90 or *.F (not *.f90 or *.f). Some compilers may behave differently depending on whether the extensions are in capital letters or small letters.*

Driver Name	Description
`drivers/music_md.F90`	Molecular dynamics simulation
`drivers/music_post.F90`	Post processing code for analyzing music results
`drivers/music_gcmc.F90`	Grand Canonical Monte Carlo simulation
`drivers/music_hgcmc.F90`	for hybrid monte carlo with or without gcmc

Of course, you can also create your own driver file to perform whatever type of simulation technique you would like. However, this will be discussed later in this document. For the time being, we will assume that you wish to perform an MD simulation, and will be using the driver `music_md.F90`.

1. Change to the top-level directory of your Music code distribution. We will assume you have placed it in `$HOME/music`.

   ```
   cd $HOME/music
   ```

2. Copy the driver file from the `drivers` directory to the source directory `src` with the command,

   ```
   cp drivers/music_md.F90 src/music.F90
   ```

   Note that the main driver has to be always named music.F90 while copied to `src` directory. This is necessary for the `makemake` script to work.

3. Change to the `src` directory,

   ```
   cd src
   ```

4. Now the Makefile needs to be created. The Makefile contains the dependencies for building the code (this tells the compiler which files are necessary to build the program), the Fortran compiler command, and compiler flags. A Perl program `makemake` is provided to aid in this. Use the command,

   ```
   makemake *.F90 *.F
   ```

   `makemake` goes through the driver (music.F90) and decides which files are to be compiled and creates the appropriate Makefile. The Makefile thus created has some options which are explained with in itself. By commenting some lines out from Makefile you could change the optimization levels etc.

5. The Music executable is built with the command,

   ```
   make
   ```

   This will create the executable `music.exe` in your top-level Music directory. *NOTE: Remember, you need GNU Make to use the generated Makefile. They have a website with help on the way "make" works and various options in Makefile.*

   NOTE: by editing Makefile you can change the name of executable if you like.

2.5 To start a run, explained using MD of butane example

This is explained for an MD run. All steps are similar for other simulations like GCMC or NVTMC. Example files for all possible simulations are given in the Possibilities (**??**) section. By now you have successfully compiled the code and created the executable (music.exe) using the driver music_md.F90. For convenience lets rename the executable as md.exe. Now let us try to run MD of Butane in gas phase at 300K using a united atom model. (Its assumed that you are molecular simulation person who has read some basic text book like Allen and Tildesely.) What are the information required so that md.exe knows what to do? we need to specify things like:

- simulation cell volume
- number of molecules
- type of forcefield
- initial configuration of system

Such information is passed to music through various inputfiles. Formats of all different inputfiles and output files are described later. Here we describe the files required for the particular case of MD of butane in gas phase.
It is convenient to copy all the required files into a separate directory. Assume you have placed them in `$HOME/music`. Change into that directory. Also copy (or link) the md.exe into this directory.
To run the Music executable, assuming you have created the control file `butane_md.ctr`, run the command

```
md.exe butane_md.ctr
```

Music will read in the information from the control file and run the appropriate simulation. Thus control file is the only file required for music executable to start. As it progresses

through the control file it might need additional information from other inputfiles too. By default, information about the simulation progress is written to the screen. If you wish, you can redirect that output to a file, say `music.log`, using,

```
md.exe butane_md.ctr > & music.log
```

Some command-line help is available with,

```
md.exe --help
```

One of the options being implemented is the ability for Music to produce a sample control file with the command,

```
md.exe --samplecf > & sample.cf
```

which will write the sample to `sample.cf`. Currently, this is only implemented for MD simulations.

2.5.1 Setting the Environment Variables

The Music code uses a number of environmental variables to find various input files(**??**) section. The different variables are described below. These have to be defined before starting a run. If none of these are defined music will look for them in the current directory

ATOMSDIR Looks in this directory to find the atom files (**??**). The atom files are kept in "/home/simsinfo/atoms".

MOLSDIR Looks in this directory to find the molecule files (**??**). The molecule files are kept in "/home/simsinfo/molecules"

PMAPDIR Looks in this directory to find the pretabulated potential map files (**??**). The pmap are stored in "/home/simsinfo/pmaps".

SMAPDIR Looks in this directory to find the sitemap files (**??**). The sitemap files are stored in "/home/simsinfo/smaps".

EMAPDIR Looks in this directory to find the electrostatic map files (**??**). The electrostatic maps are stored in "/home/simsinfo/emaps".

2.5.2 The Control File

You must tell Music what type of simulation to run, for how long, what types of atoms and molecules to use, which forces are present, and so on. This is accomplished by writing a *Music control file*. We will proceed step-by-step through the construction of the control file. If you wish, you can start with a template control file by running the command,

```
music --samplecf > & butane.cf
```

The sample control file will be written to the file `butane.cf`. It looks complicated, but we will move through each section since this file needs to be modified before it can be used. (This template generation might not work all the time, so you can also try copying a control file from the examples given in this documentaion and then edit it to your needs.) Of course, you can use any editing program you choose to modify it. *However, tabs must never be used in the control file!* This will prevent it from working.
If you are already anxious about how this "control file" thingie looks, then click here. One thing to be noted is that the control file contains different sections. The section start with a keyword(s). See Section(**??**) for full documentation of control file. full documentation of control file . Given below is a quick explanatio of different sections.
First, we will edit the "General Information" section. The general information includes the number of steps in the simulation, how often data should be recorded, how often the user should be updated, the names of the output files, and other information. Our MD simulation of butane in silicalite will be 2,000,000 steps long. Furthermore, we wish to record information during the simulation every 1,000 steps so that we may analyze the results later. The information we will record includes the butane positions and velocities. Screen output will be updated every 10,000. Our output filename will be butane.con, and the restart file will be butane.res. Using this information, the "General" section in the control file should look like the following entry.

```
--------- General Information ----------------------------------------
MD Simulation of Butane in Silicalite  # Comment line
2000000        # Number of iterations
10000          # Number of steps between writes to output/log file
1000           # Number of steps between writes to restart file
1000           # Number of steps between writes to config. file
1              # Start numbering simulations from...
839401         # Random number seed
4              # Contents of the config file (4=positions, velocities)
butane.res     # Restart file name
butane.con     # Configuration file name
---------------------------------------------------------------------
```

The basic building blocks of our system are, of course, atoms. We need to tell Music which atoms will be used in our simulation. We do this in the "Atomic Types" section in the control file. We need to use silicon and oxygen (which make up the silicalite lattice) and

united atom methyl and methylene (which make up the butane molecule). The complete "Atom Types" section is given below.

```
--------------- Atomic Types ----------------------------------------
4               # Number of atom types listed

Silicon         # Atom name
Silicon.atom    # Atom file name

Oxygen          # Atom name
Oxygen.atom     # Atom file name

Methyl          # Atom name
Methyl.atom     # Atom file name

Methylene       # Atom name
Methylene       # Atom file name
---------------------------------------------------------------------
```

From the atoms, we build the molecules. Naturally, the next section to flesh out is the "Molecule Types." Like the "Atom Types" section, we list the molecules to be included in the simulation. Also, we give the filename that contains the molecule-molecule interactions. Finally, we specify the types of *intramolecular* interactions to be included for each molecule type, indicated by 0 (off) or 1 (on).

```
-------------- Molecule Types --------------------------------------
2               # Number of molecule types

Butane          # Molecule name
Butane.mol      # Molecule file name

sili            # Molecule name
sili.mol        # Molecule file name
---------------------------------------------------------------------
```

All molecular simulations are performed in a space called the *simulation cell.* The simulation cell contains all the molecules in the simulation. Analyzing how the molecules behave in the simulation cell can yield quantitative values for physical properties, such as diffusion coefficients and heats of adsorption. The simulation cell is constructed of smaller units,

called *fundamental cells*. (The fundamental cell refers to the unit cell of the adsorbent crystal for simulation in zeolites.) Simulation cells may have boundary conditions that are open (type 2, molecules can leave), closed (type 0, molecules reflect off the walls), or periodic (type 1, molecules leave one side and return on the other). To simulate an infinite system, we use periodic boundary conditions. This information is supplied in the "Simulation Cell Information" section of the control file.
For our simulation of butane in silicalite, the fundamental cell is a cubic box of sides 10 Ang and it is specified in 'fundcell_file'

```
---------------- Simulation Cell Information -------------------------
fundcell_file  # file containing Fundamental cell info
1, 1, 1        # Number of fundamental cells repeated in x, y, z directions
1, 1, 1        # Boundary conditions (1 = periodic)
-----------------------------------------------------------------------
```

Now we need to specify how the atoms and molecules interact with each other. This info is typically given in separate files as shown below. After reading the filenames music will look in each of the files for reading the required information.

```
------ Forcefield Information -------------------------------------------
BASIC                # forcefield identifier
MOL                  # storage level
atm_atm_file         # atom-atom interaction file
spc_spc_file         # sorbate-sorbate interaction file
intramolecular_file  # intramolecular interaction file/specification
------ Ideal Parameters ------------------------------------------------
```

Now that we have given Music all the molecules we want to simulate and the type of simulation cell we wish to use, we define the type of simulation we wish to perform. Appropriately, we will be creating the "MD Information" section for the control file. For the butane in silicalite simulation, we will be using molecular dynamics. In molecular dynamics, the equations of motion are integrated forward in time to produce new positions and velocities at each time step. Thus, the properties of the integration must be specified. We will use a time step of 1 fs at a temperature of 300 K. The statistical ensemble will be NVE (constant number of particles, volume, and energy). The particular integration scheme we want is velocity verlet.

```
----------- MD Information ---------------------------------------------
1              # Number of MD move types listed
butane.res.0   # Restart file with molecule positions and velocities
```

```
INTEGRATE      # Type of move to perform
1              # Number of moves per iteration
0.001          # Time step in picoseconds (1e-12)
300.0          # Simulation temperature, K
0              # Steps between extra writes to screen (0=off)
0              # Steps between extra writes to file (0=off)
NVE            # Ensemble to simulate in
VelocityVerlet # Integrator name
-------------------------------------------------------------------
```

We need just one more piece of information for the control file: we need the initial configuration of the system. We have one molecule type, butane. The butane molecules are free to move. For the time being, we will assume that the initial configuration has been generated for us (by, say, GCMC) and the system has been equilibrated in a previous MD run, providing us with the restart file `butane.res.0`. Now we can complete the "Configuration Initialization" section.

```
------------ Configuration Initialization ----------------------------
Butane                    # Molecule name
RESTARTFILE butane.res.0 # File name with configuration information
-------------------------------------------------------------------
```

We now have a complete, working control file. We still need to construct the atom-atom interaction file, *aa.interactions*, and the molecule-molecule interaction file, *mm.interactions*. We will start with the molecule-molecule interaction file.

2.5.3 Molecule-Molecule Interaction File

The molecule-molecule interaction file is used to define the various types of interactions and potential models we wish to use in our simulation. The two general types of interactions we will consider are electrostatic (Coulombic) and non-electrostatic (non-Coulombic), such as van der Waals interactions. Each may be turned on or off in the molecule-molecule interaction file.

For the butane in ideal gas example, we will only consider van der Waals interactions described by the 12-6 Lennard-Jones potential. Coulombic interaction between the butane molecules is not considered, so they will be turned off. Our molecule-molecule interaction file `mm.interactions` should have 2 entries: an entry for Coulombic forces (which are off) and an entry for non-Coulombic forces (which are on). Butane-butane interactions will be handled with 12-6 Lennard-Jones potentials. The resulting interaction file is given below.

```
Butane Butane COUL  OFF
Butane Butane NCOUL BASIC  LJ
```

Each line consists of a molecule-molecule pair, a designation for the type of interaction (COUL or NCOUL), information about that interaction (OFF, BASIC, BASICMAP), and the name of the interaction (LJ) or, for the case of pretabulated maps, the names of the files that contain the maps. With the model types we wish to use specified, we must define the model parameters in the atom-atom interaction file.

2.5.4 Atom-Atom Interaction File

The atom-atom interaction file contains the potential model parameters. It can contain many types of model parameters, as long as each is on its own line. This allows you to reuse the atom-atom interaction file for many different simulations without the need for retyping the parameters each time.
In the molecule-molecule interaction file, we specified a Lennard-Jones interaction for butane-butane pairs. In the atom-atom interaction file, we must provide the parameters for *each* type of atom-atom pair in the molecule-molecule pair. For our specific Butane-Butane interaction, we have 2 atom types: methyl and methylene. Therefore, we should have 3 types of atom-atom interactions defined: methyl-methyl, methylene-methylene, and methyl-methylene. Since we are using the Lennard-Jones potential model, we must specify LJ as the model type, an ϵ (EPS) and a σ (SIG) for each, as well as a low (LOCUT) and high (HICUT) cutoff. Our atom-atom interaction file `aa.interactions` should look like the following:

```
Methyl    Methyl    LJ SIG@3.775 EPS@104.282 HICUT@13.0 LOCUT@0.0001
Methylene Methylene LJ SIG@3.923 EPS@72.048  HICUT@13.0 LOCUT@0.0001
Methyl    Methylene LJ SIG@3.923 EPS@72.048  HICUT@13.0 LOCUT@0.0001
```

Any atom pairs that are separated by a distance larger than the high cutoff specified by HICUT are ignored. Any atom pairs whose distance is less than the low cutoff LOCUT will cause Music to stop because the atoms are far too close to one another, and something has gone wrong.

2.5.5 Atom Files

The atom files are described elsewhere in this documentation. For this simulation, we will be using the following atom files: `Silicon.atom`, `Oxygen.atom`, `Methyl.atom`, and `Methylene.atom`.

2.5.6 Molecule Files

We will not describe the content of the molecule files, since that information can be found in other parts of this document. For this simulation, we need the following molecule files: `Butane.mol` and `sili.mol`.

2.5.7 Running the Simulation

With the control file, atom files, molecule files, atom-atom interaction file, molecule-molecule interaction file, and the restart file, we are ready to run the simulation. You can copy a working copy of all these files from link here. First, we must set the environment variables that give the location of the maps, molecule files, and atom files. If these are all located in the same directory as your Music executable, use the commands,

```
setenv ATOMSDIR ./
setenv MOLSDIR ./
```

If the files are in a different directory, then replace `./` with the appropriate path and directory name. For example, if the molecules are stored in a directory /home/molecules, then use,

```
setenv MOLSDIR /home/molecules
```

Once the environment variables are set, we can run the simulation using the command,

```
music butane.cf
```

Information will be written to the screen as the simulation progresses. If you wish to record this information in a file for later reference (often referred to as a log file), then you can redirect the screen output to a file using the command,

```
music butane.cf > & butane.log
```

for some simulations you can get a verbose output by typing

```
music -VERBOSE butane.cf > & butane.log
```

get a full list of options by

```
music --help
```

If everything goes well, you will have completed the butane in silicalite molecular dynamics simulation in a few hours. The information recorded during the simulation (butane positions and velocities) will be stored in the output file `butane.con.1`, which we specified in the control file. Later, we can analyze the data in that file to calculate physical properties using a post-processing program.

Chapter 3

Possibilities.....

3.1 Possible types of simulations

The current version of music is music-3-0. An earlier version of music which was not released on web is music-2-2. The ctrlfiles for music-2-2 are not well documented.
Possible simulations with music-3-1 (08-Feb-2003)

```
Bulk phase or porous materials
GCMC of rigid molecules, single- or multi- component
MD single- or multi- component
MD with thermostat ( velocity rescaling, Nose Hoover thermostat,
   but no Nose Hoover chains, Availiable integrators: gear, velocity Verlet)
Generation of a potential map - a binary file, used for
   simulation in zeolites or other microporous materials)
Configurational biased GCMC of n-alkanes in zeolites (multi component, can be mixed withreg
Hybrid GCMC methods ( combination of MD and GCMC for flexible molecules )
```

Possible simulations with music-2-2

```
Bulk phase or porous materials
GCMC of rigid molecules, single- or multi- component
MD single- or multi- component
MD with thermostat ( velocity rescaling, Nose Hoover thermostat,
   but no Nose Hoover chains)
Generation of a potential map( a binary file, used for
   simulation in zeolites or other microporous materials)
Configurational biased GCMC of alkanes (multi component)
```

3.2 Examples and sample files

3.2.1 Example-1 MD of butane

This is the same example used in section Basics(??)
The full list of files can be found in `music/tests` directory.
In this simulation butane molecules in gas phase is considered. This is a very simple example for demonstrating music MD. The butane molecules are modeled using the united atom model with methyl and methylene groups. Molecules interact with each other through lennard jones interactions. There are intramolecular potentials for bond stretching, bond bending and bond torsion. The post code analysis is used to get diffusivities energy averages etc.
Some of the input files and output files are listed below.

- Control File
- Atom-Atom File
- Molecule-Molecule File
- Intramolecular File
- Log File- short run
- Log File- long run
- Butane molecule
- Methyl atom
- Methylene atom
- Restart File

3.2.2 Example-2 MD of butane in silicalite

This example will show you how to use Music to run a simulation of flexible butane molecules in the zeolite silicalite. We consider specifically n-butane, which is a 4-carbon straight-chain alkane. In terms of the united atom model, butane is composed of 2 methyl atoms and 2 methylene atoms. The butane molecule is fully flexible, and we use potential models to describe the bond stretching, bond bending, and torsion angles in the molecule. Interactions between the butane molecules are described by a 12-6 Lennard-Jones potential.
The zeolite, silicalite, has a 3-dimensional pore structure. Typically, the pores are classified as either zig-zag, straight, or intersections where the zig-zag and straight meet. When

calculating the zeolite-butane interactions, we only consider interactions between the zeolite lattice oxygen atoms and the butane molecules. Furthermore, we will use a rigid zeolite lattice.
Some of the input files and output files are listed below.

- NOT YET READY

3.2.3 Example-3 Use of post-code

this section needs to be filled in

3.2.4 Example-4 GCMC in zeolites: Methane

This is a simple example of GCMC. Methane molecules (United atoms) are inserted and deleted from the zeolite frame work. Zeolite is treated as rigid.

- Control File
- Atom-Atom File
- Molecule-Molecule File
- Intramolecular File
- Log File- short run
- Control File: Long Run
- Log File- long run
- Methane molecule
- Silicalite molecule
- Methane atom
- Oxygen atom
- Silicon atom

3.2.5 Example-5 GCMC in zeolites: Hexane+Butane+Methanol (configurational biased)

This is a complex example of GCMC which shows many possible branches. Her the alkanes are treated using CB-GCMC whereas Methanol is treated with regualr GCMC. Zeolite is treated as rigid. Methanol contains partial chanrges which interact with other Methanol and also silicalite.

- Control File
- Atom-Atom File
- Molecule-Molecule File
- Intramolecular File
- Log File- short run
- Butane molecule
- Hexane molecule
- Methanol molecule
- Methyl atom
- Methylene atom
- Hydrogen atom

Chapter 4

Formats of files created and used by music

4.1 Input Files

4.1.1 Control File

This file contains the various user specified parameters for the different simulations. To keep the file general and extensible the file is divided into a number of sections. Each section is marked by a tag and is used to initialize a particular part of the simulation. A typical format for a tag is "———- TAGNAME ——-".
For example to initialize info about atoms , music looks for the tag "—— Atomic Types ——" in the control file. Then the lines immediately following this line are read for initializing all info about atoms. The order in which the sections are given does not really matter. Also, the number of dashes ("—") in a tag also does not matter. Anything written in the control file after the symbol "#" is neglected by music
Certain sections of the control file are almost always present (whether it is MD or GCMC or some other simulation) and these sections are given below: to get a fully functional ctrlfile for MD of butane in gas phase click here. More explanation for all possible individual sections and their formats are given later (see section (**??**)). We explain the basic sections in the above ctrlfile here

General Section

```
# this section is necessary for MC, MD type of simulations
#
#
#
```

```
------ General Information ---------------------------------------- LINE:01
 my simulation       # file description                               LINE:02
100000               # No. of iterations( MD steps, or MC steps)      LINE:03
10000                # No. of steps between writes to std output      LINE:04
10000                # No. of steps between writes to restart file    LINE:05
100                  # No. of steps between writes to config. file    LINE:06
1                    # Start numbering simulations from ...           LINE:07
22111971             # Iseed       LINE:08
1                    # Contents of the Configuration File( obsolete)  LINE:09
butane.res           # Restart File       LINE:10
butane.con           # Configuration File                             LINE:11
```

- LINE:01 - as shown above, the tag for general section is " – General Information –".
- LINE:02 - a description of the simulation for later book keeping.
- LINE:03 - number of iterations. this is the number of calls to Monte Carlo or Molecular dynamics rotuines. Within each call there might be more than one iteration.
- LINE:04 - tells how often is the output written onto the screen. During each output to the screen quantties like number of particles, energy etc. are displayed. By specifying the verbose flag (music.exe -VERBOSE music.ctr) sometimes you can get more information dumped on your screen
- LINE:05 - tells how often the coordinates of atoms in the system are written to 'restart file'. This file is a text file, and can be useful for looking at a crash, or starting a new run from where an old run stoped.
- LINE:06 - tells how often the configuration of the system are written to 'data file'. (data file used to be referd to as confile, configuration file, etc). This file is a binary file, and is used for doing the post code analysis. it can contain position, velocity, energies etc. The contents of datafile is specified in datafile section.
- LINE:07 - specifies how you want your output files numbered. For exampe if the basename of your restart files is basename.res and if the integer that you specify here is 2 then your output files will be named as basename.res.2, basename.res.3 ....
- LINE:08 - Seed for random number generator used
- LINE:09 - OBSOLETE, an integer which used to dictate what goes into datafile, not important any more.

- LINE:10 - basename for restart files
- LINE:11 - basename for datafiles

Next line tells how often is the output written onto the screen. The output contains usualy number of particles nrg etc.

Atomic Types Section

```
# This section lists all atom types that exist in a simulation
#
#
------ Atomic Types ------------------------------------------- LINE:01
natoms                    # number of atomic types    LINE:02
               LINE:03
atom1                     # Name of the atom    LINE:04
atom1.atm                 # Name of file containing info about this LINE:05
   LINE:06
atom2                     # Name of the atom    LINE:07
atom2.atm                 # Name of file containing info about this LINE:08
              .
.
.
atomn                     # Name of the atom
atomn.atm                 # Name of file containing info about this
```

Here we specify all atoms that exist in a simulation.

- LINE:01 - as shown above, the tag for atoms section is " – Atomic Types –".
- LINE:02 - Number of types of atoms. For example if you are running a water simulation this will be 2 (Hydrogen and Oxygen). (Note : If you are using united atom forcefield then a united atom is considered as 1 atom)
- LINE:03 - blank line, should be always blank.
- LINE:04 - Name of the first atom type. In case of water simulation 'Hydrogen' or 'Oxygen'. The order in which atoms are specified does not matter.
- LINE:05 - The name of the file which contains detailed info about the atom specified on the previous line.

- LINE:06 - blank line, should be always blank.
- these lines (LINE:04, LINE:05, and LINE:06) are repeated depending on how many atoms are specified in LINE:02.

Molecule Types Section

```
# This section lists type of molecules that exist in the simulation
#
#
------ Molecule Types ------------------------------------------------  LINE:01
nsorbs                  # number of molecule types        LINE:02
       LINE:03
molec1                  # molecule name        LINE:04
molec1.mol              # molecule information file        LINE:05
       LINE:06
molec2                  # molecule name        LINE:07
molec2.mol              # molecule information file        LINE:08
              .
.
.
molec_n                 # molecule name
molec_n.mol             # molecule information file
```

Here we specify all types molecules that exist in a simulation. In case of porous solid (e.g., zeolite) it counts as one molecule. (e.g., *For a gas phase mixture of octane, pentane and hexane, number of molecule types is 3. For adsorption of water into zeolite silicalite, number of molecule types is 2.*)

- LINE:01 - as shown above, the tag for molecules section is " – Molecule Types –".
- LINE:02 - Number of types of molecules.
- LINE:03 - blank line, should be always blank.
- LINE:04 - Name of the first molecule type. The order in which molecules are specified in this section ususally does not matter. However, if the simulations include a p[orous solid that is kept fixed (rigid) then that should be specified as the last Molecule Type
- LINE:05 - The name of the file which contains detailed info about the molecule specified on the previous line.

- LINE:06 - blank line, should be always blank.
- these lines (LINE:04, LINE:05, and LINE:06) are repeated depending on number of molecules specified on LINE:02.

Simulation Cell Section

"simcell" tells what is the size of a fundamental cell and the simulation cell. The concept of 'fundamental cell' arises becauses we may need to do simulations in crystall structures like zeolites. A fundamental cell is equuivalent to a unit cell. The lattice atom positions, potential maps etc are specified nly for the fundamental cell. By repeating this fundamental cell in 3D Music creates the simualtion cell. For regular bulk simulatons this does not matter. See below for two types of initializations :
Type.1 Initialization (Suitable for bulk simulations)

```
# This section has information about the simulation cell, pbc
#
#
------ Simulation Cell Information ----                          LINE:01
NULL                 # Filename with simulation cell info   LINE:02
30.00000      40.00000      50.00000   # unit cell edge lengths   LINE:03
90.00000      90.00000      90.00000   # unit cell angles   LINE:04
0.00000       0.00000       0.00000    # origin of unit cell   LINE:05
30.00000      40.00000      50.00000   # effective bound box size LINE:06
1, 1, 1    # No. of unit cells in x, y, z direction   LINE:07
1, 1, 1    # (1 = Periodic) in x, y, z, (0=non-periodic)   LINE:08
```

- LINE:01 - as shown above, the tag for simcell section is " – Simulation Cell Information –".
- LINE:02 - Name of file which contains fundamental cell info. "NULL" if fundamnetal cell info is added here itself
- LINE:03 - Edge legth of fundamental cell x,y,z directions
- LINE:04 - Angles of fundamental cell (only 90 Deg is tested)
- LINE:05 - Origin of unit cell (only [0.0, 0.0, 0.0] is tested)
- LINE:06 - Effective bound box size of fund. cell (x,y,z directions)
- LINE:07 - No of units to be repeated in x,y,z directions.

- LINE:08 - Apply periodic boundary conditions or not ? (only option [1, 1, 1] is tested
- NOTE : This will give you an orthorhombic simulation cell of 30.0, 40.0 50.0 Ang edgelenths

Type.2 Initialization (Suitable for zeolite simulations)

```
# This section has information about the simulation cell, pbc
#
#
------ Simulation Cell Information ----                              LINE:01
sili       # File (or Molecule ) name with simulation cell info   LINE:02
2, 3, 4    # No. of unit cells in x, y, z direction   LINE:03
1, 1, 1    # (1 = Periodic) in x, y, z, (0=non-periodic)   LINE:04
```

- LINE:01 - as shown above, the tag for simcell section is " – Simulation Cell Information –".
- LINE:02 - Name of file (or Molecule) which contains fund. cell info.
- LINE:03 - No of units to be repeated in x,y,z directions.
- LINE:04 - Apply periodic boundary conditions or not ? (only option [1, 1, 1] is tested
- NOTE : It is assumed that you have a molecule called "sili" whose molecule file contains a fund cell section. (Other wise you could have a file called "sili"). This will give you a simulation cell with 2 unit cells of silicalite in x-direction, 3 in y-direction and 4 in z-direction

Configuration Initialization Section

```
# this section describes how to initialize the system configuration
# for example we can read the coordinates of each atom of each molecule from
# the restartfile
------ Configuration Initialization -------------------------------------
Butane                      # Molecule_Type              LINE:01
RESTARTFILE butane.res.0    # Source Filename  LINE:02
..
```

- LINE:01 - as shown above, the tag for simcell section is "— Configuration Initialization –"
- LINE:02 - Name of Molecule

- LINE:03 - Source of initial configuration for this molecule. For example the file butane.res.0 might contain 32 butane molecules with their positions and velocities listed there. Other options are : a) GCMC NULL - implies that initially no molecules, but GCMC will add molecules. b) FIXED NULL - this molecule is "FIXED" like a rigid zeolite lattice.

- LINE:01 and LINE:02 are repeated depending on how many molecule types we have in the simulation.

Datafile Section

```
--------  Main Datafile Information -----------------------------------
Energy, position, pair_energy # contents of datafile: other options- velocity, time
```

Forcefield Section

```
------ Forcefield Information -------------------------------------------
BASIC                # forcefield identifier
SPC                  # storage level : other options - MOL
atom_atom_file       # atom-atom interaction file
sorb_sorb_file       # sorbate-sorbate interaction file
intramolecular_file  # intramolecular interaction file/specification
```

MD Information Section

Till now we described sections that are common for basic simulations (GCMC, MD etc.) Now let us have a look at MD section of the control file. This section will be read only by the md driver. For gcmc there is a gcmc section given later.
The MD section specifies the parameters for any molecular dynamics (MD) moves that will be performed during the simulation. It may contain two sections: regular section or/and equilibration section . An example is given below.

```
# this is the sectin of the control file specific to
# Molecular dynamics
#
------ MD Information ------------------------------------------- LINE:01
1          # Number of MD Move types listed( usually 1)            LINE:02
Generate   # Initial velocities (options=Generate, 'filename')    LINE:03
   LINE:04
INTEGRATE  # Type of move    LINE:05
```

```
1          # No. of MD steps within a move    LINE:06
0.002      # Time step, ps    LINE:07
300.0      # Simulation temperature    LINE:08
1000       # Steps between writes to std IO                          LINE:09
100000     # Steps between writes to datafile    LINE:10
NVT        # Ensemble to simulate in (NVT, NVE)    LINE:11
Gear6      # Integrator to use (Gear6 or VelocityVerlet)    LINE:12
10         # Steps between penalty function calls for constraints  LINE:13
NoseHoover # Type of thermostat for NVT                             LINE:14
100    # Mass of Nose-Hoover    LINE:15
```

- LINE:01 - as shown above, the tag for simcell section is "– MD Information – ".
- LINE:02 - Usually we will have just one type of MD move
- LINE:03 - Source of initial velocities. This should be consistent with the coonfiguration initialization section. We also have the option of using the generate key word to freshly generate velocities.
- LINE:04 - Should be blank
- LINE:05 - Move Tag (always needed)
- LINE:06 - suppose this move is repeated 1000 times. (we can have an inner loop, say of another 10 iterations within the outer loop)
- LINE:07 - Time step, ps
- LINE:08 - Simulation temperature
- LINE:09 - Steps between writes to std IO
- LINE:10 - Steps between writes to datafile
- LINE:11 - Ensemble to simulate in (NVT, NVE)
- Next few lines will depend on what you are doing
- NVT-GEAR-NoseHoover
  - LINE:11 NVT # Ensemble to simulate in (NVT, NVE
  - LINE:12 Gear6 # Integrator to use (Gear6 or VelocityVerlet)

  - LINE:13 10 # Steps between penalty function calls for constraints
  - LINE:14 NoseHoover # Type of thermostat for NVT
  - LINE:15 100 # Mass of Nose-Hoover Q (units ?)

- NVE-GEAR
  - LINE:11 NVE # Ensemble to simulate in (NVT, NVE
  - LINE:12 Gear6 # Integrator to use (Gear6 or VelocityVerlet)
  - LINE:13 10 # Steps between penalty function calls for constraints

- NVT-VelocityVerlet-VelocityRescale
  - LINE:11 NVT # Ensemble to simulate in (NVT, NVE
  - LINE:12 VelocityVerlet # Integrator to use
  - LINE:13 VelocityRescale# Type of thermostat for NVT
  - LINE:14 2 # Type of rescale

- NVE-VelocityVerlet
  - LINE:11 NVE # Ensemble to simulate in (NVT, NVE
  - LINE:12 VelocityVerlet # Integrator to use

- OTHER COMBINATIONS ARE NOT TESTED , FOR EXAMPLE: NVT-VelocityVerlet-NoseHoover COMBINATION WILL NOT WORK!!!!!!!

The MD section always begins with the "MD Information" tag followed by the number of MD move types listed. The source of the initial configuration must be specified next. If it is the MD restart file, then the filename is given here (and also in the Configuration Initialization section) (**??**). If it is another type of restart file, say GCMC for example, the option "Generate" is specified. After these items are specified, the individual MD moves are listed, each separated by a blank line from the previous.
The MD move types are each listed separately. Each must give the number of moves per iteration, the time step in picoseconds, the simulation temperature, the number of iterations between writes to the standard output, the ensemble to simulate in, and the integrator to use. Additional options may be needed depending on the choice of ensemble and integrator.
Valid Ensembles :
Music can perform MD simulations in two ensembles: microcanonical (NVE) and canonical (NVT). If NVE is specified, no additional information needs to be given. If NVT is specified, then a thermostat must be specified. However, since the thermostat is intimately connected

to the integrator, it is specified following the integrator information, not following the ensemble type.

Integrator Types :

Currently, there are two types of integrators available: VelocityVerlet and Gear6.

VelocityVerlet Velocity Verlet algorithm.

Gear6 6th-Order Gear Predictor-Corrector algorithm. Gear6 requires that the number of iterations between penalty function calls be specified after the keyword "Gear6."

If the NVT ensemble was specified, the thermostat name and options must be given as well.

VelocityRescale Velocity rescaling. It may be used by either Gear6 or VelocityVerlet. Following the VelocityRescale keyword, the scaling type must be defined and the number of iterations between scalings must be given. Valid scaling types are (1) scale by molecule temperature or (2) scale by the average of atom+molecule temperature.

NoseHoover Nose-Hoover thermostat. It may only be used with the Gear6 algorithm. Following the NoseHoover keyword, the thermal inertia Q must be specified in kcal/mol ps^2. The expected range for Q is 10-10000. You may need to experiment to find the best value.

Generate Option

If the "Generate" option is specified, this indicates that MD should generate a velocity distribution for the stored configuration and allow the system to equilibrate before running the production MD. With the "Generate" option specified, you must also include a section that details how to equilibrate the system.

MD Equilibration Information Section—

```
# IF    we gave ‘‘Generate’’ option above, we have to generate initial
#  velocities and equilibrate that. This section is responsible for that
#
#
------ MD Equilibration Information -----------------------------------
1000000              # Maximum number of iterations
0.002                # Time step, ps
300.0                # Simulation temperature
1000                 # Steps between writes to std IO
1000                 # steps between writing to confile?
NVT                  # Ensemble to simulate in (NVT, NVE)
```

```
Gear6                # Integrator to use
10                   # No. of steps between penalty calls
NoseHoover           # Thermostat to use
100                  # Q, kcal/mol ps^2 for Node-Hoover
```

GCMC Section

The GCMC section specifies the parametrs for a grand canonical Monte Carlo simulation. It consists of a common section specifying the overall parameters and some specific sections for individual molecules. The sections for individual molecules are given one after another separated by "———" . An examples is shown here :

```
------ GCMC Information ------------------------------------------- LINE:01
1                  # No. of iterations per call                 LINE:02
343.0              # temperature        LINE:03
Ideal Parameters   # Tag for the equation of state (NULL = Ideal Gas)  LINE:04
1                  # No. of simulation points ( >1 for isotherm        LINE:05
100                # Block size for statistics        LINE:06
1                  # no. of sorbates subjected to GCMC         LINE:07
          -------------------------       LINE:08
Benzene            # Sorbate Name        LINE:09
1.01               # Fugacity (kPa)(Range)        LINE:10
Null               # sitemap filename (Null = no sitemap)        LINE:11
4                  # no of gcmc movetypes        LINE:12
1.0, 1.0, 1.0, 1.0 # move type weights        LINE:13
BINSERT            # type of move        LINE:14
sili.Carbon.pmap   # Bias Potential File        LINE:15
300                # Bias temperature
BDELETE            # type of move
RROTATE            # type of move
0.2, 0             # Delta Rotate, adjust delta option (0=NO, 1=YES)
RTRANSLATE         # type of move
0.2, 0             # Delta, adjust delta option (0=NO, 1=YES)
          --------------------------
```

The "sitemap name" specifies the name of the sitemap to be used "Null" implies no sitemap is being used.
Fugacity(pressure) can be specified as *"a,b"* or by giving a filename. Here a is the lower limit and b is the upper limit of fugacity values. If a filename is specified the file should have the following format :

```
LCycloButane                # Name of molecule
5                           # No of pressure points
0.00001, 0.00005, 0.0001, 0.0005, 0.001 # Pressure Kpa
```

The movetypes available now are

```
RINSERT : inserts a molecule into the random position
BINSERT : inserts with a bias towards low energy areas( as per a bias map) and/or cavity bi
CBINSERT : inserts with a bias towards low energy areas( as per a bias map)
LINSERT : same as BINSERT, the config's are taken from a library (energy and cavity bias)
RDELETE : opposite Monte Carlo move for RINSERT (deletes a molecule)
BDELETE : opposite Monte Carlo move for RINSERT,LINSERT (deletes a molecule)
RTRANSLATE : moves a molecule by a random vector
FTRANSLATE : moves a molecule by a specified amount ( F=fixed)
RROTATE    : rotates a molecule by random eularian angles
INTEGRATE  : The heart of hybrid Monte Carlo, MD. Integrates the Whole system
```

sample control file - gcmc

sample control file - md

sample control file - hgcmc

sample control file - post

sample control file - mapmaker

Minimum Energy Paths with Simulated Annealing

Minimum Energy Paths with Baker Algorithm

******* NOT UP TO DATE **********

```
------ Baker Algorithm Parameters -----------------------------------------
1                       # 1=Find TS, 0=Find Minima
10000                   # Maximum steps allowed
1.0d-3                  # Tolerance on step size
1.0d-5                  # Tolerance on norm of the gradient
1.0d-1                  # Maximum step magnitude at each iteration
10                      # Hessian calculated every so many steps
------ Baker Minimum Energy Path ------------------------------------------
1000                            # Number of iterations
Initial Monte Carlo             # Monte Carlo to generate the Initial Config
```

```
Baker Algorithm Parameters        # Tag for Baker Parameters Section
mepbaker.benz.str                 # Base filename for storing various stats
3                                 # No. of molecules
             ---------------------
Methane                   # Molecule Name
Benzene1          # Molecule Name
Benzene2                  # Molecule Name
```

Caveat
To keep things simple the code assumes that the molecules in this section are listed first in the **Molecule Types** (**??**) section as well.

4.1.2 The Equilibrium File

OUT DATED ???????

4.1.3 The Restart File

This file is required for starting a simulation from a previous configuration. It contains the principal coordinates, velocity in cartesian coordinates and generalised coordinates. The restart file generated by a previous run can be used as a starting point for a new simulation. This is very useful while dealing with runs which might take long time to equilibrate. When a program is being executed, the restart file is written after particular number of steps; the velocities and positions of all particles are stored. This file is in the ascii format. The restart file should be written and re-written frequently, using the subroutine 'config_writerestartfile' . The frequency of writing to restart file can be specified in the general parameters section in the control file.

4.1.4 Old Crash File

OUT DATED ???????????

4.1.5 The MEP Restart File

OUT DATED ?????????????

4.1.6 The CUSTOM Restart File

OUT DATED ???????????????? This kind of file is specified by using the CUSTOM tag in the control file. An example of the specification in the control file is:

```
------ Configuration Initialization ---------------------------------------
Methane                                  # Molecule_Type
CUSTOM STDESCENT                         # Source Filename
Benzene                                  # Molecule_Type
CUSTOM STDESCENT                         # Source Filename
```

The idea behind this kind of an input file is that the format is dependent on the routine responsible for parsing this kind of file. Depending on the problem being solved a generic restartfile may not be convenient or even possible. Then the module which is custom designed to solve that particular problem can be made responsible for even parsing the kind of file that it finds most convenient to work with. Just as a way of informing a reader of the control file which routine is going to be used to parse the input file, we give the parsing module name in place of the "Filename". So, in this case the input file is going to be parsed by the module "stdescent.f90". Also note that the actual name of the input file should be given in the section where the particular module is initialized.

4.1.7 Atom File

The first part of initialization of music is initialization of atoms. Since we are using UA model we have two types of atoms : Methyl and Methylene. The control file says that the info for these atoms are stored in Methyl.atm and Methylene.atm files.
The atom file is mandatory for running the program. It defines the some basic properties of each "atom" in the simulation. The file is made up a number of TAG NAME: VALUE lines. The program searches for the atom files in the directory specified by the environment variable ATOMSDIR (**??**). The different tags that can be used are:

Atom_Name This name is used in the molecule file as a reference to this atom

Atom_Symbol This symbol is used for Rasmol or other visualization programs which depend on the atomic symbol to define connectivity.

Atom_SS_Charge Obsolescent. Not used in the current code.

Atom_SZ_Charge Obsolescent. Not used in the current code.

Atom_Mass The molecular weight in grams.

Atom_Valency The maximum number of bonds to this atom.

Sample file: Carbon.atm

```
##### Basic Atom Information
Atom_Name:  Carbon
Atom_Symbol:C
Atom_SS_Charge:0.0
Atom_SZ_Charge:0.0
Atom_Mass: 12.0
Atom_Valency: 4
```

4.1.8 Molecule File

This is the format specification for the molecule file. It consists of a number of required and optional sections. Each section is marked by a label followed by a series of options for that section. It must contain AT LEAST the molecule construction information (unless it is a single-atom molecule, in which case only the "basic molecule information" is required). Depending upon which intramolecular forces the molecule should be subject to (or those you wish the molecule to feel), the corresponding intramolecular force section should be included. Use only one of either "Bond_Constraints" or "Bond_Stretch."
If you find the general format below difficult to follw then , have a look at the specific molecule files given at the end

```
#### Molecule construction information

# Basic molecule information
Molecule_Name: [MoleculeName]

# Atom coordinates
Coord_Info: {listed, filename} {cartesian, other} {rigid, rigidrot, nalkane, branched, none
[n] # number of atoms in the molecule
[Atom#] [x coord] [y coord] [z coord] [Atom Name] [Charge] [Set] [Type]

# Connectivity info format
Connect_Info: {listed,generate,filename}
[n] # number of bond types {listed, to generate}
if "listed"
    [Atom Name] [Atom Name] [Bond Length]
    [Atom#] [List of Connected Atom#]
if "generate"
    [Atom Name] [Atom Name] [distance] [optional tolerance]
```

```
# Degrees of freedom (optional)
Molecule_DOF: [degrees of freedom]

#### Intramolecular Interaction Parameters

# torsion info format
Torsion_Info: {listed,generate,filename} {cosexpansion} {fast,slow}
[n] # number of torsion {listed, types to generate}
if "listed"
[Atom#] [Atom#] [Atom#] [Atom#] [torsion model parameters]
if "generate"
[Atom Name] [Atom Name] [Atom Name] [Atom Name] [torsion model parameters]

# Bond stretching format
Bond_Stretch: {listed,generate,filename} {Harmonic, Morse} {fast,slow}
[n] # number of bonds {listed, to generate}
if "listed"
[Atom#] [Atom#] [bond stretch model parameters]
if "generate"
[Atom Name] [Atom Name] [bond stretch model parameters]

# Bond constraints
Bond_Constraints: {listed, generate, filename} {Evansmorris} {fast,slow}
[n] # number of constraints {listed, types to generate}
if "listed"
[Atom#] [Atom#] [bond constraint model parameters]
if "generate"
[Atom Name] [Atom Name] [bond constraint model parameters]

# Bond bending format
Bond_Bending: {listed,generate,filename} {Harmonica} {fast,slow}
[n] # number of bending centers {listed, to generate}
if "listed"
[Atom#] [Atom#] [Atom#] [bond bending model parameters]
if "generate"
[Atom Name] [Atom Name] [Atom Name] [bond bending model parameters]
```

```
# 1-4 Interactions format
1-4_Info: {listed,generate,filename} {lj} {fast,slow}
[n] # number of 1-4 interactions {listed, types to generate}
if "listed"
[Atom#] [Atom#] [1-4 interaction model parameters]
if "generate"
[Atom Name] [Atom Name] [1-4 interaction model parameters]
```

Sample molecule file: Butane.mol

```
##### Basic Molecule Information

Molecule Name: Butane

Coord_Info: Listed Cartesian
    4     # number of atoms in molecule
    1    0.000000    0.000000    0.000000        Methyl   0.0 0   0    #x,y,z,name,set,type
    2    1.268427   -0.855565    0.000000     Methylene   0.0 0   0    #x,y,z,name,set,type
    3    2.536855    0.000000    0.000000     Methylene   0.0 0   0    #x,y,z,name,set,type
    4    3.805282   -0.855565    0.000000        Methyl   0.0 0   0    #x,y,z,name,set,type

Connect_Info: Generate
    2    # number of connection types listed
    Methyl    Methylene    1.530000
    Methylene Methylene    1.530000

##### Intramolecular forces

Bond_Stretch: Generate Harmonic Fast
    2 # number of bond types to generate
    Methyl    Methylene    1000.00 1.530000
    Methylene Methylene    1000.00 1.530000

Bond_Constraints: Listed Evans Fast
    3     #number of bond constraints  (0.0d0 bondlength ==> to be calculated from above)
    1    2    1.530000
    2    3    1.530000
    3    4    1.530000
```

```
Bond_Bending: Listed Harmonica Fast
    2     # number of bending centers (angle = 0.0d0 ==> to be calculated from above)
    1    2    3  124.400000  112.000000   #atoms, ktheta (kcal/mol), kthetaeq (degrees)
    2    3    4  124.400000  112.000000   #atoms, ktheta (kcal/mol), kthetaeq (degrees)



Bond_Torsion: Listed Cosexpansion Fast
    1     # number of torsion angles
    1    2    3    4    2.217500    2.905100   -3.135600   -0.731200    6.271200   -7.52710
```

Sample molecule file: xylene.mol

```
##### Basic Molecule Information

Molecule_Name: MetaXylene       # Obtained from QM optimization

Coord_Info: Listed Cartesian Rigid
    18     # number of atoms in molecule
1   0.6563   -3.6445  0.7615   Hydrogen  0.1  1 2 # x,y,z,Name,Charge,set,type
2  -0.9597    2.5292  1.4368   Hydrogen  0.1  1 2 # x,y,z,Name,Charge,set,type
3   0.6008    3.0064  0.7299   Hydrogen  0.1  1 1 # x,y,z,Name,Charge,set,type
4   0.5631    2.1178  2.2587   Hydrogen  0.1  1 1 # x,y,z,Name,Charge,set,type
5   0.2685   -0.3381 -2.6573   Hydrogen  0.1  1 1 # x,y,z,Name,Charge,set,type
6   0.2349   -2.4789 -1.4063   Hydrogen  0.1  1 2 # x,y,z,Name,Charge,set,type
7   0.1508   -1.5448  0.5265   Carbon    0.0  1 2 # x,y,z,Name,Charge,set,type
8   0.0594   -0.3165  2.2724   Hydrogen  0.1  1 2 # x,y,z,Name,Charge,set,type
9  -0.9120   -3.1867  1.4639   Hydrogen  0.1  1 1 # x,y,z,Name,Charge,set,type
10  0.6036   -2.7422  2.2817   Hydrogen  0.1  1 2 # x,y,z,Name,Charge,set,type
11  0.1234   -2.8449  1.2972   Carbon   -0.3  1 2 # x,y,z,Name,Charge,set,type
12  0.0813    2.2032  1.2732   Carbon   -0.3  1 2 # x,y,z,Name,Charge,set,type
13  0.1865    0.8743 -0.8787   Carbon   -0.1  1 2 # x,y,z,Name,Charge,set,type
14  0.2251   -0.3334 -1.5704   Carbon   -0.1  1 2 # x,y,z,Name,Charge,set,type
15  0.2066   -1.5350 -0.8673   Carbon   -0.1  1 2 # x,y,z,Name,Charge,set,type
16  0.1098   -0.3212  1.1859   Carbon   -0.1  1 2 # x,y,z,Name,Charge,set,type
17  0.1305    0.8965  0.5149   Carbon    0.0  1 2 # x,y,z,Name,Charge,set,type
18  0.1990    1.8134 -1.4266   Hydrogen  0.1  1 2 # x,y,z,Name,Charge,set,type

Molecule_DOF: 6
```

```
Connect_Info: Generate
    2    # number of connection types listed
    Carbon Carbon      1.3  0.3
    Carbon Hydrogen    1.1  0.15

##### Intramolecular forces

Bond_Constraints: Listed EvansMorriss Ciccotti Fast
    6    # number of bond constraints listed
  3  4   CALCULATE
  3  5   CALCULATE
  3  9   CALCULATE
  4  5   CALCULATE
  4  9   CALCULATE
  5  9   CALCULATE
```

Sample molecule file: methane.mol

```
### Methane Molecule File

Molecule_Name:  Methane

Coord_Info: Listed Cartesian Rigid
    1     # number of atoms in molecule
    1    0.000000    0.000000    0.000000      Methane   0.0  0   0   #x,y,z,name,set,type
```

4.1.9 Atom -Atom interactions File

This is a required file and it describes the atom-atom interaction parameters. The file consists of a number of columns separated by space(s). Multiple force models may be defined for each atom-atom pair, but each must exist on its own line. The particular force model that is used during the simulation is defined in the molecule-molecule interaction file(**??**). The file is parsed by the routine "pairmodel"(**??**). The meaning of each column is explained below.

columns 1, 2 The name of the atom pair.

column 3 The name of the forcefield we want to use for pairwise interactions. The different

options are listed in section (**??**). If we don't want the atoms to interaction then the value in this column should be "OFF".

columns4 - Column 4 onwards are fields specific to the forcefield type given in column 3. See the description of the respective forcefields in section (**??**) for the appropriate format of the columns.

Note: Every atom-atom pair in the simulation has to have an entry in the atom-atom file.

Sample atom-atom file

```
Silicon   Silicon   OFF
Oxygen    Oxygen    OFF
Methane   Methane   LJ SIG@3.730  EPS@148.013  HICUT@13.0 LOCUT@0.01
CF4       CF4       LJ SIG@4.662  EPS@134.048  HICUT@13.0 LOCUT@0.01
Methane   CF4       LJ SIG@LBMIX  EPS@LBMIX  HICUT@13.0 LOCUT@0.01
Methyl    Methyl    BUCK A@1550.95 C@0.0 RHO@0.30017 HICUT@10.0 LOCUT@0.01
Methylene Methylene BUCK A@1550.95 C@0.0 RHO@0.30017 HICUT@10.0 LOCUT@0.01
Methyl    Methylene BUCK A@1550.95 C@0.0 RHO@0.30017 HICUT@10.0 LOCUT@0.01
Methane   Oxygen    OFF
Methane   Silicon   OFF
CF4    Oxygen    OFF
CF4    Silicon   OFF
#################### Benzene Potentials ######################
Carbon    Carbon   BenzPot A@698828.9 B@569.7  C@8.389  D@-0.3346 Rhigh@13.0 Rlow@5.0
Carbon    Hydrogen BenzPot A@91826.0  B@112.26 C@-8.389 D@0.3346  Rhigh@13.0 Rlow@5.0
```

4.1.10 Molecule -Molecule interactions File

This is a required file and it contains parameters for molecule-molecule interactions. There are two lines for each molecular pair in the simulation. One line describes the non-coulombic interactions and the other the coulombic interactions. The file is read parsed by the routine "ssinteraction" (**??**). The format and meaning of the **first four** required columns in the file is described below.

columns 1, 2 The name of the molecules in the pair.

column 3 The string "NCOUL" for non-coulombic interactions and "COUL" for coulombic interactions.

column 4 The technique for computing the intermolecular interactions.

BASIC Does a brute force pairwise calculation (**??**). The name of the potential model to use must be specified.

BASICMAP For the non-coulombic case it gets the potential and forces by interpolating (**??**) from the pretabulated potential map (**??**, **??**). For the coulombic case its does the hermite interpolation from an emap file (**??**, **??**). Requires the user to define the names of the map interaction file using the TAG FILE.

In addition to the required columns there are a number of optional fields that can be specified. The number and type of the optional fields depends on the method of caculating the potential as specified in **column 4**

Valid Options when col4 = BASIC

FAST Default is SLOW. Used to specify the fast/slow interactions for multiple time-step MD.

SMOOTH Default is OFF. Used to smooth the molecule-molecule potential beyond the cutoff radius.

COMTRUNC Default is OFF. Use the center-of-mass distance between molecules to determine whether the pair is within the potential cutoff radius. The default used atom-atom cutoff.

NONEIGHBOR Default is TRUE. Setting this makes use of the neighbor list.

Valid Options when col4 = BASICMAP

FAST Default is SLOW. Used to specify the fast/slow interactions for multiple time-step MD.

The order of the optional fields is not important.

Sample molecule-molecule file

```
sili sili          NCOUL OFF
sili sili          COUL  OFF
Methane Methane    NCOUL BASIC LJ
Methane Methane    COUL  OFF
CF4 CF4            NCOUL BASIC LJ
CF4 CF4            COUL  OFF
Butane  Butane     NCOUL BASIC BUCK
Benzene Benzene    NCOUL BASIC BENZPOT
```

```
Benzene Benzene   COUL  OFF

Methane sili     NCOUL  BASICMAP FILE@sili.Methane.pmap
Methane sili     COUL   OFF
Methane parasili NCOUL  BASICMAP FILE@parasili.Methane.pmap
Methane parasili COUL   OFF
Methane CF4      NCOUL  BASIC LJ
Methane CF4      COUL   OFF
Butane  sili     NCOUL  BASICMAP FILE@sili.Methyl.pmap FILE@sili.Methylene.pmap
Butane  sili     COUL   OFF
Benzene sili     NCOUL  BASICMAP FILE@sili.Benzene.pmap
Benzene sili     COUL   BASICMAP FILE@sili.emap
```

4.1.11 Pmap File

This is a binary file containing the pretabulated information for molecule/zeolite interactions. It contains 8 quantities namely: $Uijk$, $\frac{dUijk}{dx}$, $\frac{dUijk}{dy}$, $\frac{dUijk}{dz}$, $\frac{d^2Uijk}{dxdy}$, $\frac{d^2Uijk}{dxdz}$, $\frac{d^2Uijk}{dydz}$, $\frac{d^2Uijk}{dxyz}$ where $Uijk$ is the potential of an "atom" at the node (i,j,k) in the zeolite. The nodes are typically spaced about 0.2 Åapart in the three dimensions. During the simulation the potential and forces at non-tabulated points are calculated by using hermite interpolation (**??**) between the nodes. The files are read in by the routine "maps_pmapinit" in the file "maps.f90"

Note: Pmap files can only be used work with a fixed zeolite

4.2 Output Files

4.2.1 data file

description of datafile *.con.*

4.2.2 restart file

description of *.res.*

4.2.3 log file

description of what is writtento log file, brief, but for both gcmc, md atleast, explain -VERBOSE too

Chapter 5

Developer Documentation

5.1 introduction

Explain how music in init'd

5.2 FLOW CHART OF GCMC CALL TREE, COOL

gcmc flowchart

5.3 FLOW CHART OF MD CALL TREE, COOL

md flowchart

5.4 Forcefield Calls

As is the case with any molecular simulation code, Music's core also is the forcefield routines. However, Music has a unique ability to store forcefield interaction results to facilitate flexible use of the information without costly reevaluations. The storage is a separate code object that is initialized at a user-selected level of detail. Greater detail means greater memory requirements and usually slower runtimes, but allows extraction of more fundamental quantities.
Here it is explained how one can make forcefield calls from any part of the code. This documention is designed for developers. Normal users will not need this information. The subroutines and functions called here are from the source code of Music. Look at their headers for more help.
Some useful terminology:

'subset' – A subset of the system is portion of the system defined by its species, molecule and atom number. It can be defined by an array of 3 integers and may refer to any degree of detail. Here's some examples:

- 0 = full system subset = (/0,0,0/)
- 1 = species level subset = (/a,0,0/)
- 2 = molecule level subset = (/a,b,0/)
- 3 = atom level subset = (/a,b,c/)

'subset-subset' interactions are then interactions between two defined pieces of the system. An example would be subset1=(/1,0,0/) (species 1) interacting with subset1=(/2,1,5/) (5th atom of 1st molecule of species 2).
'storage detail' or 'storage detail level' refers to the amount of interactions detail that is stored. 'Detail' refers to levels of the system–species–molecule–atom hierarchy. Here are the available settings:

- SPC – will store all species-species interactions, nothing more detailed
- MOL – will store all molecule-molecule as well as all species-species interactions.
- ATM – will store all atom-atom, molecule-molecule and species-species interactions.

In any case, if the number of derivatives stored in the storage structure is greater than zero, then forces on each atom will be stored. MOST IMPORTANT: if it has been stored, then you can request it from the storage at any point by using the interact$_e$*xtractroutine*!!

5.4.1 STEP.1 : Initializing interact

****

You have to first initalize an 'interact' structure which we will call imodel. imodel is capable of storing different forcefields, averages etc. The ctrlfile section for interact is given below: Usually this is done as part of most drivers (gcmc md etc).

```
Call interact_init(imodel, control_filename, simulation_cell, &
                   species, 'MC', opt_tag)
```

- a) 'MC' means you are doing Monte Carlo, so forces will not be calculated
  - other options are : 'MD', 'HMC'

- b) opt$_t$*agisastringthatmusicwilllookforinthecontrolfile,Ifitisnotpassedmusicwilluse'InteractionInfo'astl*
  InteractionModel'intheexamplesbelow.
  interact ctrlfile section:
  ———————— Example.1 :
  ————— The interaction model section for a simulation with umbrella sampling starts like this:

```
------ Interaction Model ------------------------------------------
UMBRELLA             # interaction model modifier
SAMPLER              # sampling forcefield identifier
REAL                 # averaging forcefield identifier
--------------------- end of section, this line is reqd -------------
```

  So, there's a first line that identifies which type of interaction model (DEFAULT (the one you may want), UMBRELLA, BOOST etc). The next lines just identify which forcefields to use.
  The forcefield sections look like this:

```
------ Forcefield Information -------------------------------------
SAMPLER              # forcefield identifier
MOL                  # storage detail
atm_atm_file1        # atom-atom interaction file
spc_spc_file1        # sorbate-sorbate interaction file
intramolecular_file  # intramolecular interaction file/specification
------ Forcefield Information -------------------------------------
REAL                 # forcefield identifier
MOL                  # storage detail
atm_atm_file2        # atom-atom interaction file
spc_spc_file2        # sorbate-sorbate interaction file
intramolecular_file  # intramolecular interaction file/specification
```

  The confusion comes in when you notice that most of our control files don't have an interaction model section. If the interact$_i$*nitroutinecan'tfindtheinteractionmodelidentifier,thenitassumesthatther*
  Example.2 : (SIMPLE one forcefield case) ———————— (good for most purposes)

```
------  Interaction Model ------------------------------------------
DEFAULT
BASIC
---------  Forcefield Information -----------
BASIC                # forcefield identifier
```

```
SPC                 # detail level of storage
atom_atom_file      # atom-atom interaction file
sorb_sorb_file      # sorbate-sorbate interaction file
intramolecular_file # intramolecular interaction file/specification
```

5.4.2 STEP.2 : Initializing a Result Structure

********** Once you have imodel from STEP.1 you can make interact calls. During the initialization of the interation model, results storage for the entire system was initialized. Additional storage initialization may be required for move types that do partial system evaluations and are not always accepted. In this case, it is necessary to store these temporary interactions and update the full storage structure with them if the move is accepted. The need for additional storage is dependent on the simulation type.

- Case.1: Full system interaction call (example : MD)
- Case.2: mol-system interaction call (example : MC)
- Case.3: atom-species interaction call (example : CB-GCMC, map generation)

Case.1: Full system evaluations, always accepted

No need to initialize anything because imodelstorage. If number of molecules changes drastically during the simulation follow below steps to make sure there is enough memory:

```
! set incr_size=.True. if imodel%results needs to be resized
! If the number of molecules in the system increases, need to resize
If (incr_size) Then
  !** resize imodel, this creates all kinds of memory problems
  Call interact_resize(iact%imodel, pparams%scell, pparams%species)
  max_nmoles=new_nmoles
Else
  Call interact_changeAllNmoles(iact%imodel, new_nmoles(1:nsorbs))
End If
```

Case.2: mol-system interaction calls (example : MC)
Temporary storage is necessary to hold partial system interaction evaluations. This is handled by the subinteract module which uses imodel and also stores the temporary interaction for the system. For example, gcmc$_{i}nitcontainsacalltosubinteract_{i}nitwhichinitializesspecialtemporarystorage$
$systeminteractions$.

```
 Call subinteract_init(gcmcparams%subints(spc),imodel,'Molec_System', &
   itype,(/spc,1,0/),(/0,0,0/))
```

If you absolutely MUST make your own results storage structure for partial system evaluations, then look at subinteract$_init$.$The partial system storage structures are made by copying just a part of the full ls$

```
  ! molsys_nrg is actually for a species with one molecule in it.
  ! now we need to shape these molsys_nrgs properly, copy it from storage
  ! (/spc,1,0/)  - implies for one molecule of species i
  Call storetop_initcopy(molsys_nrg , imodel%results(1), &
      (/spc,1,0/) )
  ! do some copying stuff
  Call storetop_fillsub(molsys_nrg , (/spc,1,0/) )
  Call storetop_setmap(molsys_nrg, .True., (/spc,1,0/) )
```

```
Case.3 : atom-species interaction call (example : CB-GCMC, map generation)

In this case, the storage structure provided by storetop, storesym are
store is NOT used.  We choose to do this because the results will not
be used to update the full system interaction storage and because it
is faster.

It is only necessary to initialize an EnergyPlus structure with
the correct number of derivatives:
\begin{verbatim}
      Call storebase_init(ffout,0)
\end{verbatim}
```

```
 ***************************************************************************
 ***************************************************************************
 ***************************************************************************
```

```
\subsection{STEP.3 : Make the call and get extract the energy values}
```

```
Three example cases are given here.  In each case, the results from
interact_extract are inserted into an EnergyPlus structure, which
can be initialized like:
      Call storebase_init(ffout,number_derivatives)

Case.1 Full system :
\begin{verbatim}
  ! (/0,0,0/) means whole system
  success = interact_int(imodel, imodel%results(1), species,&
      scell, fast, recalc, nointra, accels, &
      aux_params,(/0,0,0/),(/0,0,0/))

  ! makes sure that everything in storage is summed correctly
  Call storetop_fastsum(imodel%results(1))

  ! would return spc-system interactions in ffout
  success = storetop_extract(imodel%results(1), want_intra, &
            want_ncoul, want_coul, ffout, (/spc,0,0/), (/0,0,0/) )

  ! would return interactions for mol1 of spc1 with mol2 of spc2 in ffout
  ! NOTE: if you want interactions with molecule-depth detail, then you
  ! must initialize the storage at MOL or ATM levels.
  success = storetop_extract(imodel%results(1), want_intra, &
            want_ncoul, want_coul, ffout, (/spc1,mol2,0/), (/spc2,mol2,0/) )
\end{verbatim}
Case.2 mol-system :
\begin{verbatim}
  ! (/1,1,0/) means molecule 1 of species 1
  success = interact_int(imodel, imodel%results(1), species,&
      scell, fast, recalc, nointra, accels, &
      aux_params,(/1,1,0/),(/0,0,0/))

  ! would return interactions for mol1 of spc1 with entire system in ffout
  success = storetop_extract(imodel%results(1), want_intra, &
            want_ncoul, want_coul, ffout, (/spc1,mol2,0/), (/0,0,0/) )

\end{verbatim}
```

```
Case.3 atom-species :
\begin{verbatim}
 subset1=(/ spc1, mol, a /)
 subset2=(/ spc2, 0, 0 /)
 successflag = interact_simpleint(imodel,ffout,subset1,subset2,species, &
                scell,fast)
 write(*,*) "nrg of spc1, mol, a with spc2 : " ffout%nrg
\end{verbatim}

\section{Conventions in coding}
explain typical words , coding conventions

\section{explain main modules}

\subsection{config}
\subsection{moves}
\subsection{datafile}
\subsection{post}
\subsection{simcell}
\subsection{atoms}
\subsection{moleciles}
\subsection{interact}
\subsection{forcefield}
\subsection{storage}
\subsection{sscoul}
\subsection{ssnoncol}
\subsection{coul}

\section{explain of utilities}
\label{developer:utils}
\subsection{utils}
\subsection{file}
\subsection{vector}
\subsection{matrix}
```