The GIMIC program

User manual (v. 0.1)

Jonas Jusélius

University of Tromsø
Department of Chemistry
N-9037 Tromsø

1 Introduction

The GIMIC program is program to calculate magnetically induced current densities in molecules. The following features have been implemented in the program

- Current densities in 2D or 3D
- The modulus of the current
- The divergence of the current (this is useful for checking gauge invariance vs. gauge independence)
- Vector representation of the current in 2D or 3D
- Integration of the current flow through defined cut-planes in molecules
- Parallel execution through MPI (optional)

GIMIC has so far been interfaced to ACES2 and Turbomole. A small utility program to extract the AO density and perturbed densities from ACES2 calculations are included in the GIMIC source distribution. The Turbomole interface is rather primitive (only HF/SCF and DFT densities), and unfortunately still requires a specially patched version of Turbomole.

2 Installation

GIMIC is written in pure Fortran 90/95, and thus requires a good F95 compiler to compile. At the moment of writing GNU gfortran (4.0.2) is not able to compile GIMIC, partly due to missing language features and partly due to an incomplete standard f95 library.

GIMIC is dependent on the library `libgetkw` to handle input file parsing. Both libgetkw and GIMIC rely on GNU autoconf generated `configure` scripts to set up the system dependent build parameters. If the ACES2 interface is to be built, one needs to ensure the availability of `liblibr.a` from the ACES2 distribution.

The `configure` scripts shipped with libgetkw, GIMIC and ACES2 are quite flexible, and for a full list of options run

```
$ ./configure --help
```

for each of the programs.

The recommended procedure to build GIMIC is to first build libgetkw and install it. libgetkw can conveniently be installed in its build directory, e.g. (note the back quotes!)

```
$ ./configure --prefix=`pwd`
$ make install
```

To build GIMIC

```
$ ./configure --prefix=/usr/local --with-getkwdir=/path/to/libgetkw
$ make install
```

or optionally if the ACES2 interface is required

```
$ ./configure --prefix=/usr/local --with-getkwdir=/path/to/libgetkw \
--with-aces2=/path/to/aces2/lib
```

Furthermore, if BLAS and LAPACK are installed in non-standard locations you might need to specify their locations (`./configure -help` is your friend).

3 Usage

GIMIC needs three input files to calculate the current density:

1. A file containing the effective one-particle density, and the magnetically perturbed densities in AO basis
2. A MOL file, with information on molecular geometry and basis sets
3. A GIMIC input file

Currently the density file (usually called XDENS) can be obtained using either ACES2 or [a special version of] Turbomole. Using ACES2, the special driver script 'xgimic2.sh' must be used to run the NMR shielding calculation. Modify the script to suit your needs (and set the paths correctly). If the NMR calculation is done with symmetry, the MOL file must be converted to C1 symmetry using the script MOL2mol.sh, prior to running GIMIC. Similarly there is a script turbo2mol.sh for converting Turbomole control/coord/basis into a MOL file for GIMIC.

Example ZMAT:

```
CO2
O    2.14516685791074   0.00000000000000      0.00000000000000
C    0.00000000000622   0.00000000000000      0.00000000000000
O   -2.14516685791393   0.00000000000000      0.00000000000000

*ACES2(CALC=CCSD,BASIS=tzp,UNITS=BOHR
COORD=CARTESIAN
MEMORY=250000000
REFERENCE=RHF
SYMMETRY=ON
PROPERTY=NMR
MULTIPLICITY=1
CHARGE=0
SCF_MAXCYC=200,CC_MAXCYC=150,CC_EXPORDER=40
CC_CONV=10,SCF_CONV=10,LINEQ_CONV=10,CONV=10
LINEQ_EXPAN=30)
```

Run ACES2 via xgimic2.sh to produce the XDENS file:

```
$ xgimic2.sh --cc >aces2.out &
```

Convert the symmetry adapted MOL file to C1 symmetry:

```
$ MOL2mol.sh
```

The new MOL file is now called mol.

Create a GIMIC input file, and run GIMIC

```
$ gimic [--mpi] [gimic.inp] >gimic.out
```

4 The GIMIC input file

The GIMIC input file is parsed by libgetkw, which defines a parser based on sections and keywords in a recursive manner. The input consists of sections containing keywords and/or other sections, and so on. The input is in principle line oriented, but lines may be continued using a '
' at the end of a line. Furthermore, blanks and tabs are insignificant, with the exception of strings. Lines may be commented until end-of-line with a hash sign (#).

Sections are delimited by an opening '' and closing '', and may have a keyword argument enclosed between '(' and ')'.

Keywords come in two different types; simple keywords consisting of integers, reals or strings (enclosed in “ ”), and array keywords. Array keywords are enclosed in '[' ']' and elements – integers, reals or strings – are delimited by ','.

4.1 Keywords

The top level section is CDENS. This section defines a few global parameters:

title [str] Useless keyword, but since every program with a bit of self respect has a title, GIMIC also has one…

rerun [boolean] Experts only.

basis [str] Name of the MOL file (eg. MOL or mol or whatever)

density [str] Name of the density file (eg. XDENS)

spherical [boolean] Use spherical cartesians (i.e. 5d/7f/10g…). This is usually handled automagically. Experts only.

debug [int] Set debug level. The higher the number, the more useless output one gets.

calc [array(str)] This keyword determines what is to be calculated, and in what order. Possible options are: 'cdens' – calculate current densities, 'integrate' – integrate the current flow through a cut-plane, 'divj' – calculate the divergence of the current. Each of these options have their own respective sections to specify options and grids.

4.1.1 cdens

jtensor [str] Name of output file containing the current tensors (optional)

jvector [str] Name of output file containing the current vectors (optional)

orthogonal_magnet [boolean] Automatically make magnetic field perpendicular to the computational grid

magnet [array(3*real)] Vector which specifies the direction of the magnetic field.

scale_vectors [real] Scaling factor for plotting purposes.

diamag [boolean] Annihilate the diamagnetic contribution to the current. Experts only.

paramag [boolean] Annihilate the paramagnetic contribution to the current. Experts only.

plot([boolean)] [subsection] Produce files suitable for plotting with 'gnuplot' or 'gopenmol'

> **plots** [int] Number of plots when doing 3D calculations. Best to set equal to 1, and ignore altogether.
>
> **vector** [str] File to contain the current vector field (gnuplot friendly)
>
> **modulus** [str] File to contain the modulus of the current density (gnuplot friendly)
>
> **nvector** [str] File to contain the normalized current vector field (gnuplot friendly). Mostly useful for debugging purposes.
>
> **gopenmol** [str] File to contain the current density in a gopenmol friendly format.

grid [subsection] Grid to be used for calculating the currents. See below for a description of the 'grid' section.

4.1.2 Grids

Grid definition in GIMIC is a tad bit messy currently. There are a number of different ways to define grids (most of them pretty experimental), but for simple plots it's (fortunately) pretty straight forward.

There are two principal types of grids; the simple 'std' grid, which is defined by a pair of (orthogonal) basis vectors, and the 'bond' grid which is mostly useful for defining cut-planes for integration.

The 'std' grid is defined by giving an origin where the grid should start, and then two vectors 'v1' and 'v2' which define a plane. The length of the vectors v1 and v2 determine the size of the grid plane. The two vectors implicitly define a third vector 'v3', orthogonal to v1 and v2, and which length is given by the 'l3' parameter. Thus, for 3D grids l3 is non-zero. The number (the spacing of points) of grid points in the directions v1, v2, and v3 is given by the 'step' vector.

The 'bond' grid is defined in the following way: v1 is the coordinate of atom 1 and v2 is the coordinate of atom 2, and l3 is the distance from atom 1 to the grid (usually one wants l3 to be half of the bond distance between atom 1 and 2). The grid will be orthogonal to the vector between atoms 1 and 2, and it's size is determined by the 'height' and 'width' parameters, which are 2 dimensional vectors to be able to specify unsymmetrical grids. The 'origin' parameter is only used to fix the "torsional angle" (or what ever one wants to call it).

GIMIC automatically output a number of .xyz files containing dummy points to show how the grids defined actually are laid out in space. The 'gridplot' parameter gives the (approximate) number of dummy points to use in each direction.

grid(type) [(std|bond)]

type [str] (even|gauss) Determines the distribution of grid points; even spaced or suitable for Gaussian quadrature

origin [array(3*real)]

v1 [array(3*real)]

v2 [array(3*real)]

l3 [real]

height [array(2*real)]

width [array(2*real)]

step [array(3*real)] Spacing between grid points in x,y,z directions

map [array(2*real)] Origin for the plotable data in 2D space

gridplot [int] Approximately the number of points in each direction to be used in the .xyz file which shows the grid.

4.2 Integration

interpolate [boolean] If a calculation has been preformed on a even spaced grid, generate a grid suitable for Gaussian integration by doing Lagrange interpolation

lip_order [int] Polynomial order of the Lagrange Interpolation Polynomials

grid(std|bond)

4.3 The divergence

plots [array(int)] produce plots for v3 indices defined by the vector

gopemol [str] filename of gOpenMol plot

grid(std|bond)