Fukui v220331

A Program to Compute Fukui Indices and Atomic Overlap Matrices

Author:	e-mail:
Nicolás Otero Martínez	`nom05@uvigo.es`
Marcos Mandado Alonso	`mandado@uvigo.es`
Ricardo A. Mosquera Castro	`mosquera@uvigo.es`

Nicolás Otero Martínez

Marcos Mandado Alonso

Ricardo A. Mosquera Castro

April 24, 2022

UniversidadeVigo

Contents

1 Fukui: scheme and software

This document is a short guide to use Fukui, a program developed as a multipurpose tool to perform chemical reactivity indices called "Fukui indices", σ/π separation of atomic populations (reading a input file with the molecular orbital symmetry) and the calculation of atomic overlap matrices as a previous step to compute the N-electron delocalization indices (employed for aromaticity studies, for example) and condensed 2-center Fukui indices, used for studies of reactivity and resonance effects. The program reads points grids from several sources and, therefore, it is independent of the type of real-space atomic electron density partitioning employed.

1.1 Scheme of the program

Fukui is a tool designed to perform several tasks in a parallelized and efficient way using a grid of points:

- Compute the total population from the aforementioned grid.
- Real space molecular partitioning schemes, such as QTAIM[1] and Hirshfeld-based[2–4] methodology, can be employed for the calculation of condensed Fukui indices using the fragment of molecular response (FMR) approach [5, 6].
- Real space atomic overlap matrix (AOM) can be computed within the grid domains as a previous step to obtain bond orders and N-electron delocalization indices through a grid of points following Mandado's implementation[7, 8].
- σ/π molecular orbital (MO) contributions can be separated from the total population.

1.2 Program details

- Programming language: Fortran 2008
- Operating systems: Any with Fortran (compilers), CMake and, optionally, OpenMP (for parallelization support).
- List of source files: fukui.f90 goon.f90 help.f90 modules/computation.f90 modules/grid.f90 modules/main.f90 modules/output.f90 modules/pi.f90 modules/wfn.f90 openmp/init_par.f90

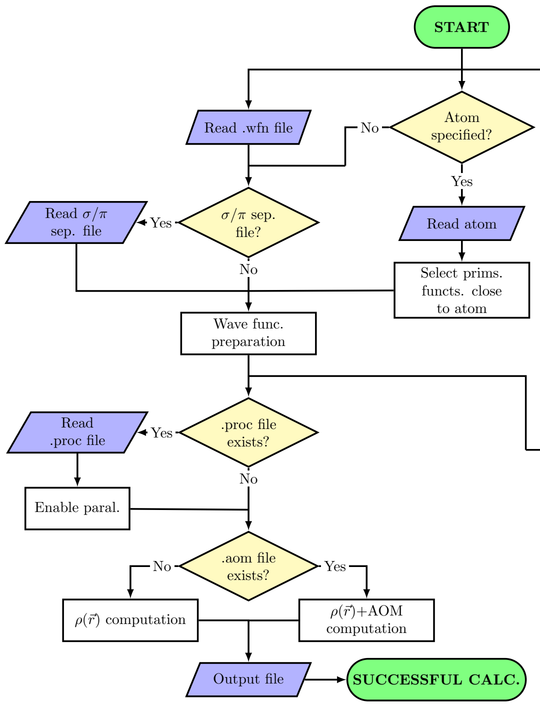

Figure 1.1: Simplified flowchart of the FUKUI 220331 program execution. Terminal blocks are represented by a green stadium shape, while the input/output and decision blocks correspond to blue rhomboid and yellow rhombus shapes, respectively. Last, the rectangle is used to stand for process blocks in the program. The acronym of atomic overlap matrix (AOM) and $\rho(\vec{r})$, the electron density, are also employed in this chart.

- List of CMake files: CMakeLists.txt modules/CMakeLists.txt openmp/CMakeLists.txt
- List of utilities: bstk_astk.f90 cube2stk.f90 fukui.sh mwfn2piorbs.sh som2sg.sh

1.3 Units

If not stated otherwise, FUKUI program outputs its results in atomic units (electrons) for populations.

1.4 Structure of the program

The structure and steps of the program are summarized in Figure 1.1.

1.5 Terms of use

FUKUI is free software under MIT license. We refer to the GitHub web page for more details about the license:

`https://github.com/nom05/fukui`PUBLICAR CÓDIGO

2 Program installation

NOTE: The preferred platform or, more specifically, the platform the developers use is GNU/Linux. MS Windows is an untested alternative. Through WSL or WSL2 the compilation would be equivalent to the following instructions. It is beyond the scope of this manual to explain how to activate these options in MS Windows.

The program source code will be available in `https://github.com/nom05/fukui` **CHECK. To install the program, follow the instructions:

1. Clone firstly the repository:

   ```
   ▷ git clone github.com/nom05/fukui **CHECK
   ```

2. Enter in the directory:

   ```
   ▷ cd fukui **CHECK
   ```

3. Verify your current CMake version[1] is equal or greater than 2.8.12.

   ```
   ▷ cmake --version
   ```

   ```
   cmake version 3.22.3
   CMake suite maintained and supported by Kitware (kitware.com/cmake).
   ```

4. Choose the Fortran compiler. By default, GNU Fortran (gfortran) will be employed. To use an alternative, edit the file called "CMakeLists.txt", in the parent directory of the source code, with your favorite editor (nano, vi, vim, Visual Studio Code, etc.). Comment (adding the symbol '#') or uncomment (remove '#') with the purpose of disabling or enabling, respectively, your personal options:

[1] From Wikipedia: In software development, CMake is cross-platform free and open-source software for build automation, testing, packaging and installation of software by using a compiler-independent method. CMake is not a build system but rather it generates another system's build files. It supports directory hierarchies and applications that depend on multiple libraries. It is used in conjunction with native build environments such as Make, Qt Creator, Ninja, Android Studio, Apple's Xcode, and Microsoft Visual Studio. It has minimal dependencies, requiring only a C++ compiler on its own build system.

▷ **vim CMakeLists.txt**

```
## >> DEFAULT COMPILER << ## Force compiler you want
set(CMAKE\_Fortran\_COMPILER gfortran)
#set(CMAKE\_Fortran\_COMPILER ifort)
set(CMAKE\_GENERATOR\_FC gfortran)
#set(CMAKE\_GENERATOR\_FC ifort)
## Comment the line in which ``gfortran'' is set and uncomment the corresponding lines
## using ``ifort'':
## >> DEFAULT COMPILER << \#\# Force compiler you want
#set(CMAKE\_Fortran\_COMPILER gfortran)
set(CMAKE\_Fortran\_COMPILER ifort)
#set(CMAKE\_GENERATOR\_FC gfortran)
set(CMAKE\_GENERATOR\_FC ifort)
```

5. Create a new directory called, for example, “build” and enter inside:

▷ **mkdir build; cd build**

6. Now, create the compilation environment:

▷ **cmake ..**

```
-- The C compiler identification is GNU 11.2.0
-- The CXX compiler identification is GNU 11.2.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- The Fortran compiler identification is GNU 11.2.0
-- Detecting Fortran compiler ABI info
-- Detecting Fortran compiler ABI info - done
-- Check for working Fortran compiler: /usr/bin/gfortran - skipped
-- Found OpenMP_C: -fopenmp (found version "4.5")
-- Found OpenMP_CXX: -fopenmp (found version "4.5")
-- Found OpenMP_Fortran: -fopenmp (found version "4.5")
-- Found OpenMP: TRUE (found version "4.5")
OPENMP FOUND
-- Configuring done
-- Generating done
-- Build files have been written to: [...]/fukui/src/build
```

7. And finally compile the code:

▷ **make**

```
Scanning dependencies of target fukui.x
[  9%] Building Fortran object CMakeFiles/fukui.x.dir/modules/main.f90.o
/home/nicux/Documentos/research/code/fortran/fukui/src/modules/main.f90:206:45:

198 |       do i = 1,imos
|                   2
......
206 |                    write(charintd,'(I7)') moc(i-1)
|                                             1
Warning: Array reference at (1) out of bounds (0 < 1) in loop beginning at (2) [-Wdo-subscript]
[ 18%] Building Fortran object CMakeFiles/fukui.x.dir/modules/grid.f90.o
[ 27%] Building Fortran object CMakeFiles/fukui.x.dir/modules/wfn.f90.o
[ 36%] Building Fortran object CMakeFiles/fukui.x.dir/modules/computation.f90.o
[ 45%] Building Fortran object CMakeFiles/fukui.x.dir/modules/pi.f90.o
[ 54%] Building Fortran object CMakeFiles/fukui.x.dir/modules/output.f90.o
[ 63%] Building Fortran object CMakeFiles/fukui.x.dir/fukui.f90.o
[ 72%] Building Fortran object CMakeFiles/fukui.x.dir/help.f90.o
[ 81%] Building Fortran object CMakeFiles/fukui.x.dir/goon.f90.o
[ 90%] Building Fortran object CMakeFiles/fukui.x.dir/openmp/init_par.f90.o
[100%] Linking Fortran executable fukui.x
[100%] Built target fukui.x
```

8. You will find the executable in the current compilation directory:

▷ **ls -tr**

```
        CMakeCache.txt  openmp  modules  Makefile  cmake_install.cmake  main.mod
grid.mod  wfn.mod  computation.mod  pi.mod  output.mod  fukui.x  CMakeFiles
```

```
fukui.x
```

TIP: To compile faster use several processor threads (increase the number of threads, #threads) through the option "-j#threads". For example, "make -j3" will set three (3) threads.

3 How to run FUKUI

FUKUI has a special way to specify the options of the calculation. Without arguments, the executable prints a short help:

▷ ./fukui.x

```
PROGRAM FUKUI 220301

** WFN file is not included as argument **

Type: ./fukui.x wfnfile stk_file/cube_file [# atom]
* Extension for stk and wfn files are optional, extension for cube file is mandatory.
* stk_file can be either a FORTRAN binary file or a ordinary text file.
They have to include, following this order: x, y, z, Gauss quadrature
weight and electron density values.
If the ordinary text format is considered, then the format to read must
be specified. the label free can be used to use free format specification.
* [# atom] is optional and enables gauss. func. skipping.
If wfnfile.skd exists, it reads cut-off1 inside.
* Sigma/pi separation: create .sg or .som (if both files exist, the first file sought is the .sg file).
* Only pi MOs: .sg/.som + .pi (command> touch wfnfile.pi).
* Skip points with gauss. quad. weight < cut-off2: touch wfnfile.skg
* Skip points with ref. dens < cut-off3: touch wfnfile.skr.
* cut-off2 and cut-off3 can be changed by editing the values in the
previously created files.
* Enable parallelization: Max         ->touch wfnfile.proc
Set # procs->edit  wfnfile.proc
* Atomic Overlap Matrix (AOM) calculation: touch wfnfile.aom
* Change max dim. of array allocation: edit  wfnfile.mxal
Include in this file (all lines are mandatory):
o Line 1: Max # atoms.
o Line 2: Max #    mol. orbs.
o Line 3: Max # points.
o Line 4: Max # prim. funcs.
```

As you can observe in the previous execution output without arguments, the program needs a file containing wave function information (see Section A.1 and Section A.2 in the Appendix for instructions to create it and convert among wave function files, respectively) and another file including the points grid.

3.1 File including grid of points

FUKUI supports two types of points grid files:

- Cube files, following the file format used by Gaussian utilities reference (`https://gaussian.com/cubegen/`). This grid is based on points set using a step size in the three directions of the space, forming a cube (strictly speaking, a parallelepiped) containing the molecule. These cube files are built from the following scheme:

Line 1 and 2	Computational details and a comment line.
Line 3	Number of atoms (NA), the grid origin in Cartesian coordinates and the number of values per point. In the case of electron density, the value is equal to 1.

Line 4	Number of steps in the slowest running direction and the vector whose modulus corresponds to the distance between two points of this direction.
Line 5	Number of steps in the intermediate running direction and the vector whose modulus corresponds to the distance between two points of this direction.
Line 6	Number of steps in the fastest running direction and the vector whose modulus corresponds to the distance between two points of this direction.
Line 7 and the NA following	Atomic number, charge, and coordinates of the atoms.
Line 7+NA+1 until the end	Values of the electron density at each point in the grid. The total number of values is obtained as the product of the number of the steps in the three directions.

Cube file structure:

```
 tetracene B3LYP/6-31G** Density
 Electron density from Total SCF Density
   30   -6.512752  -15.750103   -9.176246    1
   53    0.248108    0.000000    0.000000
  128    0.000000    0.248108    0.000000
   75    0.000000    0.000000    0.248108
    6    6.000000    0.000000    9.237351    1.351534
    6    6.000000    0.000000    7.012694    2.663494
[...]
    1    1.000000    0.000000  -11.027453   -2.355229
  1.41965E-18  4.15287E-18  1.16752E-17  3.15458E-17  8.19189E-17  2.04457E-16
```

- A points grid obtained by means of a gaussian quadrature, containing x, y, z, numerical integration weights and electron density values, strictly in this order. The program is able to read:
  - Fortran binary files, saved in unformatted files. They are the fastest way to read the points set.
  - Text files with a Fortran specific format. The format to read must be specified. The label "free" can be used to use Fortran free format specification. In contrast, the latter is the slowest way to read a file in Fortran.

Example 1: text files including a set of points with a specific Fortran format:

```
(3(F16.8),2(E18.8))
       0.00000000       2.62795893       0.00031152     0.12858975E-09
0.11765072E+03
       0.00000000       2.62795893      -0.00031152     0.12858975E-09
0.11765072E+03
       0.00000000       2.62827045       0.00000000     0.12858975E-09
0.11765075E+03
       0.00000000       2.62764741       0.00000000     0.12858974E-09
0.11765069E+03
       0.00031152       2.62795893       0.00000000     0.12858975E-09
0.11765072E+03
      -0.00031152       2.62795893       0.00000000     0.12858975E-09
0.11765072E+03
```

Example 2: text files including a set of points with free Fortran format:

```
Free
       0.00000000       2.62795893       0.00031152     0.12858975E-09
0.11765072E+03
       0.00000000       2.62795893      -0.00031152     0.12858975E-09
0.11765072E+03
       0.00000000       2.62827045       0.00000000     0.12858975E-09
0.11765075E+03
       0.00000000       2.62764741       0.00000000     0.12858974E-09
0.11765069E+03
       0.00031152       2.62795893       0.00000000     0.12858975E-09
0.11765072E+03
      -0.00031152       2.62795893       0.00000000     0.12858975E-09
0.11765072E+03
```

3.2 How to perform a calculation

The mandatory files needed for the FUKUI program are the aforementioned wave function and grid files. For example, if the files are called "wavefunction.wfn" and "grid_file_name.stk", respectively, the way to run the calculation is:

```
▷ executable_path/fukui.x wavefunction.wfn grid_file_name.stk
```

The "executable_path" is the directory containing the executable, for example "./" if you are in the current compilation directory.

The extensions are optional with the exception of the cube grid file.

3.3 Output file name

In the case the user had the following file names: wavefunction.wfn, grid_file_name.stk, FUKUI will use the following output file name: "wavefunction_grid_file_name.fuk".

3.4 Optional: specifying the corresponding atom where the points grid is centered on

In the case the set of points is centered on an atom, it is recommended to activate the “primitive gaussian functions skipping”. This option avoids to include primitive functions far from the specified atom. The default distance is 18 au (a very safe default, according to our tests) but can be modified creating a file “file.skd” employing the wave function file name and replacing the extension by “skd”. There, a new value can be set. For example:

```
▷ echo 4.000 > wavefunction.skd
```

3.5 Other options

In general, the Fukui program is employed massively in a large set of atoms. Instead of creating a unique input and performing one parallelized calculation, each grid can be processed in a independent way and, therefore, the user can submit several parallelized calculations in different nodes of a high-performance computing cluster, obtaining the results faster. In this way, to facilitate the common options for the different files, the program reads or detects the existence of files to activate its options.

Increasing the maximum default employed resources

In the opinion of the developers, it is recommended to define consciously the maximum array dimensions instead of allocating the necessary memory without human control. The user must know if the available computational resources are enough in the case of (very) large systems instead of observing how the system kernel stops the program execution or the computer crashes. For this reason, the Fukui program includes these default cut-offs modifiable through a configuration file:

# atoms	# MOs	# points	# primitives
50	250	1000000	1000

To increase these modest cut-offs (50 atoms are not a lot), the user will create a file including the following lines: Line 1: Max number of atoms. Line 2: Max number of molecular orbitals. Line 3: Max number of points. Line 4: Max number of primitive functions.

Enabling parallelization

Fukui was developed with OpenMP parallelization (a calculation cannot be shared with other machines). To activate the full parallelization, employing all the threads of the computer, we only need to create an empty file with the wave function file name and replacing the current extension by the text “proc”:

```
▷ touch wavefunction.proc
```

In the case a specific number of threads is required, the user can set it in the aforementioned file:

```
▷ echo 8 > wavefunction.proc
```

Modifying the distance in the primitive gaussian functions skipping

See Section 3.4.

Skipping points with negligible Gauss quadrature weights

If the grid file was not processed previously removing negligible Gauss quadrature weights, the user can set the option to skip them. Typing the wave function file name and replacing the extension by “skg”, this option will be enabled:

```
▷ touch grid_file_name.skg
```

The default value is 1.0×10^{-10}, a very reasonable cut-off according to our tests, but it is modifiable adding a new value in the file:

```
▷ echo 1.e-9 > grid_file_name.skg
```

The program will print on screen and in the output file the number of points skipped and the total electron density with and without these negligible points.

Skipping points with negligible reference electron densities

If reference electron densities of the grid file were not processed previously, the user could be interested in skipping them. Typing the wave function file name and replacing the extension by “skr”, this option will be set:

```
▷ touch grid_file_name.skr
```

The default value is 5.0×10^{-9}, a very reasonable cut-off according to our tests, but it is modifiable adding a new value in the file:

```
▷ echo 1.e-9 > grid_file_name.skr
```

Setting σ/π separation or only including π spin orbitals in the calculation

For a lot of years in our research group, we have used a file that includes the identification of the σ/π symmetry of the spin orbitals obtained by a program we called “simon.x”, and the outputs use the “.som” extension. It includes some text including the identification and it is not the easiest way to save a list of π spin orbitals for an user from out of our group. For this reason, FUKUI is also compatible with a more simplified format:

Structure and format of a .sg file

#MO #piMO #sigmaMO #no_identifiedMO
π spin orbitals specification
σ spin orbitals specification
Specification of unidentified spin orbitals

Here is an example of the tetracene:

▷ **cat tetracene.sg**

```
60 9 51 0
46 48 51 55 56 57 58 59 60
1 2 3 [...] 42 43 44 45 47 49 50 52 53 54
```

Therefore, the tetracene has 60 MOs, 9 of which have π symmetry, and 51 are σ ones. The contents of the previous example can be saved in a file with the wave function file name and ".sg" extension.

When the user needs to obtain populations or AOMs defined by π MOs only, we can create an empty file with the extension ".pi" and the wave function file name:

▷ **touch wfn_file_name.pi**

The calculation will be notably faster in addition.

Activating the calculation of atomic overlap matrices (AOMs)

The atomic overlap matrices (AOMs) are needed as a previous step to calculate bond orders, exchange-correlation Fukui functions and/or N-delocalization indices. This program allows computing it for any region of the molecular space and/or atomic partitioning while a points grid is provided.

Similarly to other options, the user will create an empty file to activate this computation using the wave function file name and the ".aom" extension.

▷ **touch wfn_file_name.aom**

4 Output of a correct calculation

In this section we will discuss the program output on screen (Section 4.1) and the differences with the saved file (Section 4.2).

4.1 Output of a correct calculation on screen

The structure of the output on screen will be explained below. We will use all the options enabled to obtain the population, σ/π separation and the atomic overlap matrices of a .stk file for the optimized tetracene molecule obtained at B3LYP/6-31G** level. The first text we will observe is the version, in this case 220301. After that, the wave function and grid files we are using, that is, "tetracene.wfn" and "tetracene_C001.stk", respectively. To enable the parallelization, a .proc file is needed. If you will not use the full available computer threads, specify the number in the file. Next, we will see simple information about the parallelization and a "Hello" message coming from each thread. The command we use is:

▷ ./fukui.x tetracene.wfn tetracene_C001.stk 1 (part 1 of 6)

```
PROGRAM FUKUI 220301


>> Wave function file found: tetracene.wfn
>> Grid file using .stk format: tetracene_C001.stk
>> tetracene.proc was found. Parallelization enabled!!

**************** THREAD INFORMATION ******************
>> Job running using OpenMP.
>> The number of processors is .....    8
>> The number of threads is ........    8
Hello from process        0
Hello from process        7
Hello from process        5
Hello from process        2
Hello from process        3
Hello from process        4
Hello from process        1
Hello from process        6
>> Elapsed wall clock time ........ 0.1845E-02

******************************************************
* # threads to be used .................... 8
```

Next, the default maximum resources will be printed. If you need to change them, create the .mxal file following the instruction in Section 3.5.

▷ ./fukui.x tetracene.wfn tetracene_C001.stk 1 (part 2 of 6)

```
>> Using default max array allocations:
natomx      nommx       npmx      nprimx
50        250    1000000       1000
```

The program detects a .sg file. It will read the orbital symmetry specification. Due to the fact we add the third argument to specify the atom in which the grid is centered on, Fukui will skip gaussian functions far from this atom (with distance greater than 18.0 au). Next, we have created the .skg and .skr files and, therefore, the program will check the Gauss quadrature weight functions and density values lower than the cut-off printed. In addition, we set the calculation of the AOM creating the “tetracene.aom” file.

▷ ./fukui.x tetracene.wfn tetracene_C001.stk 1 (part 3 of 6)

```
>> .sg file detected. Sigma/Pi separation will be performed by using tetracene.sg
>> Atom 1 was specified as 3rd argument. Skipping gaussian functions.
* Distance in AU from this nucleus ........ 18.000
>> .skg file detected (tetracene.skg). Skipping points with low weight( 1.000E-10=cut-off)
>> .skr file detected (tetracene.skr). Skipping points with negligible ref. dens.( 5.000E-09=cut-off)
>> .aom file detected (tetracene.aom). Atomic overlap matrix will be calculated
```

The program will detect if the .stk file uses a Fortran unformatted file. Next, it will read both wave function and grid files, verifying all the components are correct and printing how many points will be skipped with our current settings (defined in the .skg and .skr files or using the default cut-off values). The following step, if a .sg or .som file is available, is to identify the π MOs. Before the real calculation starts, the program will print the percentage of gaussian functions considered. This option is very interesting to reduce drastically the dependency with the molecule size.

▷ ./fukui.x tetracene.wfn tetracene_C001.stk 1 (part 4 of 6)

```
>> Opening grid file: tetracene_C001.stk
* unformatted grid file detected after 23 iterations

[TT] Settings prepared in ...................... 0.00 seconds

>> Reading wave function file:
* 60 MOs, 588 gaussian functions, 30 atoms.
* Coordinates and nuclear charges ........ OK
* Primitive centers ...................... OK
* Primitive types ........................ OK
* Primitive exponents .................... OK
* MO coefficients ........................ OK

[TT] Wave function read in ..................... 0.02 seconds

>> Reading grid file with a set of points:
* 82466 points (total)
* Binary stk file
* 29154 of 82466 points were considered ->  64.65% skipped

[TT] Grid file read in ......................... 0.03 seconds

>> Opening sigma/pi separation file:
* PI orbitals read ....................... 46,48,51,55-60 PI(9)
>> Preparing calculation:
* Percentage of gaussian functions used ... 88.10%

[TT] Intermediate steps in ..................... 0.00 seconds
```

We will see how the program splits the grid points among the threads of the computer we are requested:

▷ ./fukui.x tetracene.wfn tetracene_C001.stk 1 (part 5 of 6)

```
>> Computing grid points:
* CPU 2: from 10933 to 14577 of 29154
* CPU 0: from 1 to 3644 of 29154
* CPU 5: from 7289 to 10932 of 29154
* CPU 1: from 14578 to 18221 of 29154
* CPU 6: from 3645 to 7288 of 29154
* CPU 3: from 25510 to 29154 of 29154
* CPU 7: from 21866 to 25509 of 29154
* CPU 4: from 18222 to 21865 of 29154

[TT] Grid points computed in ................... 0.40 seconds
```

The only results the program prints on screen are the total populations, removing the excessive data from the MO populations and the AOMs.

▷ ./fukui.x tetracene.wfn tetracene_C001.stk 1 (part 6 of 6)

```
RESULTS OF THE INTEGRATION
POP =          6.08561974959715D+00 au
REF POP =      6.08562096123212D+00 au
FUKUI = -0.000 au

Considering SKIPPING:
REF POP =      6.08562062048363D+00 au
DIF REF POP = 3.407D-07 au
FUKUI = -0.000 au

SIGMA/PI CONTRIBUTIONS:     9 molecular orbitals with PI symmetry
SIGMA    5.14519      PI    0.94043

[TT] TOTAL ELAPSED TIME ........................ 0.46 seconds
PERFORMANCE ............................... 89.84%
```

Each value represents:

POP	is the total population computed by the program from the wave function file.
REF POP	is the population from the .stk file (not computed using the wave function file).
FUKUI	corresponds to the difference between REF POP and POP. Depending on the kind of calculation we are performing could be the Fukui index.
REF POP	considering SKIPPING: this value represents the overall sum from the densities in the .stk file skipping points with negligible Gauss quadrature weights and electron density values.
DIF REF POP	is the subtraction between both REF POP values, with and without skipping. As you can observe, the condensed populations are affected in the 7th decimal position. Therefore, our settings are a very good approach.
FUKUI	considering SKIPPING is defined identically as the another FUKUI value but including only the smaller grid of points.

Finally, the results of the σ/π separation, the time spent on the execution and the performance are printed.

4.2 File output of a correct calculation

Part of the information on screen, discussed in Section 4.1, is also written in a file called following the scheme “wavefunction_grid_file.fuk”. We will discuss the new parts with respect to the output on screen (Section 4.1). Most of the additions are after printing the populations:

▷ cat wavefunction_grid_file.fuk (part 1 of 2)

```
      ORBITAL CONTRIBUTIONS:
       N( 1) =      0.000000 au
       N( 2) =      0.000000 au
       N( 3) =      0.000035 au
       N( 4) =      0.000035 au
       N( 5) =      0.000036 au
       N( 6) =      0.000036 au
       N( 7) =      0.424937 au
       N( 8) =      0.424963 au
       N( 9) =      0.337749 au
       N(10) =      0.337798 au

[...]

       N(57) =      0.065112 au
       N(58) =      0.140636 au
       N(59) =      0.101862 au
       N(60) =      0.081479 au
```

As you can see, FUKUI prints the orbital contributions for the total populations. Next, if the .sg or .som file is available, the σ/π separation and, finally, the AOM using a 8-column format of real values in scientific notation:

▷ cat wavefunction_grid_file.fuk (part 2 of 2)

```
            The Atomic Overlap Matrix



    0.525717804208E-08
    0.735609271729E-08   0.176756945118E-07
[...]
```

And finally the elapsed time and performance.

5 Utilities

Some small programs or utilities have been included. They are independent from the main program, and some of them (with .f90 extension) are compiled as the usual way (compiler source_code.f90 -o executable_name.x).

5.1 bstk_astk.f90

This small program transforms Fortran formatted .stk files (ASCII text) into unformatted ones (binary data) and vice versa. The program needs the reference .stk file (only include the file name without extension) and the type of file we are using. If no arguments, the program will ask for them.

5.2 cube2stk.f90

This utility transforms a cube file into an unformatted .stk file. It includes a short help and its use is “cube2stk.x cube_file atom_number”. The last argument is optional.

5.3 fukui.sh

This BASH script is useful to systematize the execution of a set of .stk files using the following file name scheme: “stkfilename_X001.stk”, being “stkfilename” the root file name of the .stk file, “X” the element symbol and “001” the corresponding atom number in the molecule. The script will take all the file names and arguments automatically. For example, “./fukui.sh tetracene_C001.stk” will call the Fukui program using the wave function called “tetracene.wfn”, and it will include the third argument “1” to enable the primitive skipping. The program will also create the .mxal with the resources needed.

5.4 mwfn2piorbs.sh

This BASH script will obtain the .sg file to carry out the σ/π separation using the program called Multiwfn. Edit the script to comply with the corresponding executable paths.

5.5 som2sg.sh

This BASH script is for internal use. It transforms .som into .sg files to perform the σ/π separation.

6 Examples

7 GNU/Linux benchmarks

8 How to cite Fukui 220331

- N. Otero, M. Mandado and R. Mosquera, Fukui 220331(2022), Universidade de Vigo.

Bibliography

(1) Bader, R. F. W., *Atoms in Molecules: A Quantum Theory*; International Ser. of Monogr. on Chem; Clarendon Press: 1994.

(2) Hirshfeld, F. L. *Theor. Chim. Acta* **1977**, *44*, 129–138.

(3) Bultinck, P.; Van Alsenoy, C.; Ayers, P. W.; Carbó-Dorca, R. *J. Chem. Phys.* **2007**, *126*, 144111.

(4) Geldof, D.; Krishtal, A.; Blockhuys, F.; Van Alsenoy, C. *J. Chem. Theory Comput.* **2011**, *7*, 1328–1335.

(5) Otero, N.; Mandado, M.; Mosquera, R. A. *J. Chem. Phys.* **2007**, *126*, 234108.

(6) Otero, N.; Mandado, M. *J. Comput. Chem.* **2012**, *33*, 1240–1251.

(7) Mandado, M.; González-Moa, M. J.; Mosquera, R. A. *J. Comput. Chem.* **2007**, *28*, 127–136.

(8) Mandado, M.; Otero, N.; Mosquera, R. A. *Tetrahedron* **2006**, *62*, 12204–12210.

A Appendices

A.1 Obtaining PROAIMS wave function file

We will see how to obtain a PROAIMS wave function file with an example for the Gaussian suite of programs. More concretely, we have considered the benzene molecule:

▷ cat C6H6.gjf

```
%nproc=4
%mem=1gb
%chk=C6H6.chk
# b3lyp 6-31g(d,p) 6d integral=ultrafinegrid
out=wfn

Title Card Required

0 1
C                  0.00000000    1.39499067    0.00000000
C                 -1.20809735    0.69749533    0.00000000
C                 -1.20809735   -0.69749533    0.00000000
C                  0.00000000   -1.39499067    0.00000000
C                  1.20809735   -0.69749533    0.00000000
C                  1.20809735    0.69749533    0.00000000
H                  0.00000000    2.49460097    0.00000000
H                 -2.16038781    1.24730049    0.00000000
H                 -2.16038781   -1.24730049    0.00000000
H                  0.00000000   -2.49460097    0.00000000
H                  2.16038781   -1.24730049    0.00000000
H                  2.16038781    1.24730049    0.00000000

C6H6.wfn
```

In order to print a .wfn file, the label *out=wfn* must be included. There exist equivalent forms of this label, such as *output=wfn*, *out=psi* and *output=psi*.

A.2 Converting from another file format to PROAIMS wave function

One of the most updated software with an active development to convert among formats is the Multiwfn program (`http://sobereva.com/multiwfn/`). In the version 3.7 the steps to obtain the .wfn file are shown below:

▷ **multiwfn C6H6.fchk**

```
Multiwfn -- A Multifunctional Wavefunction Analyzer
Version 3.7, release date: 2020-Aug-14
Project leader: Tian Lu (Beijing Kein Research Center for Natural Sciences)
Below paper ***MUST BE CITED*** if Multiwfn is utilized in your work:
Tian Lu, Feiwu Chen, J. Comput. Chem., 33, 580-592 (2012)
[...]
 Total/Alpha/Beta electrons:     42.0000     21.0000     21.0000
Net charge:     0.00000      Expected multiplicity:    1
Atoms:     12,  Basis functions:    120,  GTFs:    210
Total energy:    -232.113832873650 Hartree,   Virial ratio:  2.00945783
This is a restricted single-determinant wavefunction
Orbitals from 1 to    21 are occupied
Title line of this file: Title Card Required

Loaded C6H6.fchk successfully!
Formula: H6 C6
Molecule weight:        78.11206
Point group: D6h

"q": Exit program gracefully          "r": Load a new file
************ Main function menu ************
0 Show molecular structure and view orbitals
1 Output all properties at a point
2 Topology analysis
3 Output and plot specific property in a line
4 Output and plot specific property in a plane
5 Output and plot specific property within a spatial region (calc. grid data)
6 Check & modify wavefunction
7 Population analysis and atomic charges
8 Orbital composition analysis
9 Bond order analysis
10 Plot total DOS, partial DOS, OPDOS, local DOS and photoelectron spectrum
11 Plot IR/Raman/UV-Vis/ECD/VCD/ROA/NMR spectrum
12 Quantitative analysis of molecular surface
13 Process grid data (No grid data is presented currently)
14 Adaptive natural density partitioning (AdNDP) analysis
15 Fuzzy atomic space analysis
16 Charge decomposition analysis (CDA) and plot orbital interaction diagram
17 Basin analysis                     18 Electron excitation analysis
19 Orbital localization analysis     20 Visual study of weak interaction
21 Energy decomposition analysis
100 Other functions (Part 1)         200 Other functions (Part 2)
300 Other functions (Part 3)
```

▷ **Type "100" (without inverted commas) and the program will print a text similar to this:**

```
               ============ Other functions (Part 1) ============
0 Return
1 Draw scatter graph between two functions and generate their cube files
2 Export various files (mwfn/pdb/xyz/wfn/wfx/molden/fch/47/mkl...) or generate input file of quantum chemist
3 Calculate molecular van der Waals Volume
4 Integrate a function in whole space
5 Show overlap integral between alpha and beta orbitals
6 Monitor SCF convergence process of Gaussian
8 Generate Gaussian input file with initial guess from fragment wavefunctions
9 Evaluate interatomic connectivity and atomic coordination number
11 Calculate overlap and centroid distance between two orbitals
12 Perform biorthogonalization between alpha and beta orbitals
13 Calculate HOMA and Bird aromaticity index
14 Calculate LOLIPOP (LOL Integrated Pi Over Plane)
15 Calculate intermolecular orbital overlap
16 Calculate various quantities in conceptual density functional theory (CDFT)
18 Yoshizawa's electron transport route analysis
19 Generate promolecular .wfn file from fragment wavefunctions
20 Calculate Hellmann-Feynman forces
21 Calculate properties based on geometry information for specific atoms
22 Detect pi orbitals, set occupation numbers and calculate pi composition
23 Fit function distribution to atomic value
24 Obtain NICS_ZZ value for non-planar or tilted system
```

▷ **Type "2" (without inverted commas) and the program will print a text similar to this:**

```
0 Return
Export system to various formats of files:
1 Output current structure to .pdb file
2 Output current structure to .xyz file
3 Output current structure and atomic charges to .chg file
4 Output current wavefunction as .wfx file
5 Output current wavefunction as .wfn file
6 Output current wavefunction as Molden input file (.molden)
7 Output current wavefunction as .fch file
8 Output current wavefunction as .47 file
9 Output current wavefunction as old Molekel input file (.mkl)
31 Output current structure to .cml file
32 Output current wavefunction as .mwfn file
Generate input file of quantum chemistry codes:
10 Gaussian
11 GAMESS-US
12 ORCA                     13 NWChem
14 MOPAC                    15 PSI4
16 MRCC                     17 CFOUR
18 Molpro                   19 Dalton
20 Molcas                   21 Q-Chem
```

▷ **Type "5" (without inverted commas) and the program will print a text similar to this:**

```
Input path for exporting file, e.g. C:\ltwd.wfn
```

▷ **Type the file name: C6H6.wfn**

▷ **Exit following the menu codes.**

You will find the PROAIMS wave function file called "C6H6.wfn" in the current directory together with the reference file.

Another way to convert from .fchk to .wfn format is using the program AIMALL.