Manual for Aroma – 1.0

OS: Linux, Mac, Windows

Contents

This program should be cited as (a) Rahalkar, A.; Stanger, A. "Aroma", http://chemistry.technion.ac.il/members/amnon-stanger/ (b) Stanger, A. *J. Org. Chem.* **2006**, *71*, 883-893. (c) Stanger, A. *J. Org. Chem.* **2010**, *75*, 2281-2288. (d) Gershoni-Poranne R; Stanger, A. *Chem. Eur. J.* **2014**, *20*, 5673-5688.

1. Aroma: Introduction

Aroma is a utility package for evaluating aromatic properties via NICS methods. It is designed as a "Plug-In" utility for the computational chemistry package, Gaussian. It offers automated building of input files, calculations, and analysis of output files for the following methods:

(1) NICS-scan (reference b).
(2) σ-only model (reference c), including the manipulations for obtaining $\text{NICS}(1)_{\pi,\text{ZZ}}$.
(3) CMO-NICS, using the NICS-scan procedure and NCS procedure within NBO 5.G or NBO 6.
(4) NICS-XY-scan (reference d), including its σ-only model and ready-to-plot data for π-only NICS-XY-scan.

In addition, Aroma contains utilities that allow computations of other molecular properties that may be needed, such as calculations of the area of ring(s) in a system, further analysis of the data, etc. For details see utility scripts in the appendix.

Aroma is capable of handling multiple centers within a given molecule within a single run and to execute such multiple runs corresponding to different tasks by a single command.

2. Getting Started:

You can download Aroma in two versions: The source code and the binary version. The source code needs Python version 2.7, some libraries and modules which are not always installed on the computer (see below). The BIN version is a standalone version that needs only the Gaussian to run. We highly recommend using the BIN version.

▪ <u>For Linux/Mac:</u> aroma-1.0.tar.gz, aroma-1.0-bin.tar.gz

Use following command to un-tar:

"tar zxvf aroma-1.0.tar.gz" or "tar zxvf aroma-1.0-bin.tar.gz"

▪ For Windows: aroma-1.0.zip, aroma-1.0-bin.zip

Unzip and extract the folder "aroma" on the appropriate destination on your machine, or copy the downloaded file to the appropriate directory (for example, C:\G09W) and unzip it there.

▪ The BIN versions include binaries of Aroma scripts along with additional python modules (numpy, PIL etc.)

▪ If you are using the source version please note the following:

For using GUI and automated analysis of data, additional packages such as numpy, PIL are required, which can be found on:

http://www.numpy.org/

http://www.pythonware.com/products/pil/

Requirements:

✓ For using the source code only: Python (Interpreter for Python Programming Language), version 2.7. A proper version can be obtained from:

http://www.python.org/download/

✓ Gaussian with all paths properly set for Gaussian links and other supporting executables.

Integration with Gaussian: aroma_constants.py

In order to integrate Aroma with the Gaussian on your system, the following paths in the aroma_constants.py file (which is a text file and can be edited by any text editor, e.g., gedit, wordpad, gedit) must be set to appropriate values as per your system settings.

Representative setting for Linux / Mac:

```
inpdir = "/home/anuja/input/"
outdir = "/home/anuja/output/"
chkdir = "/home/anuja/chk/"
GaussCmd = "/usr/local/g09/g09 "
FormChkCmd = "/usr/local/g09/formchk "
GaussInpExt = ".in"
GaussOutExt = ".out"
```

Representative setting for Windows:

```
inpdir = "C:/G09W/input/"
outdir = "C:/G09W/output/"
chkdir = "C:/G09W/chk/"
GaussCmd = "C:/G09W/g09 "
FormChkCmd = "C:/G09W/formchk "
GaussInpExt = ".gjf"
GaussOutExt = ".out"
```

The above variable names related to Gaussian are straightforward. To be clear: inpdir, outdir and chkdir are the paths for the input, output and checkpoint files respectively. GaussCmd is the path for Gaussian executables. (Make sure to leave a space at the end of GaussCmd and FormChkCmd). GaussInpExt and GaussOutExt define the extensions to be used for Gaussian input and output files.

The aroma_constants.py file contains also all the defaults that the program uses. All these defaults can be changed in the arm file (see the keywords section). However, since usually the same machine is used (i.e., the same number of processors and the same amount of memory), the same functionals and basis sets for given tasks, a standard procedure for NICS-scan, etc., it is advisable to change the defaults in the aroma_constants.py file, so that writing the arm file becomes easy. Please pay attention especially to the default keyline section (which are Gaussian keywords for optimization, NICS-scan and NCS jobs).

Note: In the next examples of keywords and usage of utilities, the paths for linux are used. The windows user should replace them by appropriate paths on windows such as “C:/G09W”.

3. Usage:

a. Binary Version:

The GUI for Aroma can be launched by double clicking on the application named as `aroma_gui.exe` or by issuing `./aroma_gui` from command line when in aroma directory. The GUI allows user to select the .arm or .suarm file and initiate single or multiple run respectively by pressing appropriate button. Once the execution of Aroma job starts, the status and runtime output of the Aroma is displayed on the GUI.

Please note: In some versions and setups of Linux, the Aroma_gui does not work or does not interact properly with the Gaussian. If such a case happens, please open a terminal, cd to the aroma directory and activate the GUI by typing ./aroma_gui <enter>

b. Source code version:

Command Line:

A single calculation of Aroma should be executed by issuing following command from the prompt:

```
python <path for Aroma>/aroma.py <path/Prefix>
```

Here, path/Prefix = path and filename of the **.arm** file (without extension)

For instance: If Aroma package is installed at "**/home/anuja/aroma**" and the path for the arm file is "**/home/anuja/input/benz.arm**", then the command to run Aroma for **benz.arm** would be:

```
python /home/anuja/aroma/aroma.py /home/anuja/input/benz
```

For running multiple jobs of Aroma through a single command use:

```
python <path for Aroma>/aroma_su.py <path/Prefix>
```

Here, path/Prefix = path and filename of the **.suarm** file (without extension)

For instance: If Aroma package is installed at "**/home/anuja/aroma**" and the path for the arm file is "**/home/anuja/input/phenalene.suarm**", then the command to run Aroma for **phenalene.suarm** would be:

```
python /home/anuja/aroma/aroma.py /home/anuja/input/phenalene
```

GUI:

If a proper version of Python and other packages are available, the GUI for Aroma can also be started from command prompt using:

```
python <path for Aroma>/aroma_gui.py
```

4. Input:

Aroma needs two input files.

1. An arm or suarm file which contains Aroma keywords. The keywords for Aroma are explained in detail in the next section. For further details, please refer to the sample .arm and .suarm files in Sample directory of Aroma.
2. The molecular geometry in one of the following options: (1) a standard Gaussian input file (as Z-matrix or as Cartesian Coordinates). A Gaussian output file. (3) A checkpoint file. The path for this file is specified through the .arm or .suarm file.

5. Aroma Keywords:

This Section describes the keywords and control options for Aroma which are to be given through the **.arm** or .suarm file.

GEOMFILE

Description:

Compulsory.

This is a path for a file containing molecular geometry. This file can be a Gaussian input, output or checkpoint file.

Notes:

1. The extension of input, output or chk file should match with those defined in aroma_constant.py

2. Along with the geometry, the charge and multiplicity of the system are also taken from this file.

3. The geometries are considered to be in angstroms (and degrees in case that a z-matrix format is used). Even if the geometry is provided through checkpoint file, it is converted to angstrom units by Aroma.

```
Example:
GEOMFILE=/home/anuja/input/benz.in
# when GaussInpExt (in aroma_constants.py) is set as '.in'
Or GEOMFILE=/home/anuja/output/benz-opt.out
# GaussOutExt (in aroma_constants.py) is set as '.out'
```

RUN

Description:

Optional, default is always NICSSCAN.

Possible values and their meanings are described below.

NICSSCAN: Default, performs NICS-scan

The following keywords are optional and should be given in addition to NICSSCAN if the user wants.

XY: Performs NICS-Scan in XY-direction above the molecular plane at height specified by DEFAULT_XY_DISTANCE in aroma_constants.py file. The default is set to 1.7 Å (see reference d) but can be changed.

SIGMA: Performs NICS-scan calculations on a σ-only model of the original molecule (reference c)

NCS: Performs a NICS-scan and NCS analysis (included in NBO) for obtaining CMO-NICS

OPT: Performs geometry optimization (prior to NICS calculations)

Example:

`RUN=NICSSCAN,SIGMA` **# NICSSCAN calculations for original molecule and its σ-only model in Z-direction.**

Or

`RUN=XY,NICSSCAN,SIGMA` **# NICS-XY-SCAN calculations for original molecule and its σ-only model. Please note that XY,NICSSCAN is actually one keyword asking for the NICS-XY-procedure.**[d]

Or

`RUN=OPT,NICSSCAN,NCS` **# Geometry optimization followed by NICSSCAN and CMO-NICS calculation for the final optimized geometry.**

OPT_EXTERNAL

Description:

Optional

If RUN = OPT, then user may give path and filename for the input file for optimization directly using this keyword instead of automatic generation of this file by Aroma.

This is especially useful when user wants to perform constrained optimization using Z-matrix or modredundant or using an input file which does not contain a geometry (geom=CheckPoint) or use previous results (e.g., frequencies, geometry) from a checkpoint file.

Note: This file will be directly used for geometry optimization as it is. In other words, all the Gaussian keywords including method, basis set, chk, memory, processors etc. will be used from this file and not from Aroma.

Example:

OPT_EXTERNAL=/home/anuja/benz-opt.in # Geometry optimization will be performed using this file directly and the final geometry from the output will be obtained for usual proceedings of Aroma.

CENTER

Description:

Compulsory, at least one CENTER must be defined. The only exception is if NORMAL (see below) is defined.

List of atoms making up the rings or bonds for which center(s) a NICS scan is to be performed.

Notes: 1. Each ring/bond should be defined in a new line starting with the keyword Center.

2. In case of XY-Scan, the geometrical centers of bonds and rings define the direction in which BQs are generated. Therefore, it is vital that the centers are specified in proper sequence.

3. A single atom is accepted as Center only in case of XY-Scan.

4. When the input geometry is in Z-matrix format the dummy atoms are not counted for the “Center” data.

Example:

CENTER=1,2,3,4,5 # BQs will be generated starting from the geometrical center of the ring created by atoms with indices 1 to 5.

Or in case of XY Scan

CENTER=1,6

CENTER=1,2,3,4,5,6
CENTER=3,4 # The BQs are generated on lines joining the perpendiculars of GMs of the consecutive centers.

NORMAL

Description:

Optional, relevant only for NICS Scan in Z-direction

A vector along which the NICS Scan in Z-direction is to be performed.

Notes: 1. This vector is specified as a list of x,y,z coordinates of two points which defines the vector.

2. This is useful if the position for NICS Scan in Z-direction cannot be defined as center of any ring.

3. The BQs are generated starting from the first point in the list.

4. This is irrelevant for XY-Scan.

```
Example:

NORMAL=0.917,1.896,0.000,0.917,1.896,-1.000 # BQs will be generated starting from the first point (0.917,1.896,0.000) along this vector.
```

POINT

Description:

Optional, relevant for NICS Scan in XY-plane.

Cartesian coordinates of a “point” in the sequence of XY centers which is included in the trajectory of XY-Scan.

Notes: 1. This is useful if certain point which cannot be defined as an atom, center of a bond or center of a ring, is to be included in the trajectory.

2. Since there is no ring-plane to define the perpendicular direction on this point, a normal vector for this point is aligned according to those on the previous and next ring-centers. In other words, the trajectory between the previous “center” and the point and/or the point and the next center will be raised to a distance above the systems, as defined in the aroma_constants.py file (e.g., 1.7 Å).

4. “Point” is irrelevant for scan in Z-direction.

```
Example:

run = xy, nicsscan, sigma
center:3,20,21,22,23
point:-1.713,-1.680,0.000
center:1,5,6,7,8

# The trajectory is defined as vectors joining first center to point to the second center. The perpendicular for the point is average of those for the first and the second centers.
```

AROMATIC RING – END

Description:

Compulsory, if RUN = SIGMA.

If σ-only model calculations are requested, the list of all aromatic (and/or antiaromatic) rings is to be provided. The σ-only model will be generated by adding H-atoms above the atoms forming all of these rings.

```
Example:

AROMATIC RING
1,2,3,4,5,6
3,4,7,8,9
```

```
END
```

# This means there are two aromatic/antiaromatic rings in the molecule under consideration. Even if the NICS-scan calculations are to be performed only on one ring (for example, 1,2,3,4,5,6) which is defined via keyword CENTER, the keyword AROMATIC RING should list all of the rings, in order to generate the correct model. See example 3 in Illustrations.

EXOCYCLIC - END

Description:

Optional, if RUN=SIGMA

If there is an exocyclic atom attached to a ring-atom, where the user would like to add dummy H-atom while generating the σ-only model, this keyword is to be used.

Example:

```
AROMATIC RING
1,2,3,4,5,6
2,3,15,11,13
END
EXTRACYCLIC
11,12
END
```

# This means there is an exocyclic atom indexed 12 attached to ring atom 11, for which dummy H-atom is to be added. See example 4 in Illustrations. This option should be used if an exocyclic double bond (e.g., =CH_2, =O) that is part of the conjugation in the system exists.

BQSTEP

Description:

Optional. Default is DEFAULT_BQ_STEP = 0.1 in aroma_constants.py

Step size or distance between BQs in Å.

Please note that this value is applicable for NICSSCAN and XY, NICSSCAN

Example:

```
BQSTEP = 0.2 # BQs will be generated at 0.2 Å distance.
```

BQRANGE

Description:

Optional. Default is DEFAULT_BQ_RANGE = [0,4] in aroma_constants.py

The range of distance (in Å) for BQs assuming that the geometrical center

(GC) of the ring is the Origin.

Example:

BQRANGE=0,2 # The BQs are generated on one side of the ring starting from GC (i.e. Origin) going up to 2 Å.

BQRANGE=-2,2 # The BQs are generated on both sides of the ring starting from -2 Å, going up to 2 Å.

ANALYSE

Description:

Optional.

If RUN = NICSSCAN, SIGMA, Aroma processes the data and determines the chemical shifts at distance of 1 Å above the molecular plane as the average value of Δoopc and 3Δiso (see reference c).

Polynomials are fitted for data of BQs from distance 1.1 Å and above, set by DEFAULT_DISTANCE_FOR_ANALYSIS in aroma_constants.py.

Notes: (1) Only for the source code users: This keyword is functional only if the module “numpy” is available. (2) Not applicable for NICS-XY-scan.

Example:

NO ANALYSE

# Does not perform analysis

Or ANALYSE = 1.5

# For polynomial fitting, data of BQs from distance 1.5 Å and above is considered

ANALYSE AREA

# In addition to analysis, also calculates the area for all the given aromatic rings as defined in the “aromatic rings” section.

OUTFILE

Description:

Optional.

If RUN = SIGMA, the results of analysis are stored in this file. By default the results are stored in a file with the same filename as that of “.arm” file but has extension “.armlog” in the output directory “outdir”.

Example:

OUTFILE=/home/anuja/output/benz-aroma.log

# The chemical shifts for each center will be stored in this file instead of “benz.armlog”.

SONLY CHARGE

Description:

Optional.

User defined charge on σ-only model, if user does not want to use the automatically calculated charge.

```
Example:
SONLY CHARGE = -1 # Charge of -1 will be used for σ-only model
```

KEYLINE – END KEYLINE

Description:

Optional.

The lines of keywords for running Gaussian.

These have further sub-keywords :

NICSSCAN_TEMPLATE

Description:

Optional, with following defaults in aroma_constants.py.

DEFAULT_NICS_KEYLINE = "%nproc=1\n%mem=1024MB\n# B3LYP/6-311+G* NMR=GIAO\n"

The % keywords such as chk, nproc, mem etc. and the route command starting with #, specifying the level and basis set etc. These lines are attached to the NICS-SCAN input for each “Center” as specified.

NCS_TEMPLATE

Description:

Optional, with following default parameters in aroma_constants.py.

DEFAULT_NCS_KEYLINE = "%nproc=1\n%mem=1024MB\n# B3LYP/6-311+G* NMR=GIAO IOP(10/46=1) POP(NBOREAD, FULL)\n"

Template for NICS-scan run with Gaussian with NCS run by NBO package.

Note: If RUN = NICSSCAN, NCS then the keylines should be specified under **NCS_TEMPLATE** and not under **NICSSCAN_TEMPLATE**, or Aroma will use the defaults for NCS keylines.

OPT_TEMPLATE

Description:

Optional, with following default parameters in aroma_constants.py.

DEFAULT_OPTIMIZATION_KEYLINE = "%nproc=1\n%mem=1024MB\n# B3LYP/6-311G* OPT \n"

Template of keywords for optimization if RUN = OPT

NBO_TEMPLATE

Description:

Optional, with following default parameters in aroma_constants.py.

DEFAULT_NBO_KEYLINE = "$NBO NCS=0.1 <I MO XYZ> $END\n"

As per requirement of NBO package, these keywords are attached at the end of the Gaussian input file.

Note:

1. Since **KEYLINE – END KEYLINE** is a complete set of keywords, there should not be extra blank lines in this section.

2. The order of sub-keywords does not matter.

3. If your **TEMPLATE** contains "%chk" keyword, then Aroma will add an appropriate filename for each "Center". Please refer to the example below.

4. If you are using cinstant settings for optimization, NICS-scan and CMO-NICS it is advisable to change these parameters in the aroma_constants.py file and avoid the use of the KEYLINE commands.

```
Example:

KEYLINES
OPT_TEMPLATE
%chk=/home/anuja/chk/benz-opt.chk
%nproc=2
%mem=4000mb
# b3lyp/6-311g* opt
NCS_TEMPLATE
%chk=/home/anuja/chk/benz.chk
%nproc=2
%mem=4000mb
# b3lyp/6-311+g* nmr=giao iop(10/46=1) pop(nboread, full)
END KEYLINE

# Instead of the defaults the Gaussian calculations will be
performed at these specified levels of theory with specified
```

```
hardware controls.
As “chk” is specified in the keylines:
1.The specified chkfile will be generated for the optimization.
2.For NICS-scan and CMO runs, %chk=/home/anuja/chk/benz-center1.chk
will be used for center no. 1 and so on for the rest of the
centers, assuming chkdir=“/home/anuja/chk/” as in
aroma_constants.py.
```

CLEAR

Description:

Optional.

During Aroma run, a lot of input and output files are generated. This keyword removes all the intermediate files including inputs and outputs corresponding to centers.

Example:

```
CLEAR # This should be in new line anywhere in the .arm file.
```

Following are keywords for the **.suarm** file for multiple Aroma jobs.

RUNNAME

Description:

Compulsory.

These labels are used as suffix to generate .arm files.

Example:

```
See Example 3 in Illustration Section.
```

COMMON – COMMON END

Description:

Optional.

List of common keywords to be added in each of the .arm file.

Example:

```
See Example 3 in Illustration Section.
```

Notes:

1. All the keywords are **case-insensitive**.

2. The order of keywords and extra blank lines between keywords do not matter.

3. If the molecule does not possess overall planarity, then for each ring, the entire molecule will be re-oriented in such a way that the ring under consideration lies in XY plane. This allows the program to clearly define the in-plane and out-of-plane directions.

Constants:

The paths and constants in aroma_constants.py are categorized into 3 types: 1. Atom and molecular information such as relationships between atomic numbers to symbols, atomic masses, covalent radii etc. and a few tolerance thresholds to define bonds and planarity. 2. Paths, commands and file extensions related to Gaussian. 3. Defaults set for Aroma.

The important ones are described below.

- AtmSym, AtmMass, AtmCovalentRadii: are the lists of atom-information. For example,

```
AtmSym={'H':1,'HE':2,'LI':3,'BE':4,'B':5,'C':6,'N':7,'O':8,'F':9,'SI':1
4}
```

  "AtmSym" correlates atom symbol to atomic number. On similar lines, AtmMass and AtmCovalentRadii map atomic number to atomic mass and covalent radius, respectively.

  If the molecule under consideration contains an atom, which is not present in these lists, then all three of these lists should be updated with the relevant information for that atom.

- TORSION_ANGLE_TOLERANCE: Allows a defined quantity of non-planarity. User may increase this tolerance if the molecule is not planar.
- Constants related to σ-only Model:

  FIXED_SIGMA_ANGLE: The angle between the geometrical center of ring, the ring atom and the newly added H-atom in degrees.

  ATM_H_BL: List of bond-lengths at which the new H-atoms should be added for each element

- For description of Gaussian related constants please refer to Section 2 and for the rest of the Aroma-defaults (keylines, stepsize and range of BQs etc.) refer to Section of keywords

6. Outputs:

Aroma generates output files with following extensions.

1. armdat: Text files corresponding to each center and can be opened via standard software such as notepad, Microsoft excel etc.

The format of .armdat file is as follows.

```
#    oop    in1    in2    inp    iso    x    y    z
```

where, "**#**" specifies the number of BQ,

"**oop**" stands for the out-of-plane (the eigenvalue that is closest to the ZZ tensor) chemical shift,

"**in1**" and "**in2**" are the in-plane (the eigenvalues that are closest to the XX and YY tensor) chemical shifts,

"**inp**" is the average of "**in1**" and "**in2**",

while "**iso**" depicts the isotropic chemical shift and

"**x**", "**y**" and "**z**" stand for the XX, YY and ZZ tensors, respectively.

For XY-Scan, the BQs are clubbed in sets of 50 and the input-output files are respectively labeled as center1, center2 and so on. However, at last, all the data is collected and saved in a file with suffix-extension "**-allcenter.armdat**" with above-mentioned format. If it is XY-Scan with σ-only model, another file with suffix "**-alldiff.armdat**" is additionally generated.

```
r     ZZ     Sigma-ZZ     Del-ZZ
```

where, "**r**" specifies the distance of BQ from the starting BQ at 0,

ZZ and **Sigma-ZZ** are columns of ZZ-component of tensors while **Del-ZZ** is the difference between them, representing the π-only contribution to the BQs.

2. picmo: For NCS keyword, these text files are generated for each center and can be opened via standard software such as notepad, Microsoft excel etc. The contributions from canonical π-MOs to NICS are stored in the .picmo files if

RUN=NCS. If the π-MOs are identified as 16, 20, 21, then the .picmo file will have:

```
pi-MO #  16    20    21  -Sum
```

where, **-Sum** denotes the negative of the sum of contributions from all the MOs. Please note: (a) CMO-NICS is well defined only for planar systems, where the π CMOs are orthogonal to the σ CMOs. (b) Since the identification of the π CMOs is not easy, it is advised to check if Aroma has correctly identified the π-CMOs. A picmo file can be generated manually as well (see below section 5c, aroma_picmo.py).

3. armlog: These are the default output files for the Aroma runs (unless, the filename is specified differently by keyword "OUTFILE"). For NICS-Scan in Z-direction with σ-model, this file contains the details of 3rd degree polynomial, area (if requested) and final chemical shift for each center. On the other hand, for NICS-Scan in XY-direction, there is no chemical shift assigned but this file contains the information of BQs above the given (in the arm file) centers. This information is helpful to understand the minima, maxima and other features of the curve in the XY-Scan.

7. Illustrations

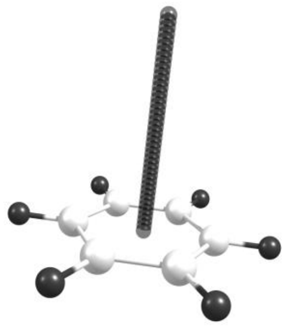
This Section explains the necessary details of the illustrative calculations which are included in Sample folder of Aroma.

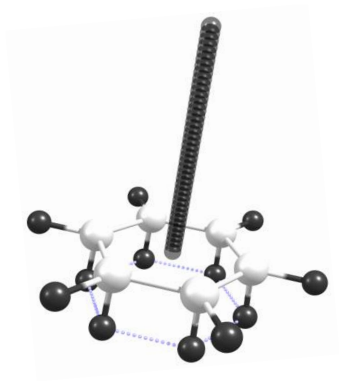
1. Benzene

Input: benz.arm, benz-opt.log

Output: benz-center1.armdat, benz-sigma-center1.armdat, benz.armlog

Description: A NICS-Scan in Z-direction with σ-only model.





benze-center1 benze-sigma-center1

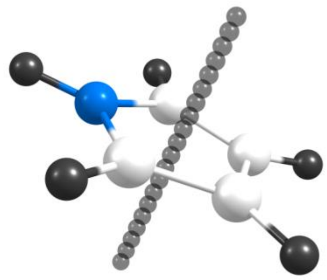
NICS$_{\pi ZZ}$ = -35.26 ± 3.75

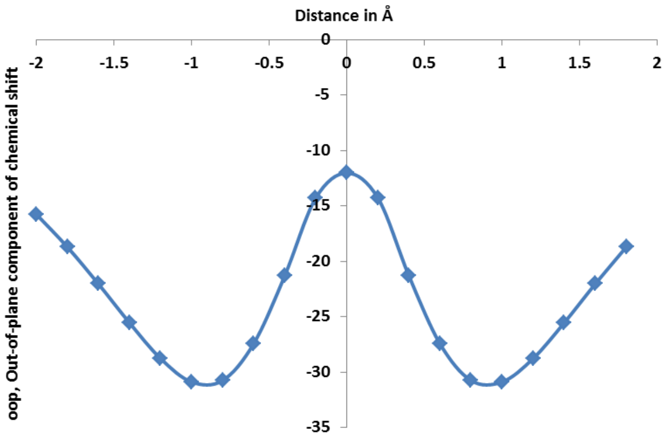
2. Pyrrole

Input: pyrrole.arm, pyrrole.in

Output: pyrrole-center1.armdat

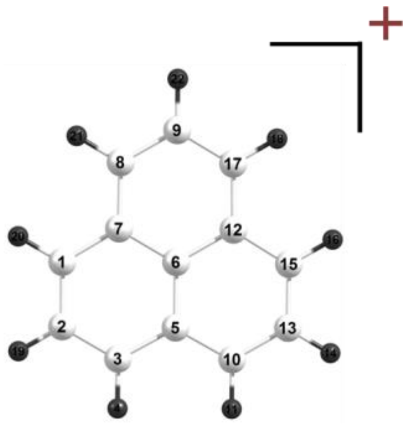
Description: A NICS-Scan in Z-direction for above and below the plane of the molecule. BQRange = -2, 2 and BQStep = 0.2 Å.



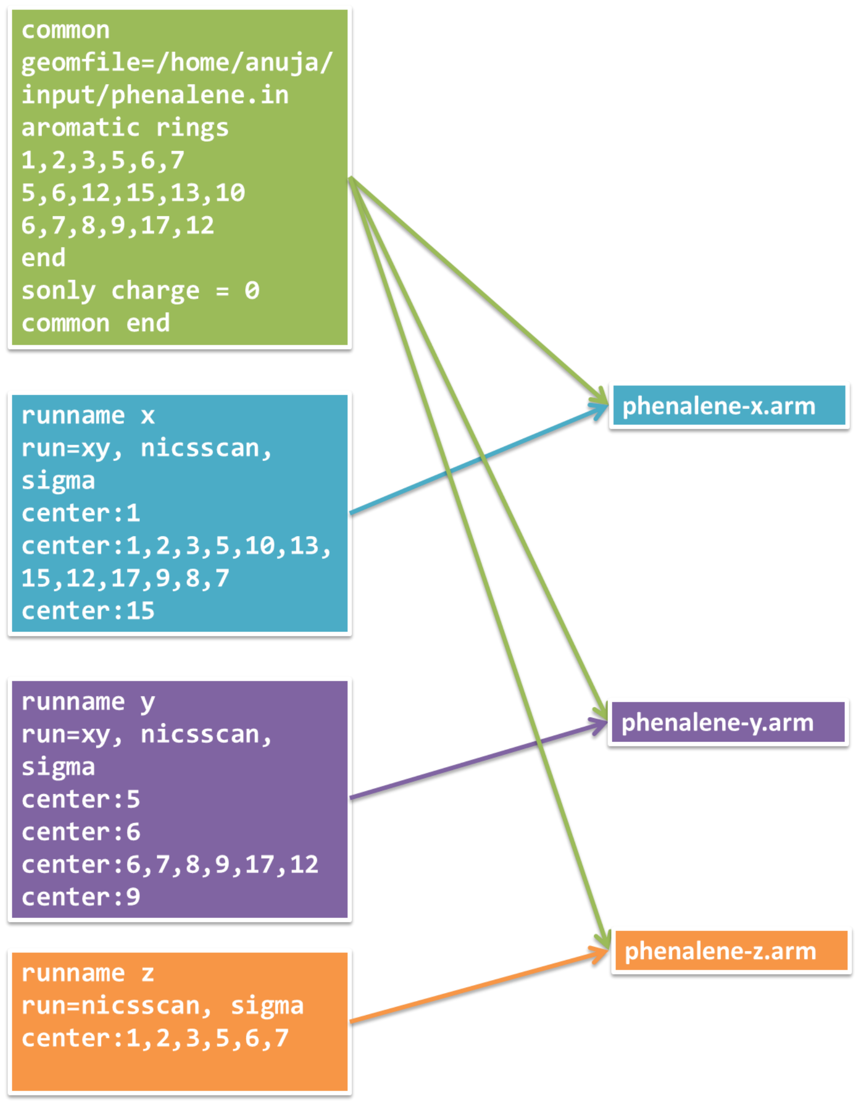


3. Phenalenyl Cation

Input: phenalene.suarm, phenalene.in



The super-arm file contains information for 3 Aroma runs named as “**x**”, “**y**” and “**z**”. Refer to following graphics for keywords “COMMON” and “RUNNAME”.



```
common
geomfile=/home/anuja/
input/phenalene.in
aromatic rings
1,2,3,5,6,7
5,6,12,15,13,10
6,7,8,9,17,12
end
sonly charge = 0
common end
```

```
runname x
run=xy, nicsscan,
sigma
center:1
center:1,2,3,5,10,13,
15,12,17,9,8,7
center:15
```

```
runname y
run=xy, nicsscan,
sigma
center:5
center:6
center:6,7,8,9,17,12
center:9
```

```
runname z
run=nicsscan, sigma
center:1,2,3,5,6,7
```

Notes:
1. For a scan parallel to molecular plane (XY-Scan), single atoms or bonds can be given as center. However, it is a must to specify at least one ring. Therefore, in this example, one **can not** specify ring centers as:
center:1
center:6
center:15

2. In this example, none of the centers of three six-membered rings is included in the scan-path. Therefore, the ring comprised of peripheral atoms is specified, which has the C-atom numbered 6 at its geometrical center.

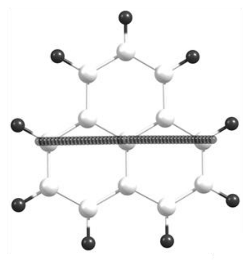
3. When multiple jobs are run using super-arm file, it is advised to use "CLEAR" keyword to avoid piling up intermediate input and output files.

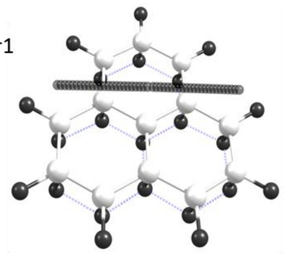
4. For XY-Scan, the sequence of centers is vital as it defines the path of scan.

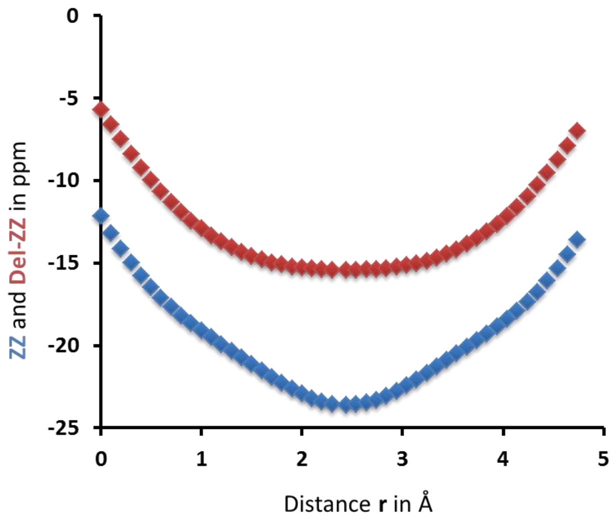
Run "**x**":

Output: phenalene-x-allcenter.armdat, phenalene-x-sigma-allcenter.armdat , phenalene-x-alldiff.armdat, phenalene-x.armlog

Description: A NICS-Scan in X-direction with σ-only model.


phenalene-x-center1

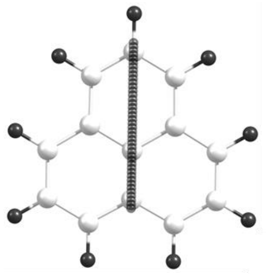

phenalene-x-sigma-center1

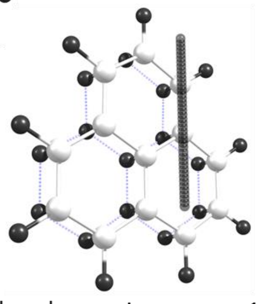


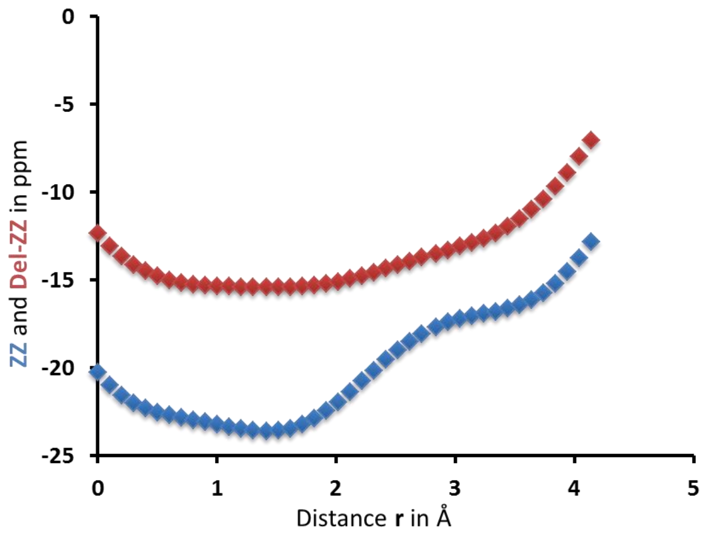
Run "**y**":

Output: phenalene-y-allcenter.armdat, phenalene-y-sigma-allcenter.armdat, phenalene-y-alldiff.armdat, phenalene-y.armlog;

Description: A NICS-Scan in Y-direction with σ-only model.


phenalene-y-center1

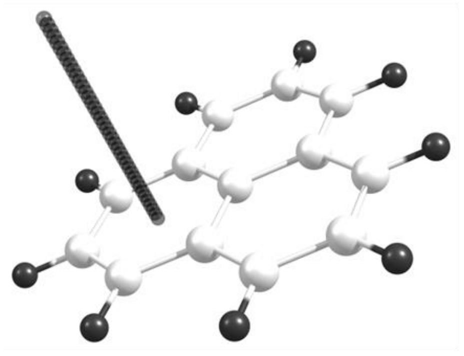

phenalene-y-sigma-center1

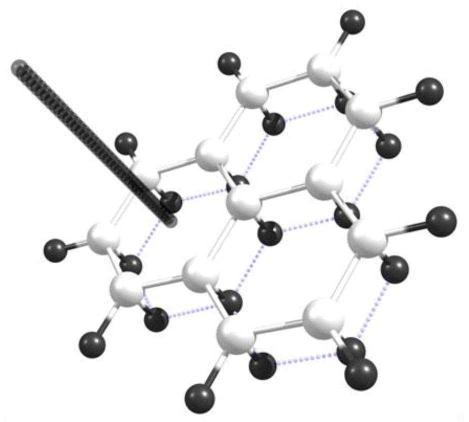


Run “**z**”:

Output: phenalene-z-center1.armdat, phenalene-z-sigma-center1.armdat, phenalene-z.armlog

Description: A NICS-Scan in Z-direction with σ-only model.

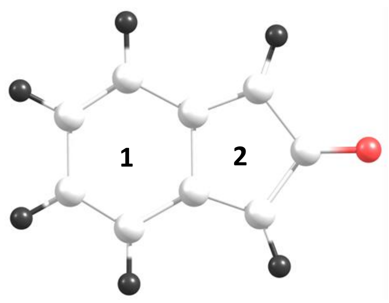

phenalene-z-center1

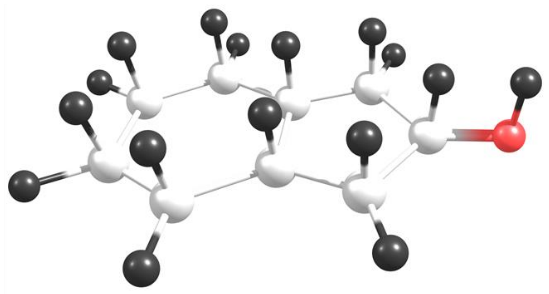

phenalene-z-sigma-center1

$\text{NICS}_{\pi ZZ}$ = -24.16 ± 2.79

4. 2H-indene-2-one

Description: Molecule of 2H-indene-2-one and its σ-only model.





Ring 1: NICS$_{\pi ZZ}$ = 28.17 ± 2.14

Ring 2: NICS$_{\pi ZZ}$ = 51.96 ± 2.37

The .arm file for this example is built as given below.

```
geomfile=/home/anuja/output/indene-opt.log
run=nicsscan, sigma
center:1,2,3,4,5,6
center:2,3,15,11,13
aromatic rings
1,2,3,4,5,6
2,3,15,11,13
end
exocyclic
11,12
end
sonly charge = 0
```

Please note the usage of “EXOCYCLIC” keyword. Atom number 11 in center #2 has an exocyclic atom attached to it which is indexed 12 in the geometry file.

8. Appendix

a. Technical Details of Program

This section describes the technical details of Aroma such as the code-structure, flow of the program and the important function calls.

The package contains a set of python scripts viz. aroma.py, aroma_su.py, aroma_parser.py, aroma_util.py, aroma_molecule.py, aroma_pinbo.py, aroma_analysis.py, aroma_ringarea.py and aroma_constants.py.

aroma.py:

This script contains the main function which drives the package. The important functions from this script are:

check(): reads .arm and checks for the validity and completeness of the keywords.
run_Optimization() and run_NICS(): functions which run Gaussian as the back-end for geometry optimization and NICS-scan for all the specified rings/bonds.
generateBQs(), generateBQs_XY(), genNicsInputs(): the former two are responsible for generating BQs respectively at the geometrical center of each ring or bond and parallel to the molecule in XY-plane while the latter generates corresponding Gaussian input files.
genSigmaModel(): this generates the σ-only model for the original molecule using the information of aromatic rings provided by the user.

aroma_su.py:

This is a wrapper script to call Aroma multiple times via a super-arm file with extension of .suarm. It contains only one function main(), which generates appropriate .arm files and runs Aroma for each of them.

aroma_parser.py:

This contains a python “class” called “FileParser” which is responsible for parsing various standard formats of Gaussian viz. input, output or checkpoint for reading the geometry of the molecule, charge, multiplicity and other relevant data.

aroma_util.py:

A few small general-purpose utility functions such as reading a file, determining geometrical center, average of values etc. are included in this script. All the geometry related functions such as determining distance, angle, dihedral angle and other vector related functions are also included in this script.

aroma_molecule.py:

This contains functions for generating the connectivity matrix for the given molecule and then identifying the rings. Also, the functions for setting planarity of the rings, reorienting the molecule into XY plane, generating z-matrix etc. are coded in this script.

aroma_pinbo.py:

This file contains a function "identifyPiMOs()" which identifies the π molecular orbitals and "grepPiCMO()" which finds the relevant data and stores it in the corresponding ".picmo" file for each ring.

aroma_analysis.py:

This file contains a function "analyse()" which processes the data for BQs from .armdat files, fits 3rd degree polynomial and determines the average $\mathrm{NICS}(1)_{\pi,zz}$ value with the possible amount of error.

aroma_ringarea.py:

This file contains a function "ring_area()" which determines the area of the requested ring(s).

aroma_constants.py:

This script sets all the constants, paths, default values for variables, and therefore, this is the only file which needs to be modified by the user. These settings are divided into atomic information, Gaussian paths and commands and defaults for Aroma etc.

b. Sigma Model

When σ-only model calculation is requested by the user, Aroma first calculates a NICS-scan for the original molecular system, which is followed by a NICS-scan for σ-only model. For this option, the user has to supply list of all aromatic rings in the molecule via .arm file, as this information is required to build the σ-only model. After adding H-atoms to involve the π-electrons in bonding with these H-atoms, Aroma also determines the charge on the system. However, it is possible to provide this information externally through the .arm file. For more details on the σ-only model, please refer to reference c.

c. Utility Scripts

aroma_picmo.py, an external utility script is a part of the Aroma package. It is a post-Aroma run utility for filtering CMO-NICS data for user-specified molecular orbitals.

Usage:

```
python <path for Aroma>/aroma_picmo.py <outdir/Prefix>
                <MO indices separated by space>
```

Example:

```
python /home/anuja/aroma/aroma_picmo.py /home/anuja/output/benz-ceter1.out 14 20 21
```

Here, a file named as "benz-center1.picmo" will be created in /home/anuja/output/ directory and it will contain the CMO-NICS data for molecular orbital no.s 14, 20 and 21.

The function analyse() from **aroma_analysis.py**, is called by default in the Aroma-run if SIGMA is requested, and the output is stored in respective file. However, the user can also call it externally for processing and fitting the existing data in form of .armdat files to save the manual efforts of analysis. The output of this script is printed on the screen and it contains the 3rd polynomial regression coefficients for Δoop and 3Δiso, the calculated chemical shifts at the standard distance of 1 Å above the molecular plane with its uncertainty (for the definition of Δoop, 3Δiso and the calculation of the chemical shift ± uncertainty see reference c in page 1 of this manual).

Usage:

```
python <path for Aroma>/aroma_analysis.py <armdat filename for main
        molecule> <armdat filename for σ-only model>
```

Example:

```
python /home/anuja/aroma/aroma_analysis.py benz-ceter1.armdat benz-sigma-ceter1.armdat
```

Here, it will consider BQs from default distance 1.1 onwards for fitting the polynomial.

```
python /home/anuja/aroma/aroma_analysis.py benz-ceter1.armdat benz-sigma-ceter1.armdat 1.5
```

Here, it will consider BQs from distance 1.5 onwards for fitting the polynomial.

The function ring_area() from **aroma_ringarea.py**, is called by default in Aroma-run in conjugation with analyse(). It can be used externally for determining the area of the ring as described below:

Usage:

```
python <path for Aroma>/aroma_ringarea.py <geomfile> <atom indices of the ring separated by spaces>
```

Example:

```
python /home/anuja/aroma/aroma_ringarea.py /home/anuja/input/benz-ceter1.in 1 2 3 4 5 6
python /home/anuja/aroma/aroma_ringarea.py /home/anuja/output/benz-ceter1.log 1 2 3 4 5 6
```

Here, the "geomfile" can be any standard Gaussian input or output. Note: 1. The sequence in which the atom indices is inputted, does not matter. 2. While calculating the area, the molecule is reoriented in such a way that the requested ring falls on XY plane and Z-coordinates are ignored. Therefore, if the ring is not planar, the determined area may be in error.

d. PBS system

This package as it is runs on PBS queuing system. Please note following:

- The usage command as described in Section 3.a should be given through the PBS script
- In case your PBS needs stagein and stageout to be specified:

1. Stagein: .arm and the input/output/chk file as specified in geomfile.

2. Stageout: Gaussian outputs, .amrdat, .picmo (in case of NCS) corresponding to all the centers, all those corresponding to sigma-model and the optimization output in case optimization is requested prior to NICS scan calculations.