py.Aroma 3

A Multi-functional tool for theoretical analyses of aromaticity.

Program User Manual

Version 3.0.0, Build on April 2nd, 2023.

– Developer –

Zhe Wang, *Ph.D.*

[Official Site] https://wongzit.github.io/program/pyaroma

1 Overview

1.1 About

py.**Aroma 3** is a multifunction tool for aromaticity analysis. It supports BLA, HOMA, POAV and NICS analysis. In addition, the basic function of **CSIgen**, which is a free open-source tool for generating computational supporting information/material, is also combined into py.**Aroma 3**.

Source code of py.**Aroma 3** including following file:

`pyaroma_main.py`: GUI code of py.**Aroma 3**.
`CONSTANT.py`: Including some constants like atom radius and HOMA parameters.
`readFile.py`: Read input files and extract geometry.
`geomAnalyzer.py`: Find connectivity and cyclic unit in molecules.
`NICSInp.py`: Create NICS input files.
`NICSout.py`: Extract shielding tensors from NICS output files.
`homaCalc.py`: Compute HOMA values.
`pathCreator.py`: Create path for 1D NICS scan.
`poav.py`: Compute POAV1 and POAV2.
`assets`: A folder including necessary images.
`config.ini`: Including program setting parameters.

1.2 Running Program

py.**Aroma 3** is built with Python 3.10 and the GUI is built with PyQt6, and relies on the libraries: NumPy, Matplotlib, NetworkX and OpenPyXL. The source code has been tested on macOS 13.2, Microsoft Windows 11 Pro (Update 22H2) and Red Hat Enterprise Linux 8.7.

To run py.**Aroma 3** with source code, please download all files in "*src*" folder on your computer, and execute command: `python3 path_to/src/pyaroma_main.py` in *Terminal* or *PowerShell*, *etc*. Please notice that, before first running of py.**Aroma 3** in source-code mode, the libraries (PyQt6, Numpy, Matplotlib, NetworkX and OpenPyXL) must be already installed on your computer, if not, please run command: `pip3 install pyqt6 numpy matplotlib network openpyxl` before running py.**Aroma 3** for the first time.

Pre-packaged executable files for macOS and Microsoft Windows are also available from homepage of py.**Aroma 3**. For macOS users, please save the *py.Aroma 3.app* to `/Applications` folder, otherwise the program may not work normally.

1.3 Startup

Main window of py.**Aroma 3** is shown in Figure 1.1. The function of each button in tool bar is described in Table 1.1.

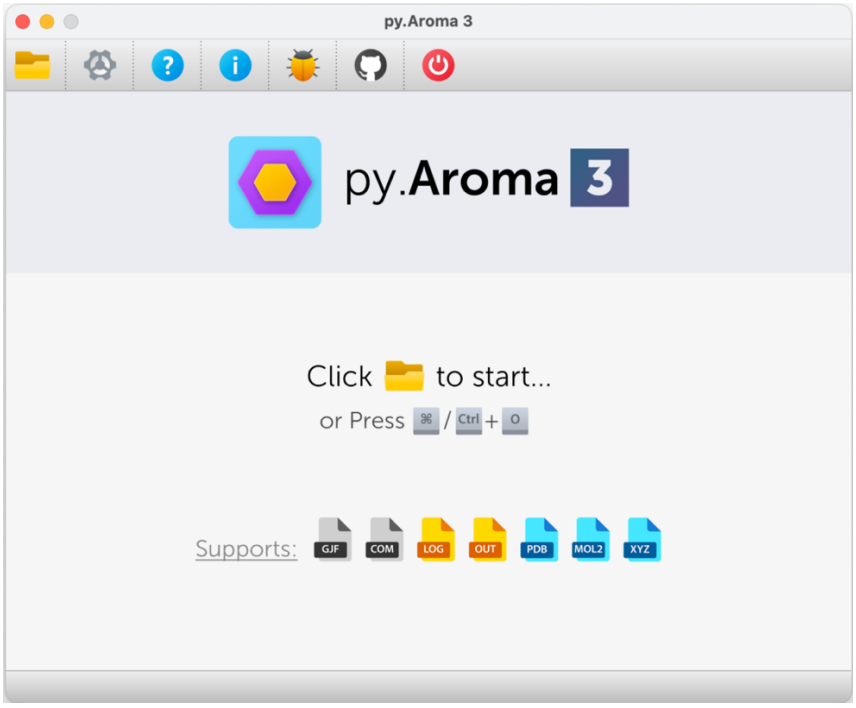


Figure 1.1 Main window of py.**Aroma 3**.

Table 1.1 Basic functions of tool bar in main window.

Functions	Icon	Description
Open		Browse a file for processing.
Settings		Open program setting panel.
Online Documents		Open online user manual.
About		About this program.
Bug Report		Report bugs to developers via e-mail.
GitHub Page		Check the GitHub page of developer.
Quit		Quit program.

2 General

This chapter introduces some basic information about py.**Aroma 3**.

2.1 File System

Three types of files could be read by py.**Aroma 3**:

Type-I: Chemical structure format: *.pdb*, *.xyz*, *.mol2*; *Gaussian*-type input file: *.gjf*, *.com*

Type-II: *Gaussian*-type output file *.log* or *.out* without ghost atom

Type-III: *Gaussian*-type output file *.log* or *.out* with ghost atom(s)

Different functions would be active when reading different files, the functions available for each type of files are summarized in Table 2.1.

Table 2.1 Basic functions of py.**Aroma 3**.

Functions	Icon	Type-I	Type-II	Type-III
Bond Length Alternation		○	○	○
HOMA		○	○	○
POAV		○	○	○
Create input file for single point NICS		○	○	–
Create input file for 1D NICS scan		○	○	–
Create input file for 2D NICS/ICSS		○	○	–
Create input file for 3D NICS/ICSS		○	○	–
Generate computational supporting information		–	○	–
Analyze output of single point NICS		–	–	△
Analyze output of 1D NICS scan		–	–	△
Analyze output of 2D NICS/ICSS		–	–	△
Analyze output of 3D NICS/ICSS		–	–	△

*△: Depends on file name, only one of these functions could be active depends on different output files.

2.2 Program Setting Panel

Program setting panel is shown in Figure 2.1. User could set some common options and HOMA parameters here. If the setting has been changed and saved, it would not go back to default even restart py.**Aroma 3**. To go back to default setting, user need to click "Default" and "Save".

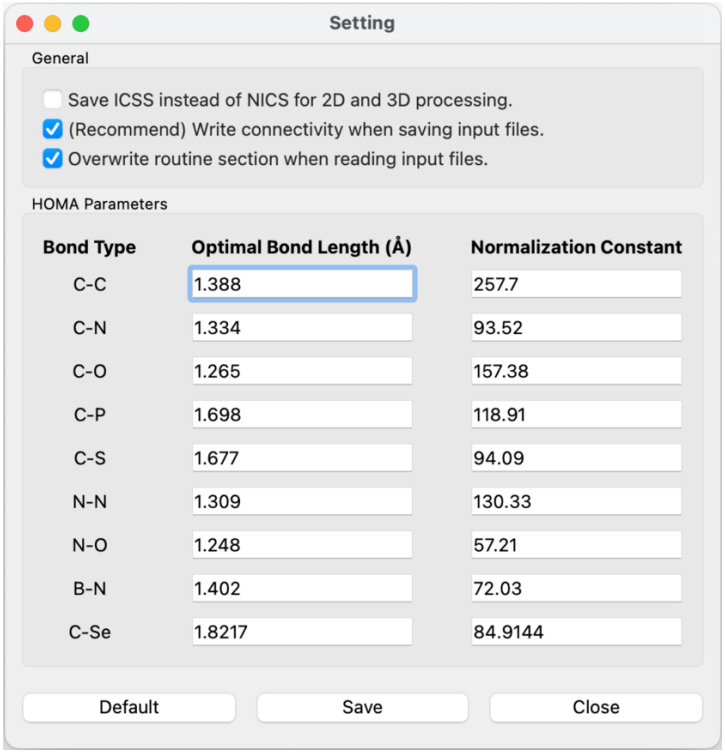


Figure 2.1 Screenshot of program setting panel.

2.3 Calculation Setup Panel

py.**Aroma 3** provides a calculation setup panel, this could be access when creating *Gaussian .gjf* input file from "Calculation Setup" button. The default setup is shown in Figure 2.2. User could preview the routine section by "Preview" button.

In calculation setup panel, user can set NMR method, computational level, Link 0 section, solvation model, and so on. The usage of this calculation setup panel is similar to the "Calculation Setup" in *GaussView*.

"Write connectivity" option is available in the program setting panel, to avoid that *GaussView* automatically make bonds between ghost atoms, I ***strongly recommend*** checking the "Write connectivity when saving input file" option in program setting panel (*Section 2.2*).

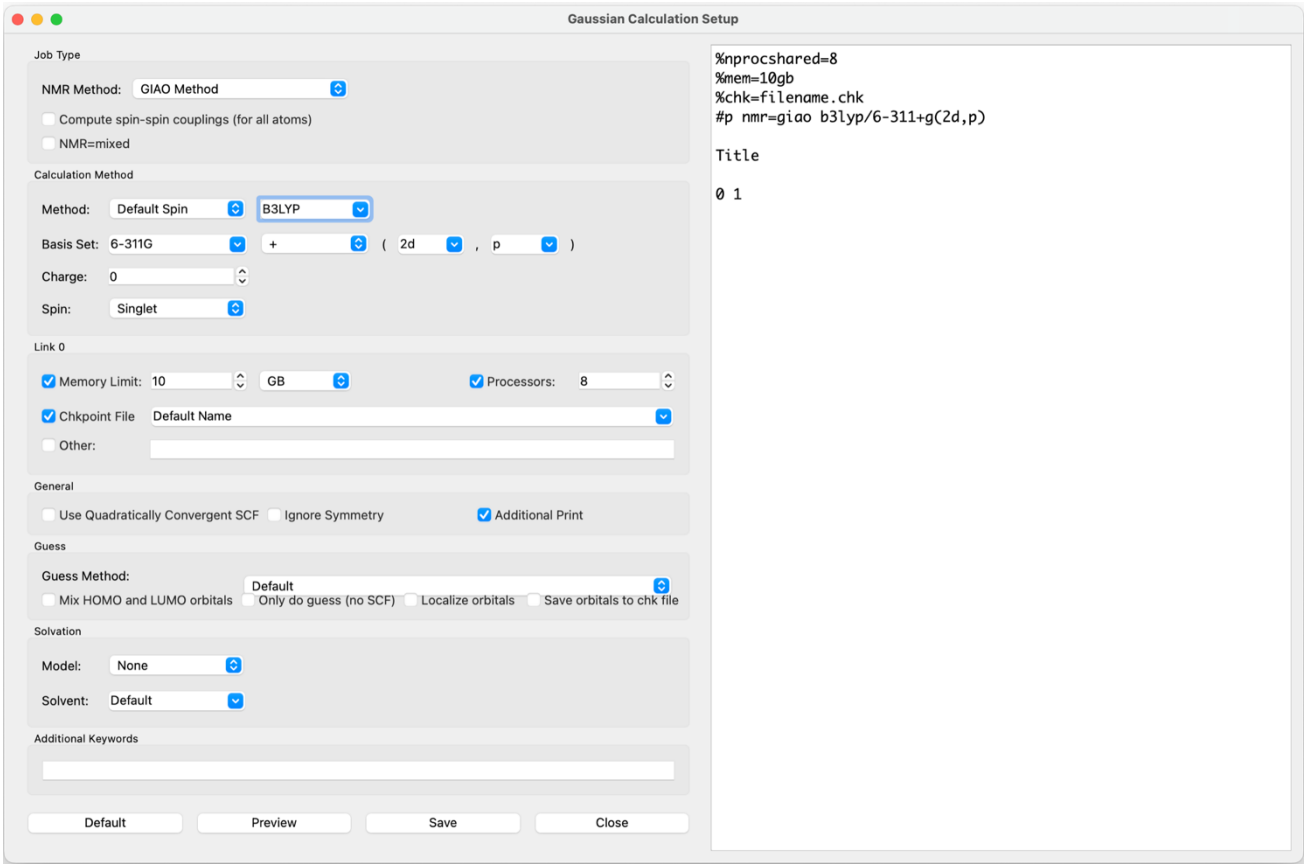


Figure 2.2 Screenshot of calculation setup panel.

3 Functions

This chapter introduces all functions of py.**Aroma 3** in detail. Different functions require different types (**Type-I**, **Type-II** and **Type-III**), please choose proper type of input file according to *Section 2.1*.

3.1 Plot Bond Length Alteration (BLA)

☞Supported file types: **Type-I**, **Type-II** and **Type-III**

3.1.1 Theory

In conjugated systems, the bond lengths in the conjugation chain show alternant character. Bond length alternation (BLA) is an important quantity in the studies of conjugation molecules. Generally, BLA is defined as *eq.* (3.1):

$$\mathrm{BLA} = \bar{R}_{\mathrm{even}} - \bar{R}_{\mathrm{odd}} \tag{3.1}$$

as the difference between average length of even bonds and odd bonds. Smaller magnitude of BLA implies better electron conjugation along the selected path.

A better way to investigate BLA is plotting bond length *vs*. bond index, the smaller the change in bond length, the smaller the BLA value. To plot bond length *vs*. bond index graph, the bond sequence in the conjugated chain should be given. An example from *J. Phys. Chem. C*, **2018**, *122*, 26777 is shown in Figure 3.1.

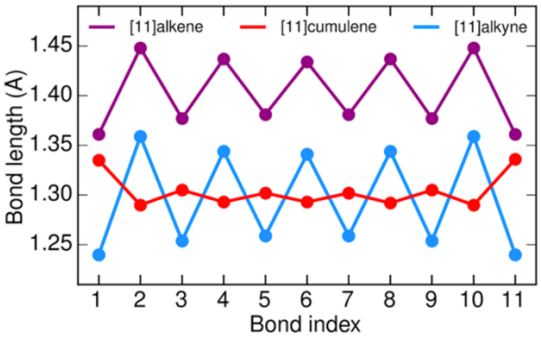


Figure 3.1 Bond length *vs.* bond index, from Figure 1c of reference literature.

3.1.2 Usage

To use this function, a bond sequence must be defined. After entering the BLA function of py.**Aroma 3**, user will be asked to input the indices of the bond that make up the sequence. After that, based on the sequence, py.**Aroma 3** automatically calculate the bond length for those bonds in the sequence and plot bond length *vs*. bond index graph in the BLA window. The graph could be saved as *.png* file and the data could be exported to *.txt* and *.xlsx* that user can import it to data plotting software such as *Origin* and *Prism*. The *.png*, *.txt* and *.xlsx* would be saved as same file name as input file, with "_BLA" suffix.

An example of using BLA function in py.**Aroma 3** is given in *Section 4.1.2* and *4.1.3*.

3.2 Calculate HOMA Index

☞Supported file types: **Type-I**, **Type-II** and **Type-III**

3.2.1 Theory

Harmonic oscillator model of aromaticity (HOMA) is one of the most popular indexes for evaluating aromaticity. This quantity was originally proposed in *Tetrahedron Lett.*, **1972**, *13*, 3839, and then the generalized form was given in *J. Chem. Inf. Comput. Sci.*, **1993**, *33*, 70. The generalized HOMA can be written as *eq*. (2).

$$\mathrm{HOMA} = 1 - \sum_{i} \frac{\alpha_{i,j}}{N} \left(R_{\mathrm{opt}} - R_{i,j} \right)^2 \tag{3.2}$$

where N is the total number of the atoms (bonds) considered, j denotes the atom next to atom i, R_{opt} and α are pre-calculated constants for each type of atom pair. If HOMA equals to 1, that means length of each bond is identical to the optimal value R_{opt} and thus the ring is fully aromatic. While if HOMA equals to 0, that means the ring is completely nonaromatic. If HOMA is a significant negative value, the ring shows antiaromatic character.

As example, HOMA indexes of benzene, cyclobutadiene and cyclooctadiene are 0.99603 (aromatic), –3.96177 (antiaromatic) and –2.40591 (nonaromatic), respectively (Figure 3.2).

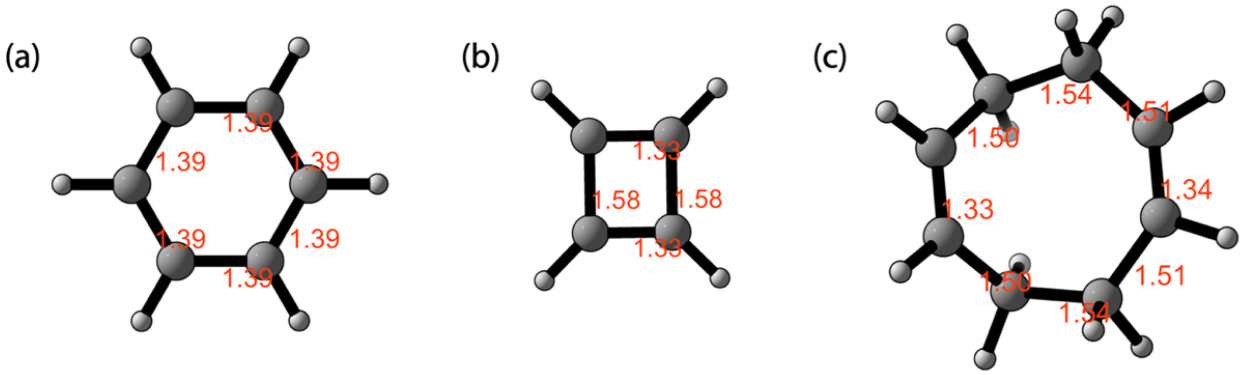


Figure 3.2 Bond length and HOMA indexes of (a) benzene, (b) cyclobutadiene and (c) cyclooctadiene, geometries were optimized at (R)B3LYP/6-311+G(2d,p) level of theory, orange values indicate bond lengths.

The built-in R_{opt} and α parameters are summarized in Table 3.1, user can modify the parameters from program setting panel in the main software window (See *Section 2.2*). The built-in R_{opt} and α parameters are taken from *Chem. Rev.,* **2014**, *114*, 6383.

3.2.2 Usage

Once entering the HOMA function, py.**Aroma 3** automatically calculate the HOMA indexes for all monocyclic structures in input geometry. User can also get HOMA index for specified cycle by input a sequence of atomic indices. The input order must be consistent with atom connectivity.

An example of using HOMA function in py.**Aroma 3** is given in *Section 4.2.2*.

Table 3.1 Built-in R_{opt} and α parameters.

Bond Type	$\boldsymbol{R_{opt}}$	$\boldsymbol{\alpha}$
C–C	1.388	257.7
C–N	1.334	93.52
C–O	1.265	157.38
C–P	1.698	118.91
C–S	1.677	94.09
N–N	1.309	130.33
N–O	1.248	57.21
B–N	1.402	72.03
C–Se	1.8217	84.9144

3.3 Calculate POAV Index

☞Supported file types: **Type-I**, **Type-II** and **Type-III**

3.3.1 Theory

The π-orbital axis vector (POAV) analysis provides a complete description of the electronic structure of nonplanar conjugated organic molecules. The POAV1 theory gives $\theta_{\sigma\pi}$ (Figure 3.3), θ_p (= $\theta_{\sigma\pi} - 90^\circ$), m [*eq*. (3.3)] and n [*eq*. (3.4)] values:

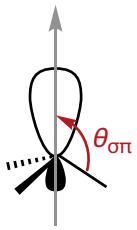


Figure 3.3 Definition of $\theta_{\sigma\pi}$.

$$m = \frac{2\cos^2\theta_{\sigma\pi}}{1 - 3\cos^2\theta_{\sigma\pi}} \tag{3.3}$$

$$n = 3m + 2 \tag{3.4}$$

where m indicates the s content of the π-orbital ($s^m p$) and n indicates the p content of the σ-orbital (sp^n).

The POAV2 hybridization provides $\theta_{1\pi}$, $\theta_{2\pi}$, $\theta_{3\pi}$, n_1, n_2, n_3 and m. The detail of POAV1 and POAV2 theory could be found in *J. Am. Chem. Soc.*, **1986**, *108*, 2837 and *J. Phys. Chem. A*, **2001**, *105*, 4164. The one-center POAV1 and POAV2 function in py.**Aroma 3** based on the three sigma bond angles.

For an ideal sp^2 hybridized carbon atom, $\theta_{1\pi} = \theta_{2\pi} = \theta_{3\pi} = 120^\circ$, $\theta_{\sigma\pi} = 90^\circ$, $\theta_p = 0^\circ$, $m = 0$ and $n = n_1 = n_2 = n_3 = 2$.

3.3.2 Usage

It is very easy to compute POAV1 and POAV2 parameters by py.**Aroma 3**. User can just input the index of target atom, then the POAV1 and POAV2 parameters would be automatically computed and displayed at the window. The inputted atom MUST and ONLY has 3 bonded neighboring atoms.

An example of using POAV function in py.**Aroma 3** is given in *Section 4.3.2*.

3.4 NICS Calculation

3.4.1 Theory

In this section, some basic information about NICS calculation will be introduced.

3.4.1.1 Single Point NICS

Nucleus-independent chemical shift (NICS) is a very popular index for aromaticity measurements by theoretical computation. NICS index is negative for aromatic system and positive for antiaromatic system. If NICS index is close to 0, it is nonaromaticity or weak aromaticity/antiaromaticity. Many literatures (such as *Org. Lett.*, **2006**, *8*, 863) showed that NICS(0)$_{ZZ}$ or NICS(1)$_{ZZ}$ is a better index than the original definition of NICS, which is current known as NICS(0).

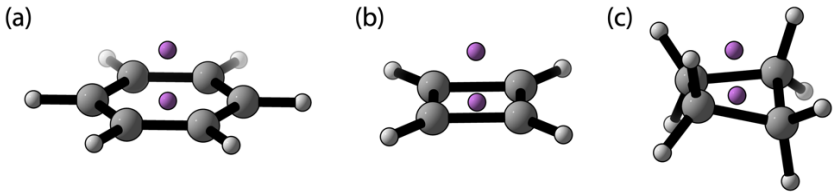
To get NICS value, the most popular way is to add ghost atom(s) (in *Gaussian*: Bq) at the target position(s) and submit it to NMR computation. The shielding tensors of all atoms would be computed and summarized in output file, a typical output for shielding tensors of ghost atom is shown in following:

```
    9  Bq   Isotropic =    -0.1133   Anisotropy =     1.6782
XX=     0.0988   YX=     0.5330   ZX=     0.3438
XY=     0.5353   YY=     0.1032   ZY=     0.3474
XZ=     0.7234   YZ=     0.7264   ZZ=    -0.5420
```

The NICS index is the reversed value of shielding tensor, thus, in the above example, NICS = 0.1133 (indicated by "Isotropic") and NICS$_{ZZ}$ = 0.5420 (indicated by "ZZ").

As example, NICS indexes calculated at (R)B3LYP/6-311+G(2d,p) of benzene, cyclobutadiene and cyclobutane are shown in Table 3.2.

Table 3.2 Input geometries and NICS indexes for (a) benzene, (b) cyclobutadiene and (c) cyclobutane.



Molecule	NICS(0)	NICS(1)	NICS(0)$_{ZZ}$	NICS(1)$_{ZZ}$
benzene	–7.6064	–9.9993	–14.9722	–29.6808
cyclobutadiene	27.0495	17.7196	110.7864	55.6224
cyclobutane	0.3766	1.2447	56.7631	3.1377

3.4.1.2 1D NICS Scan

NICS is commonly studied at some special points (*e.g.*: center of ring). By using a series of NICS probes to construct a more informative picture of the behavior of the induced magnetic field of a molecule, NICS Scan has been proposed. One of the most popular ways is called "NICS-*XY*-Scan"

(See: *Chem. Eur. J.*, **2014**, *20*, 5673), assuming the molecule is placed on the XY plane, the trajectory of NICS probes is also parallel to the XY plane, and the NICS probes are placed at 0.1 Å intervals along a line that traverses the length of system.

An example of 1D NICS-Scan of pentalene calculated at (R)B3LYP/6-311+G(2d,p) level of theory is shown in Figure 3.4, the NICS probes are aligned along the C2 to C5 with 0.1 Å intervals and placed at 1Å above the pentalene plane.

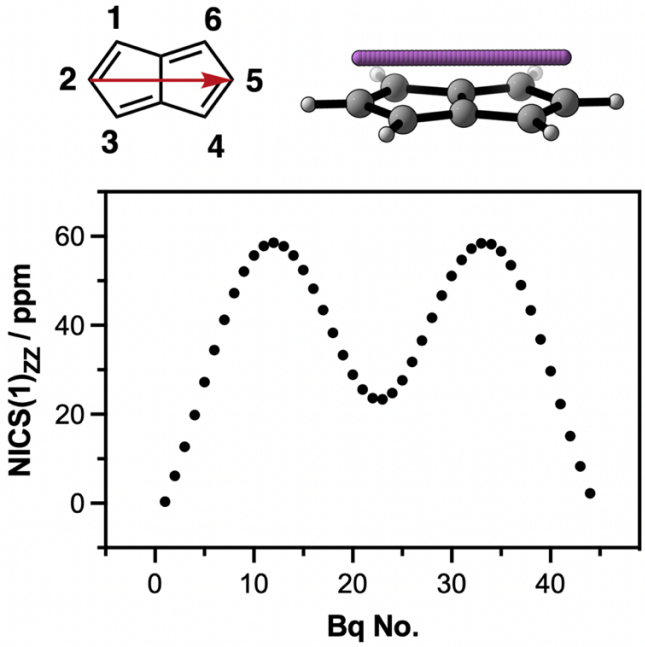


Figure 3.4 1D NICS Scan plot for pentalene. The scan trajectory is shown above the plot.

3.4.1.3 2D and 3D NICS Analysis

Like 1D NICS Scan, 2D and 3D present a more intuitive picture on aromaticity. In some cases, the 2D (3D) NICS also called as 2D (3D) ICSS, indicates iso-chemical shielding surface. From the description of ICSS, it is clearly that the values used in 2D (3D) ICSS derived from shielding tensors, rather than the reversed values those used in NICS analysis. Thus, both 2D (3D) ICSS and 2D (3D) NICS could be analyzed from same output file, they are same in the absolute values, with different sign.

Examples for 2D and 3D NICS analyses of benzene is shown in Figure 3.5, the NMR computations were carried out at (R)B3LYP/6-311+G(2d,p) level of theory.

3.4.2 Create Input File for NICS Calculation

☞Supported file types: **Type-I** and **Type-II**

User can create *Gaussian* input file for NICS calculation with py.**Aroma 3**. After opening the **Type-I** or **Type-II** input file, user can choose from "Single Point NICS", "1D NICS Scan", "2D NICS" and "3D NICS" according to the purpose.

In the input file creating windows, user can always edit the keyword for *Gaussian* calculations

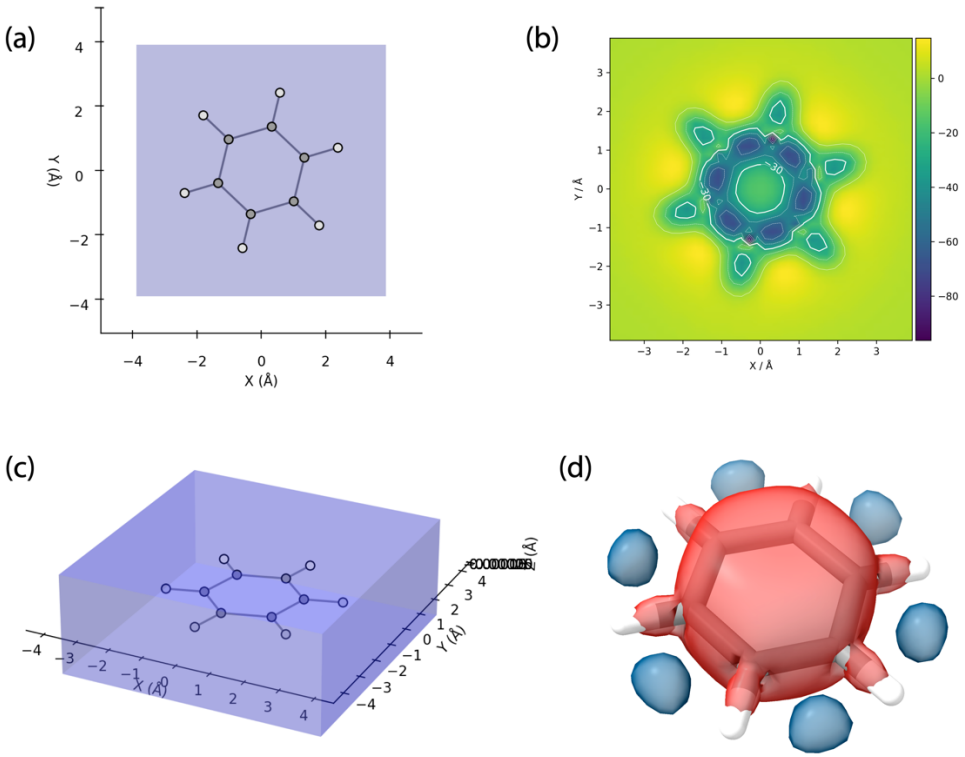


Figure 3.5 Illustrates of (a) 2D NICS and (c) 3D NICS, the violet regions indicate the positions of ghost atom. (b) give a 2D NICS(0)$_{ZZ}$ heatmap and the contour line of –30 ppm is emphasized in bold line. (d) show the iso-surface of –25 ppm (red) and 10 ppm (blue) from the data of 3D NICS$_{ZZ}$.

from "Calculation Setup" button. But, when reading a *.gjf* or *.com* as input file, the "Calculation Setup" is only valid when the "Overwrite routine section when reading input files." option is checked in program setting window (*Section 2.2*). This option is checked by default.

3.4.2.1 Single Point NICS

Once entering the single point NICS window, the py.**Aroma 3** would automatically recognize all monocycles in the geometry, thus, user can add ghost atoms for all monocycles at once. If a sequence of atom indices (more than three atoms are needed) is given, the ghost atom would be added at the center of given atoms. Users can adjust the height, too. If a height value greater than 0 is given, two ghost atoms would be added above and below the plane.

The *.gjf* file for NICS calculation would be saved as same file name as input file, with "_NICS" suffix.

An example of creating input file for single point NICS calculation with py.**Aroma 3** is given in *Section 4.4.2*.

3.4.2.2 1D NICS Scan

For creating the path for 1D NICS scan, a sequence of knot (at least two knots) is required. The

knot would be added at the center of atoms that user inputted. The path would be displayed on the screen when more than two knots have been specified, and the figure can be saved as a *.png* image with same file name as input file, with "_NICS_Scan" suffix.

py.**Aroma 3** also supports the 1D NICS scan on YZ and XZ plane. These could be edit in the parameter section, along with intervals and height above the plane. The input file would be saved as the same name as *.png* file.

An example of creating input file for 1D NICS scan calculation with py.**Aroma 3** is given in *Section 4.5.2*.

3.4.2.3 2D and 3D NICS

Creating the input file for 2D/3D NICS calculation by py.**Aroma 3** is easy. User just need to specify the region for ghost atoms. User can check the region by click "Preview" button.

The *.gjf* file for 2D/3D NICS calculation would be saved as same file name as input file, with "_2D(/3D)_NICS" suffix.

An example of creating input file for 2D NICS scan calculation with py.**Aroma 3** is given in *Section 4.6.2*, and for example of 3D NICS please check *Section 4.7*.

3.4.3 Process Output File of NICS Calculation

☞Supported file types: **Type-III**

User can process output files of NICS calculations with py.**Aroma 3**. For analyzing output files of 1D NICS scan, 2D NICS and 3D NICS, the output files must be generated by the input files which created by py.**Aroma 3**.

For 2D and 3D NICS, by default the NICS indexes (*i.e.*, reversed values of shielding tensors) would be used. If 2D and 3D ICSS is preferred, please check the "Save ICSS instead of NICS for 2D and 3D processing." option in program setting panel (*Section 2.2*).

3.4.3.1 Single Point NICS

py.**Aroma 3** would automatically find the shielding tensors from output file and display the NICS values on the screen. User can choose the component of shielding tensors. The results could be saved as *.txt* file with same file name as *Gaussian* output with "_NICS_output" suffix. An example could be found in *Section 4.8.2*.

3.4.3.2 1D NICS Scan

py.**Aroma 3** would automatically find the shielding tensors from output file, display the NICS values and plot the NICS values *vs.* indices of ghost atom on the screen. User can also choose the component of shielding tensors, the data would be updated automatically. The result could be saved as *.png* and *.xlsx* with "_ NICS_Scan_output" suffix. An example could be found in *Section 4.8.3*.

3.4.3.3 2D NICS

When reading output files for 2D NICS, the heatmap would be automatically displayed on the screen. User can choose component of shielding tensors and the heatmap would be updated. The heatmap could be saved as *.png* image with "_ 2D_NICS_output" or "_ 2D_ICSS_output" suffix in file name. The NICS (or ICSS) indexes could also be saved as *.csv* file for further investigation. An example could be found in *Section 4.8.4*.

3.4.3.4 3D NICS

After entering 3D NICS window, user could choose the component of shielding tensors and save them to a *Gaussian*-type *.cube* (also known as *.cub*) file, with "_3D_NICS_output" or "_3D_ICSS_output" suffix in file name, which could be visualized by *GaussView*, *VMD*, *ChimeraX*, *etc*.

A preview of heatmap for 2D slice would be displayed on the screen, too. User can edit it just as described in **3.4.5.3**. An example could be found in *Section 4.8.5*.

3.5 Generate Computational Supporting Material

☞Supported file types: **Type-II**

As an additional function, **CSIgen** (https://github.com/wongzit/CSIgen), which is the tool for generating computational supporting information/materials, is built in py.**Aroma 3**. When a **Type-II** file is read by py.**Aroma 3**, this function would be available.

After entering the window, user can choose which information should be included in the supporting information and save as *.txt* or *.xlsx* file.

An example for using **CSIgen** module in py.**Aroma 3** could be found in *Section 4.9*.

4 Tutorials and Examples

This chapter includes detail tutorials and some examples for using py.**Aroma 3** to carry out aromaticity analyses.

4.1 BLA Analysis

4.1.1 Basic Information

BLA function could be accessed from BLA button after open a file, the main window of BLA function is shown in Figure 4.1. The geometry of molecule in input file would be shown in **area A**, with bond indexes at the center of each bond. The **area B** give a preview of bond length *vs.* bond index graph, there is no preview unless user press the "Plot" button in **area C**. The **area C** is a functions group, including a text bar ask for bond indexes, "Plot" button for preview the graph in **area B**, "Clear" button would clear current graph in **area B**, "Save .png" button would save current graph in **area B**, "Save .txt" and "Save .xlsx" buttons would save bond date to *.txt* and *.xlsx* files.

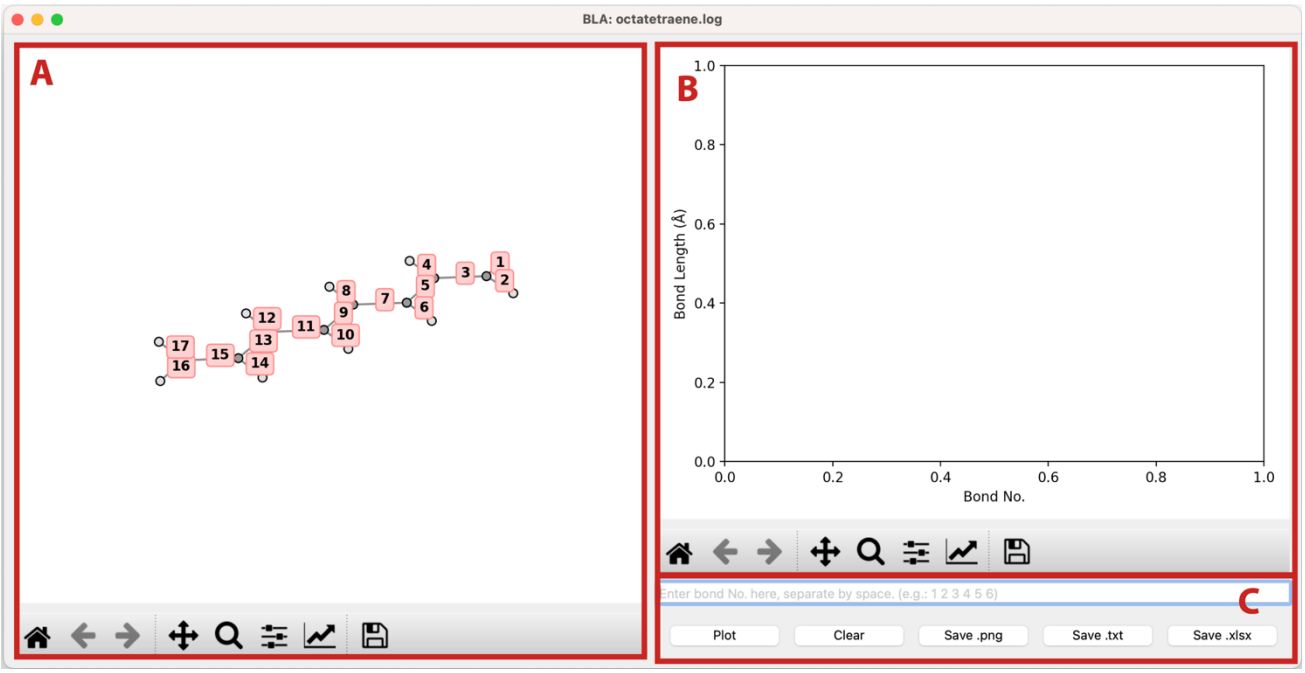


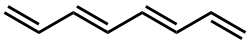
Figure 4.1 Main window of BLA function.

Since a sequence of bond indexes is needed for plot BLA graph, user need to input bond indexes separated by space in the text bar in **area C**, *e.g.*: 3 5 7 9 11 13 15. DO NOT use other separators like ",". Some bad input examples:

- 3,5,7,9,11,13,15 (*illegal separator*)
- 3.5.7.9.11.13.15 (*illegal separator*)
- 3 5 7 9 11 13 22 (*bad bond index, no bond No. 22 in molecule*)
- 3 5 7 9 11 13 1t (*only input number*)

4.1.2 BLA Analysis of 1,3,5,7-Octatetraene

Structure optimization of 1,3,5,7-octatetraene was performed at (R)ωB97X-D/6-31G(d) level of theory. The output file "*octatetraenene.log*" was read by py.**Aroma 3**.



Scheme 4.1 Chemical structure of 1,3,5,7-octatetraene.

The sequence of bond indexes "3 5 7 9 11 13 15" was inputted in the text bar in **area C**, by clicking the "Plot" button, a preview of BLA graph is shown in **area B** (Figure 4.2).

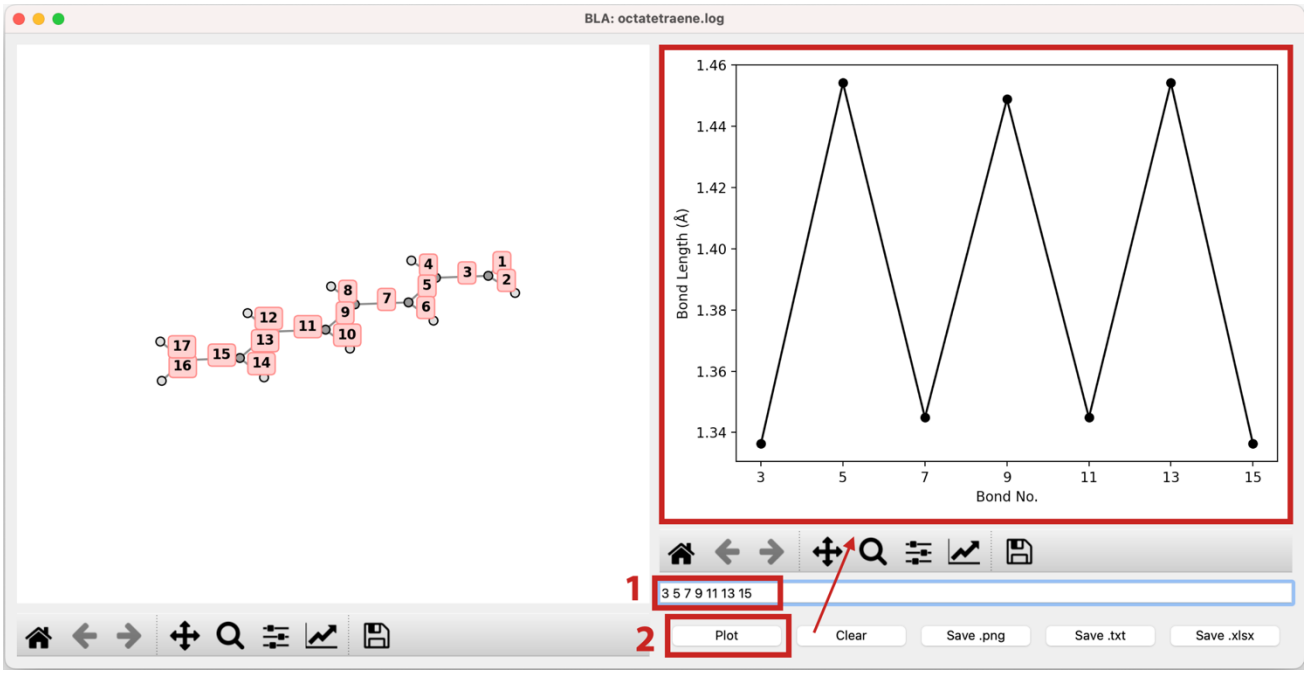
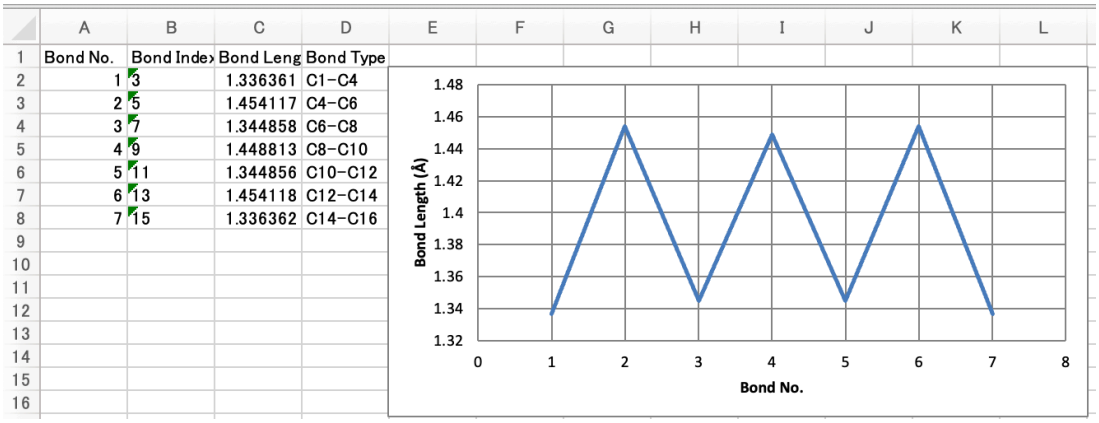


Figure 4.2 By (1) inputting bond indexes and (2) clicking "Plot" button, the graph would be displayed.

Then, if you want to save the graph and/or data, click "Save *.png*", "Save *.txt*" and "Save *.xlsx*" just by needed. The files would be saved as "*octatetraene_**BLA**.png*", "*octatetraene_**BLA**.txt*" and "*octatetraene_**BLA**.xlsx*". For example, screenshot of *.xlsx* file generated by py.**Aroma 3** is given in Figure 4.3.



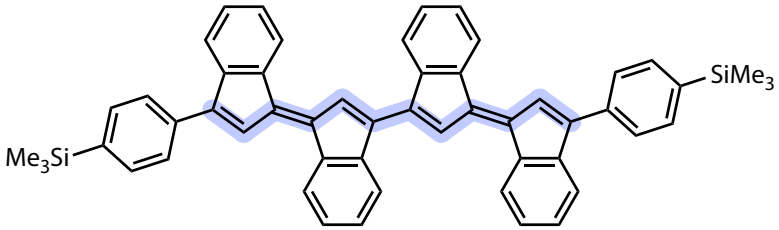
	Bond No.	Bond Index	Bond Length	Bond Type
2	1	3	1.336361	C1–C4
3	2	5	1.454117	C4–C6
4	3	7	1.344858	C6–C8
5	4	9	1.448813	C8–C10
6	5	11	1.344856	C10–C12
7	6	13	1.454118	C12–C14
8	7	15	1.336362	C14–C16

Figure 4.3 Screenshot of "*octatetraene_BLA.xlsx*".

HINT: *After inputting the bond sequence, it is possible to click "Save .png", "Save .txt" and "Save .xlsx" directly without clicking the "Plot", since the preview module of* py.**Aroma 3** *would be active before saving .png, .txt and .xlsx.*

4.1.3 BLA Analysis of Oligo(biindenylidene)

Following example is from DOI: 10.26434/chemrxiv-2022-9w59c. The structural coordinates were extracted from supporting information and saved as "natcom_4b.*xyz*" and read by py.**Aroma 3**.



Scheme 4.2 Chemical structure of oligo(biindenylidene), highlighted bonds are used for BLA analysis.

Bond indexes "13 16 17 18 21 23 46 48 49 50 53" was inputted and BLA graph was plot, as shown in Figure 4.4, the Figure 4b in original paper is well reproduced by py.**Aroma 3**.

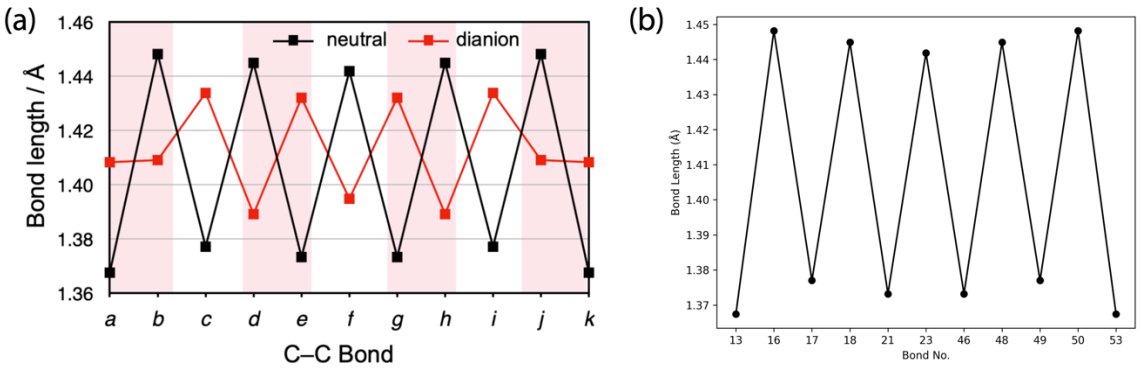


Figure 4.4 (a) BLA graph in Figure 4b of literature, black line is reproduced by py.**Aroma 3** (b).

4.2 HOMA Analysis

4.2.1 Basic Information

HOMA function could be accessed from HOMA button after open a file, the main window of HOMA function is shown in Figure 4.5. **Area A** show a 3D geometry of inputted molecule. The HOMA values for all monocycles would be automatically computed and displayed in **area B**. In **area C**, there is a text bar asking for sequence of atom indices if user has cyclic structures which the HOMA values are needed. After inputting the atom indices, user can get the HOMA value by clicking the "Calculate HOMA for specified ring." button in **area C**. The check boxes in **area C**: "Show HOMA Value" would display all HOMA values in **area B** on the center of each cycle in **area A**; "Show Atom No." would display atom indices on each atom in **area A**, this is useful when specifying rings in text bar.

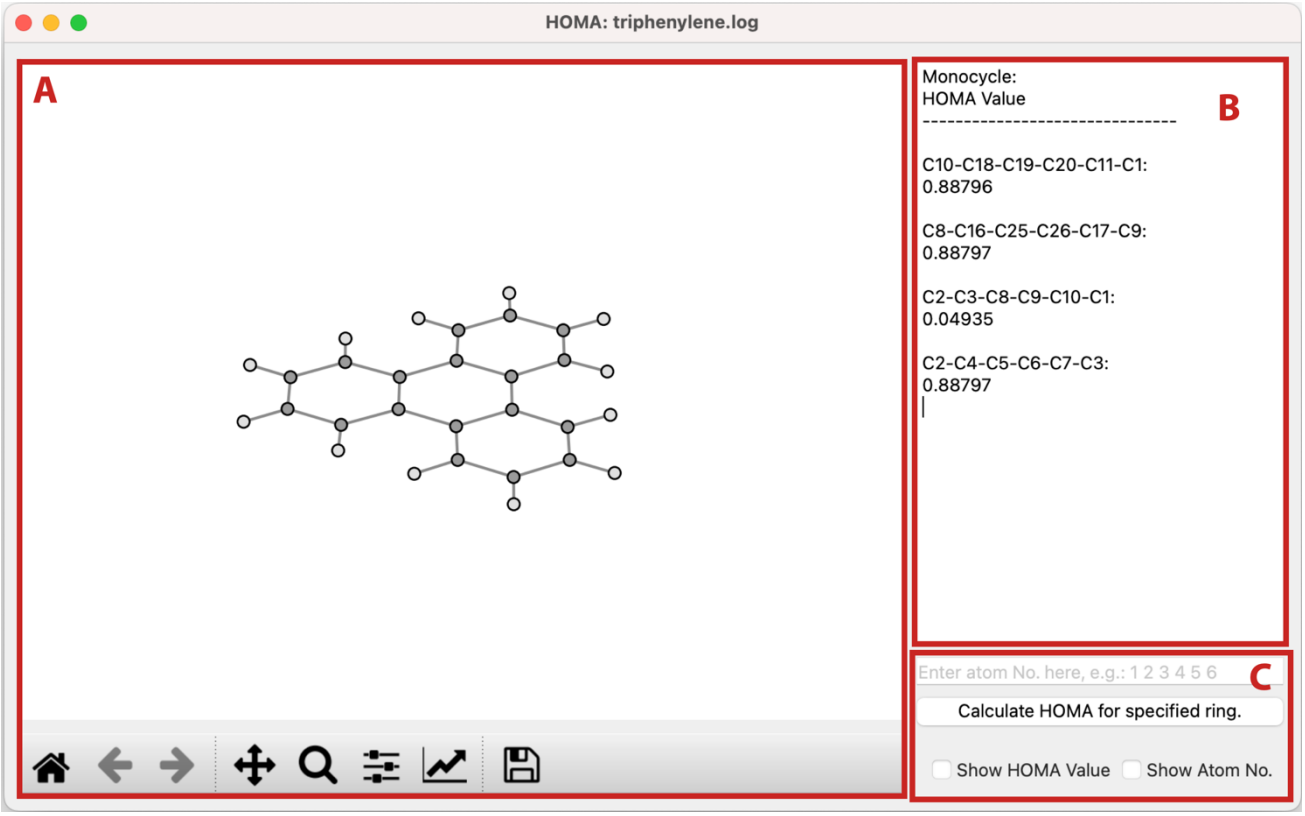


Figure 4.5 Main window of HOMA function.

The R_{opt} and α parameters used for HOMA calculations are defined in program setting panel. User can modify the parameters if necessary. More information about program setting, please refer to *Section 2.2*.

In **area B**, the arom indexes of cyclic structure and its HOMA values would be displayed. For example:

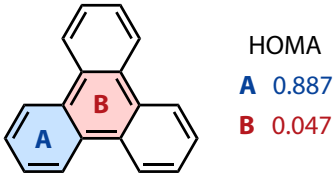
```
C10-C18-C19-C20-C11-C1:
0.88796
```

means the HOMA value of ring "C10-C18-C19-C20-C11-C1" is 0.88796.

Same as BLA function, the atom indexes in the text bar of **area C** should also be separated by space, *e.g.*: 10 18 19 20 11 1, and the order should be same as bond connectivity. For example, "10 19 18 20 11 1" is illegal since no connection between 10 and 19, 18 and 20. The sequence MUST include more than three atom indexes to construct a cyclic structure.

4.2.2 HOMA Analysis of Triphenylene

Structure optimization of triphenylene was performed at (R)B3LYP/6-31G(d) level of theory. The optimized triphenylene adapted D_{3h} point group and the HOMA values of ring A and ring B have been reported to be 0.887 and 0.047, respectively, from *Org. Lett.*, **2014**, *16*, 6116.



Scheme 4.3 Chemical structure of triphenylene and reported HOMA values for ring A and B.

The optimized geometry was read by py.**Aroma 3** for computing HOMA values. The HOMA values for all monocycles automatically displayed in **area B**, and the results are well reproduced. By checking the "Show HOMA Value" option, the HOMA values are added on the center of each monocycle.

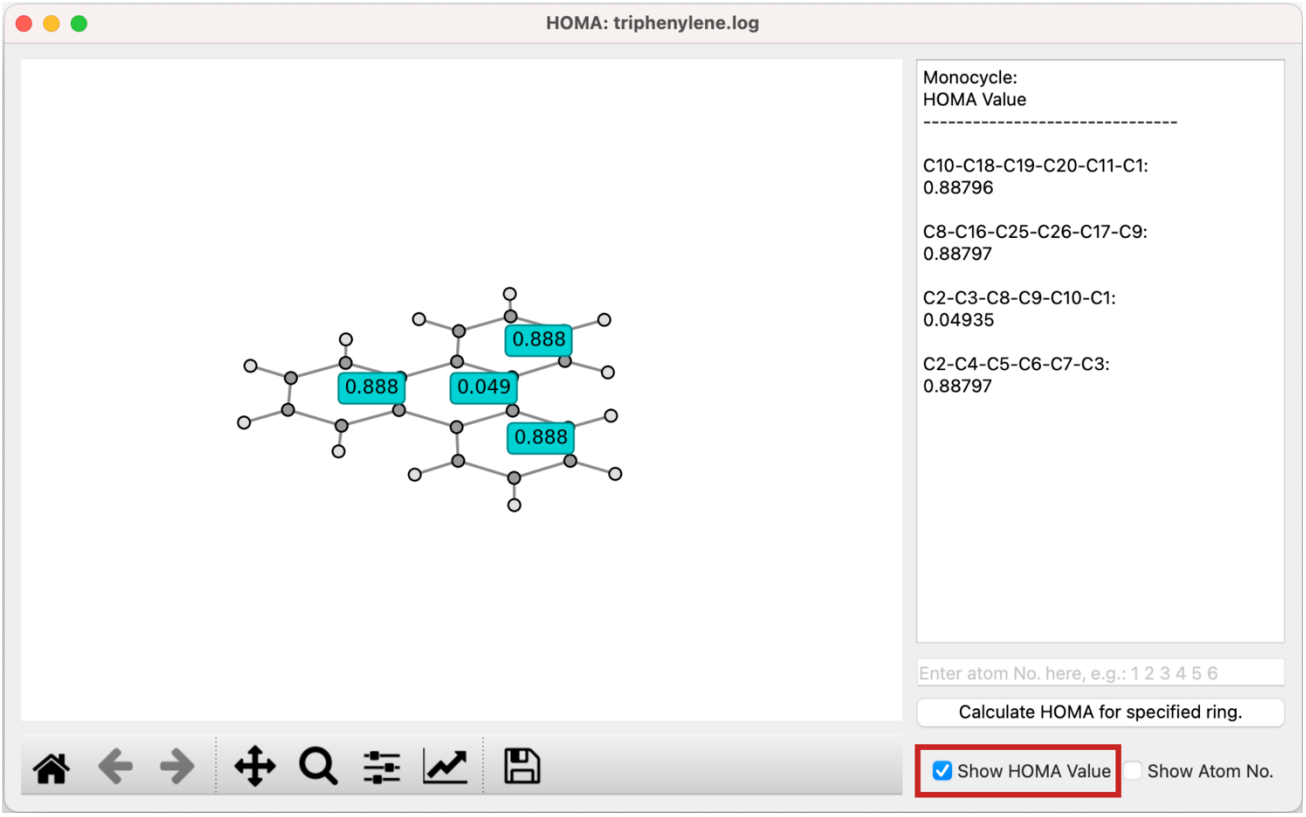
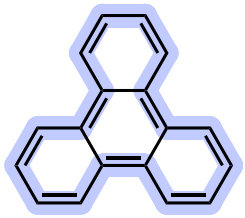


Figure 4.6 HOMA values of all monocycles.

Here, to investigate the HOMA value of outer edge, a sequence of atom indexes (3 7 6 5 4 2 1 11 20 19 18 10 9 17 26 25 16 8) is inputted in the text bar, then, click the “Calculate HOMA for specified ring.” button. After that, the HOMA value would be added in the **area B**.



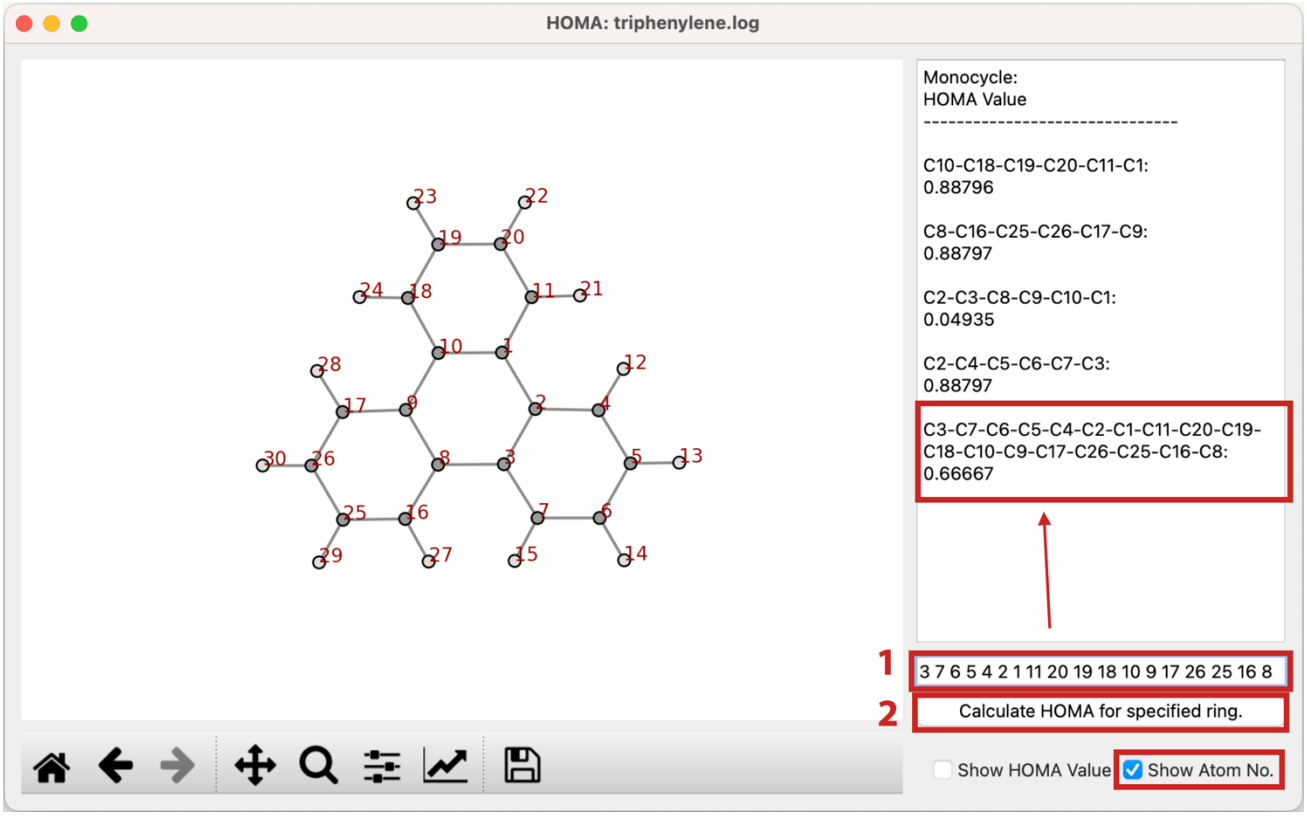


Figure 4.7 HOMA values of all monocycles, inserted structure shows the outer edge considered for HOMA.

4.3 POAV Analysis

4.3.1 Basic Information

POAV function could be accessed from POAV button after open a file, the main window of POAV function is shown in Figure 4.8. **Area A** show the input geometry with atom indexes. **Area B** includes basic function, the check box "Show Atom No." would show/hide atom indexes in **area A**, the text bar near the check box is asking for target atom index, which carried for POAV analysis. Once input the atom index, click "Compute" button, then, the POAV1 and POAV2 results would be displayed in **area C**, and the target atom would be highlighted in **area A**.

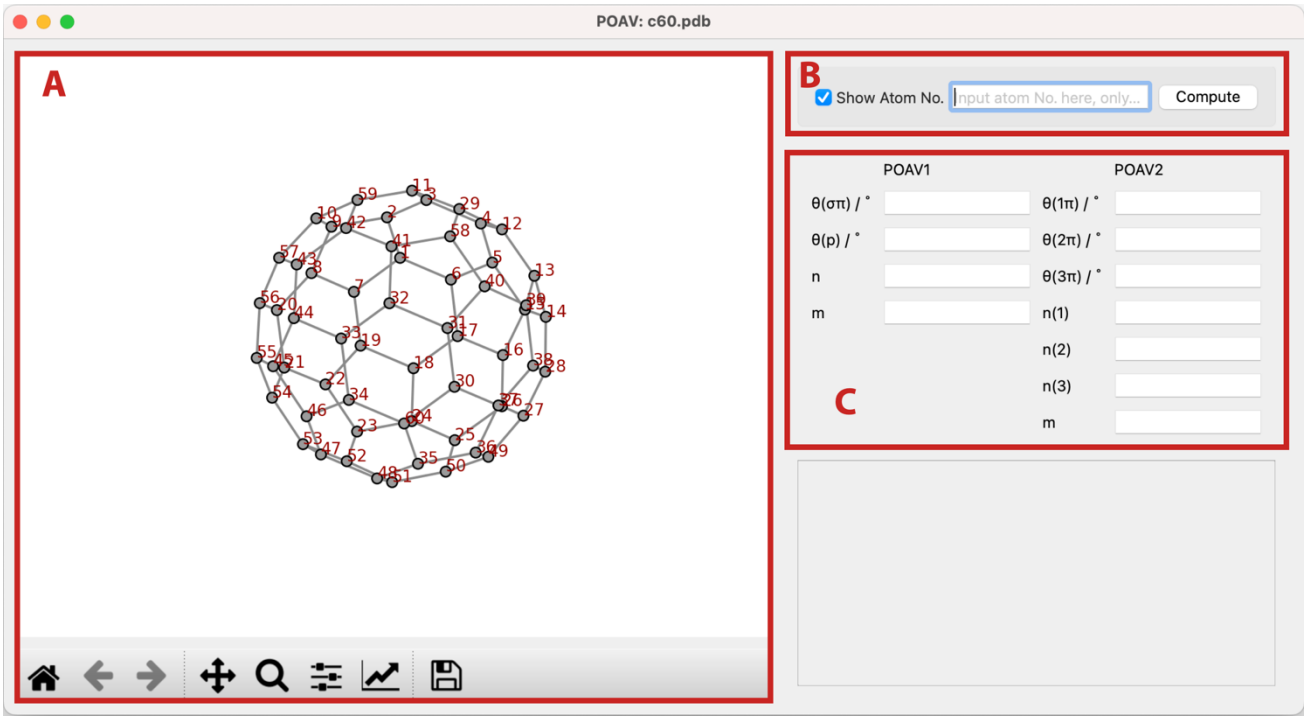


Figure 4.8 Main window of POAV function.

The text bar in **area B** could only accept one atom index, and the target atom MUST and ONLY have three bonded atoms. Otherwise, and error message would be displayed.

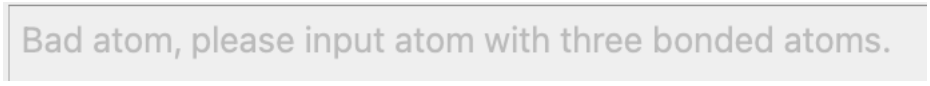


Figure 4.9 Error message if bad atom was specified.

4.3.2 POAV Analysis of Fullerene C_{60}

Fullerene C_{60} is a hot molecule used as example of POAV analysis. As example, we want to know the POAV parameters of atom C11, so, we just input 11 in the text bar and click "Compute". As shown in Figure 4.10, the atom C11 is highlighted and the POAV parameters are displayed in **area C**. There is no technic in POAV analysis.

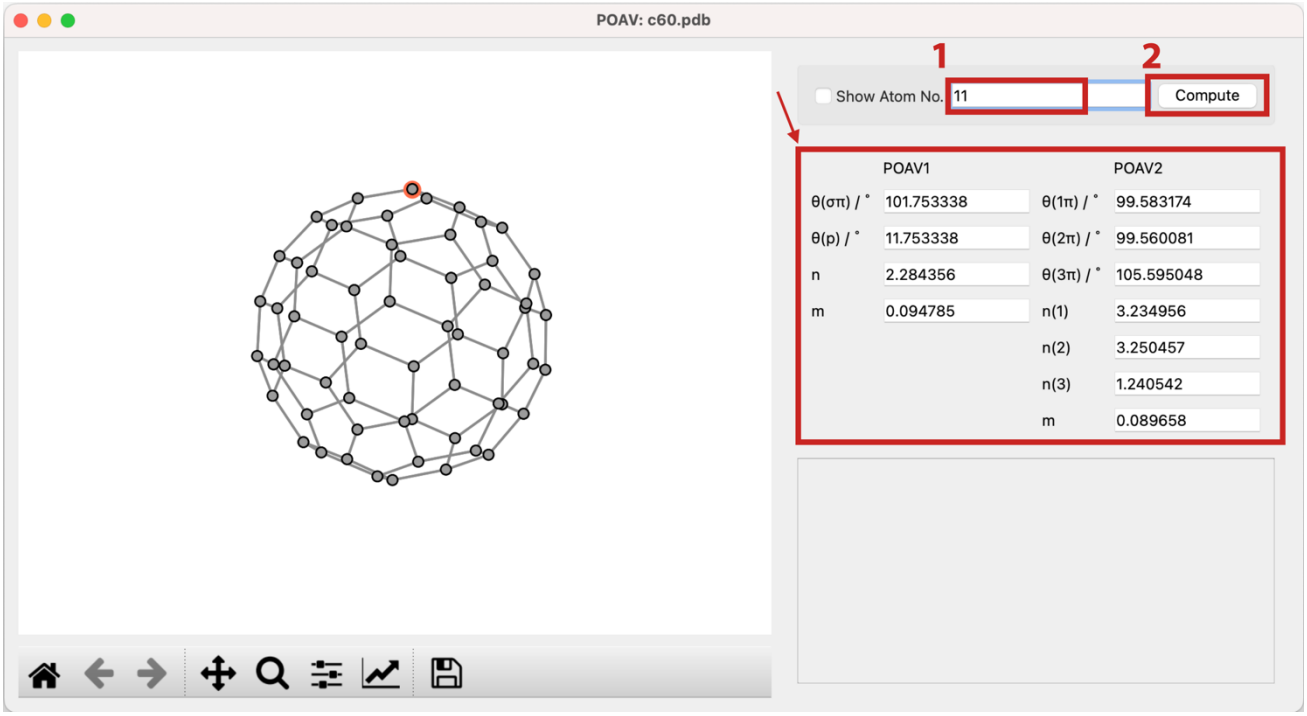


Figure 4.10 POAV analysis of atom C11 in C_{60}, the C11 is highlighted in the geometry.

4.4 Create Input File for Single Point NICS

4.4.1 Basic Information

Single point NICS function could be accessed from single point NICS button after open a file, the main window of is shown in Figure 4.11. The molecular geometry would be displayed in **area A**. The monocycle in molecule would be automatically detected, by clicking "Add Bq atoms for all monocycles" button in **area C**, py.**Aroma 3** would add ghost atoms on specified height for all monocycles (1 Å by default, the height could be modified in the "Height:" text bar in **area C**). Besides that, user can custom the position of ghost atom by following procedure:

(0) Check "Show Atom No.", this not compulsory but convenient.

(1) Inputting the atom indexes in the text bar in **area C**, (*e.g.*: 1 2 3 4 5 6).

(2) Set the height in the "Height:" text bar.

(3) Click "Add Bq atom" button. Then, the ghost atom(s) would be added in the center of atom 1, 2, 3, 4, 5 and 6, on the "height" Å.

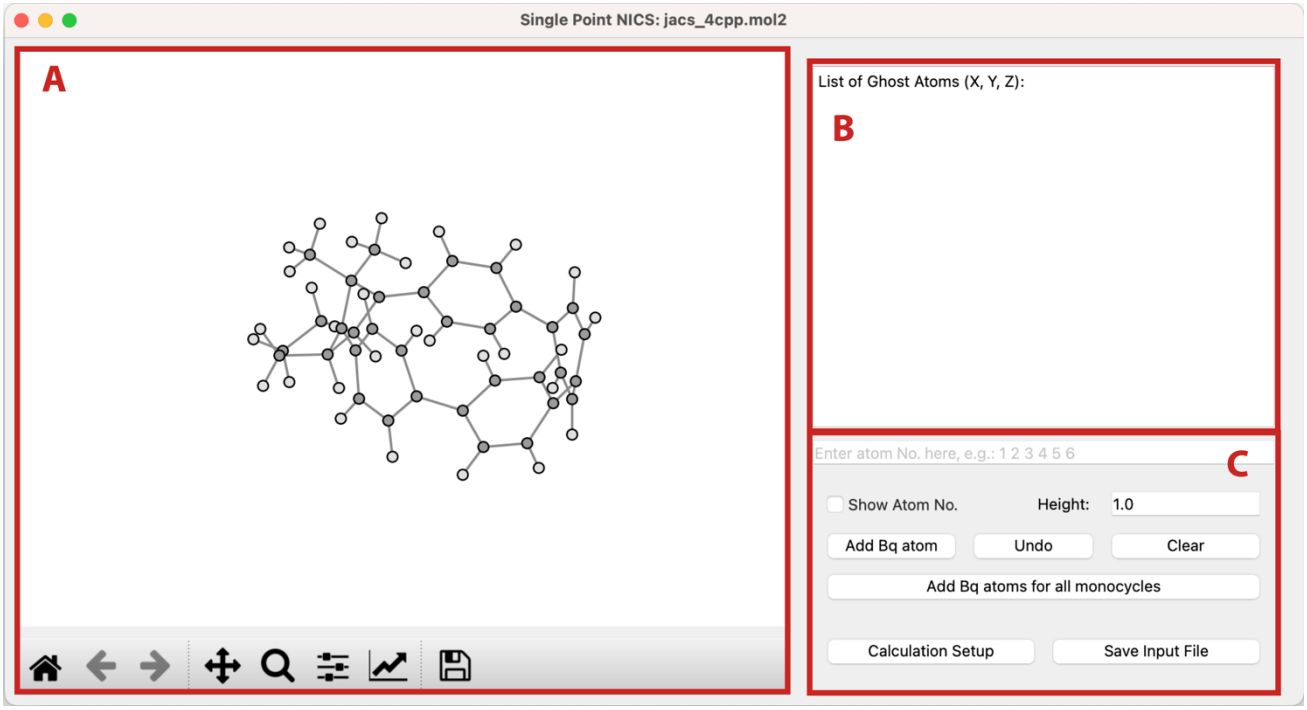


Figure 4.11 Main window of single point NICS function.

py.**Aroma 3** would ignore the monocycles those bigger than 10-membered rings, so, if your compound including large cycle and you want to compute NICS for them, please add ghost atom by specifying the atom indexes in **area C**.

User can delete ghost atoms by "Undo" and "Clear" button in **area C**. "Undo" would delete ghost atom one by one from the newest added atom. "Clear" would delete all ghost atoms that have been added. The Cartesian coordinates of ghost atoms would be displayed in **area B**.

For specifying the atom indexes in **area C**, at least three atoms are needed (to define a plane).

The order is insensitive, *e.g.*: both "1 2 3 4 5 6" and "1 5 3 6 4 2" are acceptable.

If height is not 0 (*i.e.*: NICS(*l*), $l > 0$), two ghost atoms would be added above and below the cycle.

To save *Gaussian* input file, user need to click the "Save Input File" button. Before saving, user can modify the calculation setup, more information about calculation setup, please refer to *Section 2.3*.

4.4.2 Example: DR-[4]CPP

To give a better understanding of how to use NICS module in py.**Aroma 3**, a step-by-step procedure using **DR-4CPP** (*J. Am. Chem. Soc.*, **2021**, *143*, 7426.) would be described. The target is: NICS(1) for all monocyclic rings, and NICS(0) for the whole macrocycle at its center.

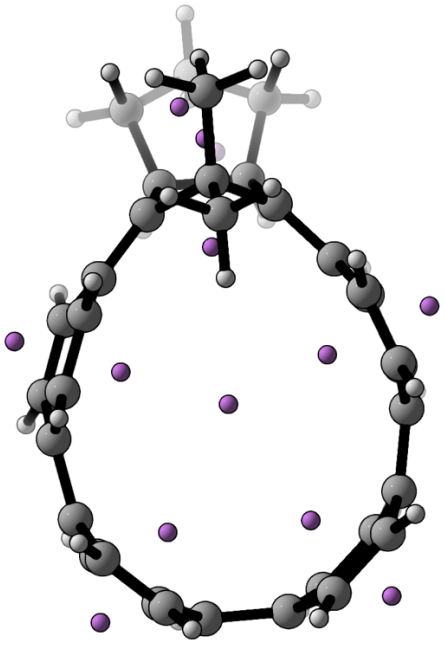


Figure 4.12 Geometry of DR-4CPP with ghost atoms.

< Procedure >

(0) Open *jacs_4cpp.mol2* in py.**Aroma 3**.

(1) Click "Add Bq atoms for all monocycles", ghost atoms for NICS(1) will be added.

(2) Check "Show Atom No."

(3) Change the "Height" value to 0.

(3) Input the atom sequence in the text bar: 2 5 6 7 34 35 44 45 16 17. In this example, the equatorial atoms are used to define the plane.

(4) Click "Add Bq Atom", the NICS(0) of the macrocycle will be added.

(5) Click "Save Input File", finished! The screenshot of final state is shown in Figure 4.13.

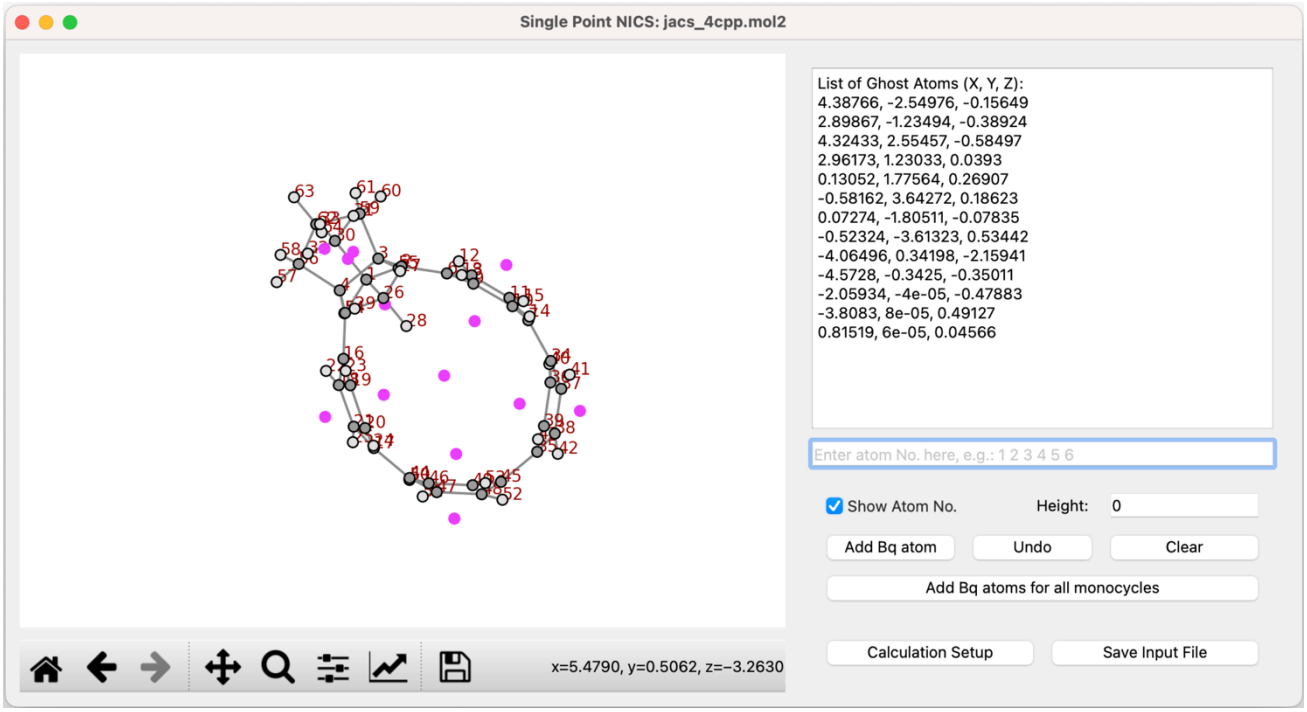


Figure 4.13 Screenshot of final state, 16 ghost atoms were added.

4.5 Create Input File for 1D NICS Scan

4.5.1 Basic Information

1D NICS scan function could be accessed from 1D NICS scan button after open a file, the main window of is shown in Figure 4.14. Once entering the 1D NICS scan function, the 3D geometry of molecule would be displayed in **area A**, and its projection on XY plane would be displayed in **area B**. User can change the plane of projection by edit the "Plane" radio button (XY, YZ, XZ) in "NICS Scan Parameters" section in **area D**.

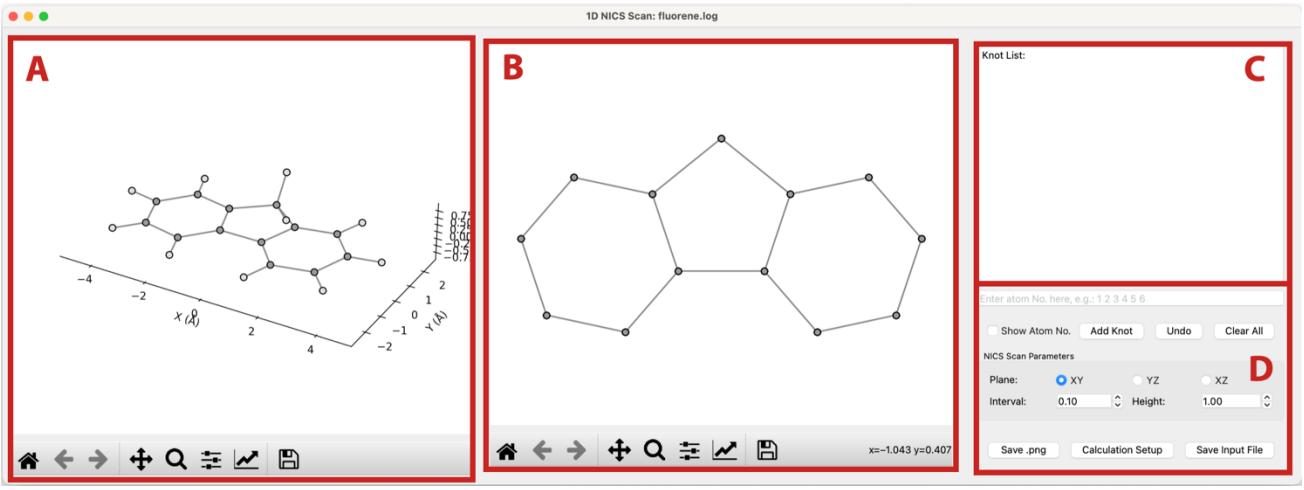


Figure 4.14 Main window of 1D NICS scan function.

A path is necessary for 1D NICS scan, user can add knots by following procedure:

(0) Check "Show Atom No.", this not compulsory but convenient.

(1) Inputting the atom indexes in the text bar in **area D**, (*e.g.*: 1 2 3 4 5 6).

(2) Click "Add Knot", the knot would be added at the center of inputted atoms, and the knot information would be updated in **area C**.

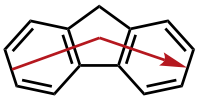
(3) Repeat step (1) and (2), until the full path has finished.

User can delete all knots by clicking "Clear All" and delete last knot by "Undo" button. To save *Gaussian .gjf* file, user can just click the "Save Input File" button, py.**Aroma 3** would add ghost atoms on specified height (1 Å by default, the height could be modified in the "Height:" text bar in **area D**) on the path of knots. The intervals of neighboring ghost atoms are 0.1 Å by default, user can modify it by the "Interval:" option in **area D**.

The figure in **area B** could be saved as *.png* file by clicking "Save .png" button in **area D**.

4.5.2 Example: Fluorene

Here is a step-by-step tutorial for how to use 1D NICS scan module in py.**Aroma 3**, with fluorene as example. The fluorene molecule is placed on XY plane, we want to put the NICS probes on the trajectory as shown in Scheme 4.4, on the height (Z axis) of 1 Å, with 0.2 Å interval of ghost atoms.



Scheme 4.4 Target path for 1D NICS scan on fluorene molecule.

< Procedure >

(0) Open *fluorene.log* in py.**Aroma 3**.

(1) Check “Show Atom No.”.

(2) Input “16 17” in the text bar in **area D** and click “Add Knot”.

(3) Input “14 5 19 6 12” in the text bar in **area D** and click “Add Knot”.

(4) Input “2 3” in the text bar in **area D** and click “Add Knot”.

(5) Change the value of “Interval” to 0.2.

(6) Click “Save Input File”.

(7) (option) Click “Save .png”.

(8) Finished! The screenshot of final state is shown in Figure 4.15, and the finial geometry is shown in Figure 4.16.

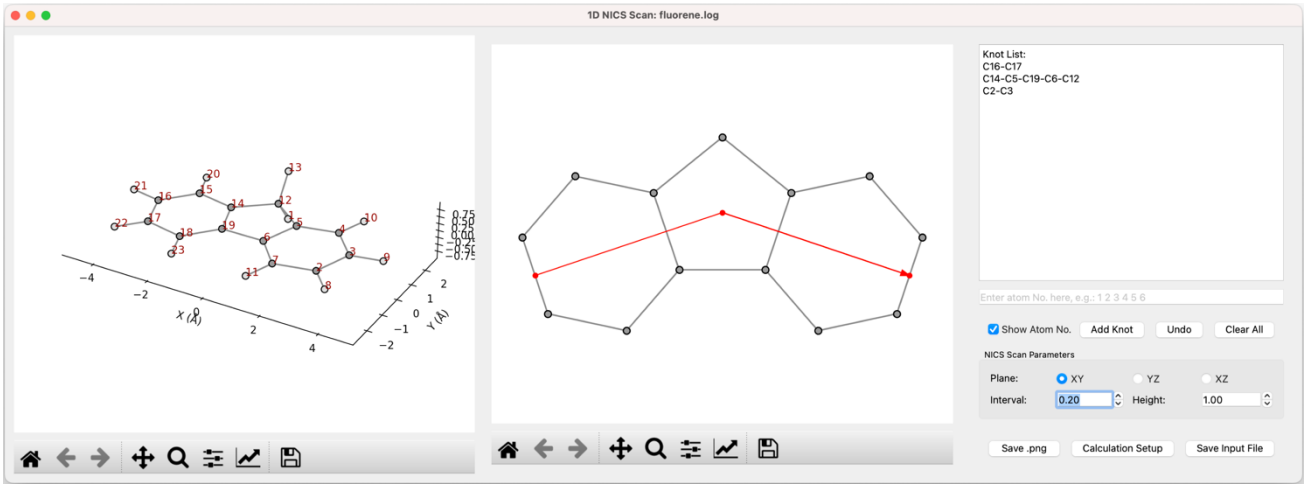


Figure 4.15 Screenshot of final state.

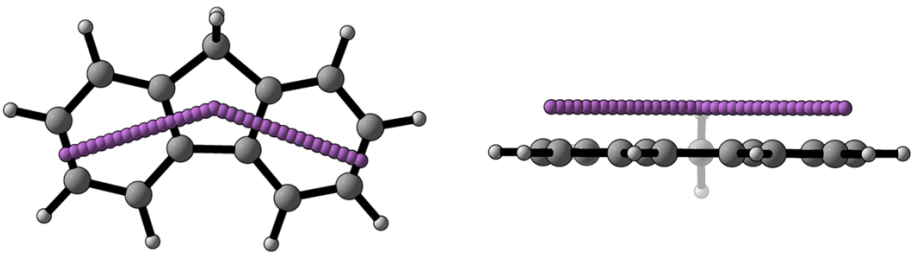


Figure 4.16 Top view (left) and side view (right) of input structure for 1D NICS scan.

4.6 Create Input File for 2D NICS

4.6.1 Basic Information

2D NICS (ICSS) function could be accessed from 2D NICS button after open a file, the main window of is shown in Figure 4.17. **Area A** give a preview of molecular structure and region for NICS probes. The information about probe region would be summarized in **area B**. In **area C**, user could adjust the probe region, modify the height, and set interval for ghost atoms.

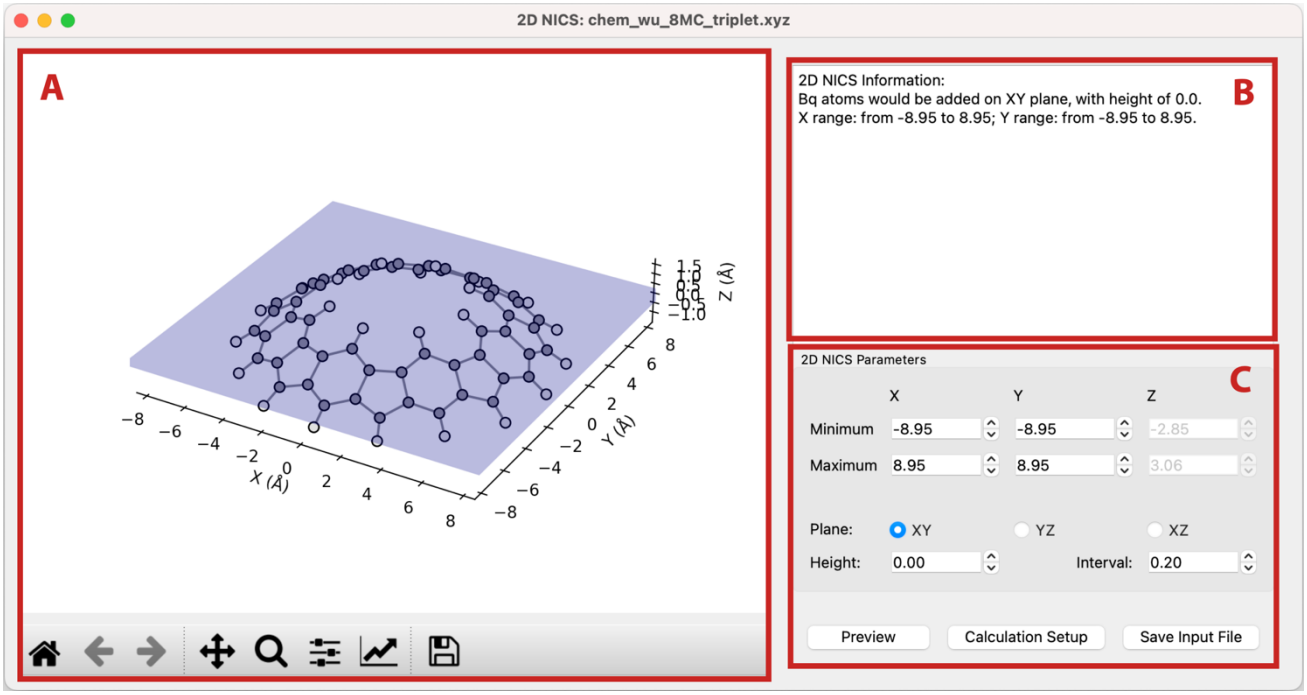


Figure 4.17 Main window of 2D NICS function.

The graph in **area A** would not be updated, unless user click the "Preview" button in **area C**.

4.6.2 Example: AWA Molecule

Here is a step-by-step tutorial for how to use 2D NICS module in py.**Aroma 3**. The molecule in this section is taken from **8MC** in *Chem*, **2018**, *4*, 1586 by Wu *et al*.

The molecule is placed parallel to XY plane, we want to put the NICS probes on the XY plane (target I) and XZ plane (target II), *i.e.,* height is 0, to reproduce the Figure 3b in the original literature. The interval of 0.2 Å (default value in py.**Aroma 3**) would be used.

< Procedure >

(0) Open *chem_wu_8MC_triplet.xyz* in py.**Aroma 3**.

(1) Click "Save Input File", by this time we can get the *.gjf* file for the target I.

(2) Choose "XZ" in "Plane:" option, and the "Y" parameters would be disabled, and "Z" parameters would be enabled.

(3) Click “Preview” to check the probe region.

(4) Click “Save Input File”, by this time we can get the *.gjf* file for the target II.

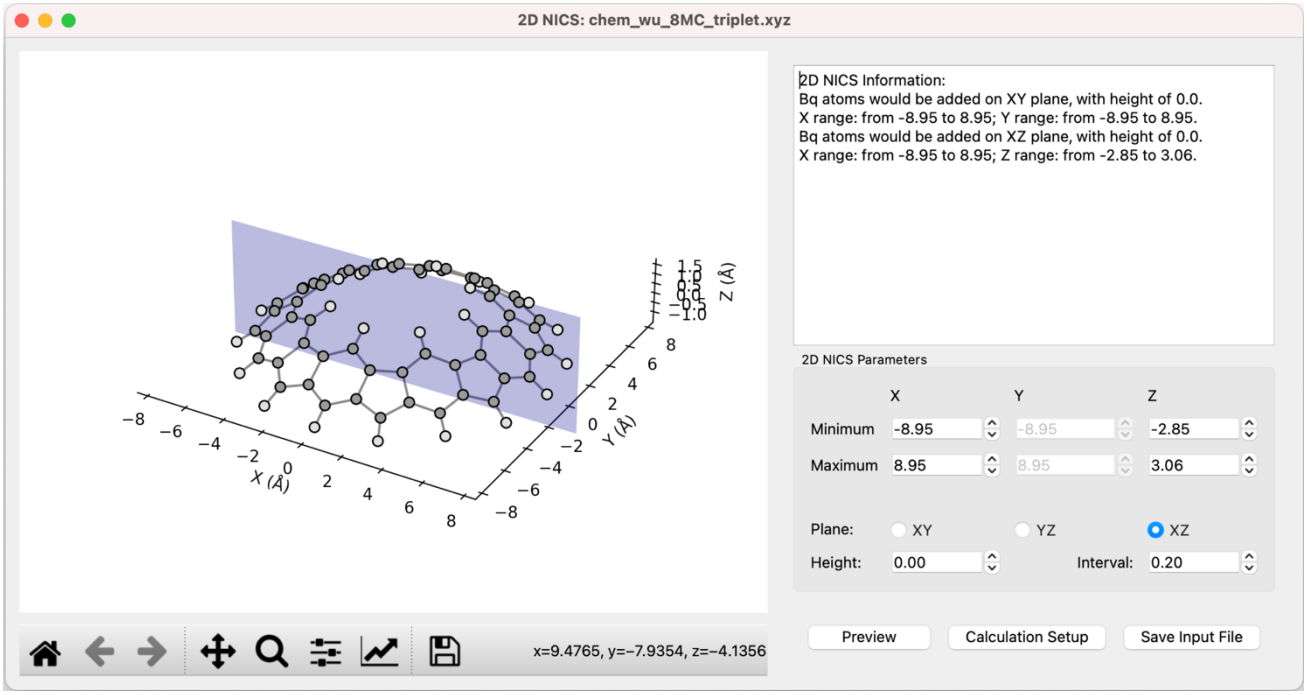


Figure 4.18 Screenshot of final state.

4.7 Create Input File for 3D NICS

3D NICS (ICSS) function could be accessed from 3D NICS button after open a file. The usage of 3D NICS module is very similar to that of 2D NICS module described in *Section 4.6*. The details would be skipped. As an example, ***c*-T12^{4+}** from *Nat. Chem.*, **2020**, *12*, 242 is shown in Figure 4.19.

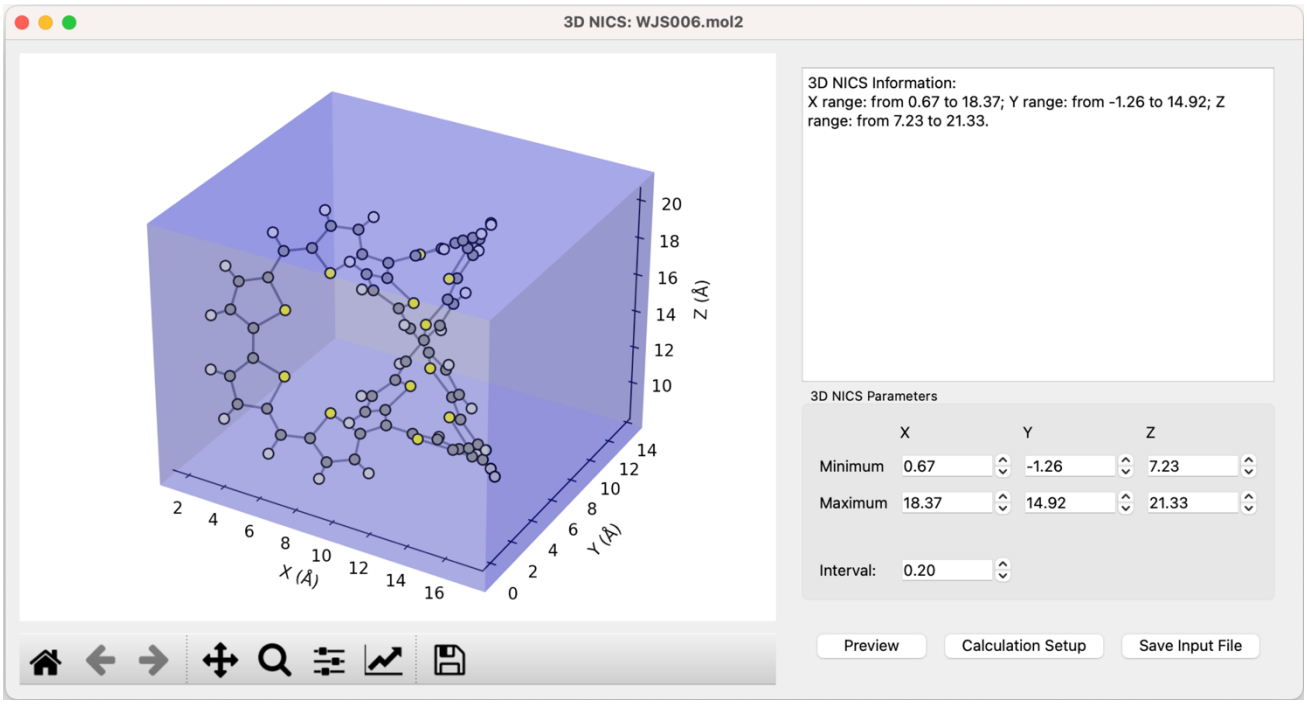


Figure 4.17 Main window of 3D NICS function.

***NOTE**: Gaussian has a limit on the number of ghost atoms, and it is somehow dependent of the version of Gaussian and the computer. To make Gaussian run normally, I set a limit to* py.**Aroma 3** *on the number of ghost atoms to 7000 in one .gjf file.*

4.8 Analyze Output File for NICS Calculations

4.8.1 Basic Information

py.**Aroma 3** could automatically extract shielding tensors from *Gaussian* output file, when an output file with ghost atoms was read, the geometry (without ghost atoms) would be displayed on the main window (Figure 4.18). User can access BLA, HOMA, POAV module from current window, and analyze the NICS output by "NICS Output" button.

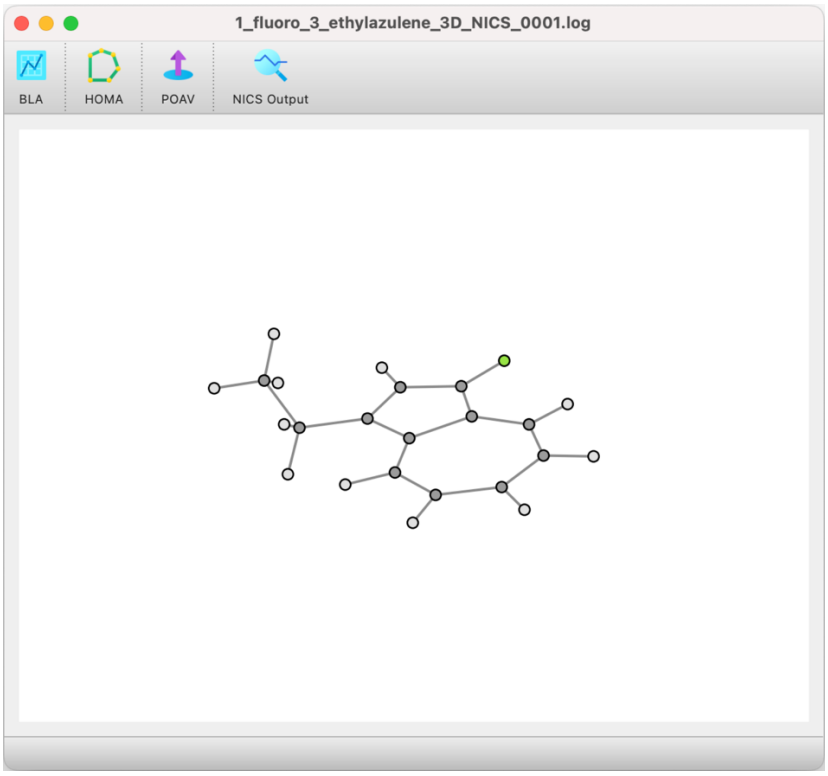


Figure 4.18 Main window when reading a output file including ghost atoms.

4.8.2 Single Point NICS

For single point NICS, as shown in Figure 4.19, the geometry with ghost atoms would be displayed in **area A**, and the shielding tensors are summarized in **area B**. The atom indices of ghost atoms could be hide by unchecking the "Show Bq No." in **area C**. By default, the ZZ component of shielding tensors would be displayed in **area B**, user can check the radio button in **area C** for other component and the data in **area B** would be updated automatically. The result in **area B** could be saved as "*.txt*" file by clicking the "Save .txt" button in **area C**.

4.8.3 1D NICS Scan

Main function of 1D NICS scan output is almost same to singlet point NICS, as shown in Figure 4.20. The atom indices of ghost atoms would be hidden by default for clarity. And the NICS scan plot would be displayed in the middle. When the component of shielding tensor is changed, the NICS scan

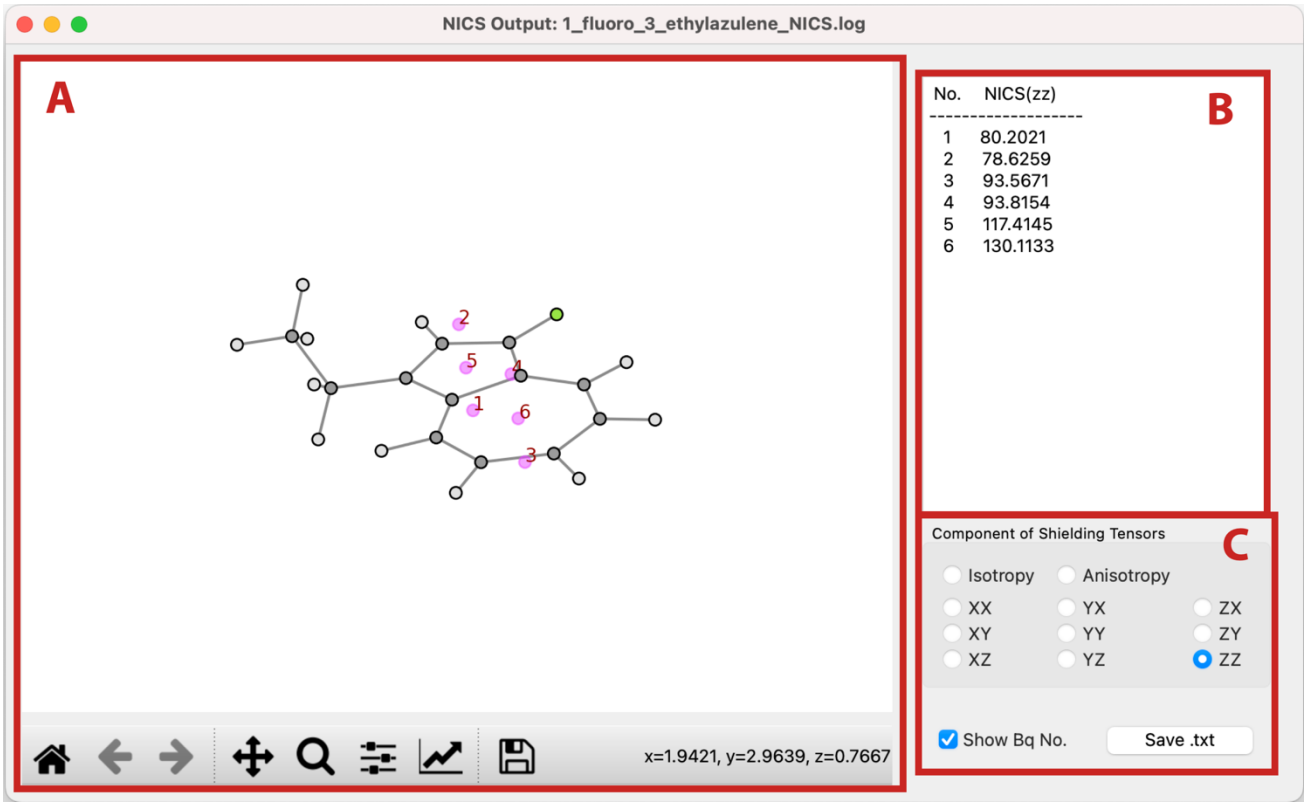


Figure 4.19 Screenshot of single point NICS output window.

plot would also be updated automatically. User could also save the plot by clicking "Save .png" button and save the data to *.xlsx* file by "Save .xlsx" button.

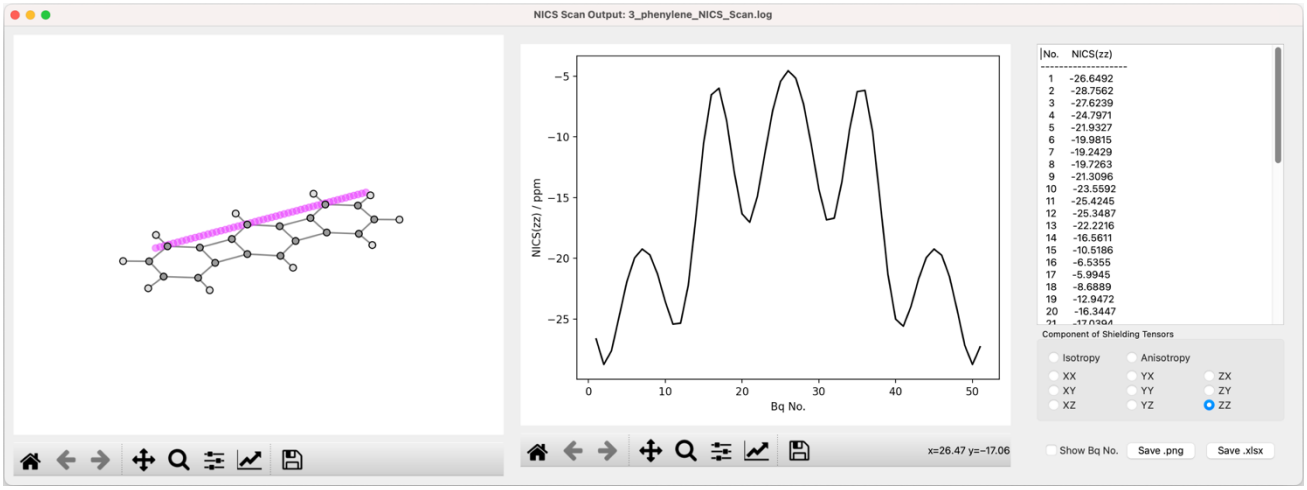


Figure 4.20 Screenshot of 1D NICS scan output window.

4.8.4 2D NICS/ICSS

Main window of 2D NICS/ICSS output is shown in Figure 4.21. In **area A** there is a preview of molecule and region of NICS probes. In **area B** a heatmap of 2D NICS/ICSS plot would be shown. User can modify the component of shielding tensors from **area C**, the heatmap would be automatically

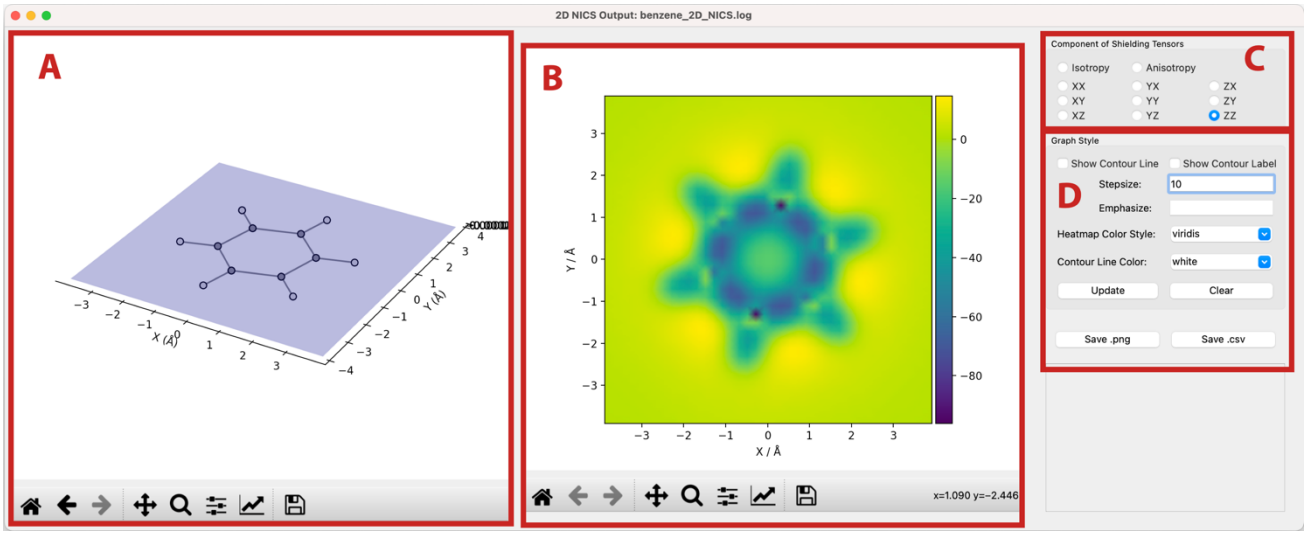


Figure 4.21 Screenshot of 2D NICS output window.

updated. User can save the plot as image by "Save .png" button and save the data to *.csv* file by "Save .csv" button in **area D**.

In **area D**, user can custom the heatmap. To show contour line in the heatmap, user can check the "Show Contour Line", the interval of contour line is controlled in "Stepsize" text bar.

If user want to highlight a specified contour line, just input the value of in "Emphasize" text bar and click "Update". The label of highlighted contour line could be added by checking the "Show Contour Label". Figure 4.22 gave an example of heatmap.

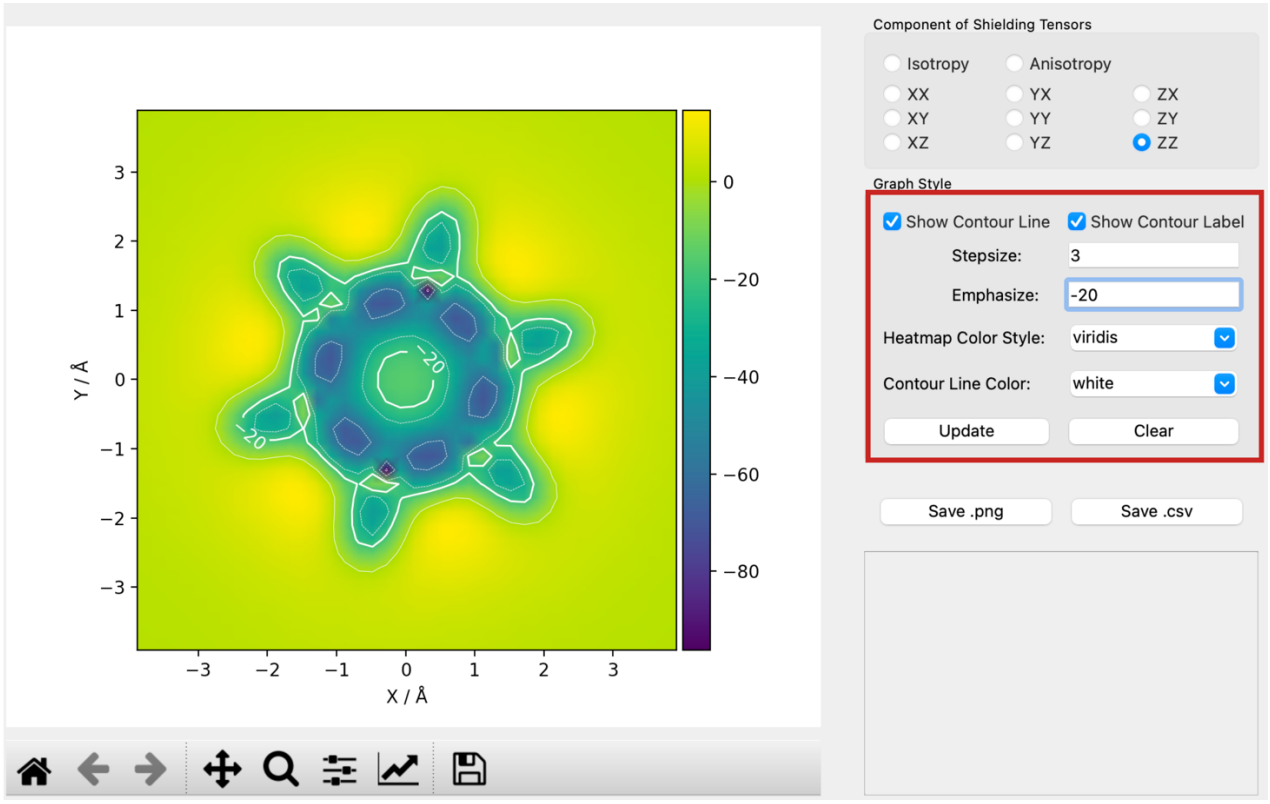


Figure 4.22 An example of heatmap and the parameters.

The heatmap function of py.**Aroma 3** is powered by *matplotlib*, thus, the color definition follows the rules of *matplotlib*. User can change the color style of heatmap and color of contour line from the list in **area D**. The colors those not listed also supported, user can just input the name of color and click "Update". For more information about color definition in *matplotlib*, please refer to or the Appendix I for colormap and Appendix II for single color specifying.

4.8.5 3D NICS/ICSS

Main window of 3D NICS/ICSS output is shown in Figure 4.23. In **area A** there is a preview of molecule and region of NICS probes. 2D NICS/ICSS function is also supported in 3D NICS/ICSS module, so, a 2D heatmap on XY plane would be displayed in **area B** by default, user could choose the plane and height in **area D** and custom the plotting style in **area E** just like those in 2D NICS/ICSS module. In **area C**, user could choose which component of shielding tensors would be investigated and the data could be saved as *.cube* file.

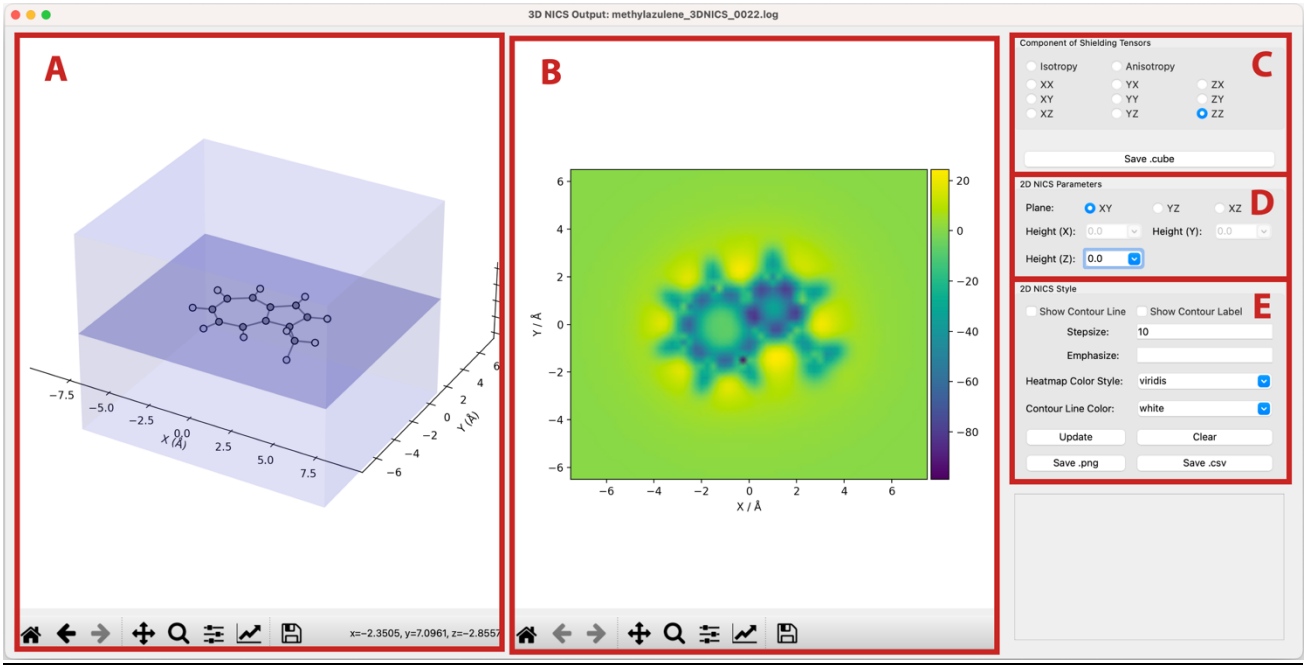


Figure 4.23 Screenshot of 3D NICS output window.

4.9 Generate Computational Supporting Material

The **CSIgen** module would be active when reading a *Gaussian* output file without ghost atoms to py.**Aroma 3**. The main window of **CSIgen** module is show in Figure 4.24.

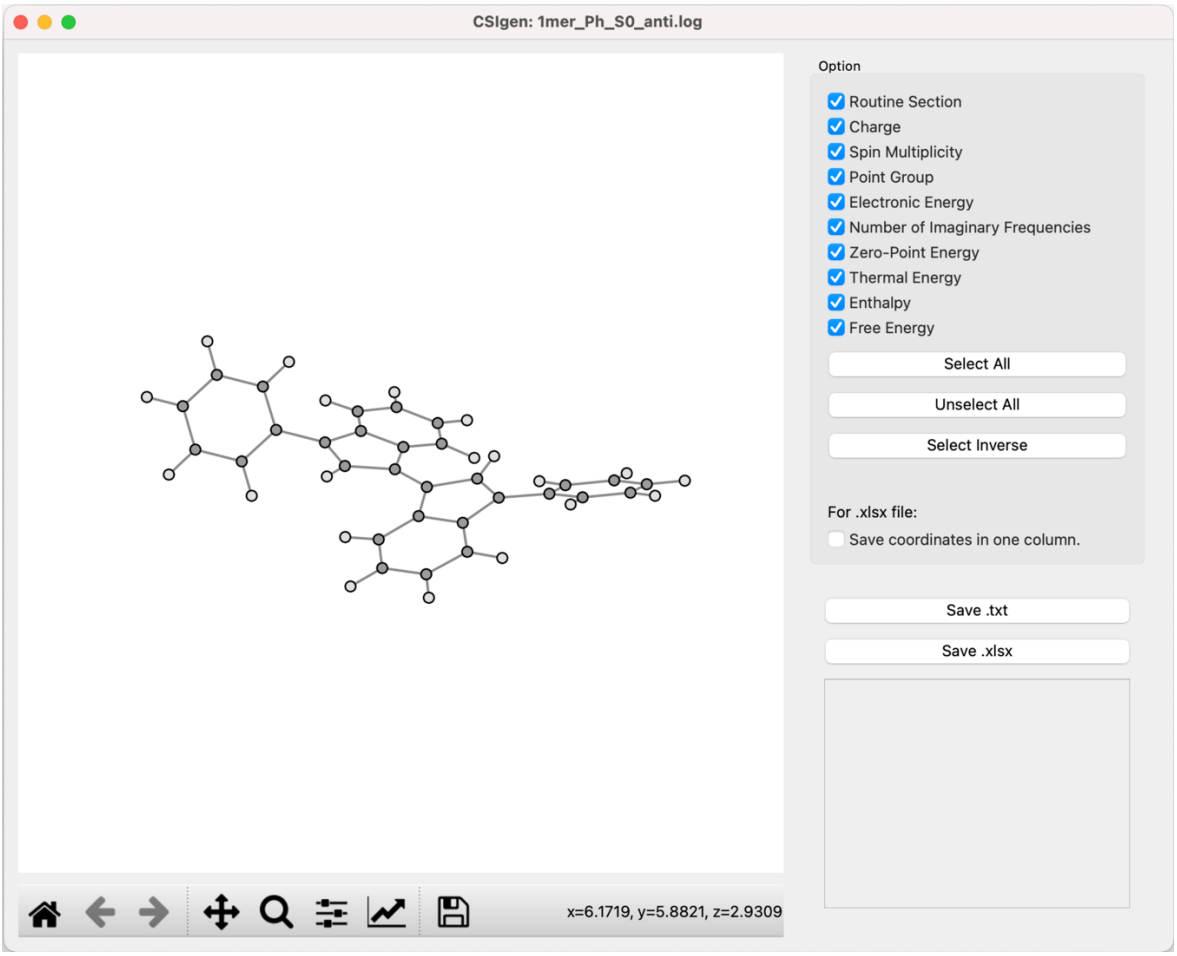


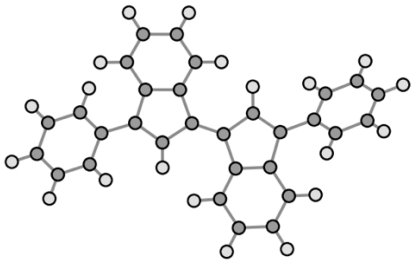
Figure 4.24 Main window of **CSIgen** module.

In **area A**, the geometry of molecule would be displayed. In **area B**, user could choose the information those are needed for the supporting information. By default, all information would be selected. The "Number of Imaginary Frequencies", "Zero-Point Energy", "Thermal Energy", "Enthalpy" and "Free Energy" only available when a *freq* job was detected.

User could choose to save the supporting information as "*.txt*" or "*.xlsx*". For saving as "*.xlsx*" file, by default, the Cartesian coordinated would be save in two columns, this could be turned off by checking the "Save coordinates in one column.".

Following page give an example using **CSIgen** module in py.**Aroma 3** to generate supporting information. The geometry in upper left was added later. ***HINT****: When saving the coordinates in two columns, the width could just fit to the "Moderate" margins setting in Microsoft Word.*

1mer_Ph_S0_anti



```
#p opt freq rwb97xd/6-31g(d,p)
```

Charge = 0, Multiplicity = 1, Point group = C1

Number of imaginary frequencies = 0

Electronic Energy = -1154.87028937 Hartree

Sum of electronic and zero-point Energies = -1154.463745 Hartree

Sum of electronic and thermal Energies = -1154.441758 Hartree

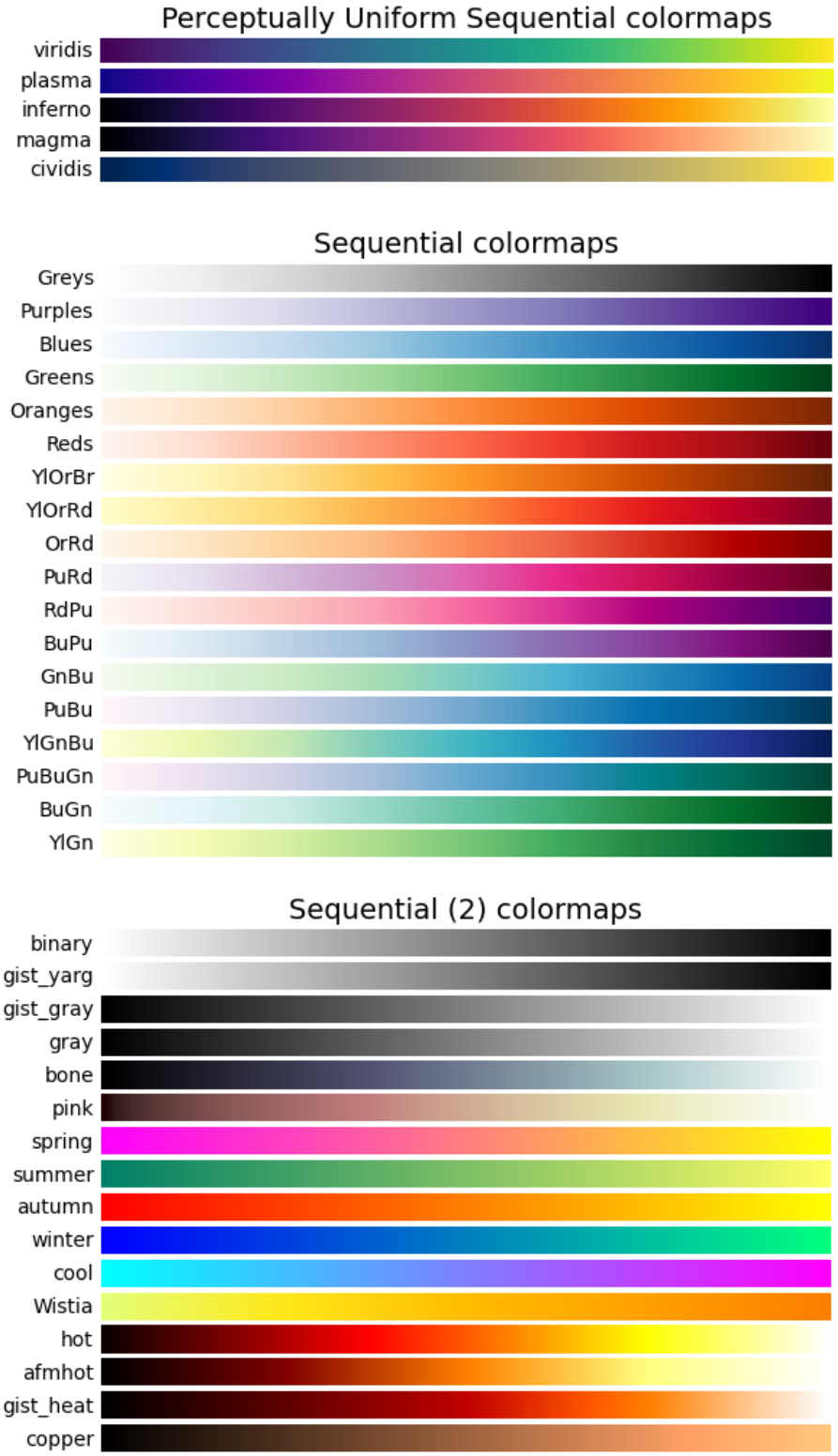
Sum of electronic and thermal Enthalpies = -1154.440814 Hartree

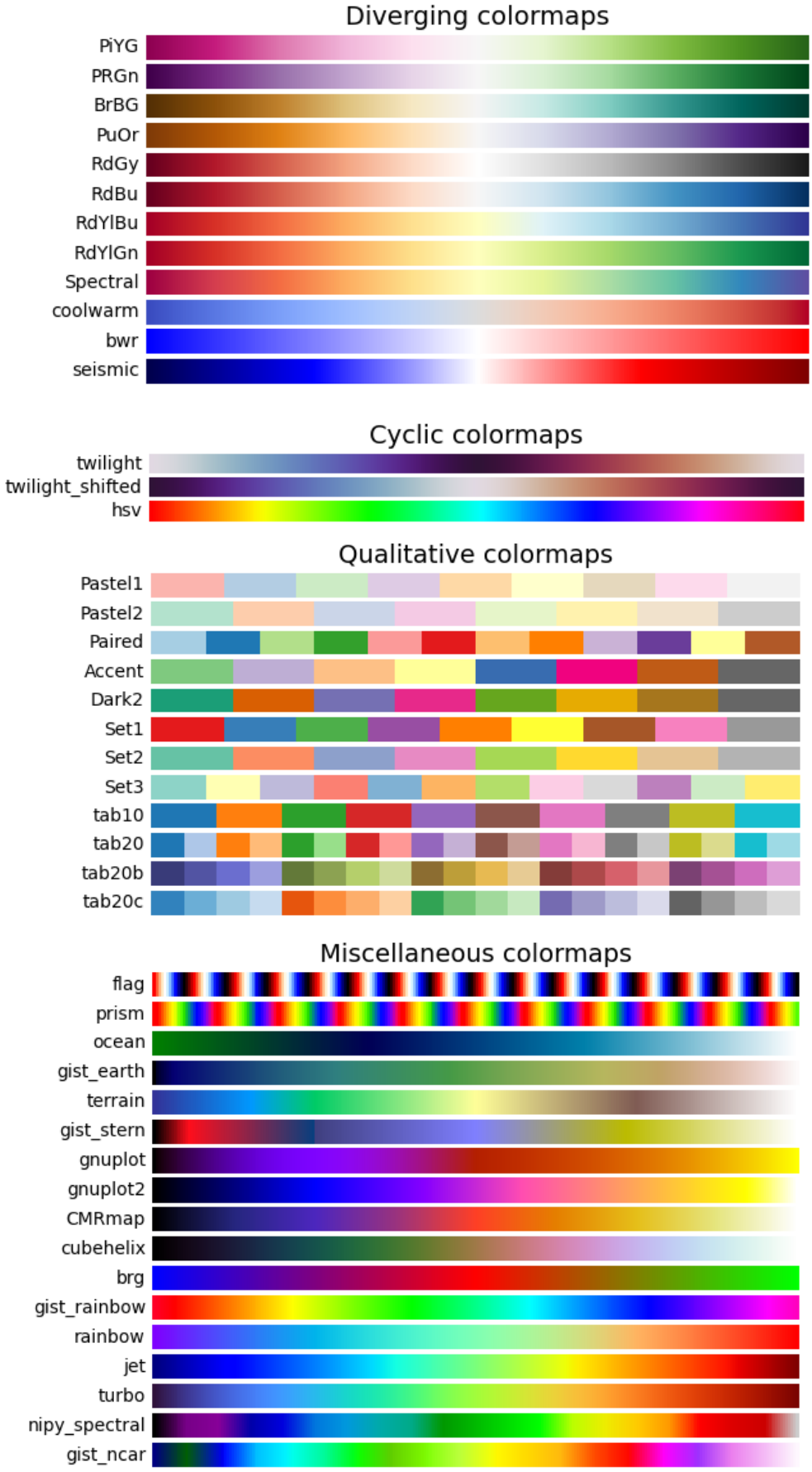
Sum of electronic and thermal Free Energies = -1154.516642 Hartree

	Cartesian Coordinates				Cartesian Coordinates		
Atoms	***X***	***Y***	***Z***	**Atoms**	***X***	***Y***	***Z***
H	4.435677	-2.606388	-0.231397	C	3.356437	-2.664594	-0.328848
C	0.569521	-2.799826	-0.699768	C	2.571309	-1.527073	-0.212788
C	2.735491	-3.884182	-0.603806	C	1.358363	-3.948847	-0.791078
C	1.171317	-1.589434	-0.381275	H	3.334927	-4.784470	-0.693673
H	0.890493	-4.899083	-1.027388	H	-0.491294	-2.867073	-0.908157
C	2.946798	-0.115707	-0.003726	C	1.825138	0.639775	-0.089644
H	1.789433	1.700971	0.109068	C	0.648524	-0.205519	-0.281946
H	-4.435780	2.606331	-0.231253	C	-3.356547	2.664605	-0.328741
C	-0.569663	2.800007	-0.699804	C	-2.571346	1.527130	-0.212737
C	-2.735690	3.884236	-0.603706	C	-1.358578	3.948982	-0.791055
C	-1.171361	1.589573	-0.381277	H	-3.335187	4.784487	-0.693533
H	-0.890777	4.899247	-1.027383	H	0.491130	2.867330	-0.908265
C	-2.946755	0.115735	-0.003721	C	-1.825064	-0.639688	-0.089728
H	-1.789301	-1.700902	0.108896	C	-0.648491	0.205678	-0.281947
C	-4.302591	-0.380861	0.272258	C	-6.864325	-1.376529	0.828160
C	-5.157951	0.298025	1.149401	C	-4.753603	-1.566015	-0.321422
C	-6.022475	-2.060311	-0.044873	C	-6.427119	-0.197450	1.425233
H	-4.812770	1.203896	1.637694	H	-4.104710	-2.086849	-1.019008
H	-6.358149	-2.977689	-0.518023	H	-7.074646	0.336285	2.113683
H	-7.856805	-1.760298	1.041443	C	4.302664	0.380807	0.272255
C	6.864458	1.376320	0.828162	C	5.158007	-0.298169	1.149344
C	4.753723	1.565971	-0.321369	C	6.022624	2.060190	-0.044816
C	6.427205	0.197228	1.425180	H	4.812791	-1.204049	1.637593
H	4.104843	2.086876	-1.018915	H	6.358333	2.977577	-0.517923
H	7.074719	-0.336576	2.113587	H	7.856959	1.760028	1.041449

Appendix

I Choosing Colormaps in *Matplotlib*




Diverging colormaps
PiYG
PRGn
BrBG
PuOr
RdGy
RdBu
RdYlBu
RdYlGn
Spectral
coolwarm
bwr
seismic
Cyclic colormaps
twilight
twilight_shifted
hsv
Qualitative colormaps
Pastel1
Pastel2
Paired
Accent
Dark2
Set1
Set2
Set3
tab10
tab20
tab20b
tab20c
Miscellaneous colormaps
flag
prism
ocean
gist_earth
terrain
gist_stern
gnuplot
gnuplot2
CMRmap
cubehelix
brg
gist_rainbow
rainbow
jet
turbo
nipy_spectral
gist_ncar

II Specifying Colors in *Matplotlib*

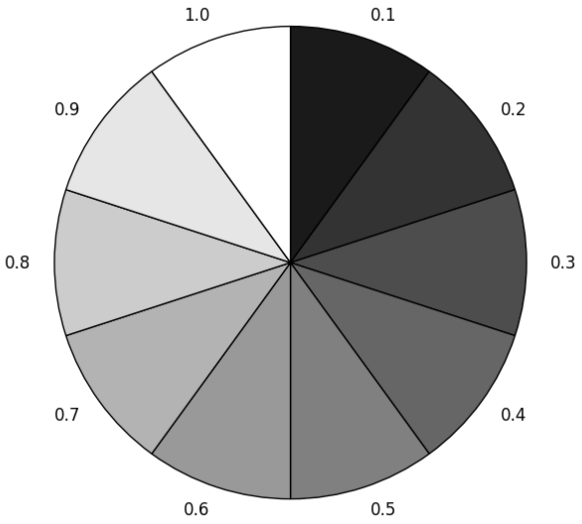
Following contents are taken from https://matplotlib.org/stable/tutorials/colors/colors.html , *Matplotlib* recognizes the following formats to specify a color.

Format	Example
RGB or RGBA (red, green, blue, alpha) tuple of float values in a closed interval [0, 1].	• `(0.1, 0.2, 0.5)` • `(0.1, 0.2, 0.5, 0.3)`
Case-insensitive hex RGB or RGBA string.	• `'#0f0f0f'` • `'#0f0f0f80'`
Case-insensitive RGB or RGBA string equivalent hex shorthand of duplicated characters.	• `'#abc'` as `'#aabbcc'` • `'#fb1'` as `'#ffbb11'`
String representation of float value in closed interval [0, 1] for grayscale values.	• `'0'` as black • `'1'` as white • `'0.8'` as light gray
Single character shorthand notation for some basic colors.	• `'b'` as blue • `'g'` as green • `'r'` as red • `'c'` as cyan • `'m'` as magenta • `'y'` as yellow • `'k'` as black • `'w'` as white
Case-insensitive X11/CSS4 color name with no spaces.	• `'aquamarine'` • `'mediumseagreen'`
Case-insensitive color name from xkcd color survey with 'xkcd:' prefix.	• `'xkcd:sky blue'` • `'xkcd:eggshell'`
Case-insensitive Tableau Colors from 'T10' categorical palette.	• `'tab:blue'` • `'tab:orange'` • `'tab:green'` • `'tab:red'` • `'tab:purple'` • `'tab:brown'` • `'tab:pink'` • `'tab:gray'` • `'tab:olive'` • `'tab:cyan'`
"CN" color spec where 'C' precedes a number acting as an index into the default property cycle.	• `'C0'` • `'C1'`

And here are some color names available in *matplotlib*.

black	dimgrey	dimgray	gray
grey	darkgrey	darkgray	silver
lightgray	lightgrey	gainsboro	whitesmoke
white	snow	rosybrown	lightcoral
indianred	brown	firebrick	maroon
darkred	red	mistyrose	salmon
tomato	darksalmon	coral	orangered
lightsalmon	sienna	seashell	chocolate
saddlebrown	sandybrown	peachpuff	peru
linen	bisque	darkorange	burlywood
antiquewhite	tan	navajowhite	blanchedalmond
papayawhip	moccasin	orange	wheat
oldlace	floralwhite	darkgoldenrod	goldenrod
cornsilk	gold	lemonchiffon	khaki
palegoldenrod	darkkhaki	ivory	beige
lightyellow	lightgoldenrodyellow	olive	yellow
olivedrab	yellowgreen	darkolivegreen	greenyellow
chartreuse	lawngreen	sage	lightsage
darksage	honeydew	darkseagreen	palegreen
lightgreen	forestgreen	limegreen	darkgreen
green	lime	seagreen	mediumseagreen
springgreen	mintcream	mediumspringgreen	mediumaquamarine
aquamarine	turquoise	lightseagreen	mediumturquoise
azure	lightcyan	paleturquoise	darkslategray
darkslategrey	teal	darkcyan	aqua
cyan	darkturquoise	cadetblue	powderblue
lightblue	deepskyblue	skyblue	lightskyblue
steelblue	aliceblue	dodgerblue	lightslategrey
lightslategray	slategray	slategrey	lightsteelblue
cornflowerblue	royalblue	ghostwhite	lavender
midnightblue	navy	darkblue	mediumblue
blue	slateblue	darkslateblue	mediumslateblue
mediumpurple	blueviolet	indigo	darkorchid
darkviolet	mediumorchid	thistle	plum
violet	purple	darkmagenta	fuchsia
magenta	orchid	mediumvioletred	deeppink
hotpink	lavenderblush	palevioletred	crimson
pink	lightpink		

About grayscale values:



京都大学
KYOTO UNIVERSITY