A Guide to the REG analysis

(and REG.py documentation)

Leonardo José Duarte and Fabio Falcioni.

REG.py
Version 1.0
August 2021

This is a guide on how to perform the REG analysis using the REG.py library. If you use it, please cite: Thacker, J.C.R. & Popelier, P.L.A. Theor Chem Acc (2017) 136: 86.

To report bugs, ask questions and send suggestions, please send an email to
fabio.falcioni@manchester.ac.uk

This manual was written under the supervision of Prof Dr. Paul Popelier.

REG.py is available to download at: github.com/FabioFalcioni

Contents

1 The IQA Energy Decomposition Scheme

From the basis of the Quantum Theory of Atom in Molecules (QTAIM), the Interacting Quantum Atoms (IQA) emerges as an energy decomposition scheme that split the molecular, or supramolecular, system energy, E_{IQA}^{sys}, into a sum of individual atomic energies composed of *intra*-atomic and *inter*-atomic energy components.

Such terms provide a very detailed description of a system's electronic density, allowing the proper quantification of chemical interactions. The IQA approach has already been applied to a variety of chemical problems, such as S_N2 reactions, hydrogen-bonded systems, the fluorine gauche effect, the rotation barriers on biphenyl, etc. In the next sections, we are going to explore the fundamentals of IQA, presenting the principal equations and how to manipulate them for use in different situations.

1.1 Intra- and Inter-atomic Contributions

The total system energy is recovered by summing each individual atom energy, E_{IQA}^{A}, according to:

$$E_{IQA}^{sys} = \sum_{A=1}^{N} E_{IQA}^{A} \tag{1}$$

where N is the total number of atoms in the system and A is the atomic label. Each atomic term can be expanded as a sum of intra- and interatomic contributions. Equation 1 becomes:

$$E_{IQA}^{sys} = \sum_{A=1}^{N} E_{Intra}^{A} + \sum_{A=1}^{N}\sum_{B>A}^{N} V_{Inter}^{AB} \tag{2}$$

with V_{Inter}^{AB} being the potential energy between atoms A and B. Alternatively, the double summation on the right-hand side of Equation 2 can be rewritten as:

$$\sum_{A=1}^{N}\sum_{B>A}^{N} V_{Inter}^{AB} \equiv \frac{1}{2}\sum_{A=1}^{N}\sum_{B\neq A}^{N} V_{Inter}^{AB} \tag{3}$$

it is important to keep such definition in mind, since both forms are used by programs like AIMAll and MORPHY.

The intratomic term can be split in its kinetic, T, electron-electron, V_{ee}, and electron-nucleus, V_{en}, potential energies:

$$E_{Intra}^{A} = T^{A} + V_{ee}^{AA} + V_{en}^{AA} \tag{4}$$

E_{intra}^{A} is a measure of the intrinsic stability of an atom in the system and behaves as the familiar steric repulsion. The total intratomic energy is obtained directly from the three-dimensional integration of the electronic density over the topological atom volume (i.e. atomic basin).The interatomic contribution can also be split into several terms:

$$V_{Inter}^{AB} = V_{nn}^{AB} + V_{en}^{AB} + V_{ne}^{AB} + V_{ee}^{AB} \tag{5}$$

where the subscripts e and n stand for *electron* and *nucleus* respectively. Note that the order of the superscripts and subscripts does matter. V_{en}^{AB} is equivalent to "potential energy between the electrons from atom A and the nucleus of atom B", whereas V_{ne}^{AB} means "potential energy between the nucleus of atom A and electrons from atom B". As a general reminder: $V_{en}^{AB} \neq V_{ne}^{AB} \equiv V_{en}^{BA} \neq V_{ne}^{BA}$.

The electron-electron potential energy can also be subdivided in two contributions: *classic Coulombic energy* and the *exchange-correlation interaction*, according to:

$$V_{ee}^{AB} = V_{coul}^{AB} + V_{xc}^{AB} \tag{6}$$

V_{xc}^{AB} is associated to the degree of covalency between the atoms A and B. All classic terms can be grouped into the classical term: $V_{cl}^{AB} = V_{nn}^{AB} + V_{en}^{AB} + V_{ne}^{AB} + V_{coul}^{AB}$. Using this definition, Equation 5 can be written as:

$$V_{Inter}^{AB} = V_{cl}^{AB} + V_{xc}^{AB} \tag{7}$$

All interatomic terms are obtained direct from a six-dimension integration of the electronic density over both, A and B, topological atoms volume.

1.2 Starting from the Reduced Density-Matrices

Once the desired multi-electronic wavefunction, Ψ, is found, the first and second-order density-matrices can be obtained following:

$$\rho_1(\boldsymbol{r_1}, \boldsymbol{r_{1'}}) = N_e \int \Psi(\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_{N_e}}) \Psi^*(\boldsymbol{x'_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_{N_e}}) d\boldsymbol{x_2} d\boldsymbol{x_3} ..., d\boldsymbol{x_{N_e}} \tag{8}$$

and;

$$\rho_2(\boldsymbol{r_1}, \boldsymbol{r_2}) = \frac{N_e(N_e - 1)}{2} \int \Psi(\boldsymbol{x_1}, ..., \boldsymbol{x_{N_e}}) \Psi^*(\boldsymbol{x_1}, ...\boldsymbol{x_{N_e}}) d\boldsymbol{x_3}, ..., d\boldsymbol{x_{N_e}} \tag{9}$$

where $\boldsymbol{x_n}$ is the vector of the spatial, denoted by $\boldsymbol{r_n}$ and spin coordinates for electron n, N_e is the total number of electrons.

According to the Born-Oppenheimer approximation, the multi-electronic Hamiltonian, in atomic units, is given by:

$$\hat{H}_{N_e} = \hat{T} + \hat{V}_{en} + \hat{V}_{ee} = -\sum_{n=1}^{N_e} \frac{1}{2} \bigtriangledown_n^2 - \sum_{n=1}^{N_e} \sum_{A=1}^{N} \frac{Z_A}{r_{nA}} + \sum_{n=1}^{N_e} \sum_{l>n}^{N_e} \frac{1}{r_{nl}} \tag{10}$$

where $\hat{T}$ is the one-electron kinetic energy operator. The total energy is given by:

$$E_{total} = E_{electronic} + V_{nn} \tag{11}$$

the potential energy between two nuclei is simply defined as:

$$V_{nn} = \sum_{A=1}^{N} \sum_{B>A}^{N} V_{nn}^{AB} = \sum_{A=1}^{N} \sum_{B>A}^{N} \frac{Z_A Z_B}{r_{AB}} \tag{12}$$

the electronic energy can be obtained solving the Schrödinger equation

$$\hat{H}_{N_e} \Psi(\boldsymbol{x_1}, ..., \boldsymbol{x_{N_e}}) = E_{Eletronic} \Psi(\boldsymbol{x_1}, ..., \boldsymbol{x_{N_e}}) \tag{13}$$

which results in

$$E_{eletronic} = \int_{\infty} \hat{T} \rho_1(\boldsymbol{r_1}, \boldsymbol{r_{1'}}) d\boldsymbol{x_1} + \int_{\infty} \hat{V}_{en} \rho_1(\boldsymbol{r_1}, \boldsymbol{r_{1'}}) d\boldsymbol{x_1} + \int_{\infty} \int_{\infty} \hat{V}_{ee} \rho_2(\boldsymbol{r_1}, \boldsymbol{r_2}) d\boldsymbol{x_1} d\boldsymbol{x_2} \tag{14}$$

In order to compute individual terms for each atom the AIM topological partitioning method is invoked.

1.2.1 The atoms within the electronic density

Using the vector field traced by the gradient, $\bigtriangledown\rho(\boldsymbol{r})$, of the electronic density, one obtains the so-called interatomic surfaces (denoted S), which are boundaries of an atomic basin (denoted Ω) and are defined by:

$$\bigtriangledown \rho(\boldsymbol{r}) \cdot \boldsymbol{n}(\boldsymbol{r}) = 0 \tag{15}$$

where $\boldsymbol{n}(\boldsymbol{r})$ is the vector normal to the interatomic surface, Ω. With all the atomic basins in the system properly identified, the atomic properties are extracted by integrating over the atomic space. The kinetic energy contribution for atom A is given by :

$$T^A = \int \hat{T}\rho_1^A(\boldsymbol{r})d\boldsymbol{r} \tag{16}$$

The mono-electronic inter-atomic terms are obtained as follows:

$$V_{en}^{AB} = \int_{\Omega_A} \hat{V}_{en}^B \rho_1(\boldsymbol{r_1}, \boldsymbol{r_{1'}})d\boldsymbol{x_1} = -\int_{\Omega_A} \frac{\rho_1(\boldsymbol{r_1})Z^B}{r_{1B}} d\boldsymbol{r_1} \tag{17}$$

The two-electron terms are obtained from the second-order density-matrix, which describes how electrons interact with each other:

$$V_{ee}^{AB} = \int_{\Omega_A}\int_{\Omega_B} \rho_2(\boldsymbol{r_1}, \boldsymbol{r_2}) r_{12}^{-1} d\boldsymbol{x_1} d\boldsymbol{x_2} \tag{18}$$

if $A = B$, then:

$$V_{ee}^{AA} = \frac{1}{2}\int_{\Omega_A}\int_{\Omega_A} \rho_2(\boldsymbol{r_1}, \boldsymbol{r_2}) r_{12}^{-1} d\boldsymbol{x_1} d\boldsymbol{x_2} \tag{19}$$

Equation 18 contains the Coulombic, exchange and correlation terms, which can be individually calculated by invoking the three types contributions to ρ_2:

$$\begin{aligned} V_{ee}^{AB} = \int_{\Omega_A}\int_{\Omega_B} \rho_2(\boldsymbol{r_1}, \boldsymbol{r_2}) r_{12}^{-1} d\boldsymbol{x_1} d\boldsymbol{x_2} = \\ \int_{\Omega_A}\int_{\Omega_B} \rho(\boldsymbol{r_1})\rho(\boldsymbol{r_2}) r_{12}^{-1} d\boldsymbol{r_1} d\boldsymbol{r_2} - \int_{\Omega_A}\int_{\Omega_B} \rho_1(\boldsymbol{r_1}, \boldsymbol{r_2})\rho_1(\boldsymbol{r_2}, \boldsymbol{r_1}) r_{12}^{-1} d\boldsymbol{r_1} d\boldsymbol{r_2} + \\ \int_{\Omega_A}\int_{\Omega_B} \rho_2^{corr} r_{12}^{-1} d\boldsymbol{r_1} d\boldsymbol{r_2} = V_{coul}^{AB} + V_{ex}^{AB} + V_{corr}^{AB} \end{aligned} \tag{20}$$

Exchange and correlation terms are typically lumped together in the exchange-correlation term. $V_{ex}^{AB} + V_{corr}^{AB} = V_{xc}^{AB}$. V_{ex}^{AB} is related to the covalency degree between atoms A and B, as well as the bond order. The classical electrostatic term, V_{coul}^{AB}, on the other hand, is related to the bond polarity and ionicity degree. The last term, V_{corr}^{AB} is responsible to increase the magnitude of the nucleus-electron potential energy and decreases the electron-electron repulsion. All these terms constitute the so-called *fine structure* of ρ_2.

1.3 Grouping Atoms

Consider a system formed by N atoms divided into three groups J, K and L. The total energy of the system is recovered as follow:

$$E_{IQA}^{sys} = \sum_{X=1}^{3} E_{IQA}^X = E_{IQA}^J + E_{IQA}^K + E_{IQA}^L \tag{21}$$

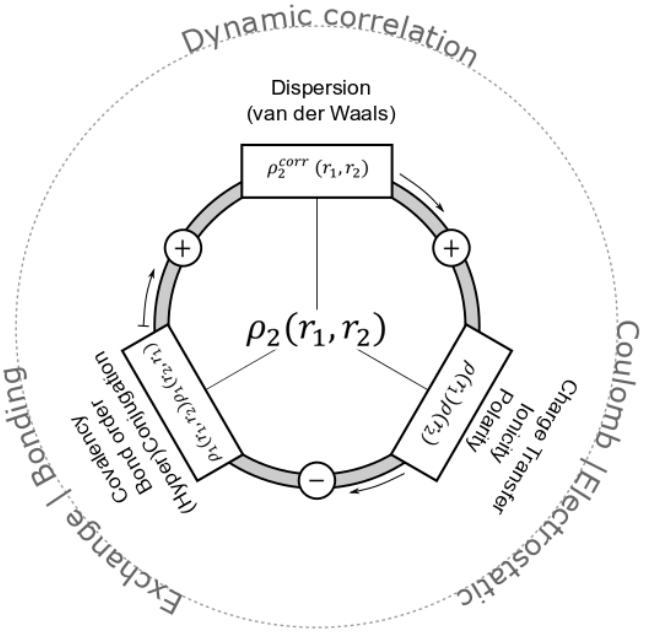


Figure 1: Representation of the fine structure of the second-order density-matrix ρ_2 . Each term is presented with its physical meaning. The "+" and "-" signs refer to the mathematical relationship between terms as shown in 20.

where $X = 1, 2$ or 3 correspond to J, K or L group. Each group contribution can be expanded in **intra**atomic-**intra**group ($Intra^2$), **inter**atomic-**intra**group($Inter/Intra$) and **inter**atomic-**inter**group($Inter^2$) terms.

The $Intra^2$ term corresponds to the sum of all intratomic energies of atoms that belong to group X and is defined as:

$$E^X_{Intra^2} = \sum_{A=1}^{N} E^A_{Intra} \qquad for\ A \in X \tag{22}$$

$Intra^2$ is associated with the intrinsic stability of the group and the steric effects.

The $Inter^2$ term is equivalent to the sum of the interatomic energy between two atoms of distinct groups,

$$E^X_{Inter^2} = \sum_{A=1}^{N} \sum_{B>A}^{N} V^{AB}_{Inter} \qquad for\ A \in X\ and\ B \notin X \tag{23}$$

alternatively, using the definition in Equation 3, one can write:

$$E^X_{Inter^2} = \frac{1}{2} \sum_{A=1}^{N} \sum_{B\neq A}^{N} V^{AB}_{Inter} \qquad for\ A \in X\ and\ B \notin X \tag{24}$$

Similarly to individual atomic terms, it is possible to split $Inter^2$ in classical and exchange-correlation interaction contributions. The chemical insights of the individual atomic terms are preserved and apply to $E^X_{Inter^2}$.

The $E^X_{Inter/Intra}$ is defined in the same way as $E^X_{Inter^2}$ with one exception: the atoms A and B MUST belong to the same group X.

$$E^X_{Inter/Intra} = \sum_{A=1}^{N} \sum_{B>A}^{N} V^{AB}_{Inter} \qquad for\ \{A, B\} \subset X \tag{25}$$

the alternative definition is also valid:

$$E^{X}_{Inter/Intra} = \frac{1}{2}\sum_{A=1}^{N}\sum_{B\neq A}^{N} V^{AB}_{Inter} \qquad for\ \{A,B\} \subset X \tag{26}$$

The principal advantage of using the grouping approach to partition the energetic terms consists of preserving the chemical information in a small number of terms.

2 Relative Energy Gradient (REG) Approach

In the last section, the IQA partition scheme was presented. It should be clear that, for a small system formed by ten atoms, a modest IQA partition will result in a set of $10 \times E_{intra}^{A}$ atomic terms, $45 \times V_{cl}^{AB}$ interatomic classical terms and $45 \times V_{xc}^{AB}$ interatomic exchange-correlation terms. With the increase of the partitioning level and/or the increase of the size of the system, the number of terms will explode and a manual analysis of such data becomes impractical, not to say impossible. The problem becomes worst when dealing with an energy surface, where the IQA terms need to be analysed over a coordinate change.

To overcome this adversity, Thacker and Popelier proposed the Relative Energy Gradient method, which compares the derivatives of each contribution with the overall energy variation over a control coordinate and ranks them in order to find the set of contributions that preserve the majority of chemical insights into the total energy profile. As stated by Thacker and Popelier, the REG method has two objectives:

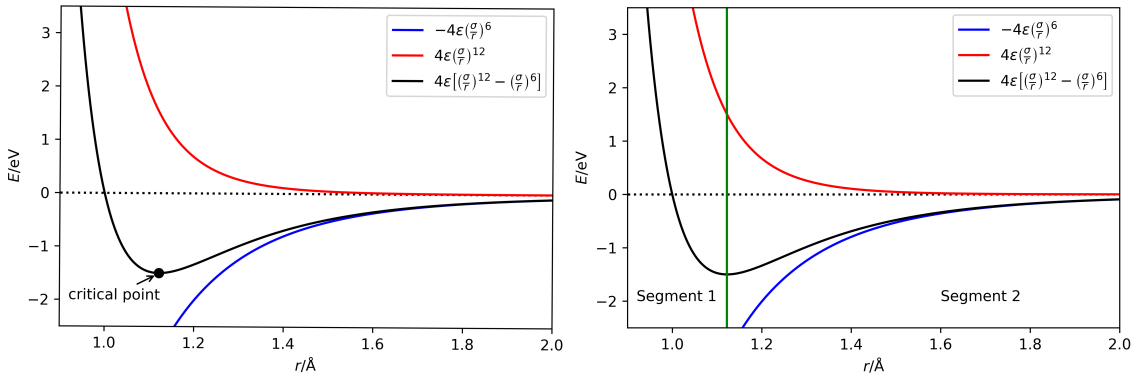
1. Determine the subset of energy contributions that preserves the chemical insights and best describes the overall profile of the total energy over any coordinate changes.

2. Extract chemical insight from the partitioned total energy.

2.1 Potential Energy Surface: Spliting into Segments

In a Potential Energy Surface (PES), each maximum critical point corresponds to an *energy barrier* separating two minimum critical points. However, the word "barrier" in this context is associated with an increase of energy, and energy segments go down when analysed from left to right. The sign of m_{REG} is independent of the direction of analysis (left to right or right to left). It is better to refer to an energy *segment* in a neutral way, hence the word "segment" instead of "barrier". A segment ideally, begins and ends at a critical (minimum, maximum and/or saddle points) or at a lateral limit of the function.

Consider the Lennard-Jones potential plot in Figure 2. The critical point corresponds to the equilibrium distance between two particles. From the critical point, the potential curve is divided in two segments, one at the left side of the critical point (segment 1), and one at the right (segment 2).



(a) Critical point

(b) Segment split

Figure 2: Lennard-Jones potential where $\sigma = 1.00$ and $\varepsilon = 1.50$. The blue, red and black lines correspond to the attractive contribution, repulsive contribution and the total energy respectively. In Figure 2a is the critical point corresponds to the equilibrium point of the system. In Figure 2b, the green line split the potential curve in two segments

In this example, segment 1 is dominated by the Pauli repulsion term, $\left(\frac{\sigma}{r}\right)^{12}$, while segment 2 is dominated by the dispersion interaction, $\left(\frac{\sigma}{r}\right)^{6}$. A way to visualise the aforementioned behaviour is to compare, for each segment, the ratio between each component

derivative and the total energy derivative with respect to changes in the control coordinate, calculating the REG coefficient, m_{REG}, as follows:

$$\frac{\mathrm{d}E_i}{\mathrm{d}s} = m_{REG,i}\frac{\mathrm{d}E_{total}}{\mathrm{d}s} \tag{27}$$

The greater the value of $m_{REG,i}$ the more important the contribution E_i to the overall behaviour of the potential energy surface.

However, Equation 27 is valid only for a perfect correlation. For all other cases, m_{REG} is approximated as the angular coefficient of the linear regression between E_i and E_{total}, as given:

$$E_i(s) = m_{REG,i}E_{total}(s) + c_i \tag{28}$$

where:

$$m_{REG,i} = \frac{n\sum_{s=1}^{n}\left(E_{total}(s)E_i(s)\right) - \sum_{s=1}^{n}E_{total}(s)\sum_{s=1}^{n}E_i(s)}{n\sum_{s=1}^{n}E_{total}^2(s) - \left(\sum_{s=1}^{n}E_{total}(s)\right)^2} \tag{29}$$

where n is the number of points in the PES and s is the control coordinate. c_i is the intercept, which is not relevant for the REG analysis. The approximation in 29 is better the closer to 1 the currently absolute value of the Pearson coefficient, r, is.

The m_{REG} values in Figure 3 confirms the statement that the segment 1 is dominated by the Pauli repulsion term, while the segment 2 is driven by the dispersion attractive term.

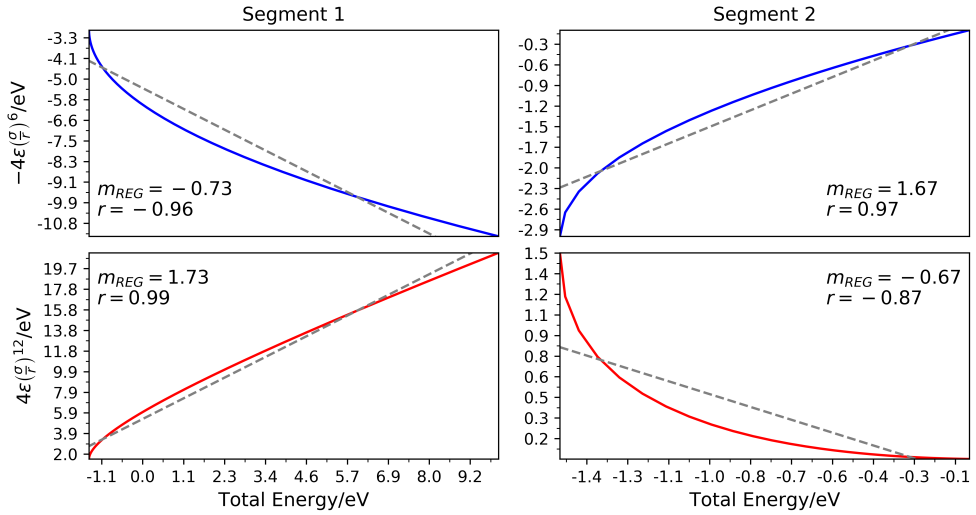


Figure 3: m_{REG} and r values for the attractive and repulsive contributions of the Lennard-Jones potential for both segments. The dashed gray lines correspond to the least square linear regression between the component and the total energy.

2.2 The Additive Property of the REG coefficient

The additive property of the REG coefficient states that if E_{total} is equal to a sum of m energy terms, $E_{total} = \sum_{l=0}^{m} E_l$, then $\sum_{l=0}^{m} m_{REG,l} = 1$. This property is easily derived from Equation 27 since the derivative of a function that is equal to a sum of other functions equals the sum of their derivatives.

$$\frac{\mathrm{d}E_{total}}{\mathrm{d}s} = \frac{\mathrm{d}\sum_{l=0}^{m}E_l}{\mathrm{d}s} = \sum_{l=0}^{m}\frac{\mathrm{d}E_l}{\mathrm{d}s} \tag{30}$$

then from Equation 27, it follows that

$$\sum_{l=0}^{m} \frac{\mathrm{d}E_l}{\mathrm{d}s} = \sum_{l=0}^{m} m_{REG,l} \frac{\mathrm{d}E_{total}}{\mathrm{d}s} \tag{31}$$

which leads to:

$$\sum_{l=0}^{m} m_{REG,l} = \frac{\sum_{l=0}^{m} \frac{\mathrm{d}E_l}{\mathrm{d}s}}{\frac{\mathrm{d}E_{total}}{\mathrm{d}s}} = \frac{\frac{\mathrm{d}E_{total}}{\mathrm{d}s}}{\frac{\mathrm{d}E_{total}}{\mathrm{d}s}} = 1 \tag{32}$$

The same is valid for the approximation in Equation 28, since:

$$\sum_{l=0}^{m} E_l(s) = \sum_{l=0}^{m} m_{REG,l} E_{total}(s) + \sum_{l=0}^{m} c_l \tag{33}$$

$$E_{total}(s) = E_{total}(s) \sum_{l=0}^{m} m_{REG,l} + C \tag{34}$$

and since we can prove that $C = 0$,(see Appendix B, section: 6) , we obtain that:

$$\sum_{l=0}^{m} m_{REG,l} = 1 \tag{35}$$

This property is useful when grouping IQA terms. As a consequence of the additive property, the REG coefficient of a group of terms equals the sum of the REG coefficient of each term, or: *The m_{REG} of a sum of each contribution is equal to the sum of each individual m_{REG} value.*

3 AIMALL Tips

AIMALL is a program used to integrate the wavefunction in order to extract the QTAIM properties of the system. Given a system of N atoms, AIMAll will create N ".int" files that contains all monoatomic properties and, if requested, $\frac{N^2-N}{2}$ ".int" files containing the interatomic properties. If you are running AIMAll via a command line, you may use the command **-encomp=4** to request all the IQA terms necessary to the REG analysis. An example is given:

```
/mnt/iusers01/pp01/n48687ld/AIMAll/aimqb.ish -nogui -encomp=4 -nproc=8
-delmog=false wavefunction.wfn
```

The **delmog** option is set to false, in order to prevent .mog files from being deleted. MOG files are used to reduce the integration error if necessary, by performing an integration with a better grid over the atoms that were poorly integrated. However, it must be noted that these files take up a lot of space.

To improve the integration grid, the following command line can be used:

```
/mnt/iusers01/pp01/n48687ld/AIMAll/aimqb.ish -nogui -encomp=4 -nproc=8
-bim=proaim -boaq=gs60 -iasmesh=superfine -delmog=false filename.wfn
```

-bim determines which integration method will be used, **-boaq** will define the basin outer angular quadrature and **-iasmesh** will define the integration mesh spacing.

To check the integration error in a specific atom, you may search for the L value in that atom's .int file. Usually a $L \leq 10^{-4}$ is sufficiently good. However, if you need to reintegrate a specific atom, you can use the following command line:

```
/mnt/iusers01/pp01/n48687ld/AIMAll/aimqb.ish -nogui -encomp=4 -nproc=8
-bim=proaim -boaq=gs60 -iasmesh=superfine -delmog=false -atoms=1,2,4
filename.wfn
```

In this example, AIMAll will run the integration over atom 1, 2 and 4. For more information about AIMAll command lines, please check the AIMAll documentation page.

3.1 AIMAll and DFT

The following DFT methods are implemented in AIMAll:

- LSDA
- B3LYP
- M06-2X
- PBE
- PBE0

However, when using DFT methods, the wavefunction file needs to be modified. In the first line of a .wfn or .wfx file, one can read:

```
GAUSSIAN              9 MOL ORBITALS    106 PRIMITIVES        8  NUCLEI
```

in order to use the DFT method, you need to write the method *exactly three spaces* after "NUCLEI". An example for B3LYP is given below:

```
GAUSSIAN              9 MOL ORBITALS    106 PRIMITIVES        8  NUCLEI    B3LYP
```

Modifying all wfn (or wfx) files manually is not recommended, since a small mistake can waste a huge amount of time. It is recommended using the following **sed** command in a terminal to modify the .wfn file:

```
$sed -i 's/NUCLEI/NUCLEI   METHOD/g' filename.wfn
```

changing METHOD to LSDA, B3LYP, M06-2X, PBE or PBE0. If the wavefunction file is not modified properly, the integration error will be huge (around 200 $kJmol^{-1}$).

At end of any .int file, there are three important notes that the user should be aware of. They are reproduced below:

2EDM Note:

For post-HF, natural orbital "wavefunctions" the Muller approximation of the two-electron density matrix (2EDM) in terms of natural orbitals of the one-electron density matrix (1EDM) is used to calculate 2EDM-dependent properties (i.e., Vee contributions and electronic localization and delocalization properties). For HF single-determinant wavefunctions the expression used for the 2EDM is exact. 2EDM-dependent properties are calculated with spin-orbital "self-interaction" terms included.

DFT Note:

VeeX represents the exchange-correlation functional for the wavefunction's underlying model. For Hartree-Fock wavefunctions, VeeX is the two-electron Hartree-Fock exchange functional. For wavefunctions of supported DFT models (currently LSDA, B3LYP and M062X), VeeX is the actual exchange-correlation functional of the corresponding DFT model and atomic contributions VeeX(A) to VeeX are explicitly calculated and unambiguous. However, since VeeX is at least partly just a one-electron functional for DFT models, the partitioning of VeeX(A) into interatomic (VeeX(A,B) and VeeX(A,A')) and intraatomic (VeeX(A,A)) contributions is ambiguous. Currently, interatomic contributions VeeX(A,B) and VeeX(A,A') are calculated using the Hartree-Fock exchange functional while the intratomic contribution VeeX(A,A) is calculated as VeeX(A) - VeeX(A,A'). For wavefunctions of non-supported DFT models, VeeX is (incorrectly) assumed to be the Hartree-Fock exchange functional and the atomic energies E_IQA(A) will not be correct and will not sum to the correct molecular energy.

Vnn(A) - W(A) Note:

For Virial-Based atomic energies, Vnn(A) - W(A) is the contribution of atom A to the nuclear repulsion energy, Vnn(A), minus the contribition of atom A to the nuclear virial of the energy-gradient-based forces on the nuclei, W(A). Vnn(A) - W(A) is defined by the 3-term sum given in equation (6.85.5) of R.F.W. Bader's 1990 AIM book. For approximate wavefunctions Vnn(A,Mol) - W(A) is origin-dependent to the extent that the atomic Ehrenfest force theorem is not satisfied. However, an atom- specific origin can always be defined so that the atomic virial theorem is satisfied. Each Vnn(A,Mol) - W(A) value given here is determined using such an origin and is equivalent to the following definition:
Vnn(A) - W(A) = -2T(A) - Vee(A) - Ven(A).

For stationary point geometries, such as equilibrium and transition- state geometries, W(A) is zero and the total atomic energy equals the atomic electronic energy, E(A) = Ee(A).

3.2 AIMAll Output

The AIMAll output presents all the IQA terms presented in the section 1 of this manual, however the notation used by the program is different from the notation utilized in our papers. Table 1 shows the correspondence between the terms:

AIMAll Notation	**Our Notation**
T(A)	T^A
Vneen(A,A)/2 = Vne(A,A)	V_{en}^{AA}
Vne(A,Mol)/2	$\frac{1}{2}\sum_{B=0}^{N} V_{ne}^{AB}$
Ven(A,Mol)/2	$\frac{1}{2}\sum_{B=0}^{N} V_{en}^{AB}$
Vneen(A,Mol)/2	$\frac{1}{2}\sum_{B=0}^{N} V_{ne}^{AB} + \frac{1}{2}\sum_{B=0}^{N} V_{en}^{AB}$
Vne(A,A')/2	$\frac{1}{2}\sum_{B\neq A}^{N} V_{ne}^{AB}$
Ven(A,A')/2	$\frac{1}{2}\sum_{B\neq A}^{N} V_{en}^{AB}$
Vneen(A,A')/2	$\frac{1}{2}\sum_{B\neq A}^{N} V_{ne}^{AB} + \frac{1}{2}\sum_{B\neq A}^{N} V_{en}^{a,b}$
Vee(A,A) + Vee(A,A')/2	$V_{en}^{AA} + \frac{1}{2}\sum_{B\neq A}^{N} V_{ee}^{AB}$
VeeC(A,A) + VeeC(A,A')/2	$V_{coul}^{AA} + \frac{1}{2}\sum_{B\neq A}^{N} V_{coul}^{AB}$
VeeX(A,A) + VeeX(A,A')/2	$V_{xc}^{AA} + \frac{1}{2}\sum_{B\neq A}^{N} V_{xc}^{AB}$
Vee(A,A)	V_{ee}^{AA}
VeeC(A,A)	V_{coul}^{AA}
VeeX(A,A)	V_{xc}^{AA}
Vee(A,A')/2	$\frac{1}{2}\sum_{B\neq A}^{N} V_{ee}^{AB}$
VeeC(A,A')/2	$\frac{1}{2}\sum_{B\neq A}^{N} V_{coul}^{AB}$
VeeX(A,A')/2	$\frac{1}{2}\sum_{B\neq A}^{N} V_{xc}^{AB}$
V_IQA(A)	$\sum_{B=0}^{N} V_{cl}^{AB} + \sum_{B=0}^{N} V_{xc}^{AB}$
VC_IQA(A)	$\sum_{B=0}^{N} V_{cl}^{AB}$
VX_IQA(A)	$\sum_{B=0}^{N} V_{xc}^{AB}$
V_IQA(A,A)	$V_{coul}^{AA} + V_{xc}^{AA}$
VC_IQA(A,A)	V_{cl}^{AA}
VX_IQA(A,A)	V_{xc}^{AA}
V_IQA(A,A')/2	$\frac{1}{2}\sum_{B\neq A}^{N} V_{cl}^{AB} + \frac{1}{2}\sum_{B\neq A}^{N} V_{xc}^{AB}$
VC_IQA(A,A')/2	$\frac{1}{2}\sum_{B\neq A}^{N} V_{cl}^{AB}$
VX_IQA(A,A')/2	$\frac{1}{2}\sum_{B\neq A}^{N} V_{xc}^{AB}$
E_IQA(A)	$T^A + V_{ee}^{AA} + V_{en}^{AA} + \frac{1}{2}\sum_{B\neq A}^{N} V_{coul}^{AB} + \frac{1}{2}\sum_{B\neq A}^{N} V_{xc}^{AB}$
E_IQA_Intra(A)	$E_{Intra}^{A} = T^a + V_{ee}^{AA} + V_{en}^{AA}$
E_IQA_Inter(A)	$\frac{1}{2}\sum_{B\neq A}^{N} V_{coul}^{AB} + \frac{1}{2}\sum_{B\neq A}^{N} V_{xc}^{AB}$
Vne(A,B)/2	$\frac{1}{2}V_{ne}^{AB}$
Ven(A,B)/2	$\frac{1}{2}V_{en}^{AB}$
Vneen(A,B)/2	$\frac{1}{2}V_{ne}^{AB} + \frac{1}{2}V_{en}^{AB}$
Vee(A,B)/2	$\frac{1}{2}V_{ee}^{AB}$
Vnn(A,B)/2	$\frac{1}{2}V_{nn}^{AB}$
VeeC(A,B)/2	$\frac{1}{2}V_{coul}^{AB}$
VeeX(A,B)/2	$\frac{1}{2}V_{xc}^{AB}$
V_IQA(A,B)/2	$\frac{1}{2}V_{Inter}^{AB} = \frac{1}{2}V_{cl}^{AB} + \frac{1}{2}V_{xc}^{AB}$
VC_IQA(A,B)/2	$\frac{1}{2}V_{cl}^{AB}$

VX_IQA(A,B)/2	$\frac{1}{2}V_{xc}^{AB}$
E_IQA_Inter(A,B)/2	$\frac{1}{2}V_{Inter}^{AB} = \frac{1}{2}V_{cl}^{AB} + \frac{1}{2}V_{xc}^{AB}$
Vne(B,A)/2	$\frac{1}{2}V_{ne}^{BA}$
Ven(B,A)/2	$\frac{1}{2}V_{en}^{BA}$
Vneen(B,A)/2	$\frac{1}{2}V_{ne}^{BA} + \frac{1}{2}V_{en}^{BA}$
Vee(B,A)/2	$\frac{1}{2}V_{ee}^{BA}$
Vnn(B,A)/2	$\frac{1}{2}V_{nn}^{BA}$
VeeC(B,A)/2	$\frac{1}{2}V_{coul}^{BA}$
VeeX(B,A)/2	$\frac{1}{2}V_{xc}^{BA}$
V_IQA(B,A)/2	$\frac{1}{2}V_{Inter}^{BA} = \frac{1}{2}V_{cl}^{BA} + \frac{1}{2}V_{xc}^{BA}$
VC_IQA(B,A)/2	$\frac{1}{2}V_{cl}^{BA}$
VX_IQA(B,A)/2	$\frac{1}{2}V_{xc}^{BA}$
E_IQA_Inter(B,A)/2	$\frac{1}{2}V_{Inter}^{BA} = \frac{1}{2}V_{cl}^{BA} + \frac{1}{2}V_{xc}^{BA}$
Vne(A,B)	V_{ne}^{AB}
Ven(A,B)	V_{en}^{AB}
Vneen(A,B)	$V_{ne}^{AB} + V_{en}^{AB}$
Vee(A,B)	V_{ee}^{AB}
Vnn(A,B)	V_{nn}^{AB}
VeeC(A,B)	V_{coul}^{AB}
VeeX(A,B)	V_{xc}^{AB}
V_IQA(A,B)	$V_{cl}^{AB} + V_{xc}^{AB}$
VC_IQA(A,B)	V_{cl}^{AB}
VX_IQA(A,B)	V_{xc}^{AB}
E_IQA_Inter(A,B)	$V_{Inter}^{AB} = V_{cl}^{AB} + V_{xc}^{AB}$
Vne(B,A)	V_{ne}^{BA}
Ven(B,A)	V_{en}^{BA}
Vneen(B,A)	$V_{ne}^{BA} + V_{en}^{BA}$
Vee(B,A)	V_{ee}^{BA}
Vnn(B,A)	V_{nn}^{BA}
VeeC(B,A)	V_{coul}^{BA}
VeeX(B,A)	V_{xc}^{BA}
V_IQA(B,A)	$V_{cl}^{BA} + V_{xc}^{BA}$
VC_IQA(B,A)	V_{cl}^{BA}
VX_IQA(B,A)	V_{xc}^{BA}
E_IQA_Inter(B,A)	$V_{Inter}^{BA} = V_{cl}^{BA} + V_{xc}^{BA}$

Table 1: IQA terms from AIMAll output

4 REG.py

REG.py is a python library that contains all functions needed to extract the IQA terms from the AIMAll outputs and perform a REG analysis. REG.py is an alternative to the ANANKE program and REG.py's advantage relies on its modular characteristics, i.e. the user can organize the functions according to his/her needs. There are five modules in the REG.py library:

1. **reg.py**: contains functions to perform the REG analysis.
2. **aimall_utils.py**: contains functions to get data from AIMAll outputs.
3. **gaussian_utils.py**: contains functions to get data from .wfn/.wfx files generated by GAUSSIAN09 or GAUSSIAN16
4. **morphy_utils.py**: contains functions to get data from MORPHY/MORFI outputs (MORFI is used to split V_{xc} into $V_{correlation} + V_{exchange}$. See MORFI manual for more details.)
5. **reg_vis.py** : contains functions to generate visualisation data for REG results. (Not up-to-date!)
6. **dftd3_utils.py** : contains functions to run and get data from the dft-d3 program by S. Grimme (DFT-D3)
7. **auto_reg.py** : main script to run a REG analysis with simple user-defined input.

The functions in each module are listed in the following sections.

4.1 List of Functions - reg.py

- **Function:** regression(A, B, mode=None)
  Perform a linear regression between A and B (B = slope*A + intercept)
  Input: A, B and mode
  A = X values
  B = Y values
  mode = None (default) : for regular linear regression
  = "norm" : use normalised values of A and B
  = "std" : use standardised values for A and B
  Output: [slope, intercept, pearson]
  slope = angular coefficient of the linear regression
  intercept = linear coefficient of the linear regression
  pearson = Pearson correlation coefficient
  Error:
  "Arrays must have the same size" : A and B have different sizes.
  "Mode not recognized. Use None, 'norm' or, 'std'" : invalid value for 'mode'

- **Function:** find_critical(Y, X, min_points=5, use_inflex=False)
  Takes the X and Y values of a function and returns the critical points that match the criteria.
  Input: Y, X, min_points, use_inflex
  Y = ordinate values of a function
  C = abscissa values of a function
  min_point = 5 (default): minimum amount of points between two critical points
  use_inflex = False (default) : Not search for inflexion points (second derivative = 0)
  = True : Search for inflexion points (second derivative = 0)

Output: critical_point_list
critical_point_list = Array containing the index of critical points (zero indexed)
Error:

"Arrays must have the same size"	:	X and Y have different sizes.
"Invalid value. Use True or False"	:	invalid value for use_inflex
"Too many points between two critical points"	:	min_points must be lower than the X array length

- **Function:** split_segm(A, critical_point_list)
Takes the A array and divide it into N arrays according to the number of critical points. Each array corresponds to a segment of the REG analysis.
Input: A, critical_point_list

A	=	Array containing the abscissa values of a function
critical_points_list	=	list of the index of the critical points (zero indexed)

Output: segm
segm = Array of arrays containing the values of A for each segment.
Error:

"Invalid critical point index"	:	index of critical point out of range of array.

- **Function:** reg(wfn_energy, control_coord, terms, critical=True, np=5, inflex=True, critical_index= [ ], mode="norm")
Perform the REG analysis over all contributions inside "terms" array
Input: wfn_energy, control_coord, terms, critical, np, inflex, critical_index, mode

wfn_energy	=	Array containing the energy values for each point (PES - Energy)
control_coord	=	Array containing the control coordinate values for each point (PES - Coordinate)
terms	=	Array of arrays, each one corresponding to one IQA contribution.
critical	=	True(default): Search for critical points
	=	False : user must provide the critical point index array
np	=	5(default): : minimum amount of points between two critical points (used if critical == True)
use_inflex	=	False (default) : Not search for inflexion points (second derivative = 0) (used if critical == True)
	=	True : Search for inflexion points (second derivative = 0 (used if critical == True)
critical_index	=	[ ](default) : list of critical points provided by the user. (used if critical == False)
mode	=	None : for regular linear regression
	=	"norm"(default) : use normalised values of A and B
	=	"std" : use standardised values for A and B

Output: [REG_values, pearson_values][segment]

REG_values	=	array of REG coefficients for each contribution. Each array correspond to a segment
pearson_values	=	array of Pearson coefficients for each contribution. Each array correspond to a segment

Error:

"PES abscissa and ordenate must have same number of points"	:	wfn_energy and control_coord have different sizes
"Contributions arrays must have same size"	:	arrays inside terms have different sizes.

- **Function:** integration_error(wfn_energies, IQA_energies)

Calculates integration error for each PES point
Input: wfn_energies, IQA_energies
wfn_energies = list of total energies form wfn files
IQA_energies = list of total energy obtained from IQA terms
Output: [error, RMSE]
error = list of wfn_energies - IQA_energies
RMSE = Root mean square error
Error:
"Arrays must have the same size" : wfn_energies and IQA_energies have different sizes.

- **Function:** group_intra(intra_energies, intra_energies_header, groups)
Group IQA intra-atomic terms into the user-defined groups
Input: intra_energies, intra_energies_header, groups
intra_energies = list of IQA intra-atomic terms
intra_energies_header = header of IQA intra atomic terms.
groups = list of list of atomic labels, each internal list correspond to a different group
Output: [new_terms, new_header]
new_terms = list of the grouped IQA terms values
new_header = list of the grouped terms header

- **Function:** group_inter(inter_energies, inter_energies_header, groups)
Group IQA interatomic terms into the user-defined groups
Input: inter_energies, inter_energies_header, groups
inter_energies = list of IQA interatomic terms
inter_energies_header = header of IQA interatomic temrs.
groups = list of list of atomic labels, each internal list correspond to a different group
Output: [new_terms, new_header]
new_terms = list of the grouped IQA terms values
new_header = list of the grouped terms header

4.2 List of Functions - aimall_utils.py

- **Function:** get_atom_list(wfn_file)
Get atomic labels from wfn file
Input: wfn_file
wfn_file = Any wfn file of the desired PES
Output: [atom_list]
atom_list = list of each atom label for all atoms in molecule
Error:
"Atomic labels not found" : Atom list does not exist in wfn_file

- **Function:** get_aimall_wfn_energies(A)
Get all wfn energies from the wavefunction files (wfn)
Input: [A]
[A] = list of all wfn files of the PES
Output: [wfn_energy]
[wfn_energy] = list of energies for each PES point
Error:

"Energy values not found in file: file_name" : No energy values in file_name

- **Function:** intra_property_from_int_file(folders, prop, atom_list)
  Get IQA intratomic properties from int files
  Input: folders, prop, atom_list
  folders = path to *_atomicfiles folders
  prop = list of IQA for each atoms e.g.: "['T(A)', 'Vee(A,A)', 'Vne(A,A)']"
  atom_list = list of atomic labels e.g.: [n1, c2, h3, ...]
  Output: [intra_properties, contributions_list]
  intra_properties = array of array containing the IQA values for each atom for each geometry
  contributions_list = list of contributions organised in the same order as in intra_properties

- **Function:** inter_property_from_int_file(folders, prop, atom_list)
  Get IQA interatomic properties from int files
  Input: folders, prop, atom_list
  folders = path to *_atomicfiles folders
  prop = list of IQA for each atoms e.g.: "['T(A)', 'Vee(A,A)', 'Vne(A,A)']"
  atom_list = list of atomic labels e.g.: [n1, c2, h3, ...]
  Output: [intra_properties, contributions_list]
  intra_properties = array of array containing the IQA values for each atom for each geometry
  contributions_list = list of contributions organised in the same order as in intra_properties

4.3 List of Functions - gaussian_utils.py

- **Function:** get_atom_list_wfn_g09(wfn_file)
  Get atomic labels from g09 wfn file
  Input: wfn_file
  wfn_file = Any wfn file of the desired PES
  Output: atom_list
  atom_list = list of each atom label for all atoms in molecule
  Error:
  "Atomic labels not found" : Atom list does not exist in wfn_file

- **Function:** get_atom_list_wfx_g09(wfx_file)
  Get atomic labels from g09 wfx file
  Input: wfx_file
  wfx_file = Any double wavefunction file of the desired PES
  Output: atom_list
  atom_list = list of each atom label for all atoms in molecule
  Error:
  "Atomic labels not found" : Atom list does not exist in wfn_file

- **Function:** get_atom_list_wfn_g16(wfn_file)
  Get atomic labels from g16 wfn file
  Input: wfn_file
  wfn_file = Any wfn file of the desired PES

Output: atom_list
atom_list = list of each atom label for all atoms in molecule
Error:
"Atomic labels not found" : Atom list does not exist in wfn_file

4.4 List of Functions - morphy_utils.py

- **Function:** read_morphy_outputs(file_list)
Get correlation data from Morphy output
Input: [file_list]
[file_list] = array that contains the path for the outputs of Morphy
Output [interactions_temp, final_values]
interactions_temp = list of all interatomic correlation energies headers
final_values = list of all interatomic correlation energies values

4.5 List of Functions - reg_vis.py

- **Function:** plot_violin(df_list, save = False, file_name='reg_violin.png')
Creates a violin plot for each segment.
Input: df_list, save, file_name
df_list = list of dataframes containing REG and R for each segment.
save = False (default) : not save figure to file.
= True : save figure to file named reg_violin.png
file_name = ./reg_violin.png (default) : output file path if save = True
Output: figure
figure : violin plot

- **Function:** generate_data_vis(df, df_list, n_term, save = False, file_name='data_vis.png', title = "REG data visualization")
Creates a graphical summary of the REG analysis
Input: df, df_list, n_term, save, file_name
df = segment dataframe [containing REG and R^2 values]
df_list = list of dataframes for all segments [containing REG and R^2 values]
n_term = number of relevant terms to be highlighted
save = False (default) : not save figure to file.
= True : save figure to file named data_vis.png
file_name = ./data_vis.png :output file path if save = True
title = "REG data visualization" (default) : output file path if save = True
Output: figure
figure : data visualization summary figure

- **Function:** plot_segment(coordinate, wfn_energy, critical_points, label = False, color=True, annotate=True, title='REG segments', y_label='Energy', x_label='Coordinate', save = False, file_name='segments.png')
Plot the PS showing the segment division
Input: coordinate, wfn_energy, critical_points, label, color, annotate, title, y_label, x_label, save, file_name

coordinate	=	control coordinate array
wfn_energy	=	array containing the energies from the wfn/wfx files
critical_points	=	array containing the list of critical points
label	=	False (default) : not label each point
	=	True : label each point
color	=	True (default) : color each segment
	=	False : not color each segment
annotate	=	True(default) : label each segment
	=	False : not label each segment
save	=	False (default) : not save figure to file.
	=	True : save figure to file named ./segments.png
title	=	"REG segments" (default) : output file path if save = True
file_name	=	./segments.png :output file path if save = True
y_label	=	y-axis label
x_label	=	x-axis label

Output: figure

figure : data visualization summary figure

4.6 Data Visualisation Examples

REG.py outputs various different data visualisation plots (Summary of results section not up-to-date) that can be used for a quick check of the REG-IQA results. However, these are not meant for publication as they are automatically generated and could present artifacts given by the different nature of the analyses pursued. A thorough output in both .csv, .xlsx and .png formats (Figure 4) is directly obtained from REG.py in the 'SYSTEM_NAME_results' folder as defined in the auto_reg.py input (see Script Example below).

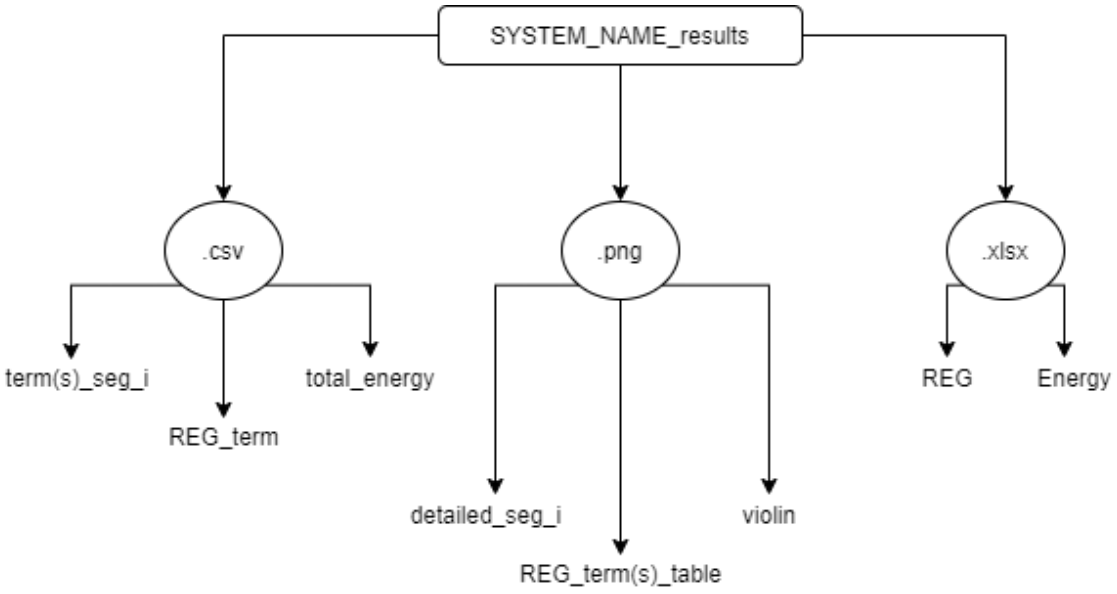
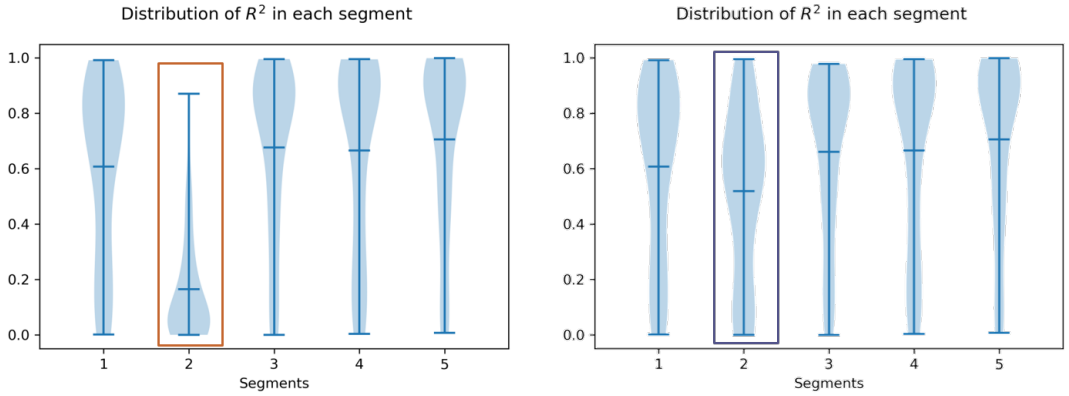


Figure 4: Flowchart of the output of REG.py for Data Manipulation and Analysis.

1. **Violin Plot:**
   The violin plots are used to compare the R^2 distribution for each segment. If the division of segments is done correctly, one can expect that the distribution of R^2 values is similar in each segment. However, a bad definition of segments will causes discrepant distributions, which can easily be seen in a violin plot. Figure 5 presents examples of violin plots, showing a bad and a good segmentation of the PES.



(a) Bad segmentation

(b) Good segmentation

Figure 5: Example of a bad segmentation (a) and a good segmentation(b). Note that for segment 2, in (a), the vast majority of the IQA presents low values of R^2.

2. **Summary of Results:**
Figure 6 contains a summary of the REG analysis for one segment. In (a): the violin plot comparing the R^2 distribution of segment 2 (green) with the R^2 distribution of all other segments. (b): histogram showing the distribution of R^2 in segment 2. (c): REG coefficient lines, showing the m_{REG} plotted against R^2. (d): The most relevant IQA terms according to the REG coefficient value.

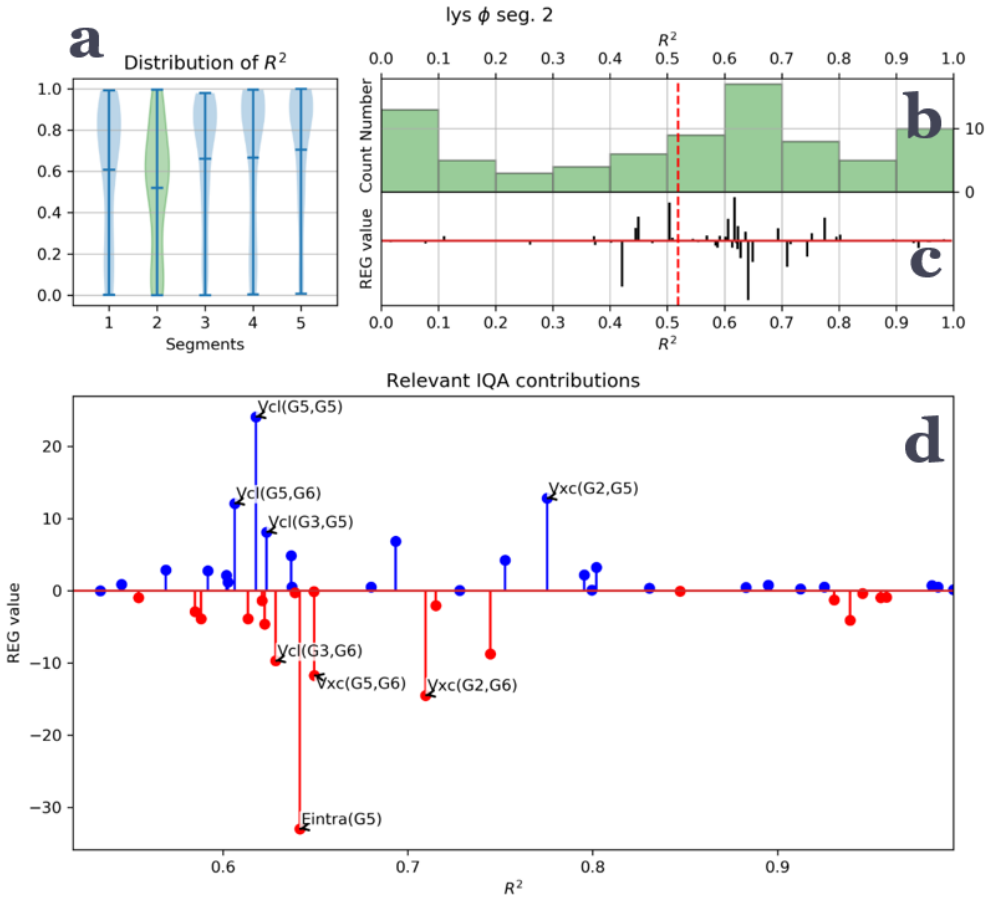


Figure 6: Data visualization summary generated by *generate_data_vis* function.

4.7 Workflow

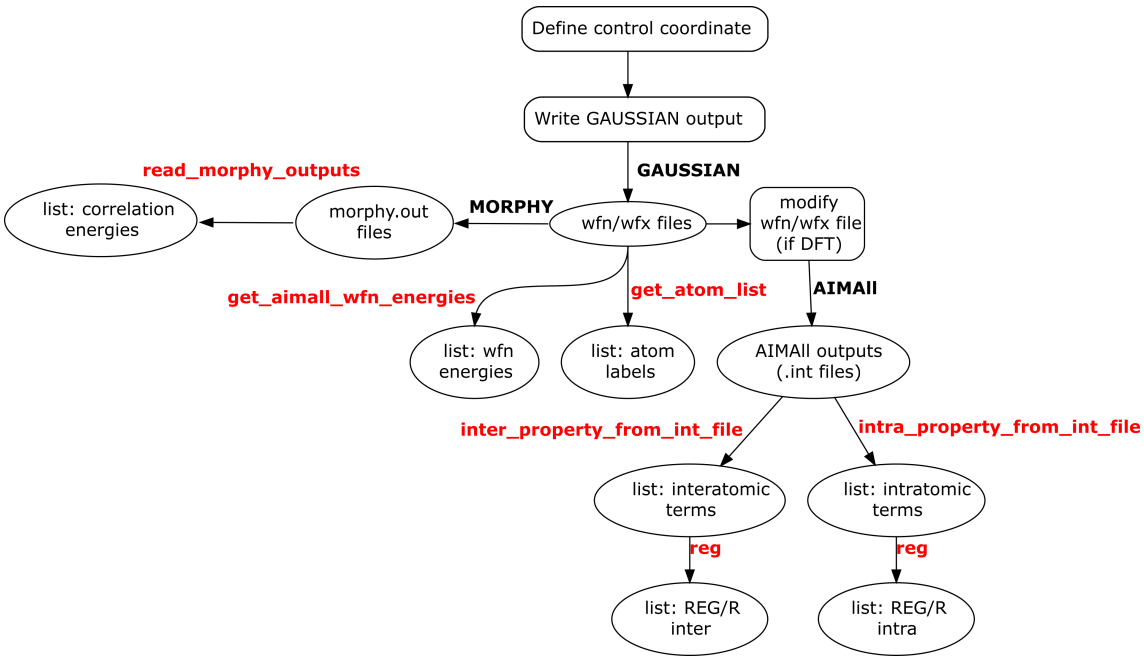


Figure 7: Basic workflow chart of how to perform a REG analysis. In red, functions available in REG.py

4.8 Script Example

In this section, an example of the *auto_reg.py* script is presented. Note that the actual script has more code that does not need to be changed by the user.

```
# IMPORT LIBRARIES
import sys
sys.path.insert(1, '/mnt/iusers01/pp01/v69787ff/REG/')  # PLEASE INSERT THE
                                        PATH OF REG.py folder installation
import reg
import aimall_utils as aim_u
import numpy as np
import pandas as pd
import reg_vis as rv
import gaussian_utils as gauss_u
import dftd3_utils as disp_u
import re
import os
import time

### STARTING TIMER ###
start_time = time.time()
##################    VARIABLES    ####################

SYS = 'SN2_ClBr_B3LYP'  # name of the system

### PES Critical points options ###
POINTS = 4  # number of points for "find_critical" function
AUTO = True  # Search for critical points
turning_points = []  # manually put critical points in the PES if necessary
# NOTE: If analysing only a single segment (i.e. the PES has no critical
                                        points) please put AUTO=False and tp
                                        =[]

# DEFINE THE DESIRED TERMS:
intra_prop = ['E_IQA_Intra(A)']  # chose the AIMAll intra atomic properties
                                        to analyse
intra_prop_names = ['Eintra']  # names of the properties shown in the
                                        output
inter_prop = ['VC_IQA(A,B)', 'VX_IQA(A,B)', 'E_IQA_Inter(A,B)']  # chose
                                        the AIMAll inter atomic properties to
                                        analyse
inter_prop_names = ['Vcl', 'Vxc', 'Einter']  # names of the properties
                                        shown in the output

REVERSE = True  # Reverse the REG points

INFLEX = False

### CONTROL COORDINATE OPTIONS ###
CONTROL_COORDINATE_TYPE = 'Scan'  # 'Scan' or 'IRC'. If empty ('') then
                                        default will be used
Scan_Atoms = [1, 6]  # list of the atoms used for PES Scan (i.e.
                                        ModRedundant option in Gaussian)
IRC_output = ''  # insert the g16 output file path if using IRC as control
                                        coordinate

CHARGE_TRANSFER_POLARISATION = True  # Split the classical electrostatic
                                        term into polarisation and monopolar
                                        charge-transfer

DISPERSION = True  # Run DFT-D3 program to consider dispersion
# NOTE: This only works if you have DFT-D3 program installed in the same
                                        machine https://www.chemie.uni-bonn.
```

```
                                        de/pctc/mulliken-center/software/dft-
                                        d3/
### DISPERSION OPTIONS ###
DFT_D3_PATH = '/mnt/iusers01/pp01/v69787ff/DFT-D3/dftd3'  # insert path of
                                        DFT-D3 program
DISP_FUNCTIONAL = 'B3-LYP'  # insert the functional used for D3 correction
BJ_DAMPING = True  # Becke-Johnson Damping

WRITE = True  # write csv files for energy values and REG analysis
SAVE_FIG = True  # save figures
ANNOTATE = True  # annotate figures
DETAILED_ANALYSIS = True
LABELS = True  # label figures
n_terms = 4  # number of terms to rank in figures and tables

#############################################################
```

5 Tutorial

The tutorial will cover all the basics to do a simple REG-IQA analysis on an SN_2 reaction. Everything will be explained as general as possible in order for the user to re-create every step of the process as pleases. For CSF3 users (i.e. any machine with the SunGrid Engine) there will be more detailed and step-by-step explanation on how to run a REG-IQA analysis from scratch and everything regarding that will be written in **orange**. Basic Bash commands are a minor requirement to work around the Linux environment so they will not be covered in this tutorial. Moreover, the input files and images shown are only for visualisation purposes (i.e. creation of the input files from scratch is recommended).

5.1 Programs and Installation

To install and run REG.py simply download the folder at github.com/FabioFalcioni and put it anywhere in the login node of the machine. It is mostly recommended to put it in the home area of the personal login node. REG.py is written in order to have little to no-dependencies. An Anaconda3 installation (anaconda installation) is more than sufficient to run the program. Installation of the latter is not necessary for CSF3 users (see next section). For electronic structure calculations the program Gaussian16 is used and interfaced with REG.py. The same applies for the IQA calculations made with AIMAll (latest version). While the latter is mandatory for a proper REG-IQA analysis, the former can be any other electronic structure software as long as it is capable of generating wavefunction files (.wfn/.wfx) from single point energy calculations. However, for the purpose of this full walk-through, Gaussian16 and GaussView6 will be employed.

5.2 Setting up the environment

CSF3 users have to load specific modules every time is needed (e.g. prior to using a software) with the following commands:

```
                                CSF3

Gaussian:  module load apps/binapps/gaussian/g16c01_em64t
Anaconda3 (i.e.  python3):  module load apps/binapps/anaconda3/2020.07
```

In CSF3, AIMAll is not installed but the whole directory can be simply copied and pasted in the home area of the login node. GaussView6 can be accessed within CSF3 by first loading the Gaussian16 module and then typing the following command:

```
qrsh -l short -V -cwd gv
```

For more information on working around the CSF3 environment see CSF3 Documentation.

5.3 Defining the Control Coordinate - An S_{N^2} reaction

The control coordinate for a REG analysis is defined by a curve connecting two points on the potential energy surface of a system. Such a line can correspond to a chemical reaction, a vibrational movement, a conformation change, a phase change, a dissociation process, etc. The definition of the control coordinate will be different for each application.

As an example, we define a control coordinate using the intrinsic reaction coordinate (IRC) of an S_{N^2} reaction using Gaussian16. To do that, we fist need to find the transition state of the reaction.

Finding transition states (TS) can be a tricky process since different systems require different approaches. In this section we describe in detail all steps used to find the TS of gas-phase S_{N^2} reactions without using the Synchronous Transit-Guided Quasi-Newton Methods (QST2, QST3). The examples given are based on the following reaction:

$$CH_3Cl + F^- \longrightarrow CH_3F + Cl^- \tag{36}$$

Input files are reproduced when necessary.

5.3.1 STEP 1 - Optimise reagents geometries

In chemical reactions, the transition state is a point in the reaction path that can only be accessed when the reagents collide at a specific angle with a determined minimum amount of energy. In this way, optimizing the reactants' geometry is a good starting point. After the optimisation step, an IR frequency calculation is performed to attest for a minimum energy structure. If all frequencies are real numbers we reached a minimum point (global or local) on the potential energy surface. Each molecule is optimised separately. The optimisation input is reproduced below:

```
%nprocshared=12
%mem=16GB
#opt b3lyp/6-311++g(d,p) nosymm

CH3Cl Optimization

0 1
 C          −0.945 825 02        0.547 442 69        −0.156 982 76
 H          −0.589 170 60       −0.461 367 32        −0.156 982 76
 H          −0.589 152 18        1.051 840 88         0.716 668 75
 H          −0.589 152 18        1.051 840 88        −1.030 634 26
 Cl         −2.705 825 02        0.547 464 37        −0.156 982 76
```

Once the optimisation is done, a frequency calculation can be run by taking the optimised geometry and run the same calculation with the "*freq*" keyword instead of "*opt*".

To run any Gaussian job on CSF3 a general job script submission is here reported:

```
#!/bin/bash --login
#$ -cwd
#$ -pe smp.pe 8

module load apps/binapps/gaussian/g16c01_em64t_detectcpu

export GAUSS_SCRDIR=/scratch/$USER/gau_temp_$JOB_ID
mkdir -p $GAUSS_SCRDIR

## Say *how many* cores to use (no need to add to input file)
export GAUSS_PDEF=$NSLOTS

## Say how much memory to use (4GB per core)
export GAUSS_MDEF=$((NSLOTS*4))GB

$g16root/g16/g16 < INPUT.gjf > OUTPUT.out

rm -r $GAUSS_SCRDIR
```

Note that the **% nprocshared** and **% mem** keywords MUST BE REMOVED from any input file. Note that the number of cores is specified in the **smp.pe** line. Any submission file can be submitted to the CSF3 by typing:

```
qsub submission_file.sh
```

5.3.2 STEP 2 - Scan through the Potential Energy Surface (PES)

Once we found the reagents optimized geometries, we can begin to explore the PES. To do that, we need to use the **Scan** method of GAUSSIAN. To access this method we add **#opt=ModRedundant** to the input file and specify what coordinate we want to scan over.

In this step we bring together both reactants geometries, building an initial guess for the TS geometry. Then, we move the nucleophile away from the electrophile (or *vice versa*. At each point of the scan coordinate we get an energy value. A maximum point within this coordinate will give an approximation of the TS geometry. We will use this approximation in **Step 3**. The GAUSSIAN scan input is reproduced below:

```
%nprocshared=12
%mem=16GB
#opt=(Modredundant) b3lyp/6-311++g(d,p) nosymm

CH3Cl + F- scan

-1 1
 C          −0.926 723 00        0.547 454 00        −0.156 981 00
 H          −0.588 083 00       −0.483 218 00        −0.156 988 00
 H          −0.588 081 00        1.062 762 00         0.735 622 00
 H          −0.588 089 00        1.062 752 00        −1.049 581 00
 Cl         −2.728 149 00        0.547 472 00        −0.156 985 00
 F           1.179 017 79        0.554 933 03        −0.173 393 80

B 1 6 S 20 0.05
A 6 1 5 F
```

The last two lines of the input file correspond to the **Modredundant** parameters. The first parameter states that the coordinate along the **B**ond between atoms **1** and **6** will be **S**canned in **20** steps with a displacement of **0.05**Å each. The second parameter states that the **A**ngle defined by atoms **6**, **1** and **5** are **F**reeze, that is, it may not be altered during the scanning process. If the nucleophile has more than one atom, more constraints will be necessary to perform the scanning.

Also, it is important to be aware of the correct basis set. Figure 8 compares the results of a scan calculation using the same geometry and constraints, changing the basis-set only. One can see that using **aug-cc-pVTZ** leads to a maximum point on the PES while the same does not occurs when using **6-31G** basis set.

5.3.3 STEP 3 - Transition State optimisation

The necessary information to make the first guess for finding the TS is obtained from the procedure described in **STEP 2**. Figure 9A shows the energy against the scan coordinate, note that there is a maximum point, which will be the initial guess geometry for the TS. Figure 9B shows the gradient of the former curve and at the maximum point the gradient is equal to zero.

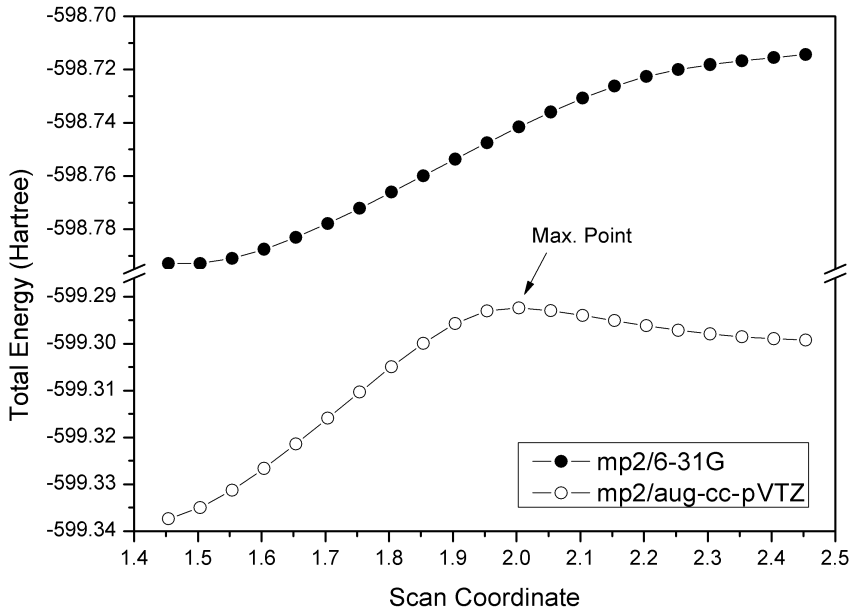


Figure 8: Total Energy versus the scan coordinate for the approximation of the fluoride ion and CH_3Cl. The black dots were obtained using the 6-31G basis set. The white dots were obtained using the aug-cc-pVTZ basis set, which leads to a maximum point.

The necessary information to make the first guess for finding the TS is obtained from the procedure described in **STEP 2**. Figure 9A shows the energy against the scan coordinate, note that there is a maximum point, which will be the initial guess geometry for the TS. Once the best guess for the TS is found, a final optimisation without constraints is made. This process should result in the correct TS geometry. The GAUSSIAN input for this step is:

```
%nprocshared=12
%mem=16GB
#opt=(TS,calcall,noeigentest) b3lyp/aug-cc-pvtz nosymm

Transition state optimization

-1 1
 C          −0.808 453 00          0.560 437 00          −0.159 765 00
 H          −0.620 700 00          −0.493 636 00         −0.164 908 00
 H          −0.623 779 00          1.090 105 00           0.753 818 00
 H          −0.649 429 00          1.090 639 00          −1.076 756 00
 Cl         −2.885 480 00          0.524 128 00          −0.148 843 00
 F           1.347 734 00          0.520 483 00          −0.261 853 00
```

The **TS** keyword means that the Berny algorithm will be used to perform the optimisation, searching for a maximum point instead of searching for a local minimum. **Calcall** means that the Hessian matrix will be recalculated in each optimization step. **Noeigentest** prevents that the occurrence of more than one negative eigenvalue causes an error. Finally, it is necessary to check for the infrared frequencies. Since the TS represents a saddle point on the PES, i.e a maximum in one direction and a minimum in all others, it should have one, *and only one*, imaginary frequency. If more than one

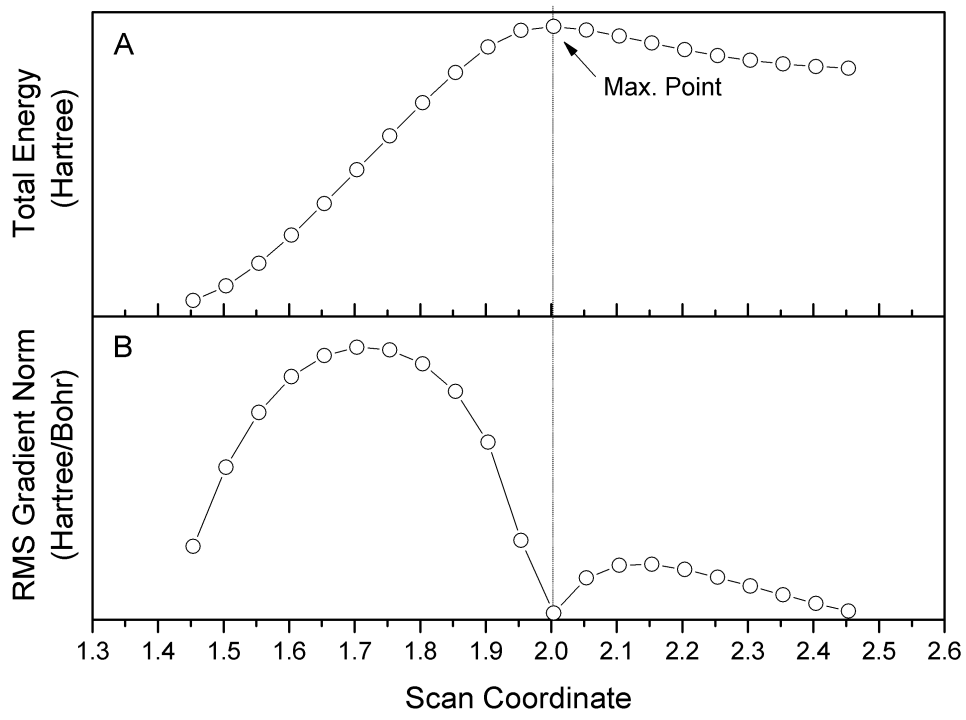


Figure 9: In A: The maximum energy point, where one can find a good approximation for the transition state geometry. In B: The first derivative norm of A.

imaginary frequency is presented, it is necessary to repeat **Step 3** with constraints. Figure 10 summarizes the results.

5.3.4 STEP 5 - Finding the IRC

Using the TS geometry as input, we request GAUSSIAN to find the intrinsic reaction coordinate in both **forward** and **reverse** directions.

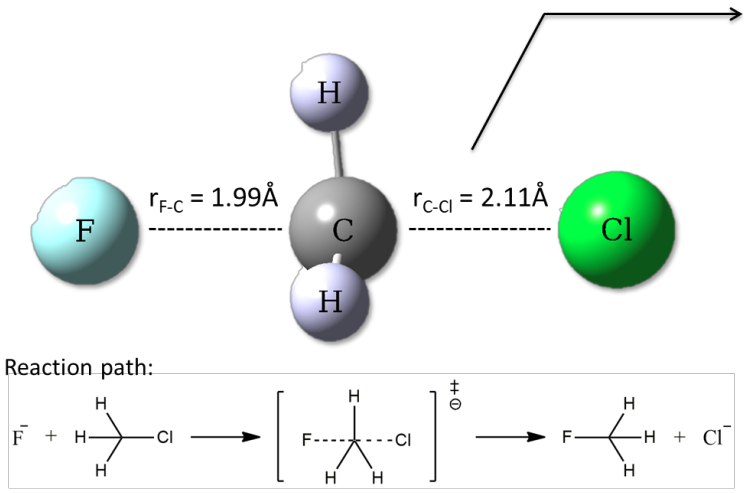
```
%nprocshared=12
%mem=16GB
#irc=(reverse,maxpoints=15,calcall) b3lyp/aug-cc-pvtz

IRC

-1 1
 C          −0.803 883 00        0.548 897 00        −0.174 019 00
 H          −0.630 563 00       −0.507 727 00        −0.179 382 00
 H          −0.611 220 00        1.076 647 00         0.737 613 00
 H          −0.655 350 00        1.078 732 00        −1.092 654 00
 Cl         −2.887 768 00        0.545 167 00        −0.123 757 00
 F           1.348 677 00        0.550 441 00        −0.226 108 00
```

If the **maxpoints** is set in the input than the output will most likely contain those number of geometries in the forward and reverse direction. Note that this number can change depending on the system analysed and a minimum on either side of the TS might

not be reached from the IRC algorithm of Gaussian. Moreover, a full IRC is possible if no specific keyword is given. In this tutorial a full default IRC is used and a total of 27 geometries are found. These define the control coordinate for the REG analysis. For each one of these geometries, it is necessary to request GAUSSIAN to perform a single point calculation writing the wavefunction (.wfn of .wfx) file. AIMALL will take the wavefunction files as input to obtain the IQA properties that will be gathered by REG.py.



Frequency (cm^{-1})	Intensity ($km.mol^{-1}$)
500.52i	1378.03
258.53	20.45
303.25	2.12
1025.03	0.29
1428.04	3.41
...	...
3213.13	0.19
3409.12	0.02

Figure 10: Transition state obtained after the proposed steps. Note that only one imaginary frequency exists, which means that we have found a saddle point in the PES.

5.4 From Gaussian to AIMAll

As previously mentioned we now need to generate a Gaussian input file for each step of the IRC (or PES scan) under study. In order to do this a simple feature of GaussView6 can be employed. First, the output file of the IRC (or PES scan) must be opened within GaussView6 under ***File*** > ***Open...*** and the ***Read Intermediate Geometries (Optimizations)*** must be ticked. Then the following steps must be taken:

- Under the ***Calculate*** > ***Gaussian Calculation Setup*** menu, a representative input file of the single point energy calculations must be generated as shown in Figure 11.
- Under the ***Preview*** tab, make sure the input is correct and then press ***Assign to Molecule Group*** (Step 1) and ***Retain*** (Step 2) as in Figure 12.
- Under ***File*** > ***Save*** > ***Advanced...*** select ***Save Molecule group : Yes, separate file for each molecule***. Then, highlight all the files and set them to have a single common name (e.g. SP_WFN.gjf) as shown in Figure 13. Finally, press ***Actions*** > ***All Items*** > ***Include molecule numbers in file names*** and then ***Save***. This will generate a single point energy file for each step of the IRC (or PES Scan).
- All the obtained single point energy files must be moved into an arbitrary folder (i.e. the folder where the REG analysis will be). Inside that folder, each file must be put into a numbered folder (e.g. SP_WFN_1.gjf inside folder **1** and so on). For any Linux user this can be done through the following bash script example:

```
#!/bin/bash
# Insert the number of steps instead of NSTEPS
```

```
for (( i=1; i<=NSTEPS); i++)); do
    cd /path/to/working/folder
    mkdir $i
    mv SP_WFN_$i.gjf $i
    cd $i
    mv SP_WFN_$i.gjf SP_WFN.gjf
    cd ..
done
```

- The last step simply involves running a single point energy calculation to obtain the wavefunction files and then feed these into AIMAll to obtain IQA partitioned energies. For CSF3 (or SGE) users this can be done through an automated script (shown below) which runs multiple similar jobs in parallel. Note that AIMAll has a command line on its own and details can be found in the respective AIMAll Documentation.

Overall, the working environment for a REG-IQA analysis should be equal to the one presented in Figure 14.

```
#!/bin/bash --login
#$ -S /bin/bash
#$ -cwd
#$ -pe smp.pe 2
#$ -t 1-NSTEPS

BASE=$HOME/scratch/

module load apps/binapps/gaussian/g16c01_em64t_detectcpu

METHOD=B3LYP
DIRS1=( reg_tutorial )
DIRS2=( REG )
DIRS3=(`echo 'for (i=1; i<=NSTEPS; i++) i' | bc` )

NUMDIRS1=${#DIRS1[@]}
NUMDIRS2=${#DIRS2[@]}
NUMDIRS3=${#DIRS3[@]}
TOTAL=$[$NUMDIRS1 * $NUMDIRS2*$NUMDIRS3]
echo "Total runs:  $TOTAL"

TID=$[SGE_TASK_ID-1]

IDX1=$[TID/$[NUMDIRS2*NUMDIRS3]]
IDX2=$[(TID/$NUMDIRS3)%$NUMDIRS2]
IDX3=$[TID%$NUMDIRS3]

JOBDIR=$DIRS1[$IDX1]/$DIRS2[$IDX2]/$DIRS3[$IDX3]

echo "Running SGE_TASK_ID $SGE_TASK_ID in directory $BASE/$JOBDIR"

#RUNNING JOB

cd $BASE/$JOBDIR
## Set up scratch dir
export GAUSS_SCRDIR=/scratch/$USER/gau_temp_$JOB_ID.$SGE_TASK_ID
mkdir -p $GAUSS_SCRDIR

## Say *how many* cores to use (no need to add to input file)
export GAUSS_PDEF=$NSLOTS

## Say how much memory to use (e.g.  3GB per core)
export GAUSS_MDEF=$((NSLOTS*3))GB

$g16root/g16/g16 < SP_WFN.gjf > SP_WFN.out

rm -r $GAUSS_SCRDIR

sed -i "s/NUCLEI/NUCLEI $METHOD/g" frame.wfn

/mnt/iusers01/pp01/v69787ff/AIMAll/aimqb.ish -nogui -iasmesh=superfine
-encomp=4 -nproc=$NSLOTS frame.wfn
```

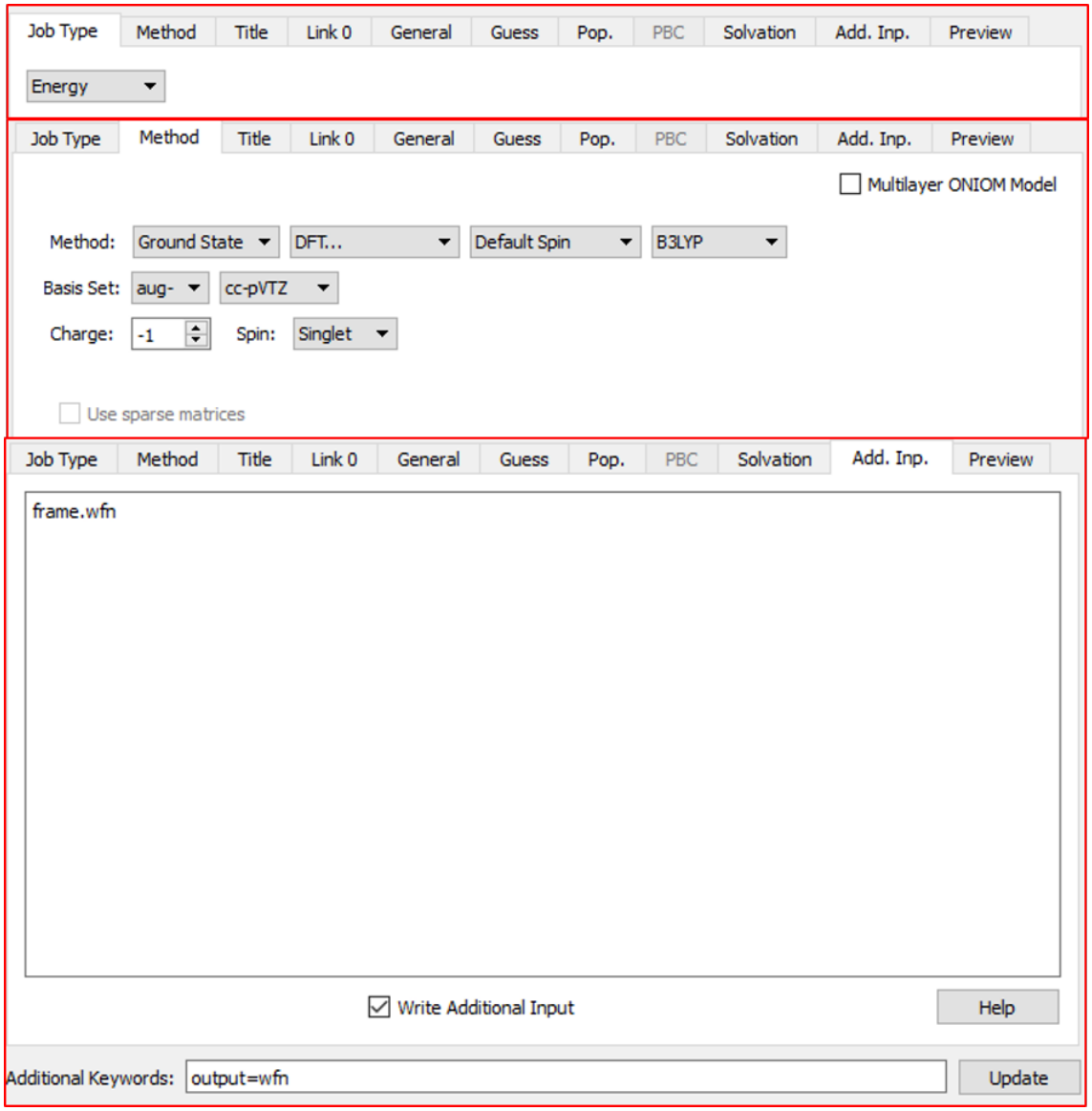


Figure 11: GaussView6 Single Point Energy calculation setup.

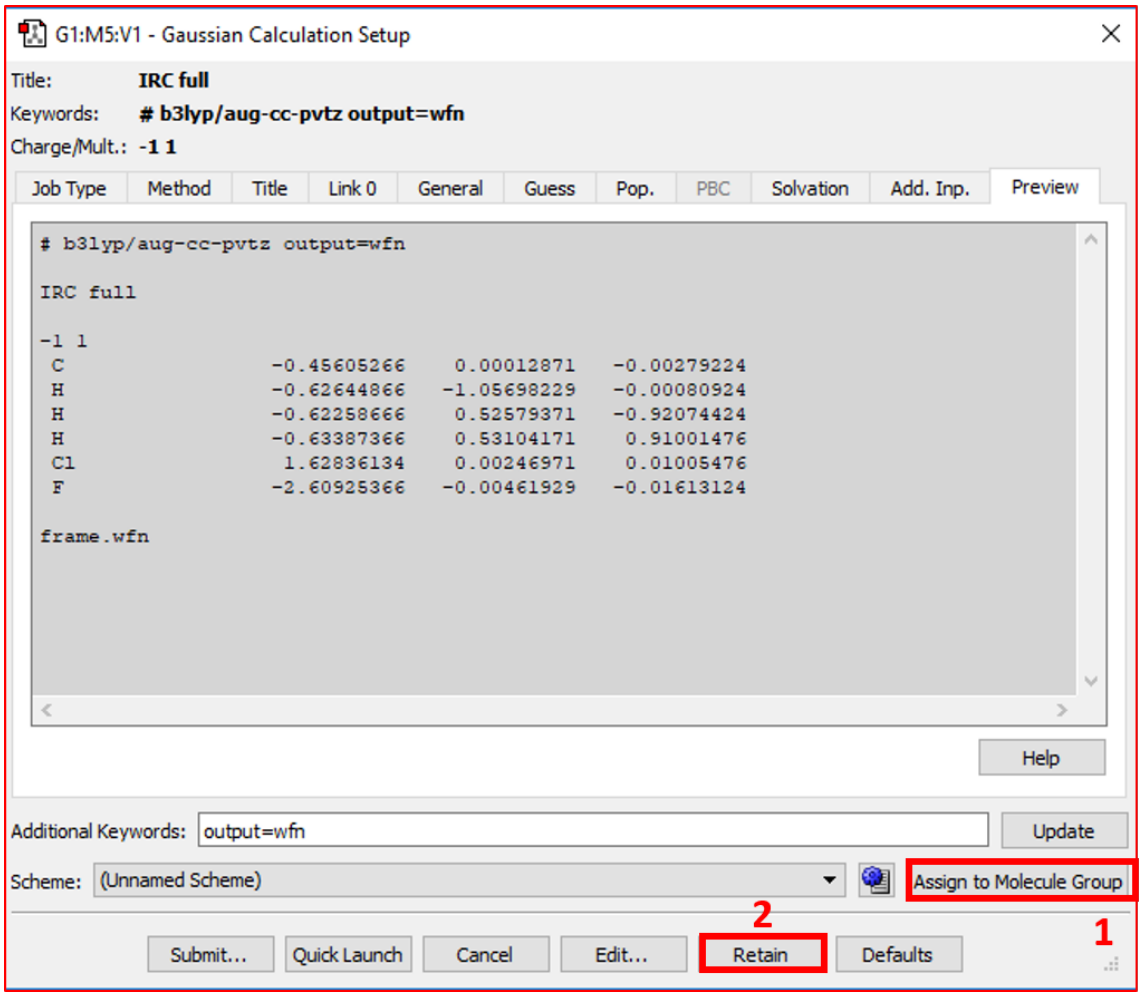


Figure 12: GaussView6 Single Point Energy calculation preview.

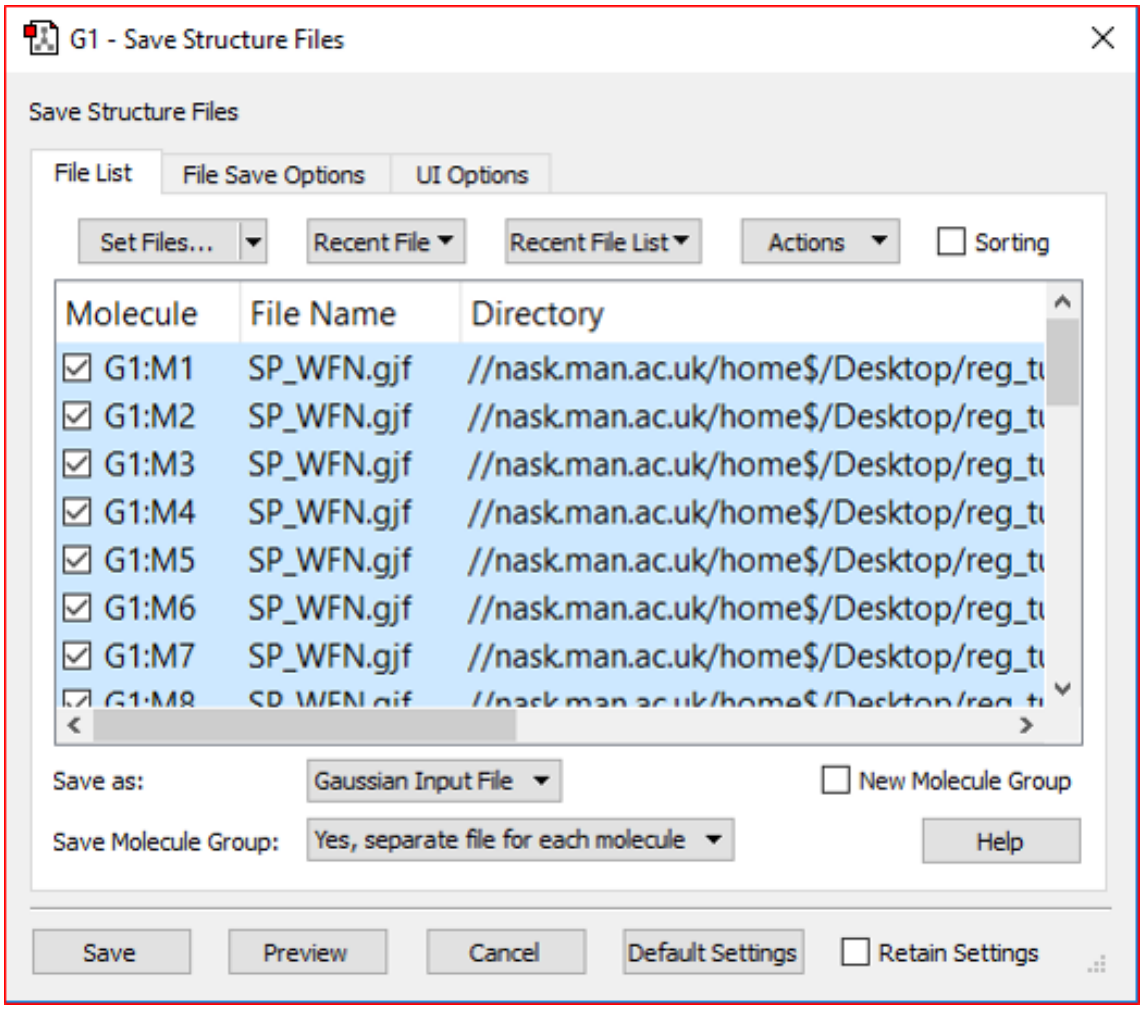


Figure 13: GaussView6 Advanced Save File options.

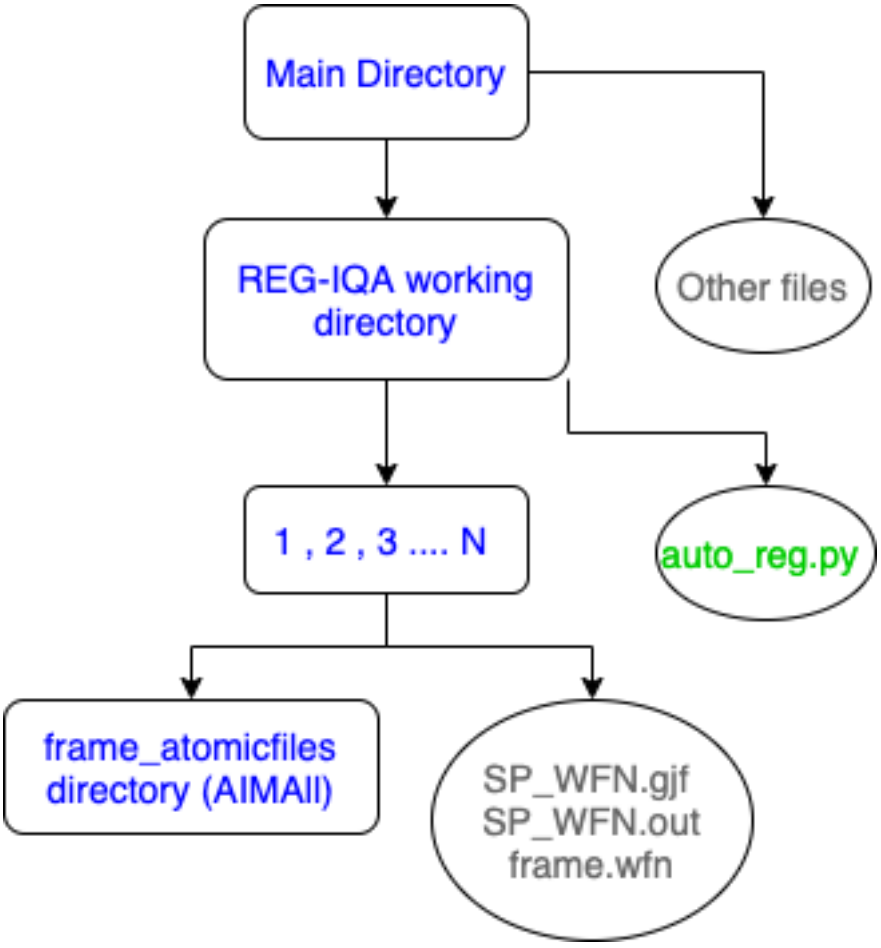


Figure 14: Flowchart of REG-IQA directory tree. Folders in blue, files in gray and scripts in green.

5.5 REG Analysis

The last step for the full REG-IQA analysis involves a simple copy and paste of the main **auto_reg.py** script into the *REG-IQA working folder* and tweaking of the parameters as shown in section 4.8. All the results will be put in a "_results" folder as described in the flowchart of Figure 4.

6 Appendix A

6.1 Demonstrating $C = 0$

Consider the plot in figure 15 where the ordinate axis corresponds to a IQA term and the abscissa axis corresponds to the total energy of the system. The dashed line corresponds to the regression line obtained by equation 28.

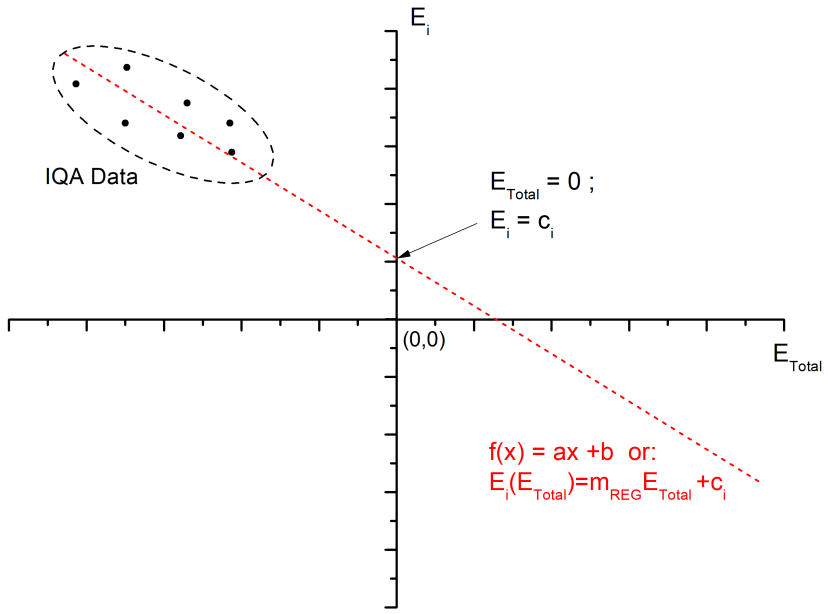


Figure 15: Example of a linear regression obtained by equation 28

Although the total energy of the system is never equal to zero, the regression line is a mathematical function and its domain corresponds to all real numbers, i.e. $(-\infty, +\infty)$. The regression line is given by:

$$E_i(E_{total}) = m_{REG,i}E_{total} + c_i$$

extrapolating the curve to the point where $E_{total} = 0$:

$$E_i(E_{total} = 0) = m_{REG,i}0 + c_i$$

$$E_i(0) = c_i$$

consider now the sum of all regression lines, one for each IQA term, at the point where $E_{total} = 0$

$$\sum E_i(0) = \sum c_i$$

since $\sum E_i = E_{total}$, and $E_{total} = 0$ then:

$$E_{total} = \sum c_i$$

$$\sum c_i = 0 \implies C = 0; Q.E.D.$$

Since C is constant, $C = 0$ is true for the entire domain $(-\infty, +\infty)$.