TOWARDS COMPUTING THE PROPERTIES OF LARGE MOLECULES FROM ATOMS-IN-MOLECULES (AIM) DENSITIES

by

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Doctor of Philosophy

Department of Chemistry

Memorial University of Newfoundland

Aug 2018

St. John's Newfoundland

Abstract

This thesis is divided into two main parts. In the first part, we proposed new three partition weights, namely; Fermi-Dirac, triangle, and Awad weights. In this part, we aim to develop new weights in which the core region close to a nucleus of an atom is assigned to that atom and not to the core or the bond of other atoms. We visually illustrate the molecular radial electron density (RDEN) and bond electron density (BDEN) using the proposed weights. We compared the molecular properties including the total number of electrons N_e, the electron-nuclear potential energy V_{ne}, and Coulomb potential energy V_{ee} which are calculated numerically using the Awad weight with those calculated by Hartree-Fock (HF) wavefunction (the exact values). Also, the computed results using the Awad weight were compared to the computed results obtained using Becke weight. Our findings show that the Awad weight gives better bonding-region representations for both RDEN and BDEN than those obtained using Becke weight. Also, the Awad weight gives better results in computing the molecular properties than those using the Becke weight.

In the second part, we proposed a new approach for calculating the total energy of molecules called atoms in molecules density (AIMD). In this part, we present the first version of AIMD. In AIMD, the molecular properties of the target molecules were computed in three steps: (i) generating small molecules (fragments) from molecules of interest, (ii) storing some properties of these fragments in a database, (iii) using the data stored in the database and by using the direct method to compute the molecular properties of the target molecule.

The electronic molecular energy (E^{ele}) is obtained using AIMD as follows,

$$E^{ele} = T^{sum} + V_{ne}[\rho] + J[\rho] - K_{HF}^{sum} \tag{1}$$

where, T^{sum} is the molecular kinetic energy density calculated by summing over the atomic kinetic energy densities, K_{HF}^{sum} is the molecular Hartree-Fock (HF) exchange energy calculated by summing over the atomic HF exchange energy densities, $V_{ne}[\rho]$ is the molecular potential energy calculated numerically using its atomic electron densities, and $J[\rho]$ is the molecular Coulomb energy calculated numerically using its atomic electron densities. The mean absolute percentage errors (MAPEs) are relatively small. The MAPE of $E_{database}$ are about 0.165%, 0.017%, and 0.008% for $\mathbf{F^1}$, $\mathbf{F^2}$, $\mathbf{F^3}$, respectively, and the MAPE of E_{direct} are about 0.164%, 0.022%, and 0.006% for $\mathbf{F^1}$, $\mathbf{F^2}$, $\mathbf{F^3}$, respectively.

Acknowledgements

I would like to appreciate my supervisor, Prof. Raymond A. Poirier, for the patient guidance, encouragement, and advice he has provided throughout my research work. I have been remarkably lucky to have a supervisor like him who cared so much about my work, and who responded to my questions and queries so promptly. He gave me unconditional support and valuable and timely advice and suggestions for the completion of my Ph.D. degree.

I would also like to thank my supervisory committee members: Dr. Christopher Flinn and Dr. Peter Warburton for the valuable advice given for improving my research, positive appreciation, counsel and for proofreading the thesis. Also, I would like to thank the Department of Chemistry for their use of their facilities and resources. I would like to thank the members of the computational and theoretical group in the Chemistry Department in Memorial University of Newfoundland; Ahmad Alrawashdeh, Matyas Csongor, Fozia Wasim, Oliver Stueker, Fatima Shaheen, Andrew Walsh, and Abd Al-Aziz Abu-Saleh. Thank you, Ahmad and Csongor for many motivational and support talks and late night pizzas. I am especially grateful to Ahmad for his effort in proofreading all this thesis.

I want to extend a special thanks to my family. To my dear wife Ghadeer Abu-Alsoud who always support, motivate, and encourage me. Thank you for being my inspiration. But most of all, thank you for being my best friend. I owe you everything. To my beloved parents Ezzeddin and Ibtisam for showing faith in me and giving me the support and for their valuable prayers. Without you, I would not be the person I am today. To my sister Reema and my brothers Mohammed, Fadi, and Shadi. To my son Abdelrahman and my

daughter Roa'a for abiding my ignorance and the patience, they showed during my study.

— Nov. 08, 2018

Contents

List of Tables

List of Figures

Abbreviations

2D two dimensional. 104, 105, 108, 110

3D three dimensional. 12–14, 20, 27, 65, 104, 105, 108–110, 112, 164

AIM atoms in molecules. 11–13, 19, 207, 208

AIMD atoms in molecules density. ii, iii, ix, xi, 163, 165, 206–208, 236, 277, 341

ALMO-EDA the absolutely localized molecular orbital energy decomposition analysis. 11

BDEN bond electron density. ii, xviii, xix, xxv–xxvii, 67, 72–74, 79–81, 83–85, 87, 88, 91–93, 95, 205, 215–220, 228–235

DFT density functional theory. 4, 20, 96

EDA energy decomposition analysis. 10

GGA Generalized Gradient Approximation. 9, 10

HF Hartree-Fock. ii, iii, xiv, 8, 10, 11, 96, 97, 146, 147, 165, 181, 205

InChI International Chemical Identifier. 128

IUPAC International Union of Pure and Applied Chemistry. 128

KM Kitaura-Morokuma. 10

KS Kohn-Sham. 6–8

LDA Local Density Approximation. 8–10

MAE mean absolute error. xii, 96–98, 100, 101, 205

MAPE mean absolute percentage error. iii, 165, 174, 177, 179, 185, 186, 192, 194, 198, 200, 201, 206

mGGA meta-GGA. 9

NNA non-nuclear attractors. 14

QTAIM quantum theory of atoms in molecules. 14

RDEN molecular radial electron density. ii, xii, xvii–xix, xxiii–xxvi, 66, 67, 72–79, 81, 82, 84–86, 88–90, 93, 94, 186, 188, 189, 205, 209–214, 220–227

RHF restricted Hartree-Fock. 10, 145, 146

SDF Structure-Data File. 128

SIE self-interaction error. 6

SLN SYBYL Line Notation. 128

SMILES Simplified Molecular-Input Line-Entry System. 127

SVD singular value decomposition. 112, 113

Chapter 1

Introduction

"Small shifts in your thinking, and small changes in your energy, can lead to massive alterations of your end result."

— Kevin Michel

1.1 Wavefunction Electronic Structure Theory

Chemistry is the science of matter that deals with construction, transformation and properties of molecules. Theoretical chemistry is the subfield where mathematical methods are combined with fundamental laws of physics to explain the structures and dynamics of chemical systems and to correlate, understand, and predict their thermodynamic and kinetic properties [1].

Due to the rapid development of theoretical techniques carrying out electronic structure calculations for many complicated systems is becoming straightforward. High level wavefunction based methods together with a large basis set have proved to be capable of achieving chemical accuracy for small molecules [2].

In wavefunction electronic structure theory, molecular structures are calculated using only the electronic Schrödinger equation (Equation 1.1),

$$\hat{H}_{elec}\Psi^{\mathbf{R}}_{elec}(\{\mathbf{r}_i\}) = E_{elec}(\mathbf{R})\Psi^{\mathbf{R}}_{elec}(\{\mathbf{r}_i\}) \tag{1.1}$$

where, $E_{elec}(\mathbf{R})$ represents the electronic energy of the molecular system, $\hat{H}_{elec}$ is the electronic Hamiltonian, which is given in atomic units as

$$\hat{H}_{elec} = -\frac{1}{2}\sum_{i=1}^{N}\nabla_i^2 - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{r_{iA}} + \sum_{i=1}^{N}\sum_{j<1}\frac{1}{r_{ij}} \tag{1.2}$$

where N is the total number of electrons, M is the total number of nuclei, Z_A is the atomic number of nucleus A, $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ is the Laplacian operator, and r_{ij} is the distance between electrons i and j.

The Born-Oppenheimer approximation [3] is applied to the non-relativistic time-independent

Schrödinger equation (Equation 1.3) to derive the electronic Schrödinger equation (Equation 1.1).

$$\hat{H}\Psi\left(\{\mathbf{r}_i\};\{\mathbf{R_A}\}\right) = E\Psi\left(\{\mathbf{r}_i\};\{\mathbf{R_A}\}\right) \tag{1.3}$$

where, $\Psi\left(\{\mathbf{r}_i\};\{\mathbf{R_A}\}\right)$ is the molecular wavefunction which is a function of electronic ($\mathbf{r}_i$) and nuclear ($\mathbf{R}_A$) coordinates, E is the total non-relativistic energy of the system; and $\hat{H}$ is the total non-relativistic Hamiltonian of the molecular system that can be written as

$$\begin{aligned}\hat{H} = &-\frac{1}{2}\sum_{i=1}^{N}\nabla_i^2 - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{r_{iA}} + \sum_{i=1}^{N}\sum_{j<1}\frac{1}{r_{ij}} \\ &-\frac{1}{2}\sum_{A=1}^{M}\frac{1}{M_A}\nabla_A^2 + \sum_{A=1}^{M}\sum_{B>A}^{M}\frac{Z_A Z_B}{R_{AB}}\end{aligned} \tag{1.4}$$

where M_A is the ratio of the mass of nucleus A to the mass of the electron and R_{AB} is the distance between nuclei A and B.

The Born-Oppenheimer approximation allows us to separate the wavefunction of a molecule into the electronic and nuclear (vibrational, rotational) components. This approximation is based on the fact that nuclei are much heavier than electrons or the electrons move much faster than the nuclei, therefore, the motion of electrons and nuclei in atomic and molecular systems can be separated. In the Born-Oppenheimer approximation, the molecular wavefunction can be express as product of the electronic and nuclear wavefunctions.

$$\Psi(\mathbf{r},\mathbf{R}) = \Psi_{elec}^{\mathbf{R}}(\mathbf{r})\Psi_{nucl}(\mathbf{R}) \tag{1.5}$$

Where, $\Psi_{elec}^{\mathbf{R}}(\mathbf{r})$ is the electronic wavefunction with fixed nuclei that depends on electronic coordinates and parametrically on nuclear coordinates, and $\Psi_{nucl}(\mathbf{R})$ is the nuclear wave-

function which depends only on nuclear coordinates.

In molecular orbital theory, each molecule contains a set of orbitals called molecular orbitals, ψ_a. Each molecular orbital can be described as a linear combination of a finite set of basis functions, φ_μ.

$$\psi_a = \sum_{\mu=1}^{K} C_{\mu a}\varphi_\mu \tag{1.6}$$

where ψ_a is a molecular orbital represented as the sum of K basis functions φ_μ, each multiplied by a corresponding expansion coefficient $C_{\mu a}$. The number of molecular orbitals is equal to the number of basis functions included in the linear expansion.

1.2 Density Functional Theory

Density functional theory (DFT) is an alternative method to investigate the molecular electronic structure of N-electron systems. Modern DFT was rooted as a promising method to study many-electron systems. Many density functional (i.e. functions of another function) approximations has become more mature for practical applications. DFT applications have increased enormously in many areas of chemistry due to the excellent performance of the functionals. Moreover, DFT's parameters are so successful because they are not system dependent. This approach is computationally very different from the direct solution of the Schrödinger equation in which the time is spent in a search over the whole of Hilbert-space to find the wavefunction. Conversely, in DFT the search is to find the three dimensional electron density [4].

In DFT, the properties of N-electron systems can be determined using functionals of electron density $\rho(\mathbf{r})$ [5]. The theory was initially developed by Hohenburg, Kohn, and Sham in

the early 1960s [6, 7]. The Hohenburg and Kohn theorem states that the exact ground-state energy E_0 is a functional of the one particle density $\rho_0(\mathbf{r})$.

$$E_0 = E_0[\rho_0] \tag{1.7}$$

However, the theorem does not tell how to construct this functional exactly. For this reason much effort has been devoted to the task of obtaining approximate functionals for the description of the ground-state properties of many particle systems [8]. The energy functional can be written as:

$$E[\rho(\mathbf{r})] = V_{ex}[\rho(\mathbf{r})] + V_{ee}[\rho(\mathbf{r})] + T[\rho(\mathbf{r})] \tag{1.8}$$

where, $V_{ex}[\rho(\mathbf{r})]$ is the external potential functional, $V_{ee}[\rho(\mathbf{r})]$ is the electron-electron potential functional, and $T[\rho(\mathbf{r})]$ is the kinetic energy functional. For molecules in the absence of any external effects, the external potential $V_{ex}[\rho(\mathbf{r})]$ is

$$V_{ex}[\rho(\mathbf{r})] = V_{ne}[\rho(\mathbf{r})] \tag{1.9}$$

where $V_{ne}[\rho(\mathbf{r})]$ is the Coulomb attraction between electrons and nuclei.

$$V_{ne}[\rho(\mathbf{r})] = -\sum_{A=1}^{M} \int \frac{Z_A}{|\mathbf{r}-\mathbf{r}_A|}\rho(\mathbf{r})d\mathbf{r} \tag{1.10}$$

The $V_{ee}[\rho(\mathbf{r})]$ is given by,

$$\begin{aligned} V_{ee}[\rho(\mathbf{r})] &= J_{ee}[\rho(\mathbf{r})] + \Delta V_{ee}[\rho(\mathbf{r})] \\ &= \frac{1}{2}\int\int \frac{\rho(\mathbf{r}_1)\rho(\mathbf{r}_2)}{|\mathbf{r}_1-\mathbf{r}_2|} d\mathbf{r}_1 d\mathbf{r}_2 + \Delta V_{ee}[\rho(\mathbf{r})] \end{aligned} \tag{1.11}$$

where the first part ($J_{ee}[\rho(\mathbf{r})]$) in the above equation is the classical Coulomb repulsion, and the second part ($\Delta V_{ee}[\rho(\mathbf{r})]$) is the non-classical corrections to the electron-electron interaction energy, such as the exchange energy (K), the correlation energy (E_{corr}), and the self-interaction error (SIE). In 1965 [7], Kohn and Sham proposed a way to compute the kinetic energy of the system $T[\rho(\mathbf{r})]$. The concept of their approach is to separate $T[\rho(\mathbf{r})]$ into the kinetic energy of the non-interacting electrons $T_{ni}[\rho(\mathbf{r})]$ and the unknown part $\Delta T[\rho(\mathbf{r})]$,

$$T[\rho(\mathbf{r})] = T_{ni}[\rho(\mathbf{r})] + \Delta T[\rho(\mathbf{r})] \tag{1.12}$$

The unknown part $\Delta T[\rho(\mathbf{r})]$ is the correction to the kinetic energy in the real system due to the interaction between the electrons. The $T_{ni}[\rho(\mathbf{r})]$ term can be computed exactly as follows,

$$T_{ni}[\rho(\mathbf{r})] = -\frac{1}{2}\sum_{i=1}^{N}\langle\psi_i^{KS}(\mathbf{r})|\nabla_1^2|\psi_i^{KS}(\mathbf{r})\rangle \tag{1.13}$$

where, ψ_i^{KS} are called Kohn-Sham (KS) orbitals. The sum of the square of KS orbitals gives the electron density of the system, $\rho(\mathbf{r})$.

$$\rho(\mathbf{r}) = \sum_{i=1}^{N}|\psi_i^{KS}(\mathbf{r})|^2 \tag{1.14}$$

By substituting Equation 1.13 into Equation 1.12, $T[\rho(\mathbf{r})]$ can be written as,

$$T[\rho(\mathbf{r})] = -\frac{1}{2}\sum_{i=1}^{N}\langle\psi_i^{KS}(\mathbf{r})|\nabla_1^2|\psi_i^{KS}(\mathbf{r})\rangle + \Delta T[\rho(\mathbf{r})] \tag{1.15}$$

and by substituting Equations 1.10, 1.11, and 1.15 in Equation 1.8, one can write the energy functional as follows,

$$\begin{aligned} E[\rho(\mathbf{r})] = &-\sum_{A=1}^{M}\int \frac{Z_A}{|\mathbf{r}-\mathbf{r}_A|}\rho(\mathbf{r})d\mathbf{r} \\ &+\frac{1}{2}\int\int \frac{\rho(\mathbf{r}_1)\rho(\mathbf{r}_2)}{|\mathbf{r}_1-\mathbf{r}_2|}d\mathbf{r}_1 d\mathbf{r}_2 + \Delta V_{ee}[\rho(\mathbf{r})] \\ &-\frac{1}{2}\sum_{i=1}^{N}\langle \psi_i^{KS}(1)|\nabla_1^2|\psi_i^{KS}(1)\rangle + \Delta T[\rho(\mathbf{r})] \end{aligned} \tag{1.16}$$

The sum of two correction terms $\Delta V_{ee}[\rho(\mathbf{r})]$ and $\Delta T[\rho(\mathbf{r})]$ is called the exchange-correlation energy functional $E_{xc}[\rho(\mathbf{r})]$.

$$E_{xc}[\rho(\mathbf{r})] = \Delta V_{ee}[\rho(\mathbf{r})] + \Delta T[\rho(\mathbf{r})] \tag{1.17}$$

Minimizing $E[\rho(\mathbf{r})]$ with respect to $\rho(\mathbf{r})$ using the Lagrange multipliers method subject to the constraint that the KS orbitals remain orthonormal, leads us to a set of equations (called KS equations),

$$\hat{h}^{KS}(\mathbf{r}_1)\psi_i^{KS}(\mathbf{r}_1) = \varepsilon_i \psi_i^{KS}(\mathbf{r}_1) \tag{1.18}$$

where

$$\hat{h}^{KS}(\mathbf{r}_1) = -\frac{1}{2}\nabla_1^2 - \sum_{A=1}^{M}\frac{Z_A}{r_{1A}} + \int \frac{\rho(\mathbf{r}_2)}{r_{12}}d\mathbf{r}_2 + \hat{v}_{xc}(\mathbf{r}_1) \tag{1.19}$$

and $\hat{h}^{KS}(\mathbf{r}_1)$ is the KS one-electron operator, ε_i is the i^{th} KS orbital energy, and $\hat{v}_{xc}(\mathbf{r})$ is the exchange-correlation potential which can be defined as the derivative of the $E_{xc}[\rho(\mathbf{r})]$ with respect to the density $\rho(\mathbf{r})$.

$$\hat{v}_{xc}(\mathbf{r}) = \frac{\delta E_{xc}[\rho(\mathbf{r})]}{\delta\rho(\mathbf{r})} \tag{1.20}$$

KS equations (Equation 1.18) must be solved self-consistently like the Hartree-Fock (HF) equations. However, KS equations differ from the HF equations only by $\hat{v}_{xc}(\mathbf{r})$, where $\hat{v}_{xc}(\mathbf{r})$ is computed in each self-consistent cycle [9]. $\hat{v}_{xc}(\mathbf{r})$ is an unknown term and an appropriate approximation of $E_{xc}[\rho(\mathbf{r})]$ is needed. A number of possible approximations may be made for $E_{xc}[\rho(\mathbf{r})]$. The simplest known as the Local Density Approximation (LDA). In general, for a spin-unpolarized system, $E_{xc}^{LDA}[\rho(\mathbf{r})]$ is written as

$$E_{xc}^{LDA}[\rho(\mathbf{r})] = \int \rho(\mathbf{r})\varepsilon_{xc}^{LDA}[\rho(\mathbf{r})]d\mathbf{r} \tag{1.21}$$

where $\varepsilon_{xc}^{LDA}[\rho(\mathbf{r})]$ is the exchange-correlation energy per electron of a homogeneous electron gas of density $\rho(\mathbf{r})$. The $\varepsilon_{xc}^{LDA}[\rho(\mathbf{r})]$ is decomposed into exchange and correlation terms linearly,

$$\varepsilon_{xc}^{LDA}[\rho(\mathbf{r})] = \varepsilon_{x}^{LDA}[\rho(\mathbf{r})] + \varepsilon_{c}^{LDA}[\rho(\mathbf{r})] \tag{1.22}$$

The exchange terms $\varepsilon_{x}^{LDA}[\rho(\mathbf{r})]$ and $E_{x}^{LDA}[\rho(\mathbf{r})]$ are known and given as [10, 11],

$$\varepsilon_{x}^{LDA}[\rho(\mathbf{r})] = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{1/3}\rho(\mathbf{r})^{1/3} \tag{1.23}$$

$$E_{x}^{LDA}[\rho(\mathbf{r})] = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{1/3}\int \rho(\mathbf{r})^{4/3}d\mathbf{r} \tag{1.24}$$

However, the correlation term $E_{c}^{LDA}[\rho(\mathbf{r})]$ is unknown and there are numerous different approximations. Vosko-Wilk-Nusair (VWN) [12], Perdew-Zunger (PZ81) [13], Cole-Perdew (CP) [14], and Perdew-Wang (PW92) [15] are common examples of $E_{c}^{LDA}[\rho(\mathbf{r})]$ functionals.

A next level of $E_{xc}[\rho(\mathbf{r})]$ approximations are the so-called Generalized Gradient Approximation (GGA),

$$E_{xc}^{GGA}[\rho(\mathbf{r}),\nabla\rho(\mathbf{r})] = \int \rho(\mathbf{r})\varepsilon_{xc}^{GGA}[\rho(\mathbf{r}),\nabla\rho(\mathbf{r})]d\mathbf{r} \tag{1.25}$$

The ε_{xc}^{GGA} is given as a functional of both the density $\rho(\mathbf{r})$ and its gradient $\nabla\rho(\mathbf{r})$. The gradient $\nabla\rho(\mathbf{r})$ is given to account for non-homogeneity of the true electron density. LDA fails in situations where the density undergoes rapid changes such as in molecules, whereas GGA shows improvement over the LDA for calculations of molecular structures and in representing weak intermolecular bonds. Becke-88 (B88) [16], Perdew-Wang-91 (PW91) [15, 17], and Perdew-Burke-Ernzerhof (PBE) [18] are common examples of GGA functionals.

The obvious logical next step in the functional improvement is to take account of the second derivative (i.e., the Laplacian) of the density ($\nabla^2\rho$). These functionals are termed as meta-GGA (mGGA). The mGGA functionals includes also the kinetic-energy density $\tau[\rho(\mathbf{r})] = \sum_{i}^{occupied} \frac{1}{2}\left|\nabla\psi_i^{KM}(\mathbf{r})\right|^2$ [19].

$$E_{xc}^{mGGA}[\rho(\mathbf{r}),\triangledown\rho(\mathbf{r}),\nabla^2\rho(\mathbf{r}),\tau] = \int \rho(\mathbf{r})\varepsilon_{xc}^{mGGA}[\rho(\mathbf{r}),\triangledown\rho(\mathbf{r}),\nabla^2\rho(\mathbf{r}),\tau]d\mathbf{r} \tag{1.26}$$

A number of mGGA functionals for exchange, correlation, or both, have been developed such as B95 [20], B98 [21], Perdew-Kurth-Zupan-Blaha (PKZB) [22] and Lee-Yang-Parr (LYP) [23].

In another approach in 1993 Becke[24] introduced a successful hybrid functional. This functional is constructed as a linear combination of the HF exchange (E_x^{HF}, Equation 1.27)

and a number of $E_{xc}[\rho(\mathbf{r})]$ functionals.

$$E_x^{HF} = -\frac{1}{2}\sum_{a>b}\int\int \psi_a^*(\mathbf{r_1})\psi_b^*(\mathbf{r_2})\frac{1}{r_{12}}\psi_b(\mathbf{r_1})\psi_a(\mathbf{r_2})d\mathbf{r_1}d\mathbf{r_2} \tag{1.27}$$

For example, the popular B3LYP (Becke, three-parameter, Lee-Yang-Parr) functional is,

$$\begin{aligned} E_{xc}^{B3LYP} =& E_x^{LDA} + a_0(E_x^{HF} - E_x^{LDA}) + a_x(E_x^{B88} - E_x^{LDA}) \\ &+ E_c^{VWN} + a_c(E_c^{LYP} - E_c^{VWN}) \end{aligned} \tag{1.28}$$

where, $a_0 = 0.20$, $a_x = 0.72$, and $a_c = 0.81$. E_x^{B88} is the Becke 88 GGA exchange functional [16], E_c^{LYP} is the Lee, Yang and Parr GGA correlation functional [23], and E_c^{VWN} is the Vosko, Wilk, and Nusair LDA correlation functional [12].

1.3 Energy Decomposition Analysis

Energy decomposition analysis (EDA) can be a valuable tool for interpreting the results of quantum chemical calculations [25]. A significant number of developments have been made in the field of EDA [26–32]. One of the earliest energy decomposition analysis schemes developed is the Kitaura-Morokuma (KM) scheme [33]. This scheme is limited to the restricted Hartree-Fock (RHF) level of theory. The KM scheme allows a partitioning of the HF energy as follows,

$$\Delta E = \Delta E_{ele} + \Delta E_{exrep} + \Delta E_{pol} + \Delta E_{chtr} + \Delta E_{mix} \tag{1.29}$$

where ΔE_{ele} is the classical electrostatic interaction, ΔE_{exrep} is the exchange-repulsion, ΔE_{pol} is the polarization interaction, ΔE_{chtr} is is the charge transfer energy, and ΔE_{mix} describes contributions to the interaction energy that are not capable of being assigned to a particular component [34]. Another scheme is the absolutely localized molecular orbital energy decomposition analysis (ALMO-EDA), in which

$$\Delta E = \Delta E_{frz} + \Delta E_{pol} + \Delta E_{chtr} \tag{1.30}$$

where ΔE_{frz} is the frozen density component that describes the exchange and electrostatic interaction of the frozen charge densities [26, 34].

1.4 Atoms in Molecules

The idea of an atom or a functional group in a molecule is principal to the science of chemistry, which existed before our understanding of quantum mechanics, atomic structure, and the nature of chemical bonding. As a result understanding properties of atoms and functional groups as molecules undergo transformations and combinations has been of interest to chemists from long time.

A functional group in different molecular environments usually gives similar properties. Therefore computing molecular properties for molecules can be accomplished by assuming the transferability of the atoms in molecules (AIM) atomic properties. Thus the additivity of AIM atomic properties gives us a better understanding of molecular properties.

A lot of scientific efforts have been devoted to AIM schemes over the past decades. The Hilbert-space analysis and the analysis in the three dimensional (3D) physical space are the

two most conceptually different approaches for defining an atom within a molecule. The Hilbert-space and 3D analyses differ conceptually, and usually give significantly different results.

1.4.1 Hilbert-Space

In the Hilbert-space analysis each atom within a molecule is characterized with the nucleus and basis functions assigned to it [30]. The Mulliken [35] and Löwdin [36] population analyses are known examples within the Hilbert-space analysis

$$q_A^{Mulliken} = Z_A - \sum_{\mu \in A} (\mathbf{PS})_{\mu\mu} \tag{1.31}$$

$$q_A^{Löwdin} = Z_A - \sum_{\mu \in A} \left(\mathbf{S}^{1/2}\mathbf{P}\mathbf{S}^{1/2}\right)_{\mu\mu} \tag{1.32}$$

where q_A is the net charge associated with an atomic nucleus A, Z_A is the nuclear charge of atom A, the index of summation indicates that the sum is over the basis functions μ centered on A, $\mathbf{S}$ is the overlap matrix, and $\mathbf{P}$ is the density matrix. The results of Hilbert-space analyses greatly depend on the basis set applied [30]. This partitioning cannot be extended to the complete basis set limit because the complete basis cannot be divided into atomic subsets. Thus, the net charge from this technique is not unique [37].

1.4.2 Three-Dimensional (3D) Physical Space

In the 3D case, one decomposes the physical space into atomic domains with sharp (non-overlapping) or fuzzy boundaries (overlapping). In contrast to Hilbert-space analysis the

results of the 3D analysis usually have moderate basis set dependence and converge well with the increasing basis sets size, but depend greatly on the definition of the 3D domains assigned to the different atoms [30].

1.4.2.1 Properties of AIM with 3D Space

In general, we can associate the atomic electron density $\rho_A(\mathbf{r})$ for every atom A in the molecule by defining a weight function $W_A(\mathbf{r})$ such that

$$\rho_A(\mathbf{r}) = W_A(\mathbf{r})\rho(\mathbf{r}) \tag{1.33}$$

where $\rho(\mathbf{r})$ is the molecular electron density at the position $\mathbf{r}$. $\rho(\mathbf{r})$ can also be represented as a summation of all AIM densities in the molecule.

$$\rho(\mathbf{r}) = \sum_A \rho_A(\mathbf{r}) \tag{1.34}$$

The weight function $W_A(\mathbf{r})$ should obey the condition

$$\sum_A W_A(\mathbf{r}) = 1 \tag{1.35}$$

Partitioning the molecular electron density into its atomic contributions is extensively used in the literature for different applications such as electron population, distribution analysis, and molecular energy components [38]. The identification methods of AIM within 3D analysis are categorized as disjunct and fuzzy methods.

1.4.2.2 Disjunct Method

In this case the partitioning weight of specific atom A is one, and only one, within its atomic basin Ω_A and zero otherwise. This situation is described by Equation 1.36 [39]

$$W_A(\mathbf{r}) = \begin{cases} 1, & \text{if } \mathbf{r} \in \Omega_A \\ 0, & \text{otherwise} \end{cases} \tag{1.36}$$

This is characteristic for the quantum theory of atoms in molecules (QTAIM), which was developed by Richard Bader [40–42]. QTAIM recovers the electron densities of atoms by partitioning the molecular space via determination of zero flux surfaces where the gradient of electron density $\nabla\rho(\mathbf{r})$ is zero for all points on the atomic surface, Equation 1.37.

$$\nabla\rho(\mathbf{r}) \cdot \mathbf{n}(\mathbf{r}) = 0 \qquad \textit{for all } \mathbf{r} \textit{ on the surface} \tag{1.37}$$

where $\nabla\rho(\mathbf{r})$ is the gradient vector of electron density and $\mathbf{n}(\mathbf{r})$ is the unit vector normal to the atomic surface.

The disadvantages of the QTAIM method are that the electronic charge at any point is attributed to only one of the atoms in the molecule. Thus it cannot reflect directly the fact that there is an accumulation of electron density in bonding regions between atoms. Also QTAIM may lead to so called non-nuclear attractors (NNA), i.e., basins that have no nuclei associated to them. In addtion, the QTAIM method requires significant computer time to investigate the topology of the electron density [38].

1.4.2.3 Fuzzy Method

In fuzzy atomic domains, the regions assigned to individual atoms have no sharp boundaries and show a continuous transition from one to another. Fuzzy atoms were first used by Hirshfeld [43] for calculating effective atomic charges in molecules.

Hirshfeld weight

The Hirshfeld weight function $W_A(\mathbf{r})$ is computed from a so-called promolecular density $\rho^0_{mol}(\mathbf{r})$,

$$W_A(\mathbf{r}) = \frac{\rho^0_A(\mathbf{r})}{\rho^0_{mol}(\mathbf{r})} = \frac{\rho^0_A(\mathbf{r})}{\sum\limits_A \rho^0_A(\mathbf{r})} \tag{1.38}$$

where $\rho^0_A(\mathbf{r})$ is the electron density of the isolated atom A. $\rho^0_{mol}(\mathbf{r})$ is computed by summing overall atomic densities $\left(\rho^0_A(\mathbf{r})\right)$ at the same point in space. The atomic electron densities for the real molecule are then obtained via

$$\rho_A(\mathbf{r}) = \frac{\rho^0_A(\mathbf{r})}{\rho^0_{mol}(\mathbf{r})}\rho(\mathbf{r}) \tag{1.39}$$

where $\rho(\mathbf{r})$ is the real molecular density.

The Hirshfeld weight has many shortcomings such as atomic charges tend to be virtually zero, depend on the choice of the promolecular density, are only available for neutral molecules, and many others [44]. Many extended Hirshfeld population analysis schemes were proposed to overcome original Hirshfeld drawbacks [45–48].

Becke weight

Becke, in his seminal paper [49], decomposed the molecular integral into atomic components by assigning a weight function $W_A(\mathbf{r})$ for each atom in the molecule. The Becke weight was originally invented for numerical integration of three-dimensional functions of molecular atomic properties. For any point $\mathbf{r}_i$ in molecular space, the confocal elliptical coordinate μ_{iAB} of nuclei A and B can be calculated as

$$\mu_{iAB} = \frac{r_{iA} - r_{iB}}{R_{AB}} \tag{1.40}$$

where r_{iA}, r_{iB} denote the distances from r_i to nuclei A and B respectively. R_{AB} is the internuclear distance between the atoms A and B. Values of μ_{iAB} are always between -1 and 1. For example, if the $\mathbf{r}_i$ point lies on atom A then $\mu_{iAB} = -1$ and if the point lies on atom B then $\mu_{iAB} = 1$. Becke introduced the following continuous cubic polynomial function that varies smoothly between -1 and $+1$,

$$h(\mu_{iAB}) = \frac{3}{2}\mu_{iAB} - \frac{1}{2}\mu_{iAB}^3 \tag{1.41}$$

where $h(1) = 1$ and $h(-1) = -1$. This function is not "step function-like", therefore Becke iterated as

$$\begin{aligned} f_1(\mu_{iAB}) &= h(\mu_{iAB}) && k = 1 \\ f_2(\mu_{iAB}) &= h[h(\mu_{iAB})] && k = 2 \\ f_3(\mu_{iAB}) &= h\{h[h(\mu_{iAB})]\} && k = 3 \\ &\vdots \end{aligned} \tag{1.42}$$

to obtain successively sharper functions. A sequence of cutoff profiles is illustrated in Figure 1.1 for k values from 1 to 5. Sharper functions are generated with increasing the iteration k value. Becke found on the basis of his experience that $k = 3$ gives the best numerical results.

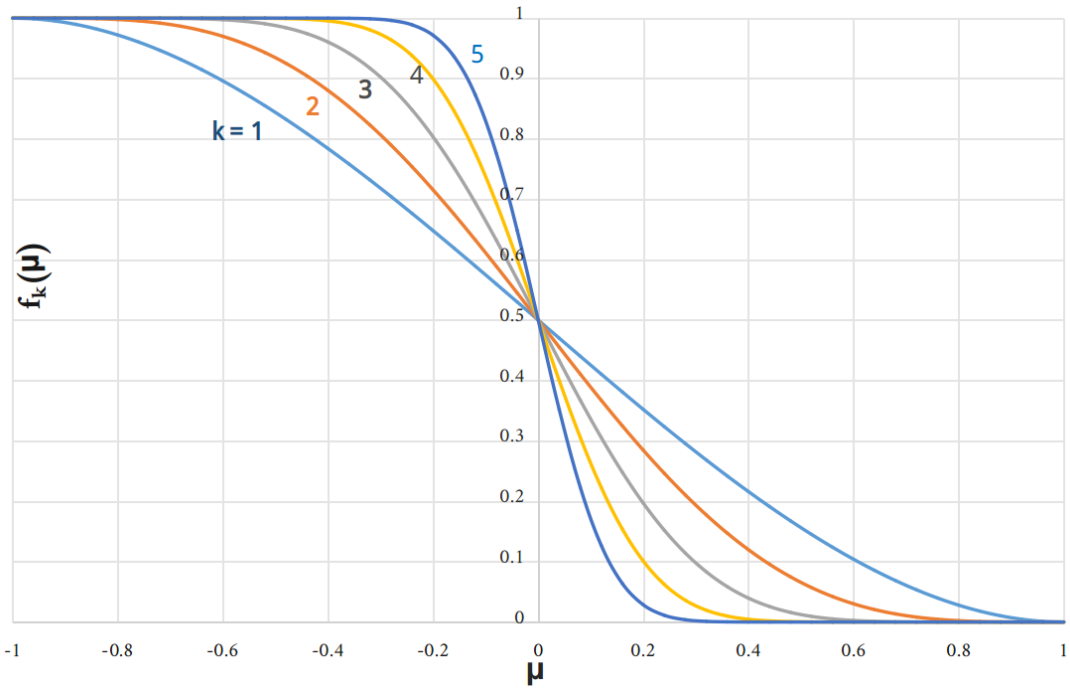


Figure 1.1: Cutoff profiles $f_k(\mu_{iAB})$ of Equation 1.42 for $k = 1$ to 5 [49].

Because the desired weight has to be in the range $[0,1]$, Becke proposed $s(\mu_{iAB})$ function

$$s(\mu_{iAB}) = \frac{1}{2}\left[1 - f_3(\mu_{iAB})\right] \tag{1.43}$$

One problem with the scheme given so far is that the space is divided equally between any two atoms. A proposed atomic size adjustment is introduced for heteronuclear molecules

by transforming μ_{iAB} into ν_{iAB} as follows,

$$\nu_{iAB} = \mu_{iAB} + a_{AB}(1 - \mu_{iAB}^2) \tag{1.44a}$$

$$a_{AB} = \frac{u_{AB}}{u_{AB}^2 - 1} \tag{1.44b}$$

$$u_{AB} = \frac{\chi - 1}{\chi + 1} \tag{1.44c}$$

$$\chi_{AB} = \frac{R_A}{R_B} \tag{1.44d}$$

where R_A and R_B are the Bragg-Slater radii for atoms A and B, respectively. Becke defined the Voronoi polyhedron on nucleus A by the product of all pairwise ownership contributions $s(\nu_{iAB})$.

$$P_A(\mathbf{r}) = \prod_{B=1, B\neq A}^{N} s(\nu_{iAB}) \tag{1.45}$$

$P_A(\mathbf{r})$ is called the cell function of atom A that has a value of one at nucleus A and vanishes at all other nuclei B. The normalized Becke weight function $W_A(\mathbf{r})$ for an atom A at a particular point $\mathbf{r}$ can now be defined as follows.

$$W_A(\mathbf{r}) = \frac{P_A(\mathbf{r})}{\sum_n P_n(\mathbf{r})} \tag{1.46}$$

The Becke weight suffers some drawbacks such as: (i) its dependence on fixed Bragg−Slater radii makes it less useful in dealing with AIM that have different charged states, (ii) if there is a large difference between the Bragg−Slater radii of the atoms, for example, in Equation 1.44d, if the value of χ_{AB} for two atoms A and B is larger than 2.4 or smaller than 0.41, then the ratio will be capped to these extremes [50].

Atoms in Molecules from Radial Density

Warburton et al. [50, 51] indicated that the radial density can be used as another method for partitioning the molecule space into atomic contributions. The results show that molecules can be partitioned into three main regions: non-bonding, core, and bonding regions. Figure 1.2 shows the radial electron density of N_2, where N_2 is partitioned into these regions using Becke weight. The core region of each nucleus is easily identified in this cross section as the symmetrical, volcano-shaped cones that surround the nuclei. The core is separated from the distorted valence by a minimum in radial density.

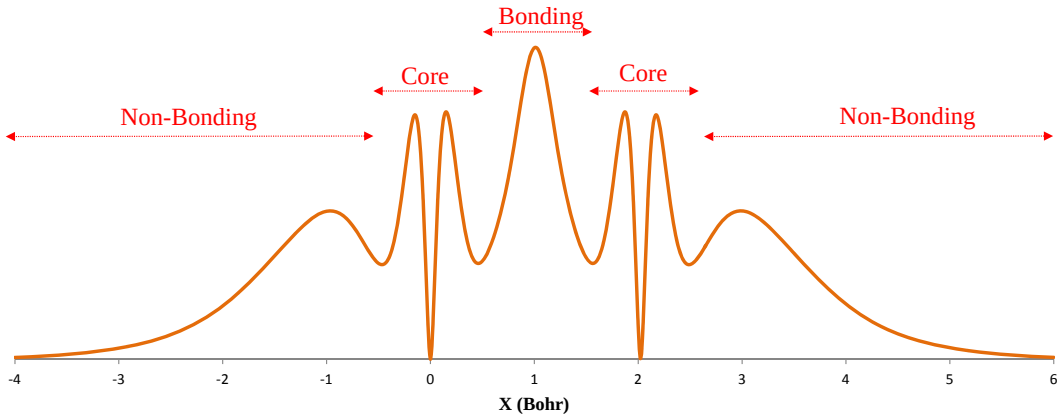


Figure 1.2: Radial electron density of N_2.

A partitioning weight can be used to calculate the atomic electron density associated with each AIM as follows,

$$\rho_A(\mathbf{r}_i) = W_A(\mathbf{r}_i)\rho(\mathbf{r}_i) \tag{1.47}$$

and the AIM radial density is expressed as

$$\rho_A^{rad}(\mathbf{r}_i) = \mathbf{r}_{iA}^2\rho_A(\mathbf{r}_i) \tag{1.48}$$

The total radial density for the molecule is the sum of each of these individual radial contributions.

$$\begin{aligned}\rho_{rad}(\mathbf{r}_i) &= \sum_{A=1}^{N} W_A(\mathbf{r}_i)\mathbf{r}_{iA}^2\rho(\mathbf{r}_i) \\ &= \sum_{A=1}^{N} \mathbf{r}_{iA}^2\rho_A(\mathbf{r}_i) \\ &= \sum_{A=1}^{N} \rho_A^{rad}(\mathbf{r}_i)\end{aligned} \tag{1.49}$$

1.5 Numerical Integration

The evaluation of the 3D integral (Equation 1.50) is important in many fields. For example, some of the applications of numerical integration in physical chemistry are to calculate number of electrons ($N_e = \int \rho(\mathbf{r})d\mathbf{r}$), dipole moment ($\mu(x) = \int x \cdot \rho(x)dx$), Coulomb potential ($V_{ee} = \int\int \frac{\rho(\mathbf{r}_1)\rho(\mathbf{r}_2)}{|r_{12}|}d\mathbf{r}_1 d\mathbf{r}_2$), potential energy ($V_{ne} = -\sum_A Z_A \int \frac{\rho(\mathbf{r})}{|\mathbf{r}-\mathbf{R}_A|}d\mathbf{r}$) and many other atomic and molecular properties.

$$I = \int f(\mathbf{r})d\mathbf{r} \tag{1.50}$$

The integration of some functions may be difficult or impossible to compute analytically, but can be computed numerically. For example, in DFT calculations, the exchange-correlation energies of many-electron approximations (Equation 1.51) cannot be evaluated analytically.

$$I = \int f(\rho, \nabla\rho, ...)d\mathbf{r} \tag{1.51}$$

In the next sections, some of the most common numerical integration methods, Newton-Cotes formulas and Gaussian quadrature, will be illustrated.

1.5.1 Newton-Cotes Formulas

Newton-Cotes rules are used to evaluate the integral, $\int_a^b f(x)dx$ at equally spaced points. Table 1.1 shows some common Newton-Cotes formulas (trapezoidal rule, Simpson's rule, and Euler-Maclaurin). Figure 1.3 shows an example for the trapezodial rule. In this figure, the blue area represents the approximate area of function $f(x) = \cos(x-0.2)+1.5$ in range $[-1,7]$ using trapezodial rule.

Table 1.1: Some common Newton-Cotes formulas within range $[a,b]$ and step size $h = \frac{b-a}{n}$ [52, 53].

Name	Formula
Trapezoidal rule	$h\sum_{k=1}^{n}\frac{f(x_{k+1})+f(x_k)}{2}$
Simpson's rule [1]	$\frac{h}{3}\left[f(x_0)+2\sum_{k=1}^{\frac{n}{2}-1}f(x_{2k})+4\sum_{k=1}^{\frac{n}{2}}f(x_{2k-1})+f(x_n)\right]$
Euler-Maclaurin [2]	$h\sum_{k=1}^{n-1}f(a+kh)-\frac{h}{2}(f(b)+f(a))-\sum_{k=1}^{m}\frac{h^{2k}}{(2k)!}B_{2k}\left(f^{(2k-1)}(b)-f^{(2k-1)}(a)\right)$

[1] n is an even integer number.
[2] B_{2k} is a Bernoulli number, the first few even numbers are $B_0 = 1$, $B_2 = \frac{1}{6}$, $B_4 = \frac{-1}{30}$.

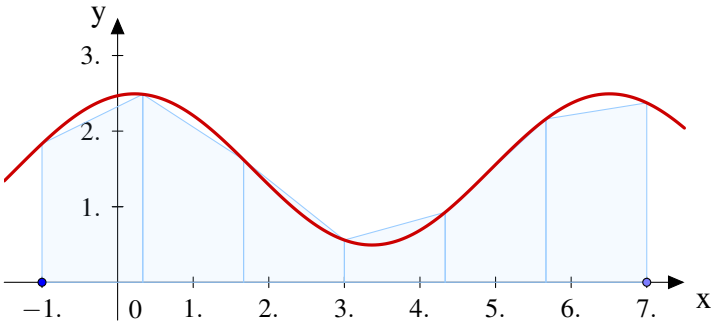


Figure 1.3: An example for trapezoidal rule, $f(x) = \cos(x-0.2)+1.5$ where $n = 6$.

1.5.2 Gaussian Quadrature

Geometrically, quadrature means finding a square equal in area to the area of interest. It refers to any numerical approximation of the integral, $\int_a^b f(x)dx$, with the fewest function evaluations on finite sets of points (Equation 1.52).

$$\int_a^b f(x)dx = \sum_{i=1}^{n} \frac{w_i}{w(x_i)} f(x_i) = \sum_{i=1}^{n} \omega(x_i) f(x_i) \tag{1.52}$$

where x_i are the base points and w_i are called quadrature points, abscissas or weight factors. Practically the weights w_i and the x_i points have to be stored as fixed data in the program. Orthogonal polynomials are required to find the quadrature points where Equation 1.53 should be obeyed for a sequence of orthogonal polynomials over the range $[a,b]$, [54]

$$\int_a^b w(x)p_i(x)p_j(x)dx = \delta_{ij} \int_a^b w(x)p_i(x)^2 dx \tag{1.53}$$

where $w(x)$ is the weight function, and $p_n(x)$ is the following polynomial function of degree n,

$$p_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \tag{1.54}$$

The roots of $p_n(x)$ must be real numbers and fall within the range $[a,b]$. Examples of orthogonal polynomials are shown in Table 1.2. For example, Laguerre functions, $L_n^{\alpha}(x)$, are orthogonal over the weight function $x^{\alpha}e^{-x}$ within $[0,\infty)$.

If $n \geq i \geq 1$ and $j = 0$ in Equation 1.53, the equation becomes,

$$\int_a^b w(x)p_i(x)p_0(x)dx = 0 \qquad \text{where} \quad n \geq i \geq 1 \tag{1.55}$$

Table 1.2: Some of the common classical orthogonal polynomials [52, 55–57].

Name	$p_n(x)$	$w(x)$	x_i	w_i	Range$[a,b]$
Legendre	$P_n(x)$	1	i^{th} zero of $P_n(x)$	$\frac{2}{(1-x_i^2)}[P_n{}'(x_i)]^2$	$[-1,1]$
Chebyshev 1st	$T_n(x)$	$1/\sqrt{1-x^2}$	$\cos[\frac{(2i-1)\pi}{2n}]$	π/n	$(-1,1)$
Chebyshev 2nd	$U_n(x)$	$\sqrt{1-x^2}$	$\cos[\frac{i\pi}{(n+1)}]$	$\frac{\pi}{n+1}\sin^2[\frac{i\pi}{(n+1)}]$	$[-1,1]$
Laguerre	$L_n^{\alpha}(x)$	$x^{\alpha}e^{-x}$	i^{th} zero of $L_n^{\alpha}(x)$	$\frac{x_i}{(n+1)^2[L_{n+1}(x_i)]^2}$	$[0,\infty)$
Hermite	$H_n(x)$	e^{-x^2}	i^{th} zero of $H_n(x)$	$\frac{2^{n-1}n!\sqrt{\pi}}{n^2[H_{n-1}(x_i)]^2}$	$(-\infty,\infty)$
Jacobi	$J_n^{\alpha,\beta}(x)$	$(1-x)^{\alpha}(1+x)^{\beta}$	i^{th} zero of $J_n^{\alpha,\beta}(x)$		$(-1,1)$

since $p_0(x)$ is constant, Equation 1.55 is written as

$$\int_a^b w(x)p_i(x)p_0(x)dx = p_0(x)\int_a^b w(x)p_i(x)dx = \int_a^b w(x)p_i(x)dx = 0 \qquad \text{where} \quad n \geq i \geq 1 \tag{1.56}$$

If $x_1, x_2, \cdots, x_n$ are denoting the roots of the polynomial, $p_n(x)$, one can obtain the following sum for any sets of x_j

$$\sum_{j=1}^{n} w_j p_n(x_j) = 0 \tag{1.57}$$

It can be shown that there is a unique solution of w_j for all polynomials $p_i(x)$ where $n \geq i \geq 1$

$$\sum_{j=1}^{n} w_j p_i(x_j) = 0 \qquad \text{where} \quad n \geq i \geq 1 \tag{1.58}$$

This can be done by solving a set of polynomial $p_i(x)$ equations simultaneously,

$$\begin{pmatrix} p_0(x_1) & \cdots & p_0(x_n) \\ \vdots & \ddots & \vdots \\ p_{n-1}(x_1) & \cdots & p_{n-1}(x_n) \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} \int_a^b w(x)p_0(x)dx \\ \vdots \\ 0 \end{pmatrix} \tag{1.59}$$

Using Equations 1.56 and 1.58, the result is

$$\int_a^b w(x)p_i(x)dx = \sum_{j=1}^{n} w_j p_i(x_j) = 0 \qquad \text{where} \quad n \geq i \geq 1 \tag{1.60}$$

In order to get a non-trivial solution for Equation 1.59, Equation 1.61 should not equal zero,

$$\int_a^b w(x)p_0(x)dx = \sum_{j=1}^{n} w_j p_0(x_j) \neq 0 \tag{1.61}$$

Writing the function $f(x)$ as a linear combination of $w(x)p_i(x)$ leads to,

$$f(x) \approx \sum_{k=1} c_k w(x) p_k(x) \tag{1.62}$$

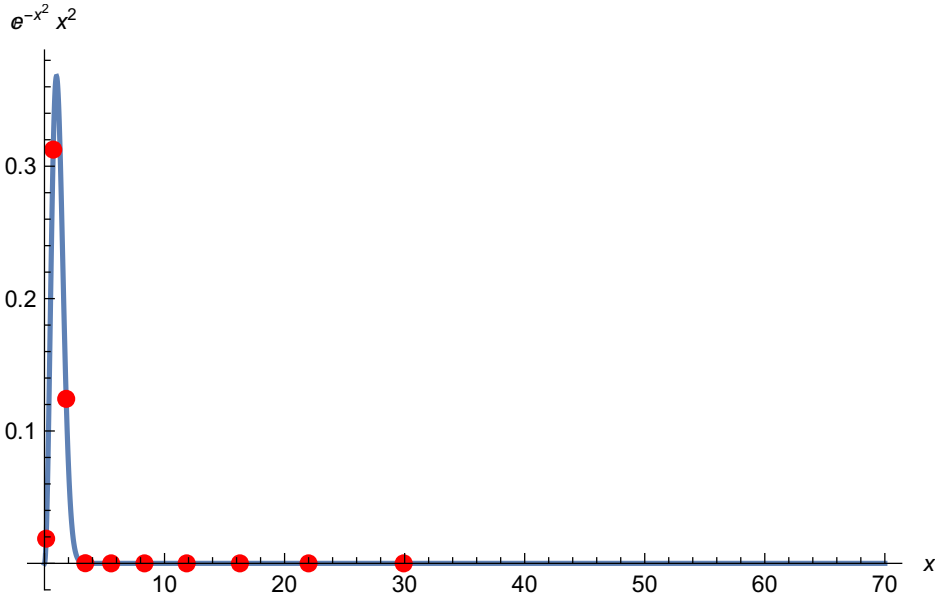
The integral of $f(x)$ (Equation 1.52) can be derived as follows:

$$\begin{aligned}
\int_a^b f(x)dx &= \int_a^b \sum_{k=1} c_k w(x) p_k(x) dx \qquad \text{using Equation 1.62} \\
&= \sum_{k=1} c_k \left(\int_a^b w(x) p_k(x) dx \right) \\
&= \sum_{k=1} c_k \left(\sum_{j=1}^{n} w_j p_k(x_j) \right) \qquad \text{using Equation 1.60} \\
&= \sum_{j=1}^{n} w_j \sum_{k=1} c_k p_k(x_j) \\
&= \sum_{j=1}^{n} \frac{w_j}{w(x_j)} \sum_{k=1} c_k w(x_j) p_k(x_j) \\
&= \sum_{j=1}^{n} \frac{w_j}{w(x_j)} f(x_j) \qquad \text{using Equation 1.62}
\end{aligned} \tag{1.63}$$

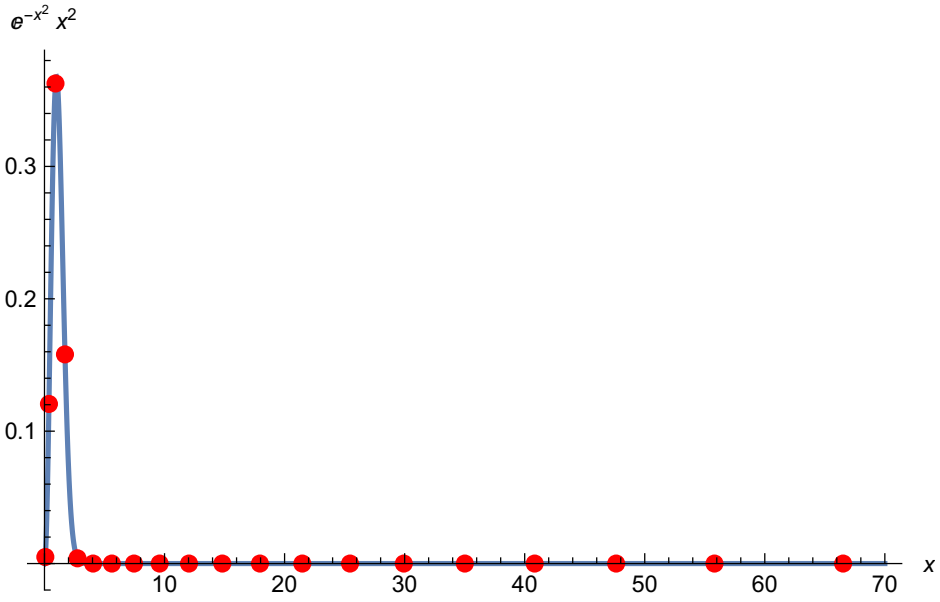
For example, the numerical integral of $x^2 e^{-x^2}$ using the Laguerre method (Table 1.2) where $\alpha = 0$, $n = 10$ and $n = 20$ points are displayed in Figures 1.4a and 1.4b, respectively. The integral is negligibly small beyond $x = 4$ while the most of quadrature roots w_i are beyond $x = 4$. The use of 20 points instead of 10 results in increasing the number of quadrature roots out of the region of interest (Figure 1.4b).

To overcome this problem, these roots should be scaled. This can be done using the u-substitution integration technique

$$\int_a^b f(x)dx = \int_{u(a)}^{u(b)} f(g(u))g'(u)du \qquad \text{where} \quad x = g(u) \tag{1.64}$$



(a) using 10 points



(b) using 20 points

Figure 1.4: Gauss-Laguerre nodes for $x^2e^{-x^2}$ function.

A new function, $u = \frac{x}{\lambda}$, is proposed where λ is a scale factor. The integration, $\int_a^b f(x)dx$, is achieved by rewriting the integral as $\int_{u(a)}^{u(b)} \lambda f(u\lambda)du$. Therefore Equation 1.52 becomes,

$$\int_a^b f(x)dx = \lambda \int_{\frac{a}{\lambda}}^{\frac{b}{\lambda}} f(\lambda u)du = \lambda \sum_{i=1}^{n} \frac{w_i}{w(u_i)} f(\lambda u_i) \tag{1.65}$$

As a result the distribution of the points in **r** space can be controlled by scaling the factor λ (Figures 1.5a and 1.5b).

1.6 Numerical Integration in Quantum Chemistry

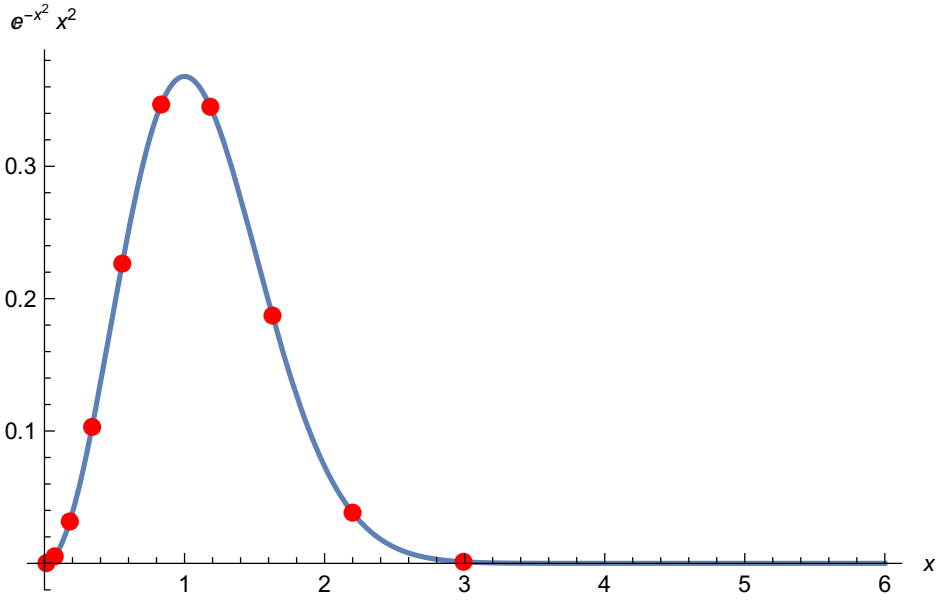
1.6.1 Atomic Numerical Integration

We are living in 3D world, thus the solution of triple integrals using spherical coordinates is considered to be important.

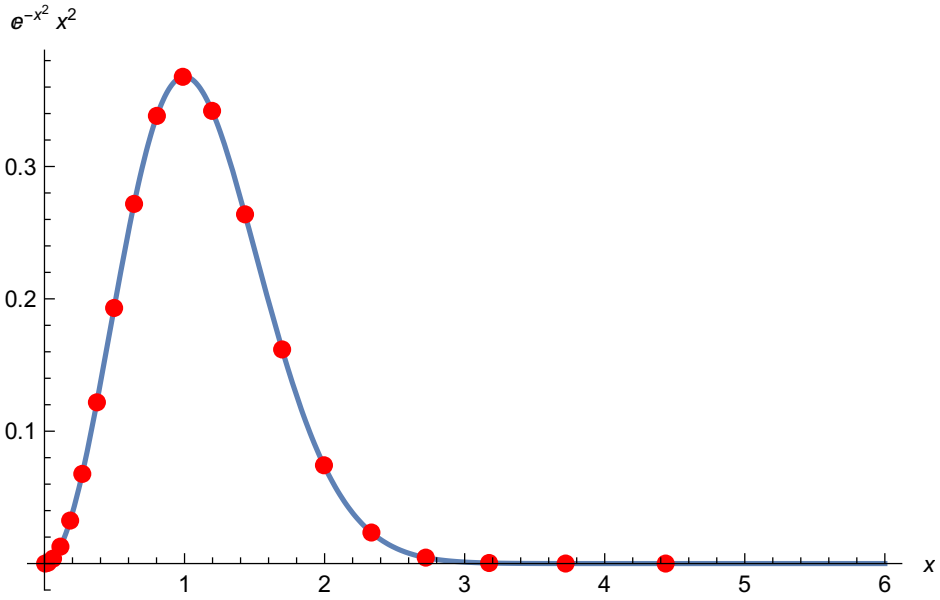
$$\int f(\mathbf{r})d\mathbf{r} = \int_0^{2\pi} \int_0^{\pi} \int_0^{\infty} r^2 \sin(\theta) f(r,\theta,\phi)d\phi d\theta dr \tag{1.66}$$

For simplicity, the function $f(\mathbf{r})$ is separated into radial and angular parts (e.g. $f(\mathbf{r})$=$f(r)f(\theta,\phi)$),

$$\int f(\mathbf{r})d\mathbf{r} = \int_0^{\infty} r^2 f(r)dr \int_0^{2\pi} \int_0^{\pi} \sin(\theta) f(\theta,\phi)d\phi d\theta \tag{1.67}$$



(a) using 10 points and rescaled using $\lambda = 0.10$.



(b) using 20 points and rescaled using $\lambda = 0.067$.

Figure 1.5: Rescaled Gauss-Laguerre nodes for $x^2 e^{-x^2}$ function.

1.6.1.1 Integration of the Radial Part

Different algorithms have been used to evaluate the radial part, $I(r) = \int_0^\infty r^2 f(r) dr$, where the intervals, $[a,b]$, (Tables 1.1 and 1.2) were mapped to $[0,\infty)$.

Gauss-Laguerre and Gauss-Hermite quadratures (Table 1.2) are the most traditional choices because their radial integration intervals are over $[0,\infty)$ and $(-\infty,\infty)$ respectively.

For example, using Laguerre quadrature for $\alpha = 0$ in Equation 1.65 the integral, $I(r)$, is evaluated as,

$$I(r) = \int_0^\infty r^2 f(r) dr = \lambda \sum_{i=1}^{n} \frac{w_i}{w(u_i)} (\lambda u_i)^2 f(\lambda u_i) = \lambda^3 \sum_{i=1}^{n} w_i u_i^2 e^{u_i} f(\lambda u_i) \tag{1.68}$$

where values of u_i are the roots of Laguerre polynomial $L_n^\alpha(x)$. Values of u_i and w_i can be calculated using Equation 1.59 or obtained using the formulas in Table 1.2. The parameter λ is utilized to adjust the radial points to a suitable physical scale.

Becke [49] used the Gauss-Chebyshev 2nd quadrature (Table 1.2) to find the radial integral $I(r)$ by mapping the interval of r from $[-1,1]$ to $[0,\infty)$. Becke used the transform coordinate formula $r = \alpha \frac{(1+x)}{(1-x)}$ with u-substitution integration technique (Equation 1.64) $\left(\text{note: } dr = \frac{2\alpha}{(1-x)^2} dx\right)$ to get:

$$\begin{aligned}
\int_0^\infty r^2 f(r) dr &= \int_{-1}^{1} \left(\alpha \frac{1+x}{1-x}\right)^2 \frac{2\alpha}{(1-x)^2} f\left(\alpha \frac{1+x}{1-x}\right) dx \\
&= 2\alpha^3 \int_{-1}^{1} \frac{(1+x)^2}{(1-x)^4} f\left(\alpha \frac{1+x}{1-x}\right) dx \\
&= 2\alpha^3 \sum_{i=1}^{n} \frac{w_i}{w(x_i)} \frac{(1+x_i)^2}{(1-x_i)^4} f\left(\alpha \frac{1+x_i}{1-x_i}\right)
\end{aligned} \tag{1.69}$$

Equation 1.69 can be simplified by using the information in Table 1.2, where $w_i = \frac{\pi}{n+1}\sin^2\left[\frac{i\pi}{(n+1)}\right] = \frac{\pi}{n+1}(1-x_i^2)$ and $w(x_i) = (1-x_i^2)^{1/2}$,

$$\int_0^\infty r^2 f(r)dr = 2\pi\alpha^3 \sum_{i=1}^{n} \frac{1}{(n+1)} \frac{(1+x_i)^{\frac{5}{2}}}{(1-x_i)^{\frac{7}{2}}} f(r_i) \tag{1.70}$$

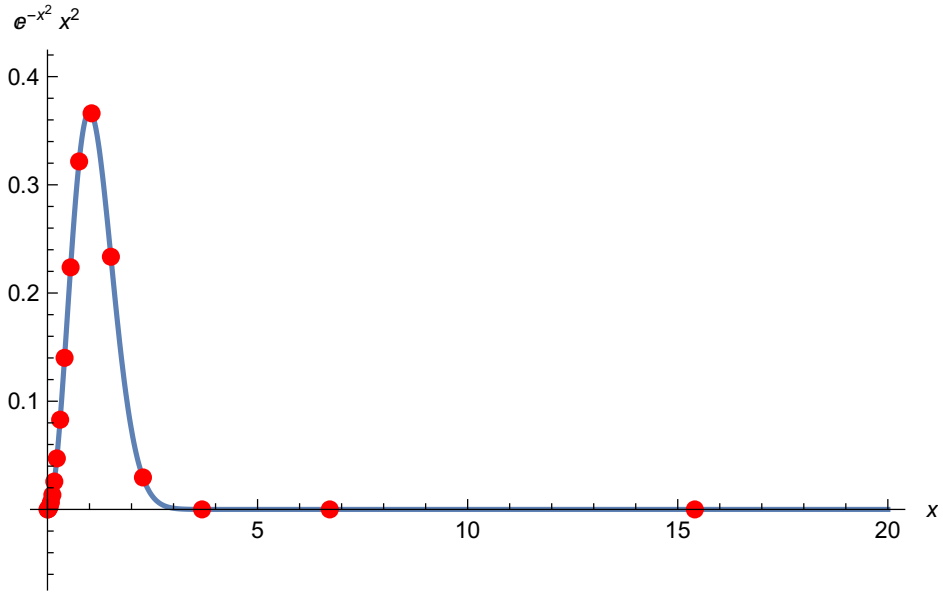
Becke assumed that parameter α is equal to half of the Bragg-Slater radius of the atom, except for hydrogen. He obtained accuracy in the order of 10^{-5} by using 110 angular grid points (see section 1.6.1.2 for more detail about the angular integration) and 20 radial grid points for calculating number of electrons in hydrogen. An extra 5 grid points were added for each additional atomic shell (i.e., 25 radial grid points for first-row atoms Li to Ne and 30 points for the second row atoms Na to Ar). Figure 1.6 shows that Becke grids have a large concentration near $r = 0$, which implies a significant waste of grid points [58].

Handy and Boys [59] approximated the Euler-Maclaurin formula within range $[0,1]$ (i.e., $a = 0$ and $b = 1$ in Table 1.1)

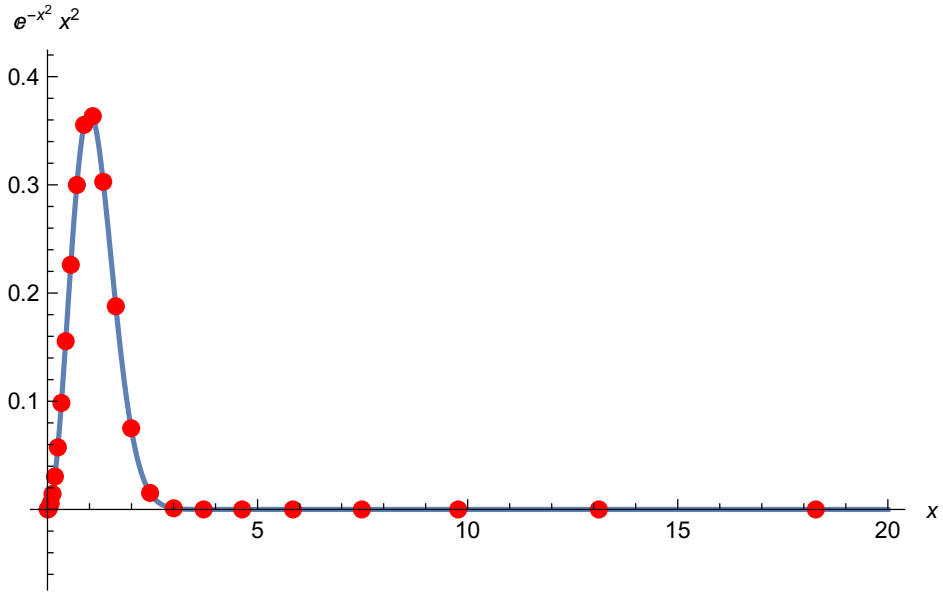
$$\int_0^1 f(q)dq = \frac{1}{n}\sum_{k=1}^{n-1} f(\frac{k}{n}) - \frac{1}{2n}\left(f(1)+f(0)\right) - \sum_{k=1}^{m} \frac{B_{2k}}{n^{2k}(2k)!}\left(f^{(2k-1)}(1) - f^{(2k-1)}(0)\right) \tag{1.71}$$

They showed that the integral $\int_0^1 f(q)dq$ can be approximated by using the transform coordinate formula $q = \alpha\left(\frac{x}{1-x}\right)^m$ to $\frac{1}{n}\sum_{k=1}^{n-1} f(\frac{k}{n})$. If the summation term is to n points then the formula becomes:

$$\int_0^1 f(q)dq = \frac{1}{n+1}\sum_{k=1}^{n} f\left(\frac{k}{n+1}\right) \tag{1.72}$$



(a) using 20 points and 0.35 scale



(b) using 30 points and 1.8 scale

Figure 1.6: Some examples for Becke scheme nodes of $x^2e^{-x^2}$ function.

They also showed that there is no significant improvement in the results when $m > 3$.

Murray et al. [60] used the Handy approximation for the Euler-Maclaurin formula (Equation 1.72) and mapped it to $[0,\infty)$ by using transform coordinate formula $r = \alpha\left(\frac{x}{1-x}\right)^2$ $(m = 2)$, $\left(\text{note: } dr = 2\alpha\frac{x}{(1-x)^3}dx\right)$,

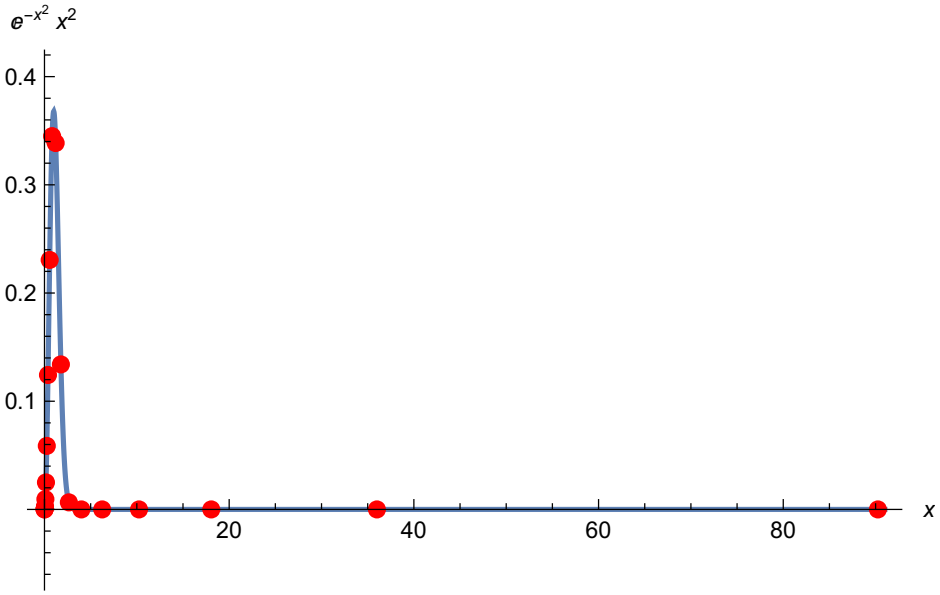
$$\begin{aligned}\int_0^\infty r^2 f(r)dr &= \int_0^1 \left(\alpha\left(\frac{x}{1-x}\right)^2\right)^2 (2\alpha)\frac{x}{(1-x)^3}f(r)dx \\ &= 2\alpha^3\int_0^1 \frac{x^5}{(1-x)^7}f(r)dx \\ &= 2\alpha^3\sum_{i=1}^{n}\frac{1}{(n+1)}\frac{{x_i}^5}{(1-x_i)^7}f(r_i) \qquad \text{where} \quad x_i = \frac{i}{n+1}\end{aligned} \tag{1.73}$$

Murray et al. assumed α as a scaling parameter depends on the Bragg-Slater atomic radii. They compared the accuracy for their calculations with Gauss-Laguerre points. They showed that the Euler-Maclaurin scheme is more accurate than the Gauss-Laguerre scheme. The Euler-Maclaurin scheme generates a number of points which are very far from the nucleus (Figure 1.7). These wasted points can be eliminated by dynamic pruning of the grid points [61].

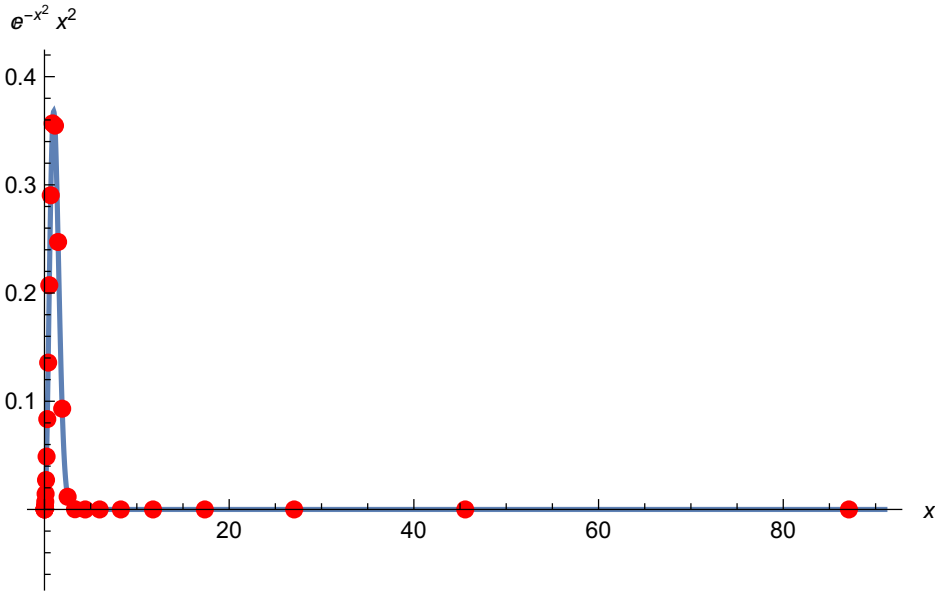
Treutler and Ahlrichs [58] used the Gauss-Chebyshev 2nd quadrature (Table 1.2) with transform coordinate formula (Equation 1.74) to find the integral $I(r)$ by mapping the interval of r from $[-1,1]$ to $[0,\infty)$.

$$r = \alpha\frac{(1+x)^\beta}{\ln 2}\ln\left(\frac{2}{1-x}\right) \tag{1.74}$$

The best performance was obtained at $\beta = 0.6$. The values of α for H to Kr atoms are given in reference [58]. By using the previous u-substitution integration technique (Equa-



(a) using 20 points



(b) using 30 points

Figure 1.7: Some examples for Euler-Maclaurin scheme nodes of $x^2e^{-x^2}$ function.

tion 1.64), they obtained

$$\int_0^\infty r^2 f(r)dr = \frac{\pi\alpha^3}{\ln^3 2}\sum_{i=1}^{n}\frac{(1+x_i)^{3\beta}}{n+1}\left[\sqrt{\frac{1+x_i}{1-x_i}}\ln^2\left(\frac{1-x_i}{2}\right)-\beta\sqrt{\frac{1-x_i}{1+x_i}}\ln^3\left(\frac{1-x_i}{2}\right)\right] \tag{1.75}$$

By comparing the Treutler and Ahlrichs with Becke grids, they proved that the new scheme is superior to the Becke scheme by more than one magnitude in accuracy [58].

Mura and Knowles [61] proposed another transform coordinate formula from $[0,1]$ to $[0,\infty)$ using the Euler-Maclaurin formula approximation (Equation 1.72). The new transform formula is

$$r = -\alpha ln(1-x^m) \qquad \text{where} \quad dr = \frac{m\alpha x^{m-1}}{1-x^m}dx \tag{1.76}$$

By combining Equation 1.72 and their transform coordinate formula (Equation 1.76) they got,

$$\begin{aligned}\int_0^\infty r^2 f(r)dr &= \int_0^1 (-\alpha ln(1-x^m))^2\frac{m\alpha x^{m-1}}{1-x^m}f(r)dx \\ &= m\alpha^3\int_0^1\frac{x^{m-1}ln^2(1-x^m)}{1-x^m}f(r)dx \\ &= m\alpha^3\sum_{i=1}^{n}\frac{x_i^{m-1}ln^2(1-x_i^m)}{(n+1)(1-x_i^m)}f(r_i) \qquad \text{where} \quad x_i = \frac{i}{n+1}\end{aligned} \tag{1.77}$$

where the values of α are given in Mura and Knowles paper [61]. The best performance was obtained when $m = 3$. The results are slightly better than the Treutler and Ahlrichs grids (Equation 1.75).

Gill and Chien [62] constructed their orthogonal polynomial for weight function $w(x) = ln^2(x)$ by using Gram-Schmidt process (Equation 1.78) on the range of $[0,1]$ starting from

sequence of $\{1,\ x,\ x^2,\ ...\}$

$$p_n = x^n - \sum_{j=0}^{n-1} \frac{\langle x^n, p_j(x)\rangle}{\langle p_j(x), p_j(x)\rangle} p_j(x) \qquad \text{where} \qquad \langle f(x), g(x)\rangle = \int_a^b w(x) f(x) g(x) dx \tag{1.78}$$

where the nodes x_i and the weights w_i for $n = 1-6,\ 8,\ 10,\ 15,\ 20$ are given in their paper [62]. They mapped the interval of r from $[0,1]$ to $[0,\infty)$ by using the transform coordinate formula $r = -\alpha lnx$ (note $dr = -\alpha x^{-1} dx$) as follows,

$$\begin{aligned} \int_0^\infty r^2 f(r) dr &= \int_1^0 -\alpha x^{-1} (-\alpha ln(x))^2 f(r) dx \\ &= \alpha^3 \int_0^1 ln^2(x) \frac{f(r)}{x} dx = \alpha^3 \sum_{i=1}^{n} \frac{w_i}{x_i} f(r_i) \end{aligned} \tag{1.79}$$

The new Gill and Chien scheme is called the "MultiExp quadrature".

1.6.1.2 Integration of the Angular Part

One of the most common angular integration quadrature (Equation 1.80) is the Lebedev quadrature [63–66].

$$I(\theta, \phi) = \int_0^{2\pi} \int_0^{\pi} \sin(\theta) f(\theta, \phi) d\theta d\phi \tag{1.80}$$

Lebedev developed a high performance angular quadrature. The grid points are constructed to lie on the surface of the unit sphere $S:\ x^2 + y^2 + z^2 = 1$. The integral can be written as

$$I(\theta, \phi) = 4\pi\, I(f) = \int_S f(s) ds \tag{1.81}$$

where s represents the Cartesian coordinates of vectors pointing to the surface of a unit sphere, $ds = \sin(\theta)d\theta d\phi$ and $I(1) = 1$. Lebedev used the quadrature scheme below to evaluate the integral $I(f)$.

$$I(f) = A_1 \sum_{i=1}^{6} f(a_i^1) + A_2 \sum_{i=1}^{12} f(a_i^2) + A_3 \sum_{i=1}^{8} f(a_i^3) + \sum_{k=1}^{N_1} B_k \sum_{i=1}^{24} f(b_i^k) + \sum_{k=1}^{N_2} C_k \sum_{i=1}^{24} f(c_i^k) + \sum_{k=1}^{N_3} D_k \sum_{i=1}^{48} f(d_i^k). \quad (1.82)$$

The weights $(A_1, A_2, A_3, B_k, C_k, D_k)$ and the base points $(a_i^1, a_i^2, a_i^3, b_i^k, c_i^k, d_i^k)$ are invariant with respect to the inversion of the octahedral group. Table 1.3 shows how the base points are represented geometrically on the surface of the unit sphere.

Table 1.3: The Lebedev grid points distribution on octahedral point group

	location on octahedron	coordinates (permutations)	constraint	number of points
a_i^1	vertices	$(\pm 1, 0, 0)$		6
a_i^2	mid-point of edges	$2^{-\frac{1}{2}}(\pm 1, \pm 1, 0)$		12
a_i^3	centres of faces	$3^{-\frac{1}{2}}(\pm 1, \pm 1, \pm 1)$		8
b_i^k	bisectors of faces	$(\pm l_k, \pm l_k, \pm m_k)$	$l_k = 2^{-\frac{1}{2}}(1 - m_k^2)^{\frac{1}{2}}$	24
c_i^k	edges of faces	$(\pm p_k, \pm q_k, 0)$	$q_k = (1 - p_k^2)^{\frac{1}{2}}$	24
d_i^k	general position	$(\pm r_k, \pm s_k, \pm t_k)$	$r_k^2 + s_k^2 + t_k^2 = 1$	48

1.6.2 Performance and Accuracy of Numerical Integrations

The performance and the accuracy of the integration quadrature scheme depends on the molecule of interest; the type of atoms, the type of bonds, the structure of the molecule or the desired property.

Gill et al. [67] suggested that the acceptance error in molecular energy should be ≈ 300 μhartree for a medium-sized molecule. They modified the standard grid (SG1) where different angular points were used on different radial points (i.e., partitioning the atom into regions from innermost to outermost spheres with 6, 38, 86, 194 and 86 angular points). This strategy is called "grid pruning". They showed that SG1 (which is about 3750 grid points per atom) yields acceptable grid errors compared with a benchmark grid that consists of 50 Euler-Maclaurin radial grids (Equation 1.73) and combined with 194 Lebedev angular grids (i.e., the total is 9700 grid points per atom).

Later, Chien and Gill [68] developed another standard grid (SG0) which is half of SG1 (i.e., SG0 is about 1500 grid points per atom) but somewhat less accurate than SG1. They chose the radial scheme (Equation 1.79) as a base for SG0 and they decided to select the angular grid independently for each radial point.

El-Sherbiny and Poirier [69] tested the MultiExp grids (Equation 1.79) in terms of accuracy and efficiency. They found that dividing the atomic space into three regions (core, middle, and outer) gave errors that are well within the acceptable error. They showed even for a small grid (6, 6, 8) with 6-86-194 angular grid that it performs well compared with the much larger SG1 grid, and is better than the Becke grid. Also, they found that dividing the atomic space of the Treutler scheme (Equation 1.75) into three regions improved the accuracy more than one order of magnitude even by using fewer grids for both the radial and the angular parts.

El-Sherbiny and Poirier [70] tested different numerical methods by computing different molecular properties. They found that SG1 and their new Treutler schemes gave the most accurate results. All tested grids were inaccurate in calculating the potential energy and there was inconsistent behaviour in Coulomb energy calculations.

1.6.3 Molecular Numerical Integration (Multi-Center Integration)

In molecules, the numerical integration of $f(\mathbf{r})$ is not straightforward. The popular solution for this involves partitioning of molecular space into cells or discrete regions, where the straightforward numerical integration can be carried out within each region. Bader et al. [40] have defined the atomic regions within the molecules on the basis of the topology of the electron density.

The molecular function, $f(\mathbf{r})$, can be decomposed into a sum of atomic functions, $f_a(\mathbf{r})$,

$$f(\mathbf{r}) = \sum_a f_a(\mathbf{r}), \qquad f_a(\mathbf{r}) = W_a(\mathbf{r}) f(\mathbf{r}) \tag{1.83}$$

Here W_a is the partition function that satisfies the conditions; $\sum_a W_a(\mathbf{r}) = 1$ and $W_a(\mathbf{r}) \geq 0$. The sum is over all atoms in the molecule. $W_a(\mathbf{r})$ has a value of unity near the nucleus a, but vanishes at other nuclei. The integral I in Equation 1.50 can be calculated as a sum of atomic integral contributions. Thus the problem is reduced from multicentre integration to the sum of single centre integrals (i.e., $I = \sum_{i=1}^{N} I_i$, where N is the number of atoms within the molecule).

Bibliography

[1] F. Jensen. *Introduction to Computational Chemistry*. John Wiley & Sons, Inc., 2017. ISBN 978-1-118-82599-0.

[2] J. A. Pople. Nobel Lecture: Quantum chemical models. *Reviews of Modern Physics*, 71:1267–1274, Oct 1999.

[3] M. Born and R. Oppenheimer. Zur Quantentheorie der Molekeln. *Annalen der Physik*, 389(20):457–484, 1927.

[4] A. J. Cohen, P. Mori-Snchez, and W. Yang. Challenges for Density Functional Theory. *Chemical Reviews*, 112(1):289–320, 2012.

[5] J.-L. Calais. Density Functional Theory of Atoms and Molecules. R.G. Parr and W. Yang, Oxford University Press, New York, Oxford, 1989. IX + 333 pp. *International Journal of Quantum Chemistry*, 47(1):101–101, 1993.

[6] P. Hohenberg and W. Kohn. Inhomogeneous Electron Gas. *Physical Review*, 136: B864–B871, Nov 1964.

[7] W. Kohn and L. J. Sham. Self-Consistent Equations Including Exchange and Correlation Effects. *Physical Review*, 140:A1133–A1138, 1965.

[8] E. S. Kryachko and E. V. Ludena. Density Functional Theory: Foundations Reviewed. *Physics Reports*, 544(2):123 – 239, 2014.

[9] W. Kohn, A. D. Becke, and R. G. Parr. Density Functional Theory of Electronic Structure. *The Journal of Physical Chemistry*, 100(31):12974–12980, 1996.

[10] P. A. M. Dirac. Note on Exchange Phenomena in the Thomas Atom. *Mathematical Proceedings of the Cambridge Philosophical Society*, 26(3):376–385, 1930.

[11] R. Parr and W. Yang. *Density-Functional Theory of Atoms and Molecules*. 1989. ISBN 978-0-19-509276-9.

[12] S. H. Vosko, L. Wilk, and M. Nusair. Accurate Spin-Dependent Electron Liquid Correlation Energies for Local Spin Density Calculations: A Critical Analysis. *Canadian Journal of Physics*, 58(8):1200–1211, 1980.

[13] J. P. Perdew and A. Zunger. Self-Interaction Correction to Density-Functional Approximations for Many-Electron Systems. *Physical Review B*, 23:5048–5079, May 1981.

[14] L. A. Cole and J. P. Perdew. Calculated Electron Affinities of the Elements. *Physical Review A*, 25:1265–1271, Mar 1982.

[15] J. P. Perdew and Y. Wang. Accurate and Simple Analytic Representation of the Electron-Gas Correlation Energy. *Physical Review B*, 45:13244–13249, Jun 1992.

[16] A. D. Becke. Density-Functional Exchange-Energy Approximation With Correct Asymptotic Behavior. *Physical Review A*, 38:3098–3100, Sep 1988.

[17] J. P. Perdew, J. A. Chevary, S. H. Vosko, K. A. Jackson, M. R. Pederson, D. J. Singh, and C. Fiolhais. Atoms, Molecules, Solids, and Surfaces: Applications of the Generalized Gradient Approximation for Exchange and Correlation. *Physical Review B*, 46:6671–6687, Sep 1992.

[18] J. P. Perdew, K. Burke, and M. Ernzerhof. Generalized Gradient Approximation Made Simple. *Physical Review Letters*, 77:3865–3868, Oct 1996.

[19] C. J. Cramer. *Essentials of Computational Chemistry Theories and Models*. Number 2. Second edition, 2004. ISBN 978-0-470-09182-1.

[20] A. D. Becke. Density Functional Thermochemistry. IV. A New Dynamical Correlation Functional and Implications for Exact Exchange Mixing. *The Journal of Chemical Physics*, 104(3):1040–1046, 1996.

[21] A. D. Becke. A New Inhomogeneity Parameter in Density-Functional theory. *The Journal of Chemical Physics*, 109(6):2092–2098, 1998.

[22] J. P. Perdew, S. Kurth, A. Zupan, and P. Blaha. Accurate Density Functional with Correct Formal Properties: A Step Beyond the Generalized Gradient Approximation. *Physical Review Letters*, 82:2544–2547, Mar 1999.

[23] C. Lee, W. Yang, and R. G. Parr. Development of the Colle-Salvetti Correlation-Energy Formula into a Functional of the Electron Density. *Physical Review B*, 37: 785–789, Jan 1988.

[24] A. D. Becke. A New Mixing of Hartree-Fock and Local Density Functional Theories. *The Journal of Chemical Physics*, 98(2):1372–1377, 1993.

[25] I. Mayer. Improved Chemical Energy Component Analysis. *Physical Chemistry Chemical Physics*, 14:337–344, 2012.

[26] R. Z. Khaliullin, E. A. Cobar, R. C. Lochan, A. T. Bell, and M. Head-Gordon. Unravelling the Origin of Intermolecular Interactions Using Absolutely Localized Molecular Orbitals. *The Journal of Physical Chemistry A*, 111(36):8753–8765, 2007.

[27] Y. Mo, P. Bao, and J. Gao. Energy Decomposition Analysis Based on a Block-Localized Wavefunction and Multistate Density Functional Theory. *Physical Chemistry Chemical Physics*, 13:6760–6775, 2011.

[28] E. D. Glendening and A. Streitwieser. Natural Energy Decomposition Analysis: An Energy Partitioning Procedure for Molecular Interactions With Application to Weak Hydrogen Bonding, Strong Ionic, and Moderate Donor-Acceptor Interactions. *The Journal of Chemical Physics*, 100(4):2900–2909, 1994.

[29] Y. Imamura, J. Suzuki, and H. Nakai. Kinetic Energy Decomposition Scheme Based on Information Theory. *Journal of Computational Chemistry*, 34(32):2787–2795, 2013.

[30] I. Mayer. Relation Between the Hilbert Space and "Fuzzy Atoms" Analyses. *Chemical Physics Letters*, 585:198–200, 2013.

[31] P. Salvador and I. Mayer. Energy Partitioning for Fuzzy Atoms. *The Journal of Chemical Physics*, 120(11):5046–5052, 2004.

[32] Y. Mao, P. R. Horn, and M. Head-Gordon. Energy Decomposition Analysis in an Adiabatic Picture. *Physical Chemistry Chemical Physics*, 19:5944–5958, 2017.

[33] K. Kitaura and K. Morokuma. A New Energy Decomposition Scheme for Molecular Interactions Within the Hartree-Fock Approximation. *International Journal of Quantum Chemistry*, 10(2):325–340, 1976.

[34] M. J. S. Phipps, T. Fox, C. S. Tautermann, and C.-K. Skylaris. Energy Decomposition Analysis Approaches and Their Evaluation on Prototypical Protein-Drug Interaction Patterns. *Chemical Society Reviews*, 44:3177–3211, 2015.

[35] R. S. Mulliken. Electronic Structures of Molecules XI. Electroaffinity, Molecular Orbitals and Dipole Moments. *The Journal of Chemical Physics*, 3(9):573–585, 1935.

[36] P. O. Löwdin. Approximate Formulas for Many Center Integrals in the Theory of Molecules and Crystals. *The Journal of Chemical Physics*, 21(2):374–375, 1953.

[37] A. E. Clark and E. R. Davidson. Population Analyses That Utilize Projection Operators. *International Journal of Quantum Chemistry*, 93(6):384–394, 2003.

[38] P. Salvador and E. Ramos-Cordoba. Communication: An Approximation to Bader's Topological Atom. *The Journal of Chemical Physics*, 139(7):071103, 2013.

[39] I. Mayer. *Bond Orders and Energy Components: Extracting Chemical Information From Molecular Wave Functions*. CRC press, 2016. ISBN 978-1-4200-9011-6.

[40] R. F. W. Bader, T.-T. Nguyen-Dang, and Y. Tal. A Topological Theory of Molecular Structure. *Reports on Progress in Physics*, 44(8):893, 1981.

[41] R. F. W. Bader. *Atoms in molecules : a quantum theory*. Number 22. John Wiley & Sons, Ltd, Chichester, UK, 1994. ISBN 978-0-19-855865-1.

[42] R. F. W. Bader. *The Quantum Mechanical Basis of Conceptual Chemistry*, volume 136. 2005.

[43] F. L. Hirshfeld. Bonded-Atom Fragments for Describing Molecular Charge Densities. *Theoretica Chimica Acta*, 44(2):129–138, 1977.

[44] P. Bultinck, C. V. Alsenoy, P. W. Ayers, and R. Carbó-Dorca. Critical Analysis and Extension of the Hirshfeld Atoms in Molecules. *The Journal of Chemical Physics*, 126(14):144111, 2007.

[45] T. A. Manz and D. S. Sholl. Chemically Meaningful Atomic Charges That Reproduce the Electrostatic Potential in Periodic and Nonperiodic Materials. *Journal of Chemical Theory and Computation*, 6(8):2455–2468, 2010.

[46] T. Verstraelen, P. W. Ayers, V. Van Speybroeck, and M. Waroquier. Hirshfeld-E Partitioning: AIM Charges with an Improved Trade-off between Robustness and Accurate Electrostatics. *Journal of Chemical Theory and Computation*, 9(5):2221–2225, 2013.

[47] T. A. Manz and D. S. Sholl. Improved Atoms-in-Molecule Charge Partitioning Functional for Simultaneously Reproducing the Electrostatic Potential and Chemical States in Periodic and Nonperiodic Materials. *Journal of Chemical Theory and Computation*, 8(8):2844–2867, 2012.

[48] D. E. P. Vanpoucke, P. Bultinck, and I. Van Driessche. Extending Hirshfeld-I to Bulk and Periodic Materials. *Journal of Computational Chemistry*, 34(5):405–417, 2013.

[49] A. D. Becke. A Multicenter Numerical Integration Scheme for Polyatomic Molecules. *The Journal of Chemical Physics*, 88(4):2547–2553, 1988.

[50] J. E. Besaw, P. L. Warburton, and R. A. Poirier. Atoms and Bonds in Molecules: Topology and Properties. *Theoretical Chemistry Accounts*, 134(10):1–15, 2015.

[51] P. L. Warburton, R. A. Poirier, and D. Nippard. Atoms and Bonds in Molecules from Radial Densities. *The Journal of Physical Chemistry A*, 115(5):852–867, 2011.

[52] M. Abramowitz, I. A. Stegun, and D. Miller. Handbook of Mathematical Functions With Formulas, Graphs and Mathematical Tables (National Bureau of Standards Applied Mathematics Series No. 55). *Journal of Applied Mechanics*, 32(1):239, 1965.

[53] G. Evans. *Practical Numerical Integration*. John Wiley & Sons Australia, Limited, 1993. ISBN 047193898X.

[54] V. I. Krylov. *Approximate Calculation of Integrals*. Dover Publications, 2006. ISBN 0486445798.

[55] H. V. Smith. *Numerical Methods of Integration*. Chartwell-Bratt, 1993. ISBN 0862383315.

[56] P. J. Davis and P. Rabinowitz. *Methods of Numerical Integration*. Academic Press, second edition, 1984. ISBN 978-0-12-206360-2.

[57] N. Kovvali. Theory and Applications of Gaussian Quadrature Methods. *Synthesis Lectures on Algorithms and Software in Engineering*, 3(2):1–65, 2011.

[58] O. Treutler and R. Ahlrichs. Efficient Molecular Numerical Integration Schemes. *The Journal of Chemical Physics*, 102(1):346–354, 1995.

[59] N. C. Handy and S. F. Boys. Integration Points for the Reduction of Boundary Conditions. *Theoretical Chemistry Accounts*, 31:195–200, 09 1973.

[60] C. W. Murray, N. C. Handy, and G. J. Laming. Quadrature Schemes for Integrals of Density Functional Theory. *Molecular Physics*, 78(4):997–1014, 1993.

[61] M. E. Mura and P. J. Knowles. Improved Radial Grids for Quadrature in Molecular Density Functional Calculations. *The Journal of Chemical Physics*, 104(24):9848–9858, 1996.

[62] P. M. W. Gill and S.-H. Chien. Radial Quadrature for Multiexponential Integrands. *Journal of Computational Chemistry*, 24(6):732–740, 2003.

[63] V. I. Lebedev. Values of the Nodes and Weights of Ninth to Seventeenth Order Gauss-Markov Quadrature Formulae Invariant Under the Octahedron Group With Inversion. *USSR Computational Mathematics and Mathematical Physics*, 15(1):44 – 51, 1975.

[64] V. I. Lebedev. Quadratures on a sphere. *USSR Computational Mathematics and Mathematical Physics*, 16(2):10 – 24, 1976.

[65] V. I. Lebedev. Spherical Quadrature Formulas Exact to Orders 25–29. *Siberian Mathematical Journal*, 18(1):99–107, Jan 1977.

[66] V. I. Lebedev and D. N. Laikov. A Quadrature Formula for the Sphere of the 131st Algebraic Order of Accuracy. *Doklady Mathematics*, 59:477–481, 1999.

[67] P. M. W. Gill, B. G. Johnson, and J. A. Pople. A Standard Grid for Density Functional Calculations. *Chemical Physics Letters*, 209(5):506 – 512, 1993.

[68] S.-H. Chien and P. M. W. Gill. SG-0: A Small Standard Grid for DFT Quadrature on Large Systems. *Journal of Computational Chemistry*, 27(6):730–739, 2006.

[69] A. El-Sherbiny and R. A. Poirier. An Evaluation of the Radial Part of Numerical Integration Commonly Used in DFT. *Journal of Computational Chemistry*, 25(11): 1378–1384, 2004.

[70] A. A. El-Sherbiny and R. A. Poirier. Comprehensive Study of Some Well-Known Molecular Numerical Integration Methods. *Canadian Journal of Chemistry*, 87(10): 1313–1321, 2009.

Chapter 2

Proposition of New-Partitioning Weights

"If you think you understand quantum mechanics, you don't understand quantum mechanics."

— Richard Feynman

2.1 New-Partitioning Weights

In this chapter the development of our three new partitioning models will be explained. These models are Fermi-Dirac, triangle, and Awad. The goal is to develop new weights in which the core electrons close to a nucleus of an atom should be assigned to that atom and not to the core or the bond of other atoms.

2.1.1 Model One: Fermi-Dirac Weight

The Fermi-Dirac distribution function $f(\varepsilon)$ gives the probability of occupancy of energy levels by fermions, i.e., particles that have half-integer spin and behave in accordance to the Pauli exclusion principle,

$$f(\varepsilon) = \frac{1}{e^{(\varepsilon-\mu)/k_B T} + 1} \tag{2.1}$$

where μ is the chemical potential, ε is the Fermi energy that is the energy of the top most filled level at 0 K, k_B is the Boltzmann constant which equals 1.38×10^{-23} J·K^{-1}, and T is the temperature in Kelvin. At absolute zero, the probability is equal to one for energies less than or equal to the Fermi energy ε and zero for energies greater than the Fermi energy. The chemical potential μ equals the Fermi energy at $T = 0$ K.

Figure 2.2 is a schematic representation of two atoms A and B in a molecule. In this figure, A denotes the atom of interest and B denotes the other atoms in the molecule. Green and orange circles are the cores of atoms A and B, respectively. r_0^A and r_0^B are the radii of the cores of atoms A and B, respectively. r_i^A and r_i^B are the distances between the grid point (i) and nuclei of atoms A and B, respectively. R_{AB} is the distance between A and B nuclei.

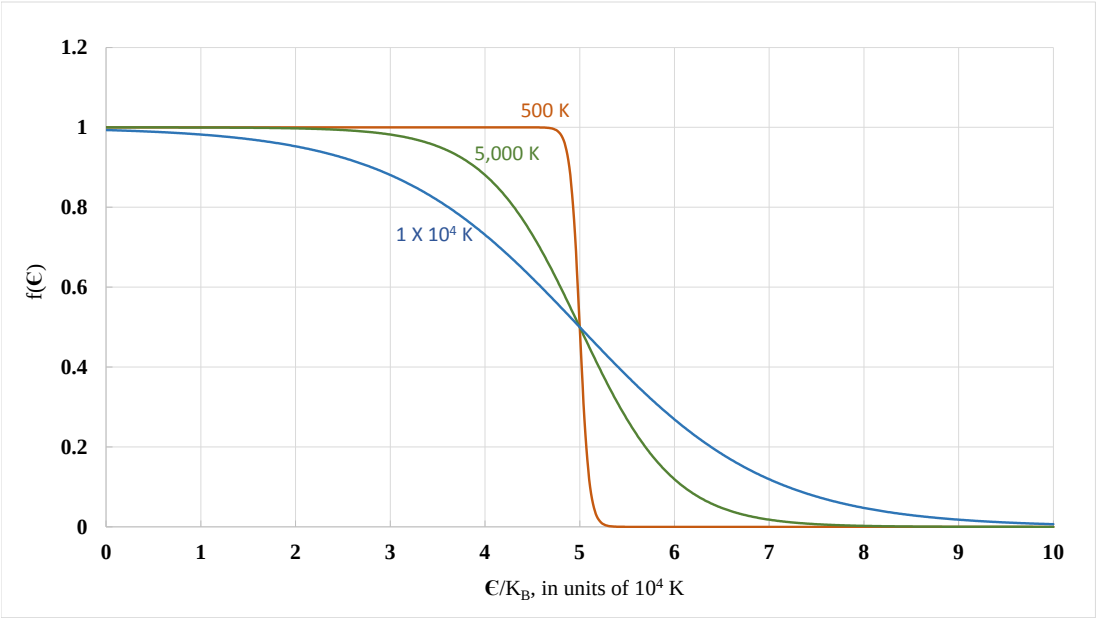


Figure 2.1: Fermi-Dirac distribution function (Equation 2.1) at the various labeled temperatures. The total number of particles is a constant and independent of temperature [1].

For simplicity, atom B is shifted to a new position in which coordinates of the grid point (i) and nuclei A and B lay on a straight line. The distance r_i^B must remain the same before and after the shifting. β^{AB} is the distance between the outer surfaces for cores of atoms A and B after shifting. It is clear from Figure 2.2 that β^{AB} can be written as,

$$\beta^{AB} = r_i^A + r_i^B - r_0^A - r_0^B \tag{2.2}$$

The derivation of our new weight (model one) starts with an expression similar to the Fermi-Dirac distribution function (Equation 2.1), i.e.,

$$w^{AB}(r_i^A) = \frac{1}{e^{\alpha(r_i^A - v^{AB})^n} + 1} \tag{2.3}$$

Here $w^{AB}(r_i^A)$ is the our partitioning weight for the atom A at specific radial grid point (i), r_i^A is the distance between the nucleus of atom A and the grid point i, α is a real constant, n is an integer number, and ν^{AB} is the distance between the nucleus of atom A and the half distance between the core of atom A and the core of other atoms B after shifting

$$\nu^{AB} = r_0^A + \frac{\beta^{AB}}{2} \tag{2.4}$$

By substituting ν^{AB} into the Equation 2.3, we get

$$w^{AB}(r_i^A) = \frac{1}{e^{\alpha\left(r_i^A - \left(r_0^A + \frac{\beta^{AB}}{2}\right)\right)^n} + 1} \tag{2.5}$$

Figure 2.3 shows the partitioning weight of atom A as a function of the distance from the nucleus A along the AB axis. In this figure, $w^{AB}(r_0^A)$ and $w^{AB}(r_0^A + \beta^{AB})$ are the weights of atom A on the surfaces of the cores for atoms A and B, respectively.

The different between $w^{AB}(r_0^A)$ and $w^{AB}(r_0^A + \beta^{AB})$ is defined as follows,

$$C = w^{AB}(r_0^A) - w^{AB}(r_0^A + \beta^{AB}) \qquad \text{should be equal to one} \tag{2.6}$$

In order to define a correct partitioning weight function, the following conditions must be met $w^{AB}(r_0^A) = 1$, $w^{AB}(r_0^A + \beta^{AB}) = 0$ and $C = 1$. Using Equation 2.5, the weight of atom

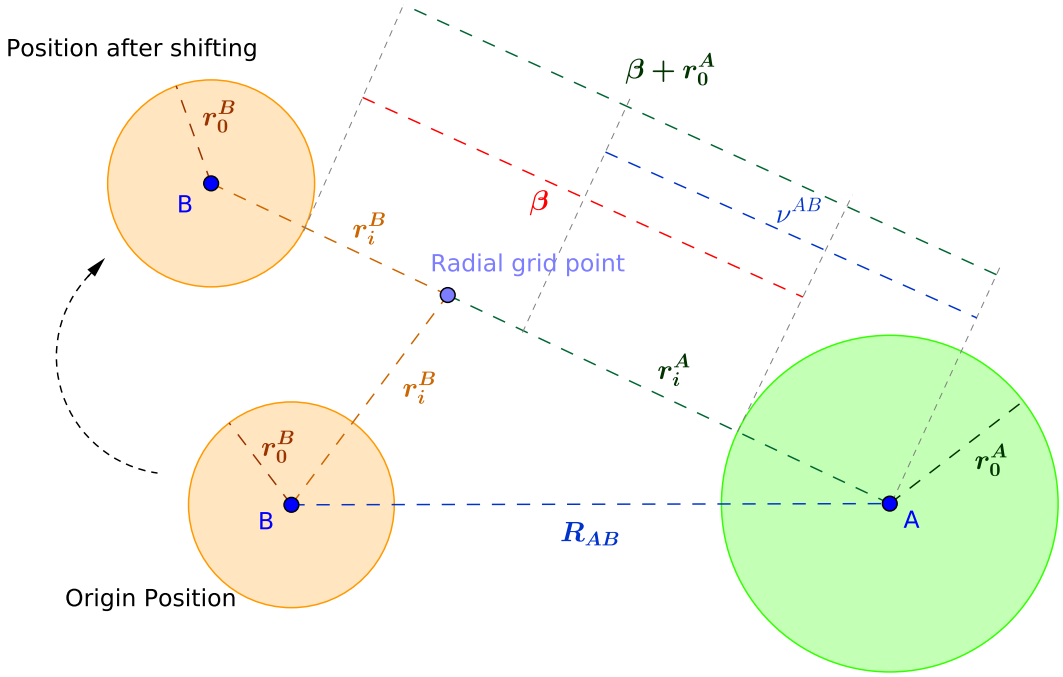


Figure 2.2: Fermi-Dirac model (model one), atom A represents the atom of interest and atom B represents the other atoms. The green and the orange circles are the cores of atoms A and B respectively.

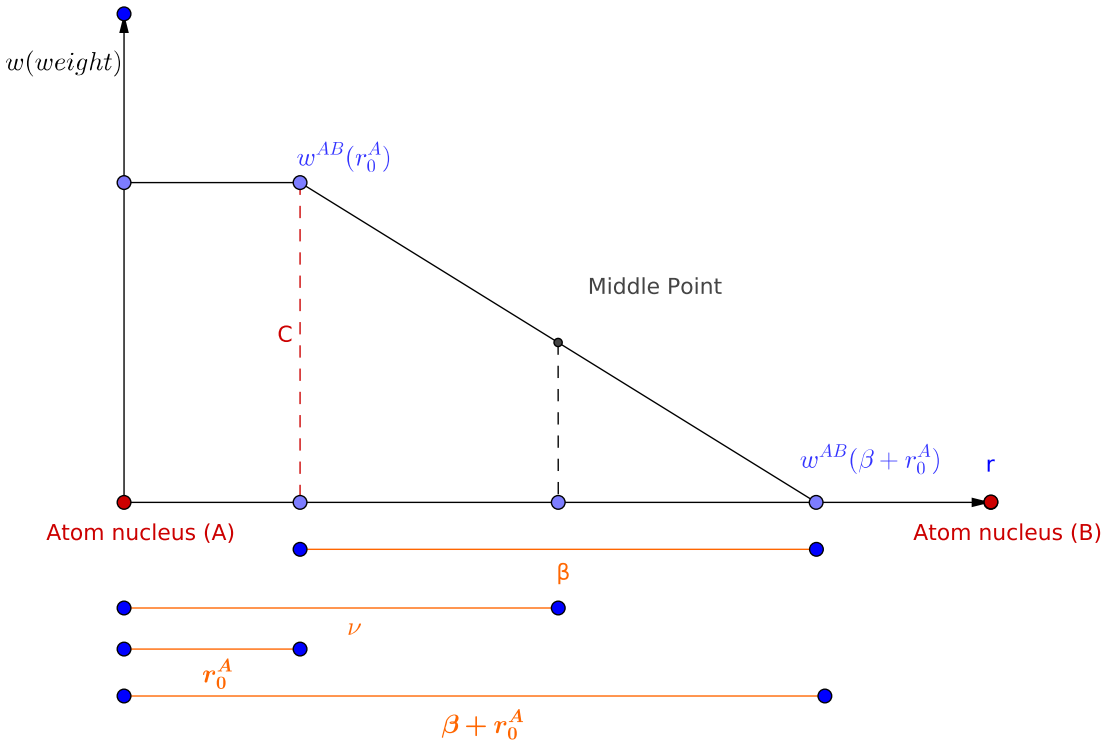


Figure 2.3: Fermi-Dirac model (model one).

A on its core's surface (i.e., at position r_0^A) is,

$$w^{AB}(r_0^A) = \frac{1}{e^{\alpha\left(r_0^A - \left(r_0^A + \frac{\beta^{AB}}{2}\right)\right)^n} + 1} = \frac{1}{e^{(-1)^n \alpha\left(\frac{\beta^{AB}}{2}\right)^n} + 1} \tag{2.7}$$

and the weight of atom A on the core's surface of atom B (i.e. at position $r_0^A + \beta^{AB}$) is,

$$w^{AB}(r_0^A + \beta^{AB}) = \frac{1}{e^{\alpha\left(\frac{\beta^{AB}}{2}\right)^n} + 1} \tag{2.8}$$

The substitution of Equations 2.7 and 2.8 into Equation 2.6 leads to,

$$C = \frac{1}{e^{(-1)^n \alpha \left(\frac{\beta^{AB}}{2}\right)^n} + 1} - \frac{1}{e^{\alpha \left(\frac{\beta^{AB}}{2}\right)^n} + 1} \tag{2.9}$$

It is clear that if n is an even number then $C = 0$, whereas if n is an odd value C can be computed as follows:

$$\begin{aligned} C &= \frac{1}{e^{-\alpha \left(\frac{\beta^{AB}}{2}\right)^n} + 1} - \frac{1}{e^{\alpha \left(\frac{\beta^{AB}}{2}\right)^n} + 1} \qquad n \text{ is an odd number} \\ &= \frac{e^{\alpha \left(\frac{\beta^{AB}}{2}\right)^n} - 1}{e^{\alpha \left(\frac{\beta^{AB}}{2}\right)^n} + 1} \end{aligned} \tag{2.10}$$

From the above equation, α is,

$$\begin{aligned} C\left(e^{\alpha \left(\frac{\beta^{AB}}{2}\right)^n} + 1\right) &= e^{\alpha \left(\frac{\beta^{AB}}{2}\right)^n} - 1 \\ Ce^{\alpha \left(\frac{\beta^{AB}}{2}\right)^n} + C &= e^{\alpha \left(\frac{\beta^{AB}}{2}\right)^n} - 1 \\ e^{\alpha \left(\frac{\beta^{AB}}{2}\right)^n} - Ce^{\alpha \left(\frac{\beta^{AB}}{2}\right)^n} &= 1 + C \\ e^{\alpha \left(\frac{\beta^{AB}}{2}\right)^n}(1 - C) &= 1 + C \\ \alpha &= \left(\frac{2}{\beta^{AB}}\right)^n ln\left(\frac{1+C}{1-C}\right) \end{aligned} \tag{2.11}$$

As mentioned before the variable C must be one. However, from Equation 2.11 C can not equal one, and so we assume C is a number very close to one.

For simplicity, α can be written as,

$$\alpha = k\left(\frac{2}{\beta^{AB}}\right)^n \qquad where \qquad k = ln\left(\frac{1+C}{1-C}\right) \tag{2.12}$$

Substituting α into Equation 2.5 gives

$$w^{AB}(r_i^A) = \frac{1}{e^{k\left(\frac{2(r_i^A - r_0^A)}{\beta^{AB}} - 1\right)^n} + 1} \tag{2.13}$$

Because n must be an odd number and for simplicity through this thesis, we will consider $n = 1$. Note that all variables in Equation 2.13 are given or easily calculated. Equation 2.13 is the final form of the partitioning weight for atom A. In order to determine the amount of space owned by atom A in the entire molecule, the following pairwise ownership has been defined between atom A and all other atoms.

$$w^A(\mathbf{r}) = \prod_{B=1, B\neq A}^{N} w^{AB}(r_i^A) \tag{2.14}$$

The normalized weight function $W^A(\mathbf{r})$ for an atom A at a particular point $\mathbf{r}$ can now be defined as follow.

$$W^A(\mathbf{r}) = \frac{w^A(\mathbf{r})}{\sum_n w^n(\mathbf{r})} \tag{2.15}$$

2.1.2 Model Two: Triangle Weight

Figure 2.4 is a schematic representation of two atoms A and B in a molecule, where A denotes the atom of interest and B denotes the other atoms in the molecule. If we let a radial grid point i lie between these two atoms, then we can define the following partitioning ratio

Figure 2.4: Model two; first assumption.

μ_i^{AB},

$$\mu_i^{AB} = \frac{r_i^B - r_i^A}{r_i^B + r_i^A} \tag{2.16}$$

where r_i^A and r_i^B are the distances between the grid point i and the nuclei of A and B, respectively. Note that when i lies on the nucleus of atom A (i.e., $r_i^A \to 0$) then $\mu_i^{AB} \to +1$ and when i lies on the nucleus of atom B (i.e., $r_{Ai}^B \to 0$) then $\mu_i^{AB} \to -1$. If i is not on a nucleus then μ_i^{AB} takes values between -1 and $+1$. The new partitioning weight w_i^{AB} can be defined using μ_i^{AB} as follows,

$$\begin{aligned} w_i^{AB} &= \frac{1}{2}\left(\mu_i^{AB} + 1\right) \\ &= \frac{1}{2}\left(\frac{r_i^B - r_i^A}{r_i^B + r_i^A} + 1\right) \end{aligned} \tag{2.17}$$

when $r_i^A \to 0$ then $w_i^A \to +1$, and when $r_i^B \to 0$ then $w_i^A \to 0$. If the radial grid point is not on a nucleus then w_i^{AB} takes values between 0 and $+1$.

μ_i^{AB} and w_i^{AB} can be manipulated by replacing the values of r_i^A and r_i^B in Equation 2.17 with $(r_i^A - r_0^A)$ and $(r_i^B - r_0^B)$, respectively. Here r_0^A and r_0^B are the atomic core radii of A and B, respectively.

$$\mu_i^{AB} = \frac{(r_i^B - r_0^B) - (r_i^A - r_0^A)}{(r_i^B - r_0^B) + (r_i^A - r_0^A)} \tag{2.18}$$

$$w_i^{AB} = \frac{1}{2}\left(\frac{(r_i^B - r_0^B) - (r_i^A - r_0^A)}{(r_i^B - r_0^B) + (r_i^A - r_0^A)} + 1\right) \tag{2.19}$$

In this case we assume that the atomic cores have spherical shapes. When i lies on the core surface of atom A (i.e., $r_i^A \rightarrow r_0^A$) then $w_i^{AB} \rightarrow +1$, and when i lies on the core surface of other atoms B (i.e., $r_i^B \rightarrow r_0^B$) then $w_i^{AB} \rightarrow 0$.

In order to make the weight w_i^{AB} of A equals to one within its core and equal to zero within the core of B, Equation 2.19 can be modified by utilizing the absolute value as follows,

$$w_i^{AB} = \frac{1}{2}\left(\frac{\left|r_i^B - r_0^B\right| - \left|r_i^A - r_0^A\right|}{(r_i^B - r_0^B) + (r_i^A - r_0^A)} + 1\right) \tag{2.20}$$

All variables in Equation 2.20 are given or easily calculated. Once again, in order to determine the amount of space owned by particular atom, A, in the entire molecule, the following pairwise ownership has been used,

$$w^A(\mathbf{r}) = \prod_{B=1, B\neq A}^{N} w_i^{AB}(\mathbf{r}) \tag{2.21}$$

The normalized weight function $W^A(\mathbf{r})$ for an atom A at a particular point $\mathbf{r}$ can now be defined as,

$$W^A(\mathbf{r}) = \frac{w^A(\mathbf{r})}{\sum_n w^n(\mathbf{r})} \tag{2.22}$$

2.1.3 Model Three: Awad Weight

In this model, we use the same principle that is used in model two (subsection 2.1.2). However, the denominator of Equation 2.18 is modified as follows

$$\mu_i^{AB} = \frac{(r_i^B - r_0^B) - (r_i^A - r_0^A)}{d} \tag{2.23}$$

where d is the distance between points C and F in Figure 2.5. The points C and F represent the intersection of lines that connect the atomic nuclei and the grid point i with the atomic core surfaces of A and B, respectively. In order to derive an expression for d, we apply the

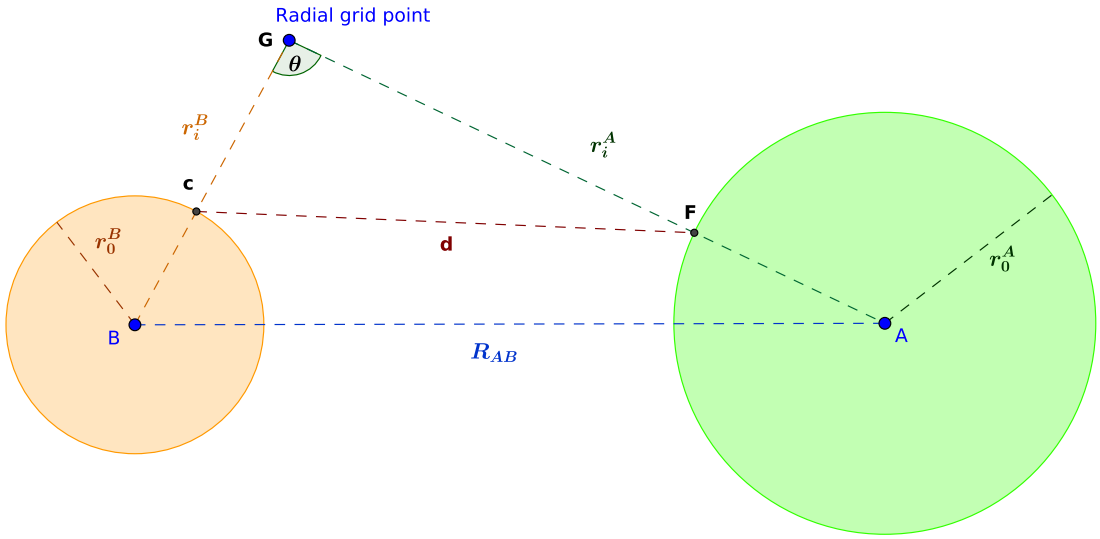
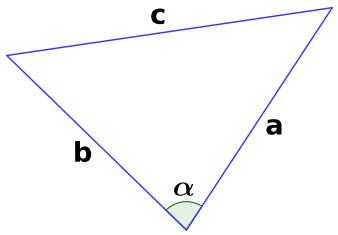


Figure 2.5: Awad model (model three).

cosine rule,

$$\cos(\alpha) = \frac{a^2 + b^2 - c^2}{2ab} \tag{2.24}$$



As shown in Figure 2.5, triangles GAB and GFC share the angle θ. By applying the cosine rule (Equation 2.24) on GAB and GFC, we get

$$\cos(\theta) = \frac{(r_i^A - r_0^A)^2 + (r_i^B - r_0^B)^2 - d^2}{2(r_i^A - r_0^A)(r_i^B - r_0^B)} = \frac{{r_i^A}^2 + {r_i^B}^2 - {R_{AB}}^2}{2 r_i^A r_i^B} \tag{2.25}$$

The rearrangement of Equation 2.25 gives,

$$d^2 = (r_i^A - r_0^A)^2 + (r_i^B - r_0^B)^2 - \frac{({r_i^A}^2 + {r_i^B}^2 - {R_{AB}}^2)(r_i^A - r_0^A)(r_i^B - r_0^B)}{r_i^A r_i^B} \tag{2.26}$$

Substitution of d in Equation 2.23 gives,

$$\mu_i^{AB} = \frac{(r_i^B - r_0^B) - (r_i^A - r_0^A)}{\left[(r_i^A - r_0^A)^2 + (r_i^B - r_0^B)^2 - ({r_i^A}^2 + {r_i^B}^2 - {R_{AB}}^2)\frac{(r_i^A - r_0^A)}{r_i^A}\frac{(r_i^B - r_0^B)}{r_i^B}\right]^{\frac{1}{2}}} \tag{2.27}$$

Note that when i lies on the core surface of atom A (i.e., $r_i^A \to r_0^A$) then $\mu_i^{AB} \to +1$ and when i lies on the core surface of atom B (i.e., $r_i^B \to r_0^B$) then $\mu_i^{AB} \to -1$. It should be noted that Equation 2.27 is invalid within the cores of atoms A and B. However, μ_i^{AB} outside the core regions is a linear function. To replace μ_i^{AB} by a smooth function, one can benefit from the sine function graph (shown in Figure 2.6).

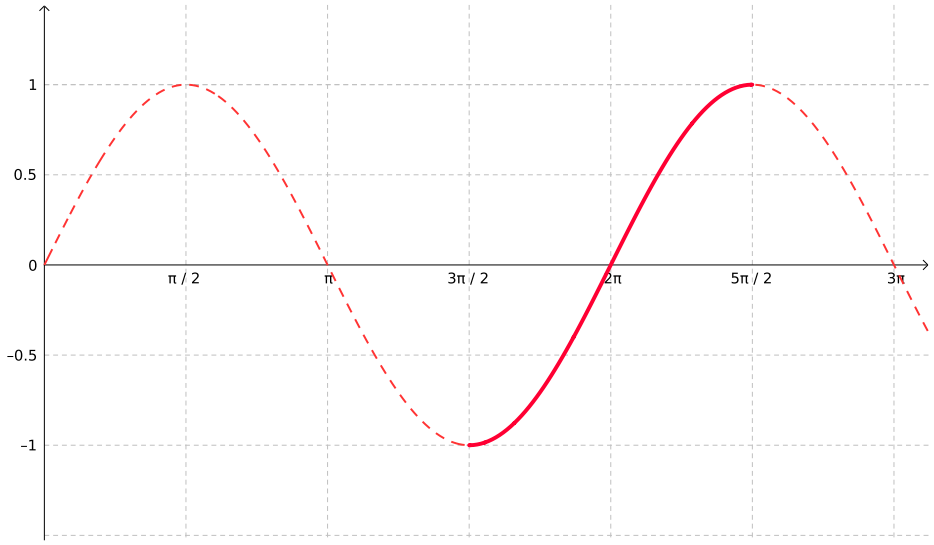


Figure 2.6: Graph of the sine function sin(x).

Using Taylor expansion for the sine function of x in radians,

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots \tag{2.28}$$

Our aim is to use the sine function shape in $[3\pi/2,\ 5\pi/2]$ (solid line in Figure 2.6). This shape can be obtained from using the first two terms in the expansion,

$$h(x) = ax - bx^3 \tag{2.29}$$

where $h(x)$ is the truncated Taylor expansion that has the desired shape with unknown coefficients a and b. To derive the desired function, the conditions: $h(1) = 1$ and $h(-1) = -1$,

$$\begin{aligned} h(\pm 1) &= a(\pm 1) - b(\pm 1)^3 = \pm 1 \\ \Rightarrow a - b &= 1 \end{aligned} \tag{2.30}$$

and the value of $h'(\pm 1) = 0$ should be held,

$$\begin{aligned} h'(\pm 1) &= a - 3b(\pm 1)^2 = 0 \\ \Rightarrow a &= 3b \end{aligned} \tag{2.31}$$

from Equations 2.30 and 2.31, $a = \frac{3}{2}$ and $b = \frac{1}{2}$, and $h(x)$ becomes,

$$h(x) = 1.5x - 0.5x^3 \tag{2.32}$$

Equation 2.32 represents the maximum smoothing for $h(x)$, whereas when $b = 0$ Equation 2.29 becomes $h(x) = x$ and represents the minimum smoothing. Therefore, we can control the steepness of the curve by changing the value of b from 0 to $\frac{1}{2}$, or mathematically,

$$h(x) = (b+1)x - bx^3 \qquad \text{where} \qquad 0 \leq b \leq \frac{1}{2} \tag{2.33}$$

The substitution of μ_i^{AB} in Equation 2.33 leads to,

$$h(\mu_i^{AB}) = (b+1)\mu_i^{AB} - (b)\mu_i^{AB^3} \tag{2.34}$$

This function $h(\mu_i^{AB})$ is still not smooth enough. Thus $h(\mu_{Bi}^{A})$ is iterated three times (i.e., same as Becke) to obtain the desired smoothness.

$$\begin{aligned} f_1(\mu_i^{AB}) &= h(\mu_i^{AB}) \\ f_2(\mu_i^{AB}) &= h\left[h(\mu_i^{AB})\right] \\ f_3(\mu_i^{AB}) &= h\{h\left[h(\mu_i^{AB})\right]\} \end{aligned} \tag{2.35}$$

The outcome of above three iterations is illustrated in Figure 2.7.

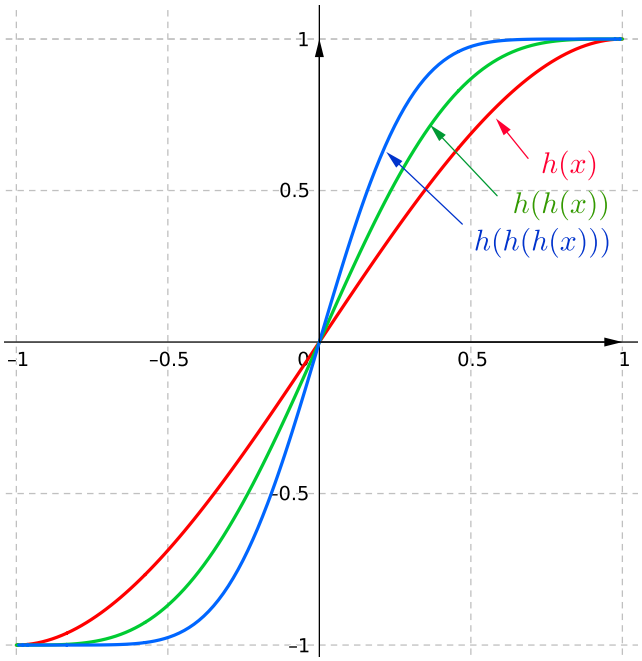


Figure 2.7: Profiles of functions $f_1(\mu_i^{AB})$, $f_2(\mu_i^{AB})$, and $f_3(\mu_i^{AB})$.

The Awad weight w_i^{AB} can be defined using $f_3(\mu_i^{AB})$ as follows,

$$w_i^{AB} = \frac{1}{2}\left[f_3(\mu_i^{AB}) + 1\right] \tag{2.36}$$

Once again, in order to determine the amount of space owned by particular atom, A, in the entire molecule, the following pairwise ownership has been used,

$$w^A(\mathbf{r}) = \prod_{B=1, B\neq A}^{N} w_i^{AB}(\mathbf{r}) \tag{2.37}$$

The normalized weight function $W^A(\mathbf{r})$ for an atom A at a particular point $\mathbf{r}$ can now be defined as,

$$W^A(\mathbf{r}) = \frac{w^A(\mathbf{r})}{\sum_n w^n(\mathbf{r})} \tag{2.38}$$

Bibliography

[1] H. Ibach and H. Lüth. *Solid-State Physics: An Introduction to Principles of Materials Science*, volume 27. Springer-Verlag Berlin Heidelberg, 2009. ISBN 9783540938033.

Chapter 3

New-Partitioning Weights Calculations and Results

"Einstein said that if quantum mechanics were correct then the world would be crazy. Einstein was right - the world is crazy."

— Daniel M. Greenberger

3.1 Computational Method

All the calculations were performed using the MUNgauss package [1]. The results were calculated at the HF/6-31G(d)//HF/6-31G(d) level unless otherwise specified. The atomic radial densities for the isolated atoms from hydrogen to argon are computed at HF/6-311++G(d,p) and employ a mesh size of 0.00001 bohr. The results of numerical integration were calculated using the 6-31G(d) basis sets on HF/6-31G(d) optimized structures, and the benchmark grid is used in the calculations to decrease the error that comes from the grid. All of the visual aids, including contour, relief, and the 3D plots were created using Mathematica Version 11.2 graphing package [2].

Throughout this thesis, the length is given in bohr, radial density ρ_{rad} is given in $e/bohr$ and electron density is in $e/bohr^3$, unless otherwise specified.

3.2 Results of the New Partitioning Weight Models

The derivation of three new partitioning weight models (Fermi-Dirac, triangle, and Awad) are shown in the previous chapter. In this chapter, we will compute some of molecular properties using the new partitioning weight models. The visualization of radial and bond electron densities using these models will be discussed, then the molecular properties will be calculated numerically using the new partitioning weight models. The results will be compared with other popular weight schemes such as the Becke weight.

3.2.1 Suggested Cores Sizes

As mentioned in chapter 2, the Fermi-Dirac, triangle, and Awad models (for example Equation 2.27) require the atomic core sizes. The question is, what are the atomic core sizes to be used in these models. There is an enormous number of definition for atomic radii, such as a covalent, ionic, metallic, and Bragee slater radii. Because the new weights are developed for molecular radial electron density, the atomic radial densities for the isolated atoms from hydrogen to argon are computed at HF/6-311++G(d,p). The minimum radial density, maximum radial density and the last core maximum were found using Mathematica's FindMaximum and FindMinimum functions, and have an accuracy of ± 0.00001 bohr based on using a mesh size of 0.00001 bohr between points. For example, the molecular radial electron density (RDEN) for Cl atom is shown in Figure 3.1, where r_{max} is the last core maximum, $r_{lastmax}$ is the last maximum and r_{min} is the last minimum. In addition, the expectation value for the radii of the last core atomic shells $r_{\langle r \rangle}$ of the isolated atoms from hydrogen to argon were calculated numerically using SG1 radial grid at HF/6-311++G(d,p). Table 3.1 shows values of r_{min}, r_{max}, $r_{lastmax}$, and $r_{\langle r \rangle}$ for isolated atoms

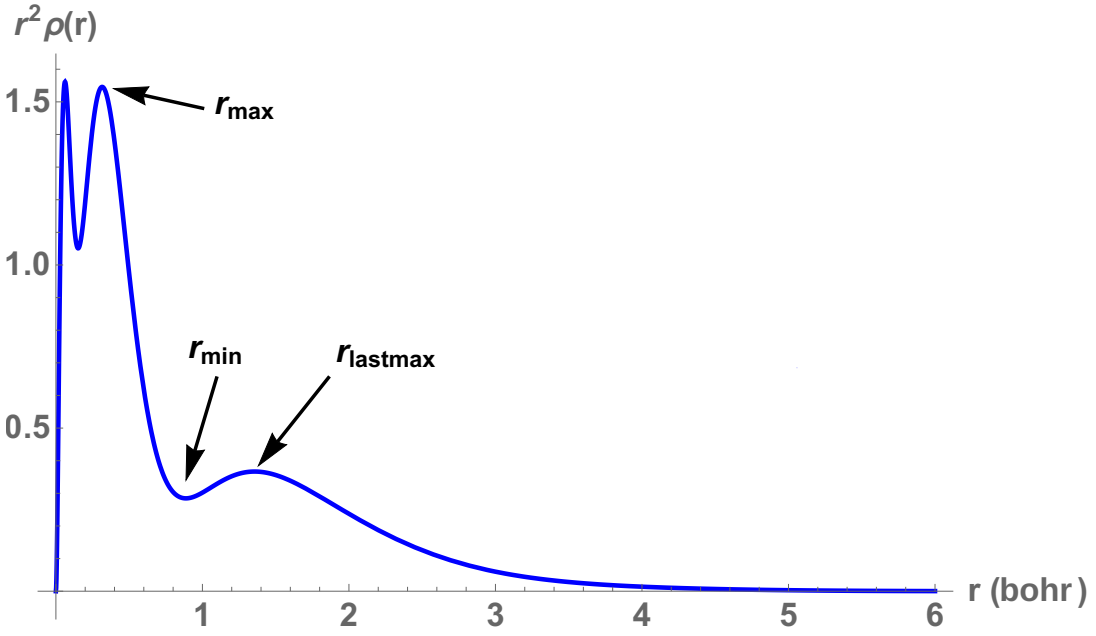


Figure 3.1: Radial electron density for Cl atom using HF/6-311++G(d,p). r_{max} is the last core maximum, $r_{lastmax}$ is the last maximum and r_{min} is the last minimum.

from hydrogen to argon in bohr. It is clear from Table 3.1 that $r_{lastmax}$ cannot be used to define atomic cores as their values are so large. For example, the bond length of the LiCl molecule is about 3.849 bohr, where the $r_{lastmax}$ of Li is 3.081 bohr and $r_{lastmax}$ of Cl is 1.356 bohr; their sum is larger than bond length of LiCl. It should be noted that differences between cores in metal atoms (e.g., Li and Na) are larger compared to non-metal atoms (e.g., F and S). Therefore, in molecules containing metal atoms the choice of the core is of great importance as it affects significantly both RDEN and bond electron density (BDEN) of atoms in the molecule.

3.2.2 Visualization of the New Weights

In this section, some of the advantages and the drawbacks of Fermi-Dirac, triangle, and Awad models will be illustrated visually. The weight of the C atom within the CO molecule

Table 3.1: The cores values (bohr), minimum radial density (r_{min}), maximum radial density (r_{max}), the last core maximum ($r_{lastmax}$), and the average radial value of the last core shell ($r_{\langle r \rangle}$) of isolated atoms calculated using HF/6-311++G(d,p)

	r_{min}	r_{max}	$r_{lastmax}$	$r_{\langle r \rangle}$
H	0.00000	0.00000	1.00784	0.00000
He	0.00000	0.00000	0.57697	0.00000
Li	1.77050	0.36501	3.08074	1.14660
Be	1.09369	0.26629	2.05757	0.83013
B	0.79765	0.21031	1.53395	0.65175
C	0.61976	0.17398	1.21493	0.53690
N	0.50088	0.14875	1.00110	0.45661
O	0.41737	0.13009	0.84941	0.39719
F	0.35450	0.11581	0.73676	0.35151
Ne	0.30578	0.10451	0.65015	0.31527
Na	3.40265	0.57464		1.59791
Mg	2.10027	0.50602	2.42357	1.38154
Al	1.63344	0.45278	2.12579	1.24024
Si	1.34635	0.40929	1.87480	1.12594
P	1.14513	0.37240	1.66778	1.03142
S	0.99901	0.34217	1.48923	0.95160
Cl	0.88750	0.31578	1.35608	0.88353
Ar	0.79563	0.29320	1.23988	0.82465

along the bond using $r_{\langle r \rangle}$ as cores for C and O atoms will be used as an example. Figures 3.2, 3.3, and 3.4 show Fermi-Dirac, triangle, and Awad models, respectively.

3.2.2.1 Fermi-Dirac Weight

Figure 3.2 shows the Fermi-Dirac weight for C atom $\left(W^C(r)\right)$ within the CO molecule along the bond at C values of 0.999 and 0.999999, i.e., $k = 7.6$ and 14.5. As it can be seen in this figure, the weight shows some drawbacks: it goes to 0.5 when r approaches $-\infty$ or ∞ along the bond, and it is not exactly one (less than one) within the atom's core. It is clear that when changing the k value from 7.6 to 14.5 the weight function becomes similar to a step function.

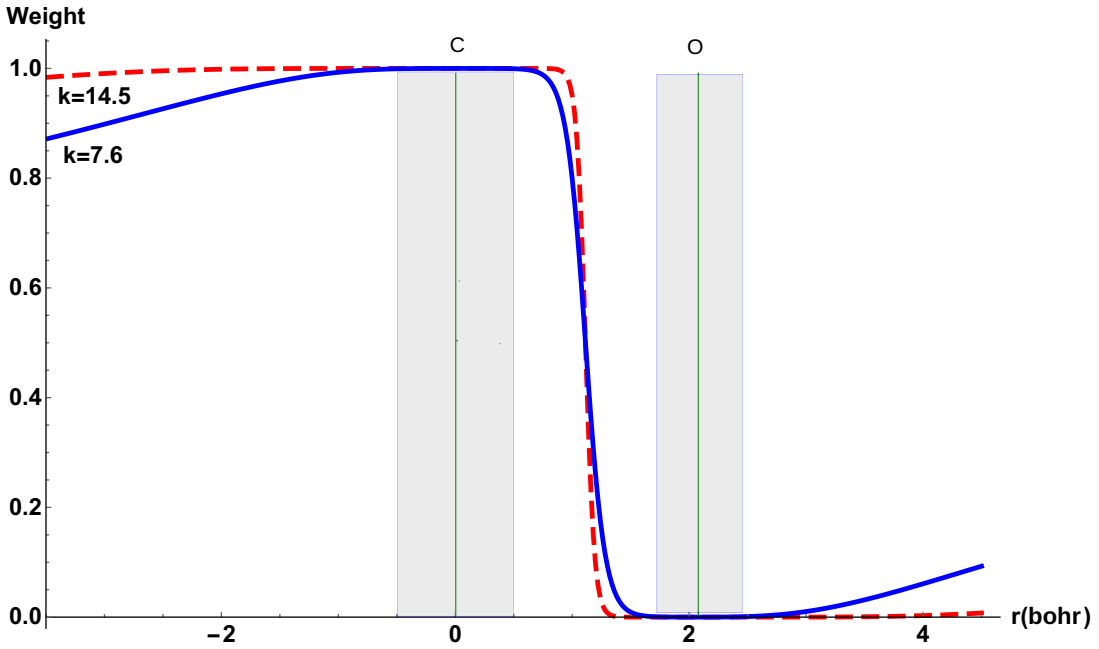


Figure 3.2: The Fermi-Dirac weight $W(\mathbf{r})$ with cores given by $r_{\langle r \rangle}$ for C atom within the CO molecule along the bond. The red dashed line is obtained at $k = 14.5$ and the blue solid line is obtained at $k = 7.6$ (Equation 2.12). The C atom located at the origin, the C-O bond length is 2.0871 bohr, $r_{\langle r \rangle}$ for C is 0.53690 bohr, and $r_{\langle r \rangle}$ for O is 0.39719 bohr.

3.2.2.2 Triangle Weight

In contrast to Fermi-Dirac weight, the triangle weight of the C atom in the CO molecule shows that the weight is exactly one within the core of C atom and exactly zero within the core of the other atom (in this case O atom) (Figure 3.3). However, similar to Fermi-Dirac weight, triangle weight goes quickly to 0.5 when r approaches $-\infty$ or ∞ along the bond. Another drawback is that the weight does not give a smooth curve because it has four singular points.

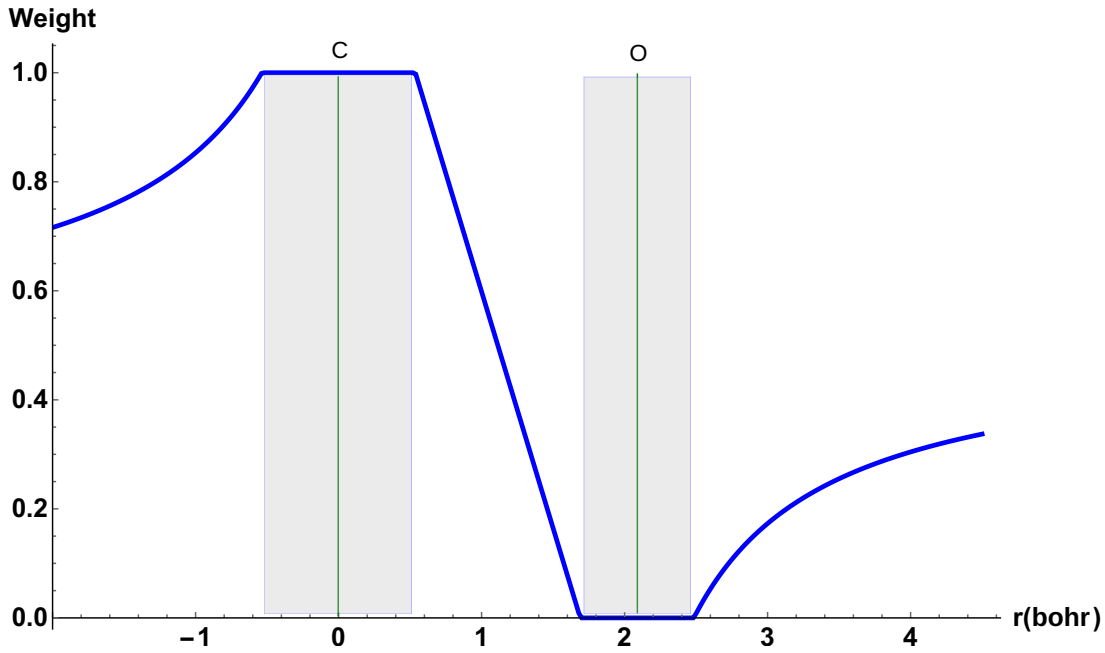


Figure 3.3: The triangle weight $W(\mathbf{r})$ with cores given by $r_{\langle r \rangle}$ for C atom within CO molecule along the bond. The C atom located at the origin, the C-O bond length is 2.0871 bohr, $r_{\langle r \rangle}$ for C is 0.53690 bohr, and $r_{\langle r \rangle}$ for O is 0.39719 bohr.

3.2.2.3 Awad Weight

The Awad weight gives exactly one within the core of C atom and exactly zero within the core of other atoms (Figure 3.4). In addition, this weight goes to one and zero when r approaches $-\infty$ or ∞, respectively. Moreover, Awad weight has the flexibility to change the steepness of the curve by controlling the value of steepness b (Equation 2.34). For example as shown in Figure 3.4, $W^C(\mathbf{r})$ within the CO molecule is computed using two values of steepness b; the red dashed line is obtained at $b = 0.5$ and the blue solid line is obtained at $b = 0$.

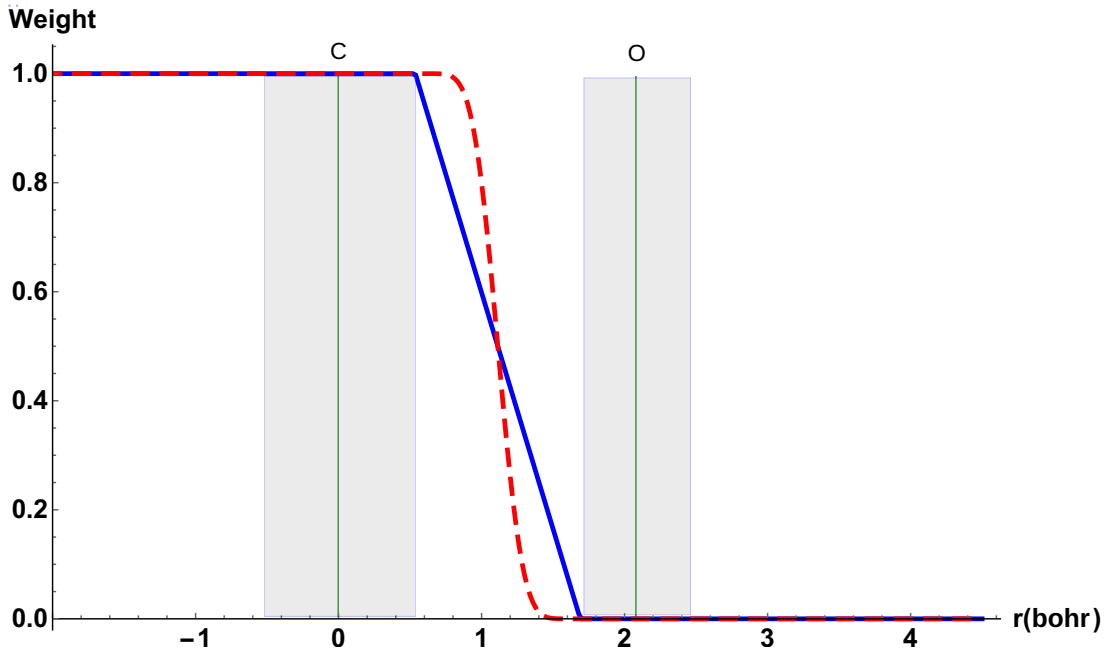


Figure 3.4: The Awad weight for C atom within the CO molecule along the bond using a difference steepness value. The red dashed line is obtained at $b = 0.5$ and the blue solid line is obtained at $b = 0$ (Equation 2.34). The C atom is located at the origin, the C-O bond length is 2.0871 bohr, $r_{\langle r \rangle}$ for C is 0.53690 bohr, and $r_{\langle r \rangle}$ for O is 0.39719 bohr.

3.2.3 Effect of Choosing the Core Size

The Fermi-Dirac, triangle, and Awad weights depend on the core sizes of the isolated atoms. In subsection 3.2.1, r_{max}, r_{min}, and $r_{\langle r \rangle}$ have been suggested as core sizes for Fermi-Dirac, triangle, and Awad models. In this section, we study the effect of choosing the core size on Awad weight visually and numerically. The visualization study includes RDEN and BDEN, whereas the numerical study includes comparing the following molecular properties: number of electrons, potential energy, and Coulomb energy.

3.2.3.1 Molecular Radial and Bond Electron Density

Besaw et al. [3] defined a bond as the region of space between a pair of atoms where the joint probability of finding electron density owned by each atom is nonzero. In the bonding region, the weighting functions of both nuclei (W_A and W_B) must be nonzero, since both the space and electron density are shared. Using W_A and W_B, the BDEN ($\rho_{A-B}(\mathbf{r})$) can be defined as,

$$\rho_{A-B}(\mathbf{r}) = \sqrt{W_A(\mathbf{r})W_B(\mathbf{r})}\rho(\mathbf{r}) \tag{3.1}$$

where $\rho(\mathbf{r})$ is the molecular electron density at $\mathbf{r}$ and the radial bond electron density ($\rho_{\text{rad}A-B}(\mathbf{r})$) is defined as follows,

$$\rho_{\text{rad}A-B}(\mathbf{r}) = \sqrt{W_A(\mathbf{r})W_B(\mathbf{r})}\rho_{\text{rad}}(\mathbf{r}) \tag{3.2}$$

where $\rho_{\text{rad}}(\mathbf{r})$ is the molecular radial electron density at $\mathbf{r}$.

We used two main ways of visualizing RDEN and BDEN in this thesis: the relief plot and the contour plot. The relief plot is a 3D graph that represents the cartesian location along

the plane of the molecule in the x and y directions, and the radial or bond electron densities in the z-direction. The contour plots are 2D figures that display curves of constant radial or bond electron densities. Since this section has many plots, some plots have been placed in Appendix A.

The studied molecules include CO, H_2, N_2, Cl_2, FCl, LiCl, LiF, HCl, HF, LiH, and NaH. These molecules are divided into two groups, the homonuclear diatomic molecules (H_2, N_2, and Cl_2) and the heteronuclear diatomic molecules (CO, FCl, HF, HCl, LiCl, LiF, LiH and NaH). Geometries of these molecules are optimized using HF/6-31G(d). The RDEN and BDEN of the studied molecules using the Awad weight are calculated on a mesh size of 0.02 bohr.

Besaw et al. [3] showed that RDEN is rich in topological features such as critical points, critical rings, and critical spheres. The classification (point, ring, sphere) and magnitude (maximum, minimum, or saddle) of a critical feature can be found by appraising the diagonalized Hessian of radial density at these points to yield three characteristic eigenvalues (λ_1, λ_2, λ_3). Table 3.2 summarizes the critical architectures of RDEN. In this table, the rank (r) and the signature (s) are the number of nonzero eigenvalues and the sum of the signs of the three eigenvalues ($+1$ for a positive eigenvalue, -1 for a negative eigenvalue, or zero for a vanishing eigenvalue) respectively. A rank of one, two, or three represents a critical sphere, ring, and point, respectively. The signature relative to the rank determines whether the topological feature is a maximum, minimum, or saddle. If $s = +r$ a minimum results; $s = -r$ a maximum results; if $-r < s < r$, a saddle results. The trivial case occurs when both s and $r = 0$.

Table 3.3 gives nuclear coordinates for atoms of the studied molecules and the magnitudes and positions of maxima and minima for the molecular radial density in the bonding region

Table 3.2: Molecular radial density: Critical architectures [3].

Architecture	Name	Acronym	λ_1	λ_2	λ_3	r	s
Point	Max point	PX	-	-	-	3	-3
	Saddle max point	PSX	-	-	+	3	-1
	Saddle min point	PSM	-	+	+	3	+1
	Min point	PM	+	+	+	3	+3
Ring	Max ring	RX	-	-	0	2	-2
	Saddle ring	RS	-	0	+	2	0
	Min ring	RM	0	+	+	2	+2
Sphere	Max sphere	SX	-	0	0	1	-1
	Min sphere	SM	0	0	+	1	+1
Trivial Case	Infinite point	IP	0	0	0	0	0

along the internuclear axis obtained at HF/6-31G(d) using the Awad weight with r_{min}, r_{max}, and $r_{\langle r \rangle}$. The maximum and minimum points along the bond have been determined accurately using Mathematica's FindMaximum and FindMinimum functions. The results in Table 3.3 show that positions and values of maximum peaks for RDENs in bonding regions for homonuclear molecules do not depend on the derived core. Whereas for heteronuclear molecules (one exception is HF), values of maximum peaks for RDENs are directly proportional to the size of cores, in which the r_{min} core have the largest values and r_{max} core have the smallest values.

Homonuclear Diatomic Molecules: Three homonuclear diatomic molecules; H_2, N_2, and Cl_2 are considered in this study. For the H_2 molecule, RDEN and BDEN do not depend on the type of the core chosen since the H atom does not have a core and only has a valence shell. Therefore, RDEN and BDEN of H_2 are composed of only distorted valence

Table 3.3: The maximum and minimum of molecular radial electron density (RDEN) along internuclear axis of the studied diatomic molecules in bonding region. These maximum and minimum are obtained using Awad weight with r_{min}, r_{max}, and $r_{\langle r \rangle}$.

Molecule (Coordinates, bohr)		r_{min}		r_{max}		$r_{<r>}$	
		Maximum (Coordinates*)	Minimum (Coordinates*)	Maximum (Coordinates*)	Minimum (Coordinates*)	Maximum (Coordinates*)	Minimum (Coordinates*)
N_2	(N 0.0, N 2.0229)	0.7403 (1.0115)	0.2225 (0.4675)	0.7403 (1.0115)	0.2258 (0.4612)	0.7403 (1.0115)	0.2225 (0.4675)
Cl_2	(Cl 0.0, Cl 3.7600)	0.5530 (1.8800)	0.2498 (0.9122)	0.5530 (1.8800)	0.2538 (0.9035)	0.5530 (1.8800)	0.2498 (0.9122)
CO	(C 0.0, O 2.0871)	0.7848 (1.1431)	0.1857 (0.5388), 0.2601 (1.6736)	0.7408 (1.1359)	0.1896 (0.5300), 0.2621 (1.6771)	0.7598 (1.1301)	0.1857 (0.5388), 0.2601 (1.6736)
FCl	(F 3.0503, Cl 0.0)	0.6978 (1.8308)	0.2278 (0.9265), 0.2486 (2.6632)	0.5980 (1.8206)	0.2349 (0.9136), 0.2489 (2.6637)	0.6974, (1.8314)	0.2278 (0.9265), 0.2486 (2.6632)
HCl	(H 2.4000, Cl 0.0)	0.4966 (1.4471)	0.2826 (0.8767)	0.4140 (1.2451)	0.2935 (0.8643)	0.4957 (1.4453)	0.2826 (0.8767)
HF	(H 1.6960, F 0.0)	0.5082 (0.8404)	0.2760 (0.3770)	0.5358 (0.7630)	0.2780 (0.3738)	0.5082 (0.8394)	0.2760 (0.3770)
LiH	(Li 0.0, H 3.0390)	0.4281 (2.2964)	0.0588 (1.2002)	0.1117 (1.8370)	0.0628 (1.1835)	0.2009 (2.0446)	0.0588 (1.2002)
Na H	(Na 0.0, H 3.6261)	3.0347 (3.4634)	0.1021 (1.7602)	0.1155 (2.2375)	0.1032 (1.8243)	0.2507 (2.5417)	0.1021 (1.7602)
LiCl	(Li 0.0, Cl 3.8492)	0.7249 (2.3201)	0.0618 (1.1697), 0.2884 (2.9704)	0.4820 (2.3546)	0.0652 (1.1462), 0.2917 (2.9772)	0.4610 (2.2367)	0.0618 (1.1697), 0.2884 (2.9704)
LiF	(Li 0.0, F 2.9777)	2.5613 (2.1282)	0.0847 (1.0296), 0.3195 (2.6159)	0.6275 (2.0087)	0.0886 (1.0114), 0.3197 (2.6162)	0.9107 (1.9374)	0.0847 (1.0296), 0.3195 (2.6158)

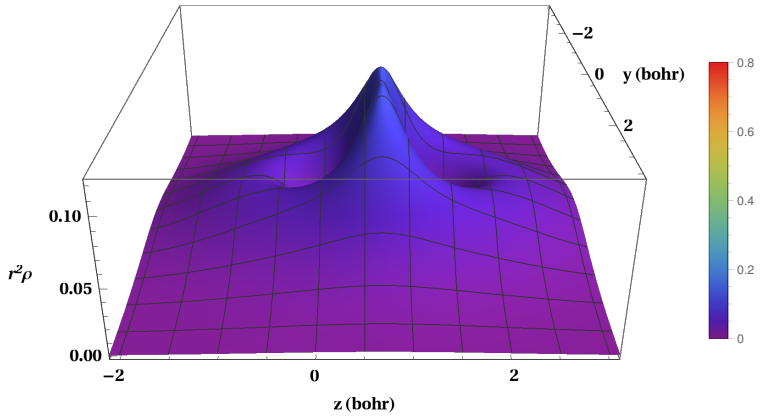
* All the coordinates (bohr) are along internuclear axis.

shells as shown in Figure 3.5. Figure 3.5a shows a relief map of the RDEN in the plane of H_2. Figure 3.5b is another way of displaying these results as a contour plot. Figure 3.5c shows 1D-plot of the RDEN along the internuclear axis. In these figures the RDEN of H_2 molecule was obtained using the Awad weight and the atomic coordinates of the two H atoms are 0.0 and 1.3895 bohr, respectively.

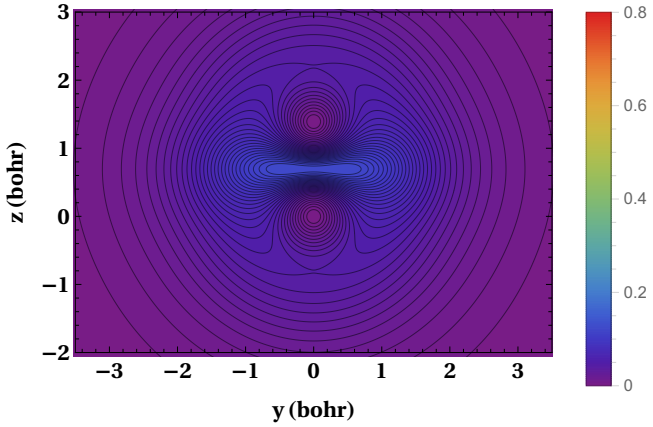
The next two molecules studied are N_2, and Cl_2. Unlike the H atom, the atomic radial density of the individual N and Cl atoms possess both valence and core shells. The radial density of a single N atom has one core shell (n = 1) and one valence shell (n = 2), whereas the radial density of a single chlorine atom has two core shells (n = 1, 2) and a valence shell (n=3). Therefore, it is expected that the RDEN of N_2 and Cl_2 would differ from that of H_2 whose atomic constituents only have a valence shell.

Figure 3.6a, b, and c represent contour plots and 1D-plots (along the internuclear axis) of molecular radial electron density for N_2 obtained using the Awad weight at r_{max}, r_{min}, and $r_{\langle r \rangle}$, respectively. For N_2, the values of r_{max}, r_{min}, and $r_{\langle r \rangle}$ are 0.148748, 0.500878, and 0.456610 bohr, respectively. In this figure, N atoms of the optimized N_2 structure are located at 0.0 bohr and 2.0229 bohr along the z axis.

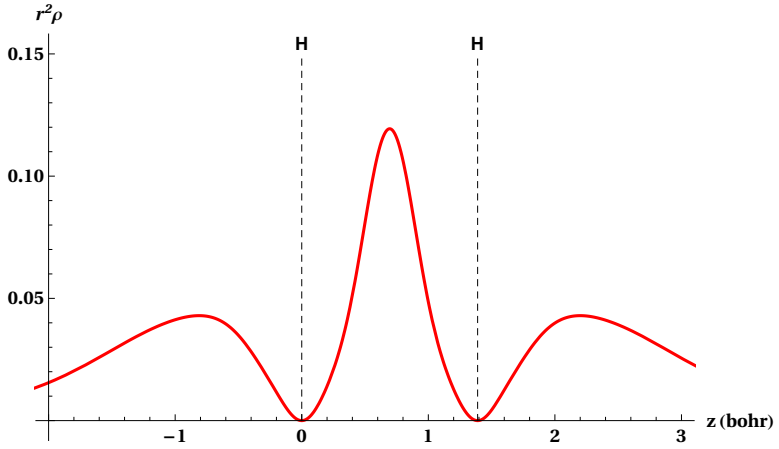
As mentioned in section 1.4.2.3, any molecule can be partitioned using RDEN to three regions: non-bonding, core, and bonding regions. As shown in Figure 3.6 the non-bonding region of N_2 molecule has one maximum point on each atom and the positions and RDEN at these points remain invariant when using r_{min}, $r_{\langle r \rangle}$, and r_{max} because of Awad weight is always one or zero in this region. The core region of N_2 molecule contains a pair of maxima peaks adjacent to each nucleus. These peaks located at z = -0.1491, 0.1504, 1.8725, and 2.1720 bohr with 0.5841, 0.5908, 0.5908, and 0.5841 e/bohr radial density, respectively by using r_{min}, $r_{\langle r \rangle}$, and r_{max}. The bonding region contains one maximum located exactly



(a) 3D plot

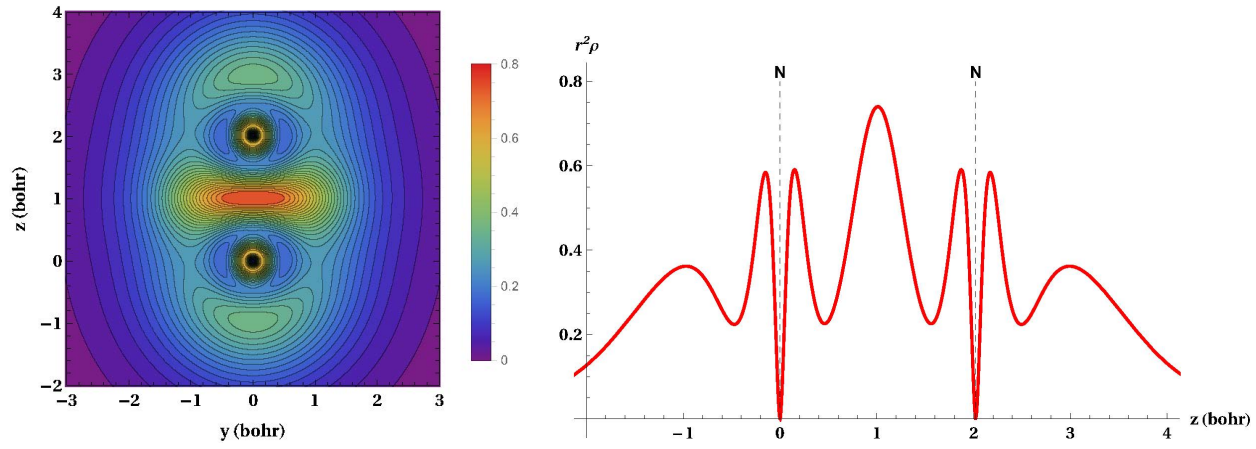


(b) Contour plot.

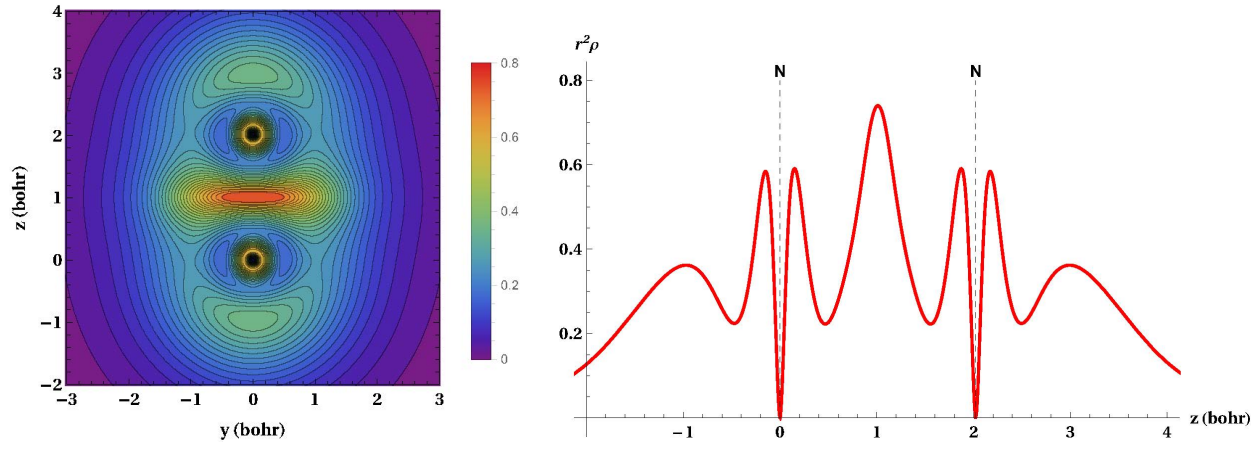


(c) Along the internuclear axis.

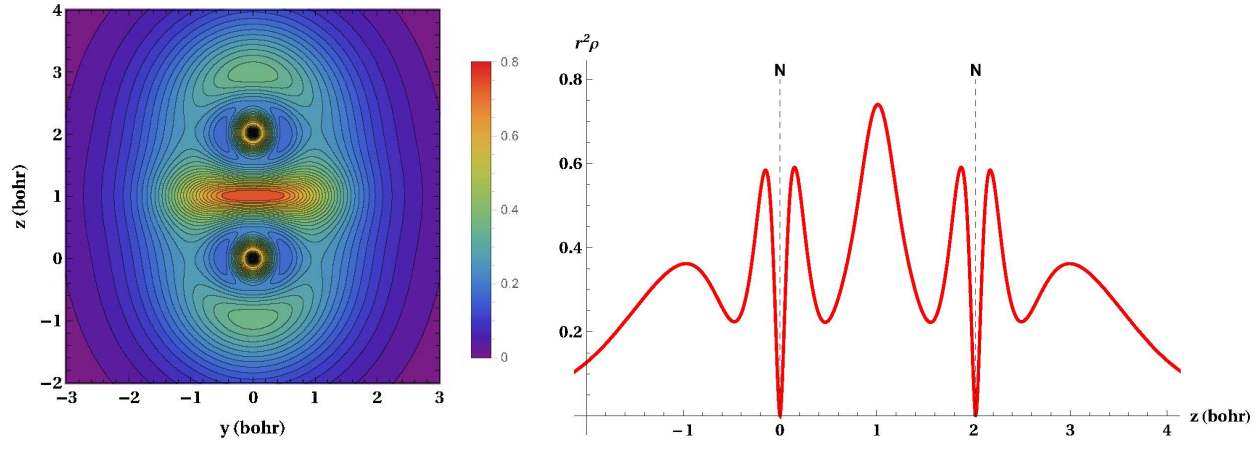
Figure 3.5: Molecular radial electron density (RDEN) of H_2 molecule. The atomic coordinates of the two H atoms are 0.0 and 1.3895 bohr along z axis.



(a) r_{max}



(b) r_{min}



(c) $r_{\langle r \rangle}$

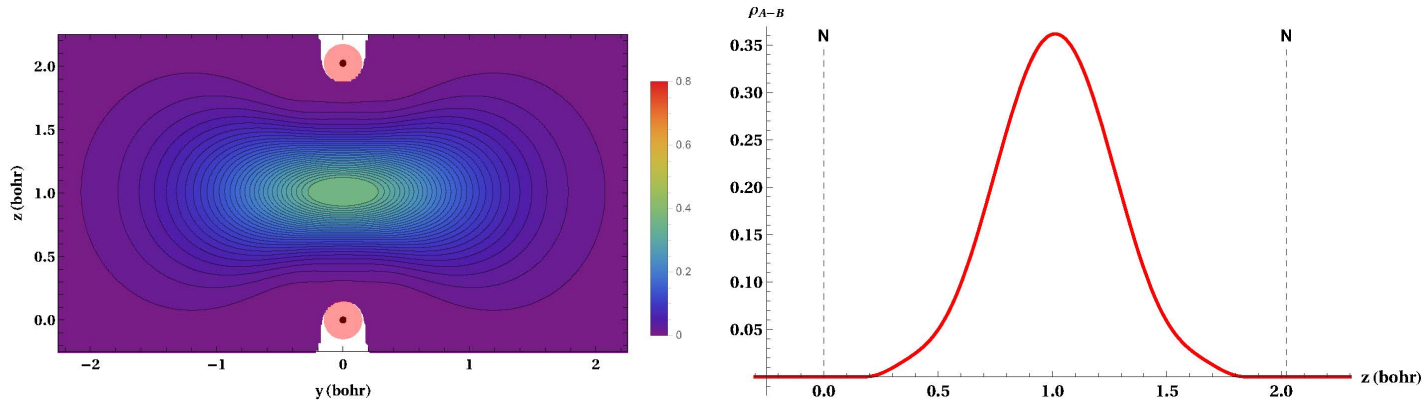
Figure 3.6: Molecular radial electron density (RDEN) for N_2 is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of the two N atoms are 0.0 and 2.0229 bohr along z axis.

between the two nitrogen nuclei at z = 1.0114 bohr with radial density of 0.7492 e/bohr using all r_{min}, $r_{\langle r\rangle}$, and r_{max}. The values and positions of these maxima are expected to be the same for r_{min}, $r_{\langle r\rangle}$, and r_{max}, due to the symmetry of N_2 and the Awad weight is exactly 0.5 at the middle distance of the N_2 bond. Unlike positions of minima located at the non-bonding region, the positions of the two minima located between the nuclei and hence their respective RDENs vary by changing the suggested cores. When r_{max} is used as a core, the points appear at z = 0.4612 and 1.5617 bohr with the same radial density of 0.2258 e/bohr, whereas their positions become at z = 0.4675 and 1.5554 bohr with 0.2225 e/bohr radial density when using $r_{\langle r\rangle}$ and r_{min}.

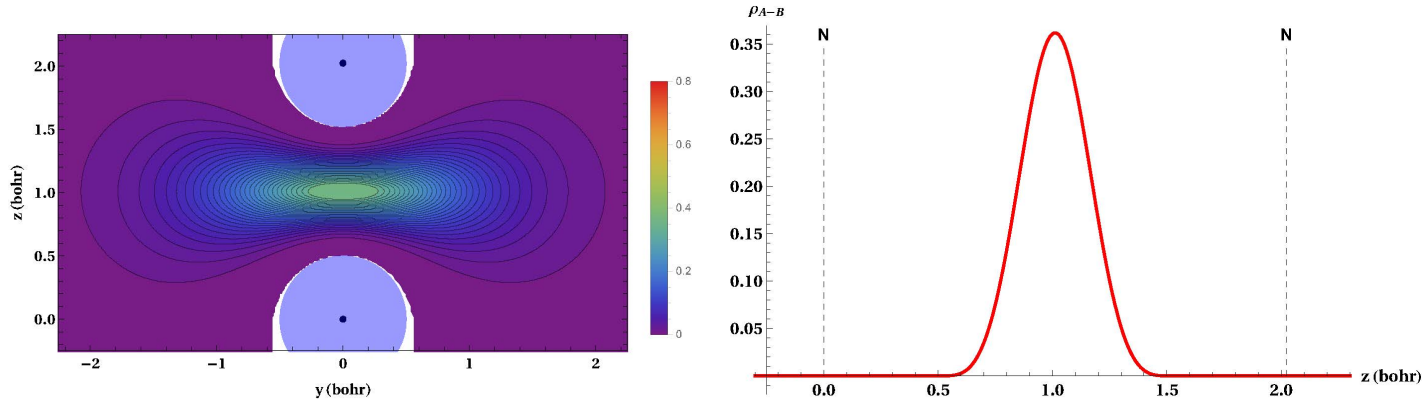
Figure 3.7a, b, and c show contour plots and 1D-plots (along the internuclear axis) of the BDEN for N_2 molecule obtained using the Awad weight with cores defined at r_{max} (red disk), r_{min} (blue disk), and $r_{\langle r\rangle}$ (green disk), respectively. It should be noted that r_{min}, $r_{\langle r\rangle}$, and r_{max} of the N_2 radial densities have smooth curves with only one maximum for each curve located exactly at the middle of the internuclear axis with 0.3662 e/bohr radial density. It is obvious from Figure 3.7 that using the smallest suggested core size (r_{max}) gives the widest BDEN's peak relative to those of the other two cores.

Figure 3.8a, b, and c represent contour plots and 1D-plots (a long the internuclear axis) of RDEN for Cl_2 obtained using Awad weight with cores defined at r_{max}, r_{min}, and $r_{\langle r\rangle}$, respectively. Since Cl_2 and N_2 are both homonuclear diatomic molecules, it is expected that they will exhibit similar molecular radial density distributions. One of the expected differences is that Cl atom has two core shells, whereas N atom has only a single core shell. As shown in Figure 3.8, the chlorine nuclei are located at $z = 0.0$ and $z = 3.7600$ bohr.

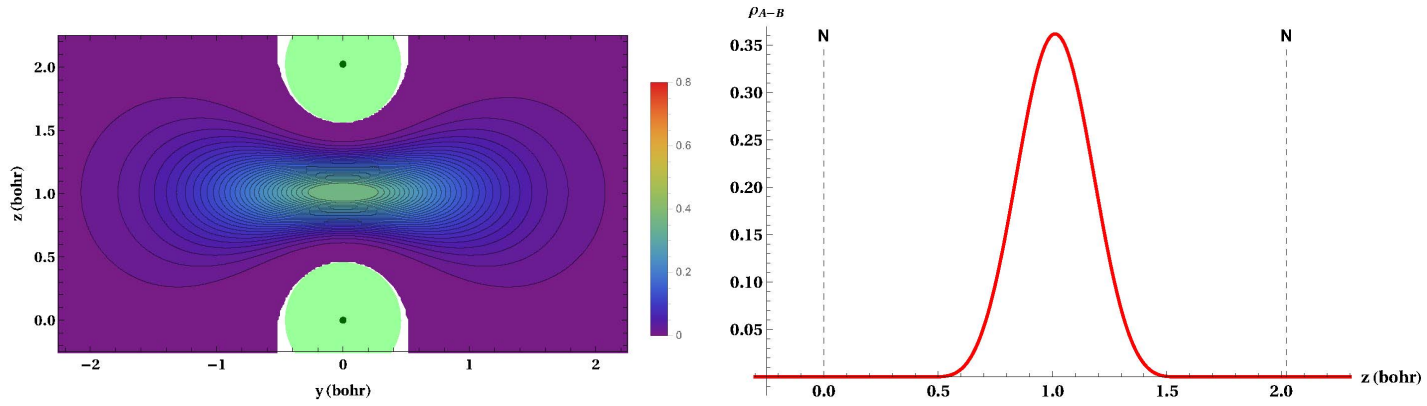
Adjacent to each nucleus along the bond, there are two pairs of maxima and one pair of



(a) r_{max}



(b) r_{min}

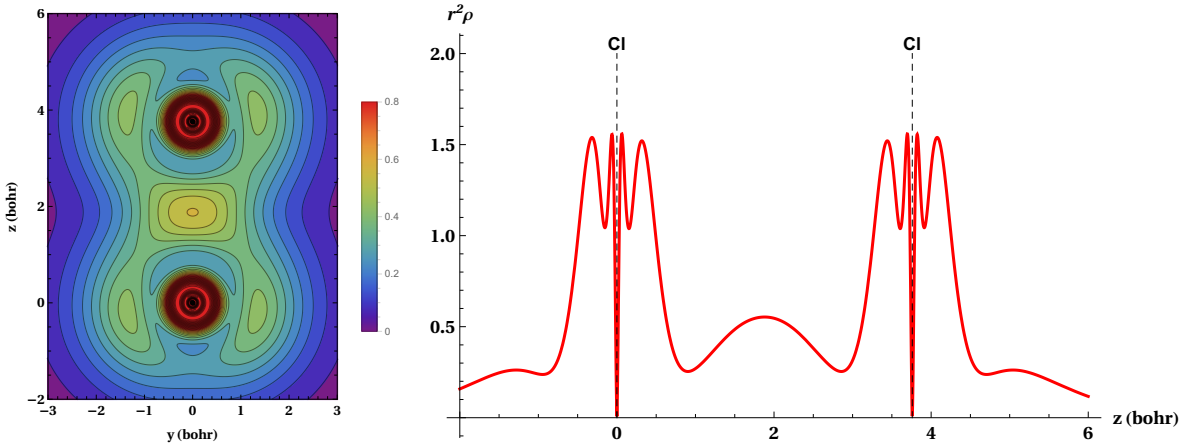


(c) $r_{\langle r \rangle}$

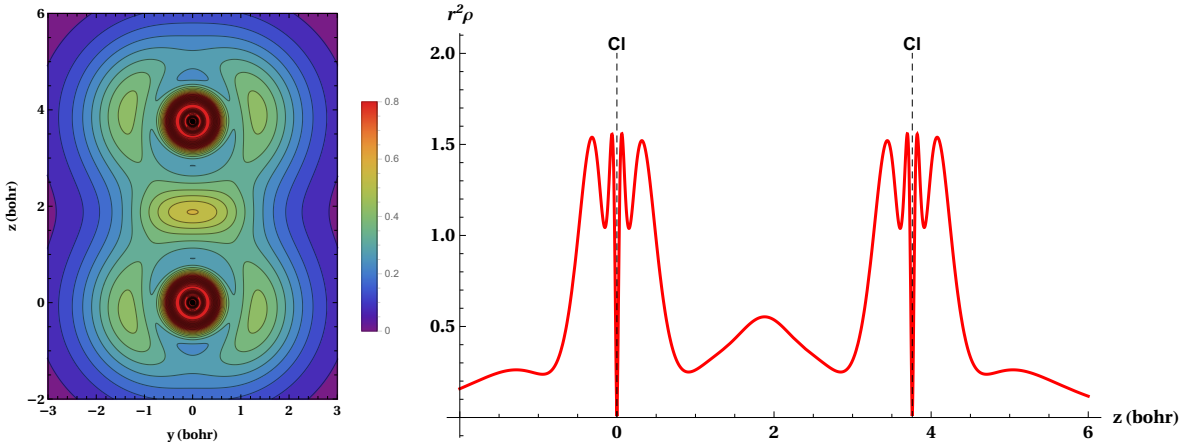
Figure 3.7: Bond electron density (BDEN) for N_2 is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of the two N atoms are 0.0 and 2.0229 bohr along z axis. The red, purple, and green disks are the core regions of r_{max}, r_{min}, and $r_{\langle r \rangle}$, respectively.

minima. For all different suggested core sizes, these maxima and minima are associated with the two core shells of a chlorine nucleus. Because the Awad weights within the core region for r_{max}, r_{min}, and $r_{\langle r \rangle}$ are similar, the maxima and minima are located exactly in same positions within this region. The maxima peaks along the bond within the cores are located at z = -0.3168, -0.0627, 0.0628, and 0.3177 bohr with 1.5394, 1.5565, 1.5591, and 1.5208 e/bohr radial density, respectively. The minima peaks along the bond within the cores are located at z =-0.1526 and 0.1537 bohr with 1.0436 and 1.0386 e/bohr radial density, respectively.

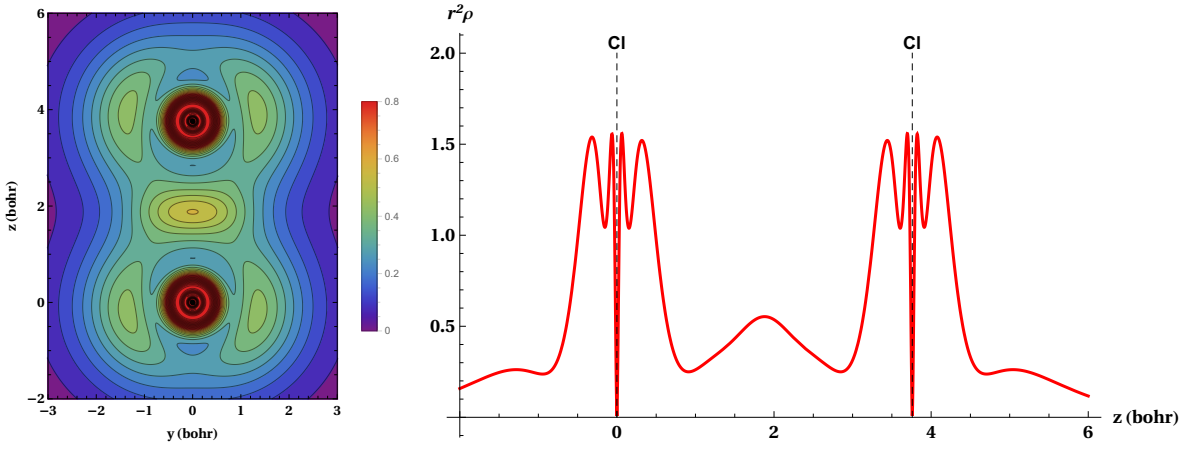
Similar to N_2 molecule, the non-bonding region has one maximum point on each atom and the positions and RDEN at these points remain invariant when using r_{min}, $r_{\langle r \rangle}$, and r_{max} because the Awad weight is always one or zero in this region. In addition, within the bonding region there is only one maximum located exactly in the middle of the bond with 0.5530 e/bohr radial density with r_{max}, r_{min}, and $r_{\langle r \rangle}$. Unlike positions of minima located at the non-bonding region, the positions of the two minima are located between the nuclei and hence their respective RDEN vary by changing the core size. For r_{max}, the minimum points are located at 0.9034 and 2.8566 bohr with 0.2538 e/bohr radial density and for both r_{min} and $r_{\langle r \rangle}$, they are located at 0.9122 and 2.8478 bohr with 0.2500 e/bohr radial density. Figure 3.9 shows the BDEN for Cl_2 molecule using the Awad weight at r_{min}, $r_{\langle r \rangle}$, and r_{max}. In this figure, the BDEN along the bond using r_{max} has minor shoulders. As r_{max} is so small to be used as a core, these shoulders may come from the electron density of the core. In addition, BDEN contour plots of r_{min}, and $r_{\langle r \rangle}$ (Figures 3.9b and c) give smoother contour lines comparing to that of r_{max} (Figure 3.9a). For these reasons r_{min} and $r_{\langle r \rangle}$ give better representations of the BDEN than r_{max}.



(a) r_{max}

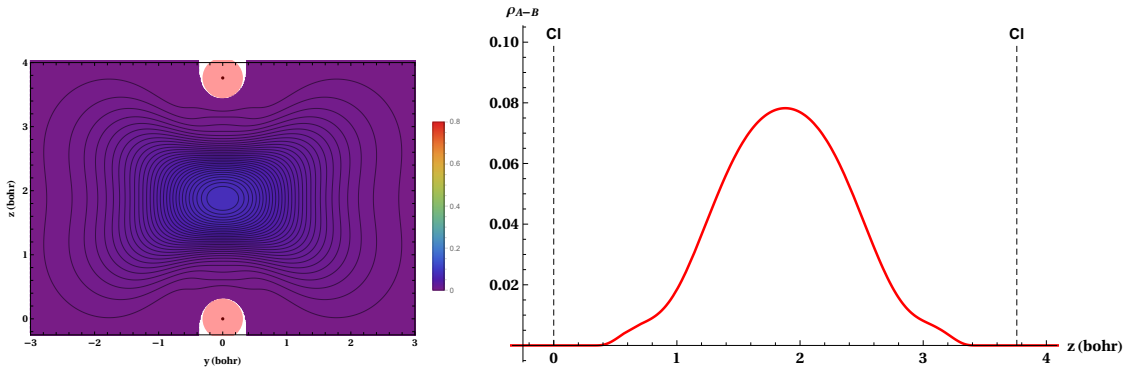


(b) r_{min}

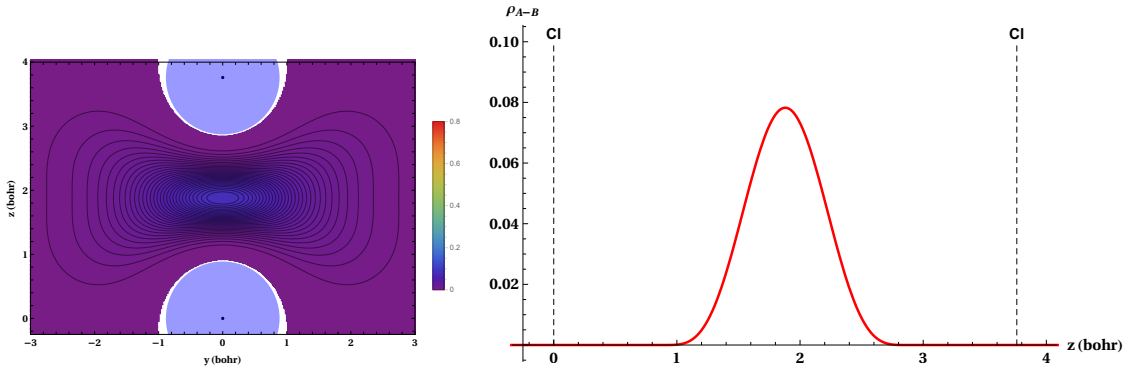


(c) $r_{\langle r \rangle}$

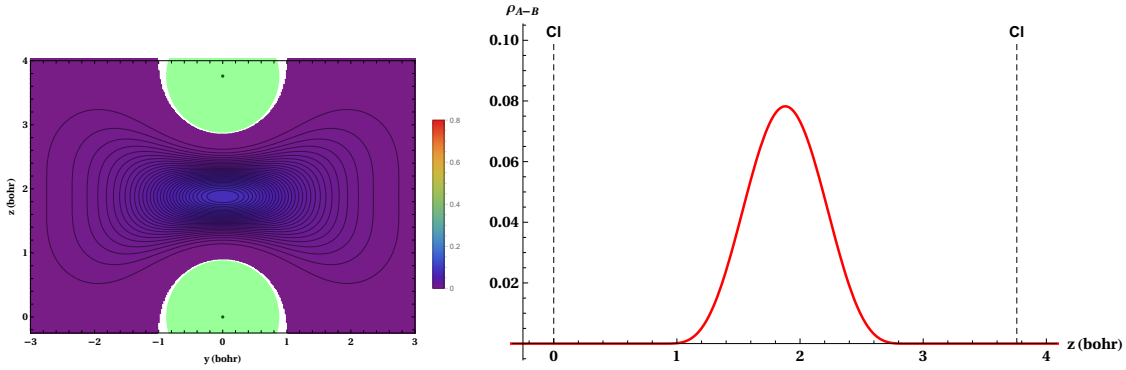
Figure 3.8: Molecular radial electron density (RDEN) for Cl_2 obtained using Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of the two Cl atoms are 0.0 and 3.7600 bohr along z axis.



(a) r_{max}



(b) r_{min}



(c) $r_{\langle r \rangle}$

Figure 3.9: Bond electron density (BDEN) for Cl_2 obtained using Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of the two Cl atoms are 0.0 and 3.7600 bohr along z axis.

Heteronuclear Diatomic Molecules: We consider the following heteronuclear diatomic molecules: CO, FCl, HF, HCl, LiCl, LiF, LiH and NaH. Although the RDENs and BDENs of the heteronuclear molecules are less uniform than those of the previously discussed homonuclear molecules (H_2, N_2 and Cl_2), some very similar features are observed. The similarities and differences between these homonuclear and heteronuclear diatomics can be observed by comparing the contour plots and 1D-plots (along the internuclear axis) of these diatomic molecules.

The topology of the RDEN and BDEN were studied along the bond. It was found that the minimum and maximum peaks within the core and non-bonding regions are the same (i.e., have the same locations and values) for r_{max}, r_{min}, and $r_{\langle r \rangle}$. This result is expected because the weight is one in the core and non-bonding regions for the target atom and zero for other atoms. In contrast, the locations and values of the maximum and minimum peaks in the bonding region depend on the core size used.

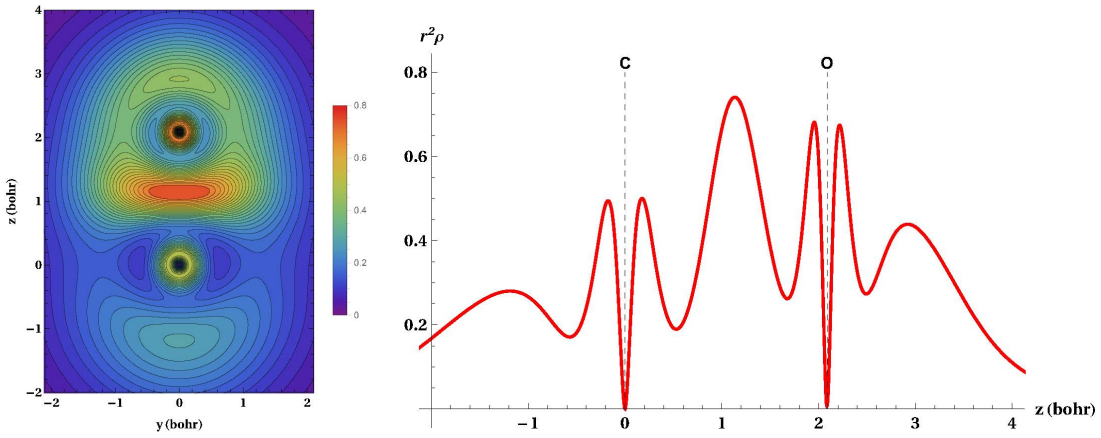
For example, Figure 3.10 shows RDEN for CO obtained using the Awad weight at r_{min}, $r_{\langle r \rangle}$, and r_{max}. In this figure, the C atom is located at 0.0 bohr and the O atom is located at 2.0871 bohr along z axis. Each core region of the CO molecule contains a pair of maximum peaks adjacent to each nucleus. Using r_{min}, $r_{\langle r \rangle}$, and r_{max}, the maximum peaks adjacent to the C nucleus are located at -0.1722 and 0.1759 bohr with 0.4948 and 0.5006 e/bohr radial density respectively, whereas the maximum peaks adjacent to O nucleus are located at -0.1313 and 0.1301 bohr (relative to O nucleus) with 0.6811 and 0.6745 e/bohr radial density respectively. The non-bonding region for each nucleus contains one maximum peak that has the same location and RDEN using r_{min}, $r_{\langle r \rangle}$, and r_{max}. The maximum peaks of the non-bonding regions are located at $z = -1.1871$ bohr with RDEN of 0.2802 e/bohr for C atom and at 2.9212 bohr with RDEN of 0.4390 e/bohr for O atom. Similar to the homonu-

clear molecules, the minima that separate the core and the non-bonding regions have the same location and RDEN for the two atoms using r_{max}, r_{min}, and $r_{\langle r\rangle}$. These minima are located at -0.5710 bohr with RDEN of 0.1710 e/bohr for C atom and at 0.4055 bohr (relative to O nucleus) with RDEN of 0.2733 e/bohr for O atom.

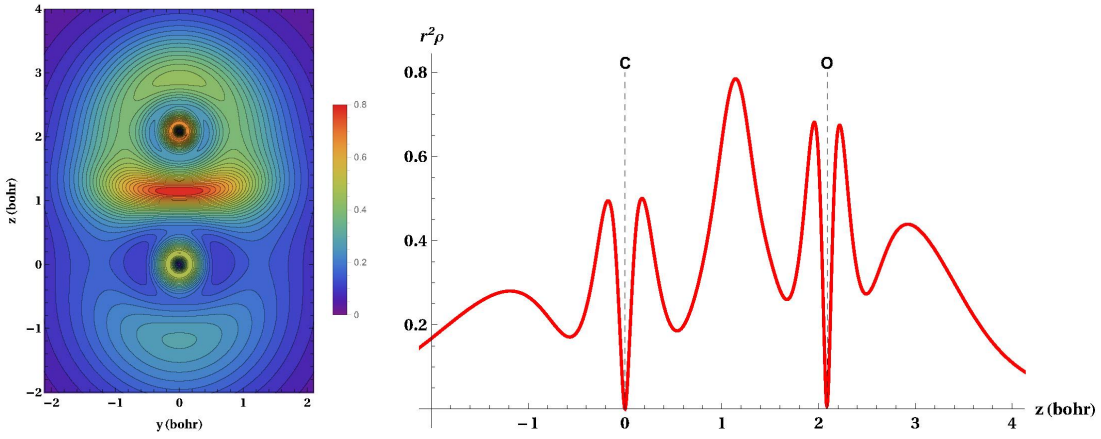
Finally, there is one maximum peak in the bonding region of the CO molecule. The location and value of this peak depend on choosing the core. For example, the maximum peaks are located at $z = 1.1301$, 1.1359, and 1.1431 bohr with 0.7589, 0.7408, and 0.7848 e/bohr radial densities for $r_{\langle r\rangle}$, r_{max}, and r_{min}, respectively. Also, the positions of the two minima that separate the core and bonding regions and hence their respective RDEN vary by changing the core sizes. For r_{max}, the minimum points are located at $z = 0.5300$ and 1.6771 with RDENs of 0.1896 and 0.2621 respectively, and for both r_{min} and $r_{\langle r\rangle}$, they are located at $z = 0.5388$ and 1.6736 bohr with RDENs of 0.1857 and 0.2601 e/bohr respectively.

Figure 3.11a, b, and c show contour plots and 1D-plots (along the internuclear axis) of the BDEN for CO molecule obtained using the Awad weight at r_{max} (red disk), r_{min} (blue disk), and $r_{\langle r\rangle}$ (green disk), respectively. Similar to the results of RDEN, the positions of maximum peaks of the BDEN for CO depends on choosing the core size. The positions of these maximum peaks are located at 1.16985, 1.14291, and 1.13562 bohr with 0.36153, 0.35101, and 0.34142 e/bohr electron densities for r_{min}, $r_{\langle r\rangle}$ and r_{max}, respectively. The resulting curves for 1D-plots of BDEN along the internuclear axis of CO molecule for r_{min}, $r_{\langle r\rangle}$ and r_{max} are smooth. In contrast, the contour plot lines of r_{max} core in Figure 3.11a are less smooth compared to the contour plot lines of r_{min} and $r_{\langle r\rangle}$ core sizes. Once again, the use of the smallest core size (r_{max}) shifts these minima slightly toward nuclei resulting in a wider bonding region than those obtained with r_{min} and $r_{\langle r\rangle}$.

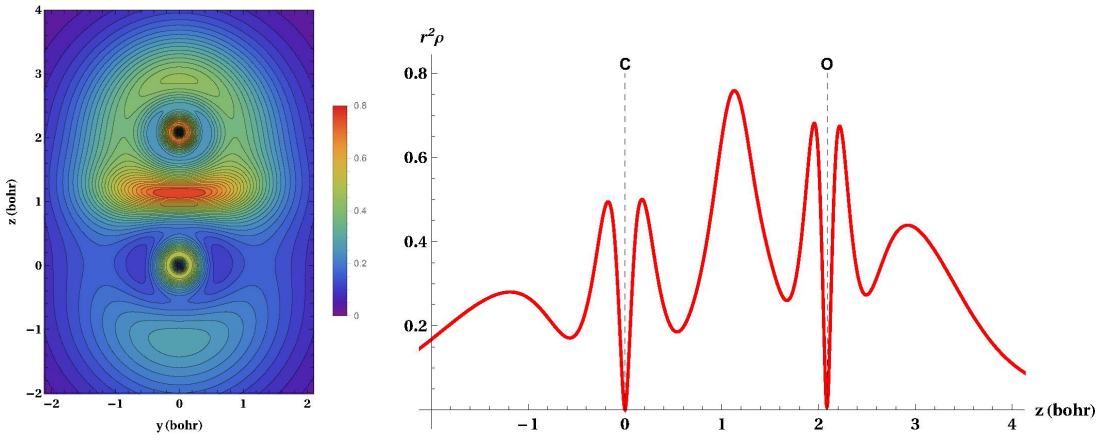
Figures 3.12 and 3.13 show RDENs for LiH and NaH molecules using the Awad weight



(a) r_{max}

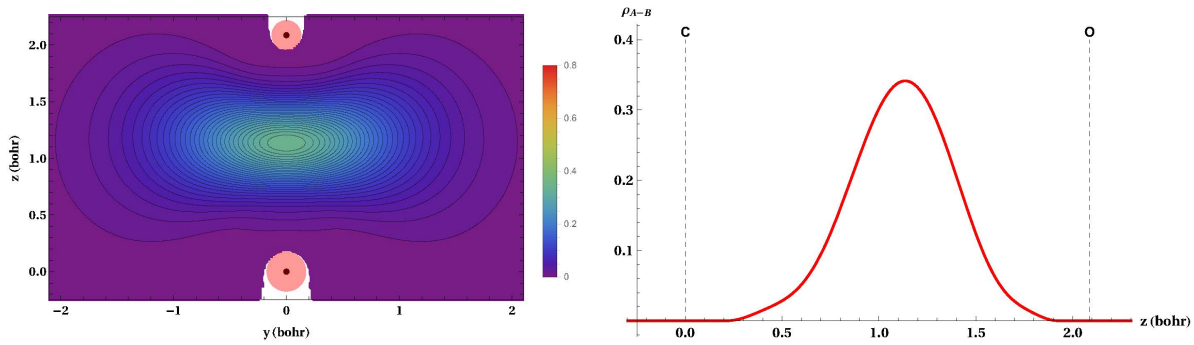


(b) r_{min}

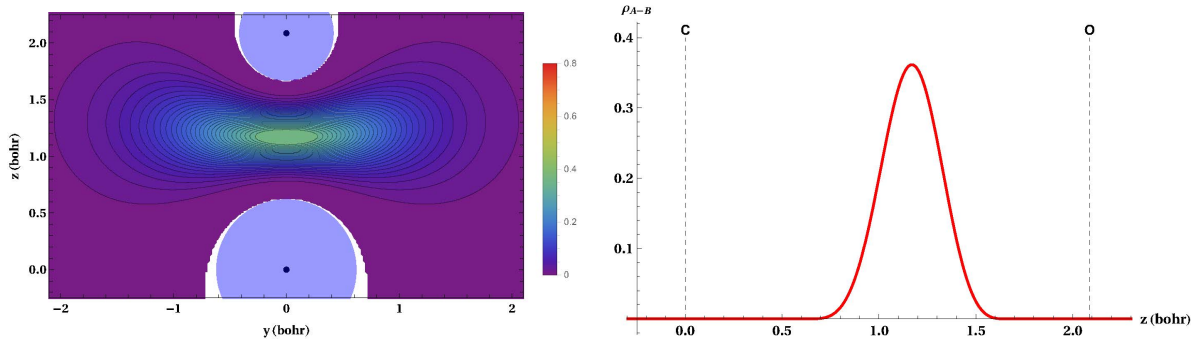


(c) $r_{\langle r \rangle}$

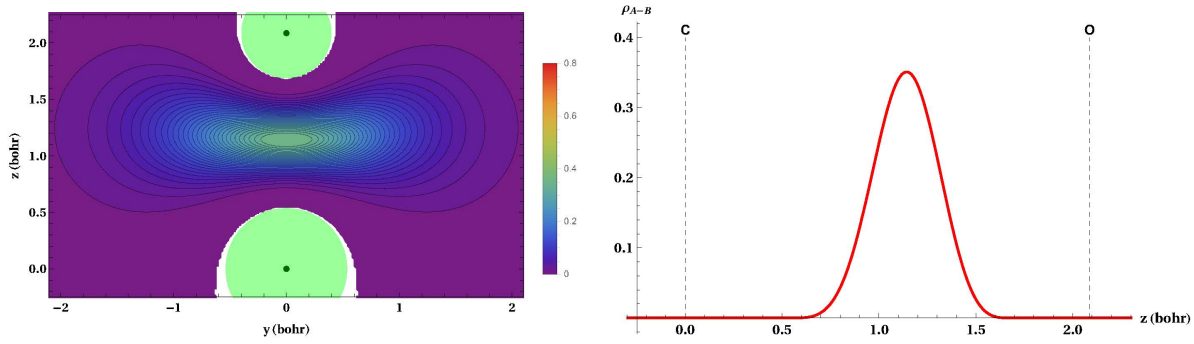
Figure 3.10: Molecular radial electron density (RDEN) for CO is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of C at 0.0 bohr and O at 2.0871 bohr along z axis.



(a) r_{max}



(b) r_{min}



(c) $r_{\langle r \rangle}$

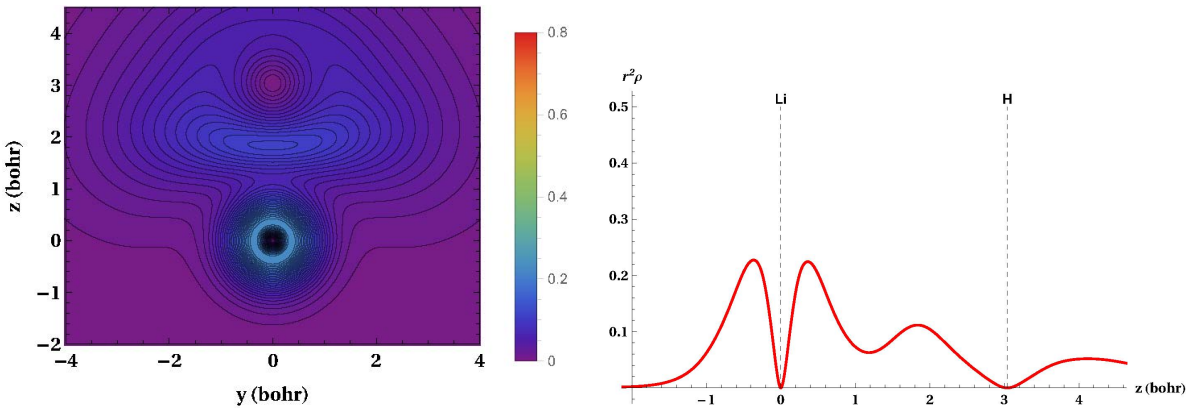
Figure 3.11: Bond electron density (BDEN) for CO is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of C at 0.0 bohr and O at 2.0871 bohr along z axis. The red, purple, and green disks are the core regions of r_{max}, r_{min}, and $r_{\langle r \rangle}$, respectively.

at different core sizes respectively. In these figures, positions and values of minimum and maximum peaks in the bonding region of the RDEN depend remarkably on choosing of the core. For NaH molecule, positions of these maximum peaks are located at 3.4634, 2.5417, and 2.2375 bohr with 3.0347, 0.2507, and 0.1156 e/bohr electron densities for r_{min}, $r_{\langle r \rangle}$ and r_{max}, respectively, and for LiH molecule they are located at 2.2964, 2.0446, and 1.8367 bohr with 0.4281, 0.2009, and 0.1166 e/bohr electron densities for r_{min}, $r_{\langle r \rangle}$ and r_{max}, respectively. Figures 3.14 and 3.15 depict BDENs for NaH and LiH molecules using the Awad weight at different cores respectively. These figures show clearly the significance of choosing the core size as the position, shape, and value of BDENs change remarkably in both NaH and LiH molecules. These changes are more pronounced in molecules containing metal atoms such as Li and Na than molecules contain only non-metallic atoms. Generally for heteronuclear molecules, values of maximum peaks for RDENs are affected by changing the core size, in which the r_{min} has the highest RDEN values and r_{max} has the smallest RDEN values. In addition, the curves of BDEN using r_{max} are wider in width and less smooth than using other core sizes.

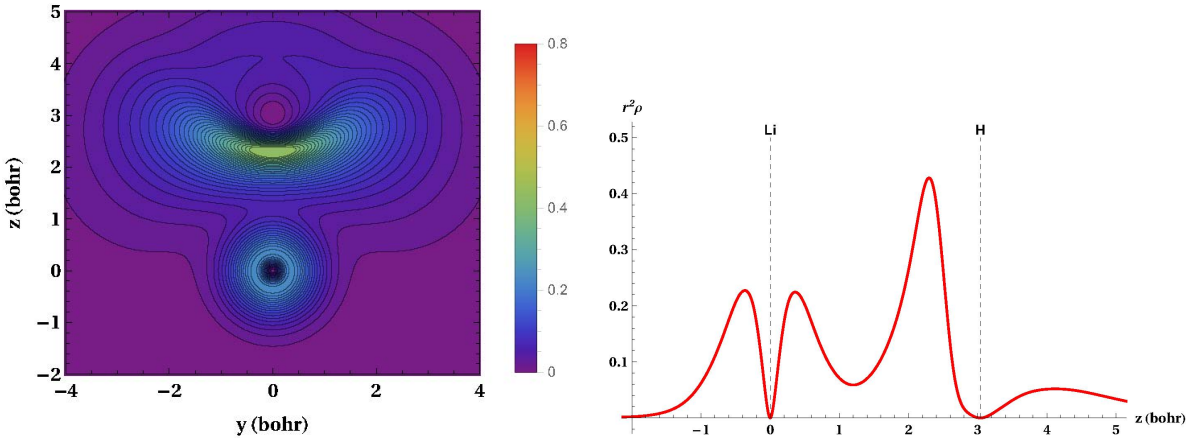
So far we discussed two homonuclear molecules (N_2 and Cl_2) and three heteronuclear molecules (CO, LiH, and NaH). The results of the remaining molecules (H_2, FCl, LiCl, LiF, HCl, and HF) are given in Appendix A.

3.2.4 Comparing the Awad and Becke Weights

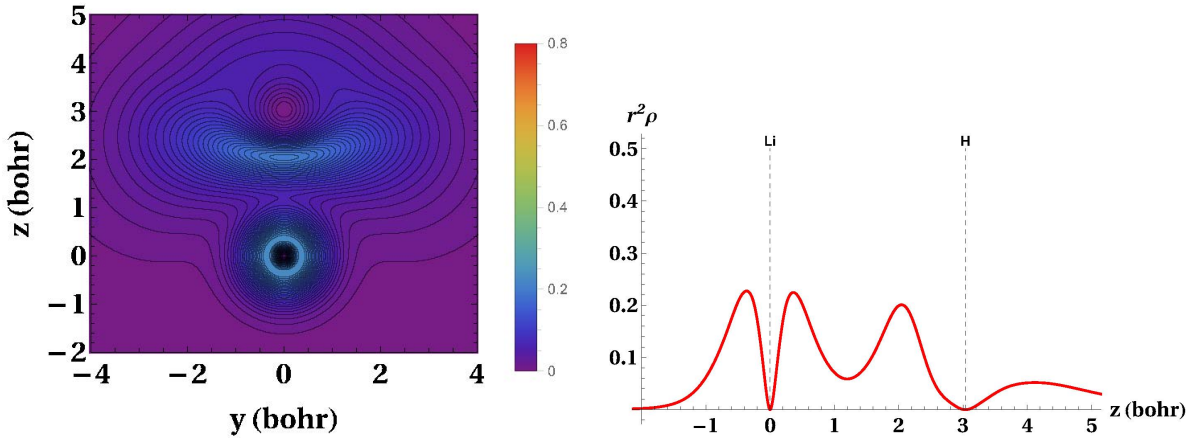
In this section, we compare the molecular RDENs and BDENs obtained using the Awad weight with those obtained using Becke weight. The RDEN and BDEN for LiF molecule using both Becke and Awad weights are shown in Figures 3.16 and 3.17 respectively. We



(a) r_{max}

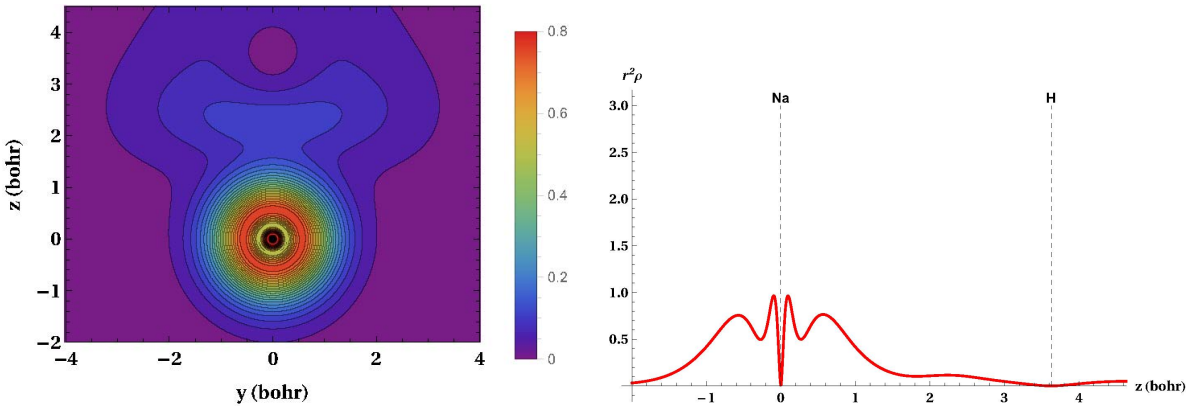


(b) r_{min}

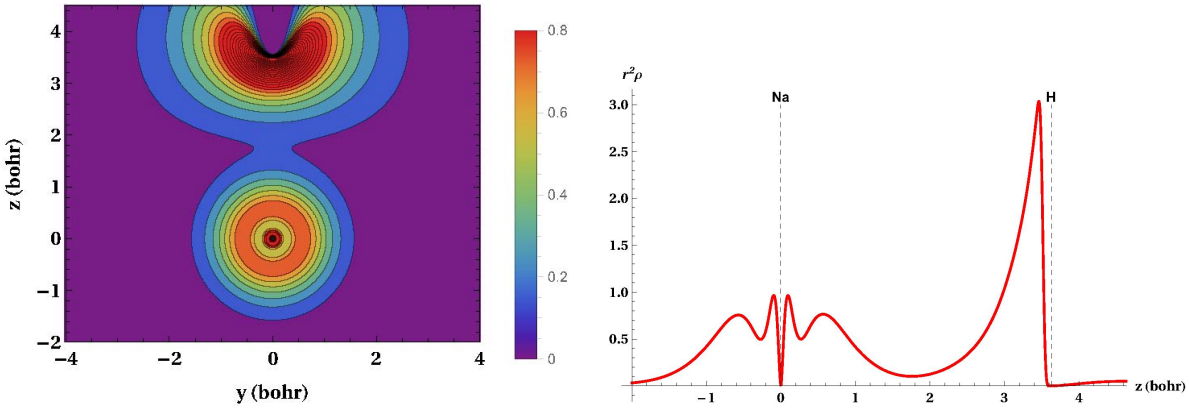


(c) $r_{\langle r \rangle}$

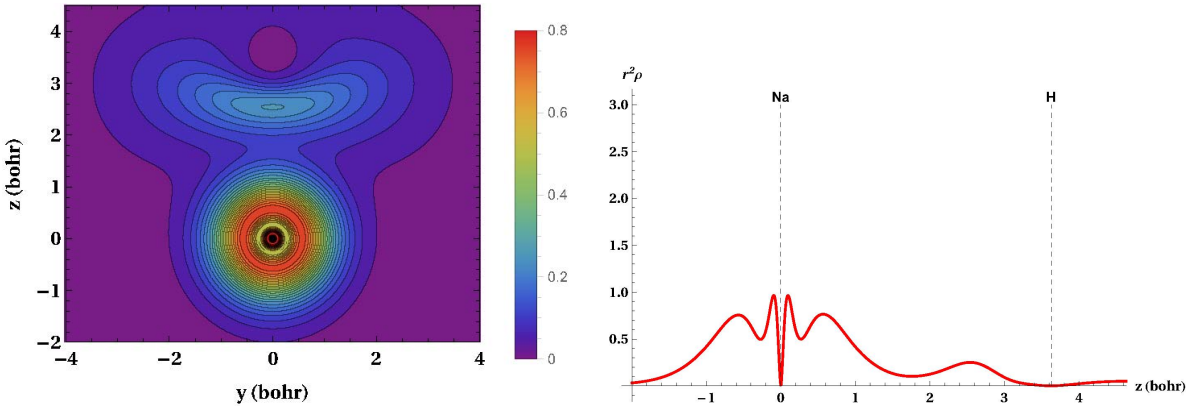
Figure 3.12: Molecular radial electron density (RDEN) for LiH is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of Li at 0.0 bohr and H at 3.0390 bohr along z axis.



(a) r_{max}

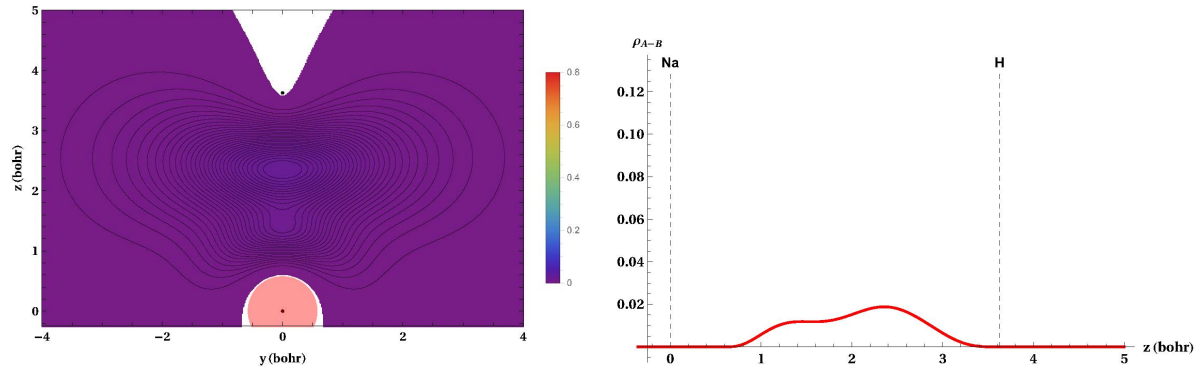


(b) r_{min}

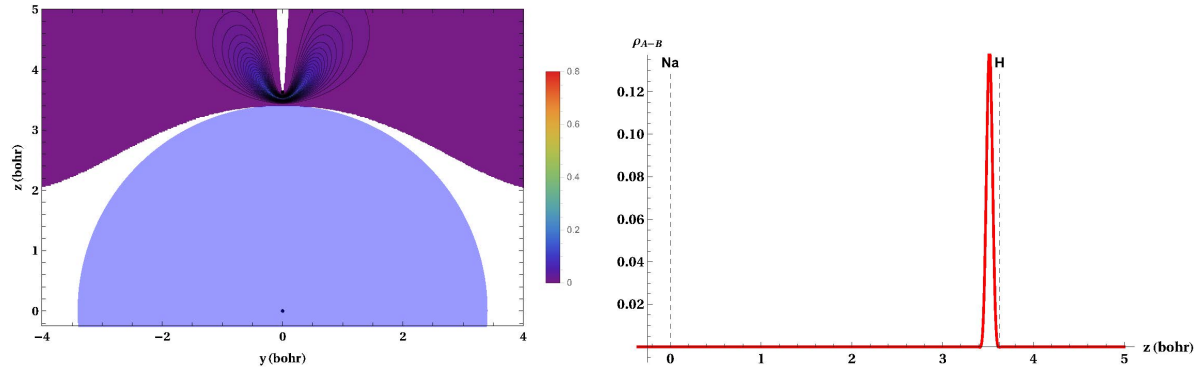


(c) $r_{\langle r \rangle}$

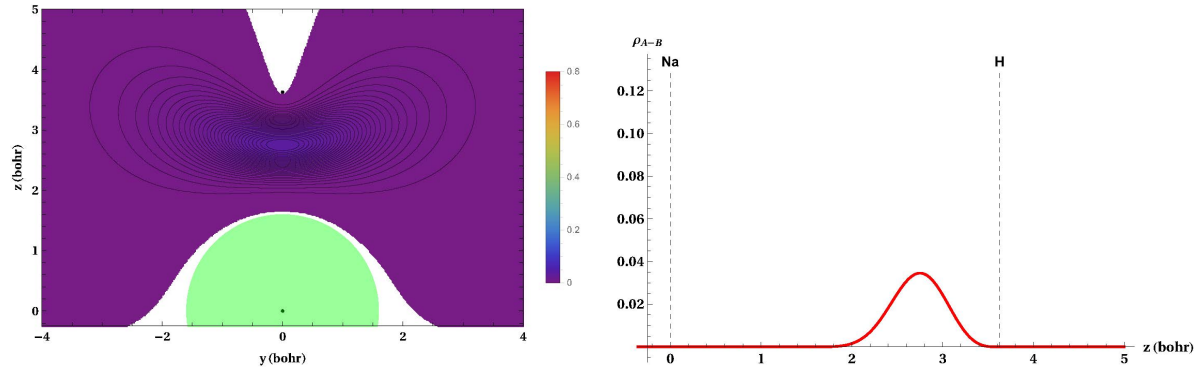
Figure 3.13: Molecular radial electron density (RDEN) for NaH is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of Na at 0.0 bohr and H at 3.6261 bohr along z axis.



(a) r_{max}

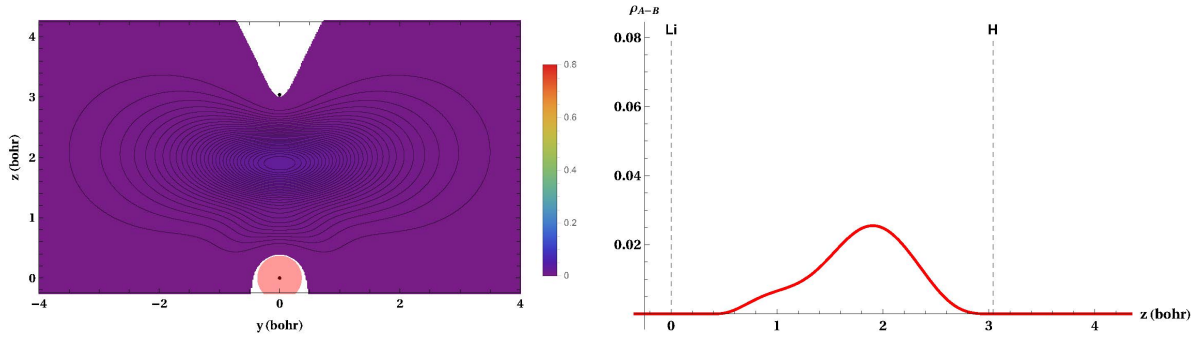


(b) r_{min}

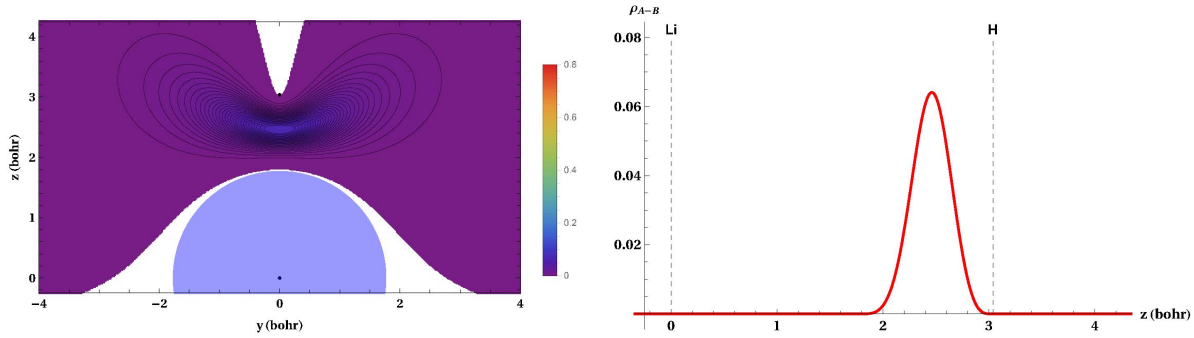


(c) $r_{\langle r \rangle}$

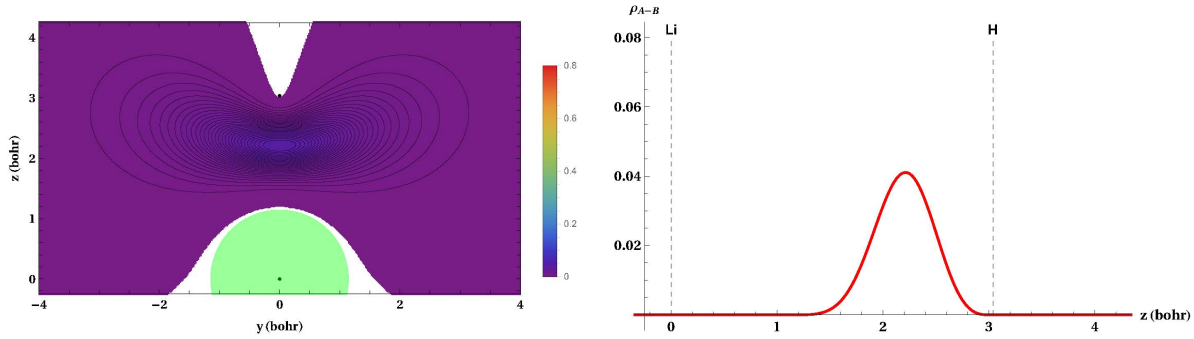
Figure 3.14: Bond electron density (BDEN) for NaH is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of Na at 0.0 bohr and H at 3.6261 bohr along z axis. The red, purple, and green disks are the core regions of r_{max}, r_{min}, and $r_{\langle r \rangle}$, respectively.



(a) r_{max}



(b) r_{min}



(c) $r_{\langle r \rangle}$

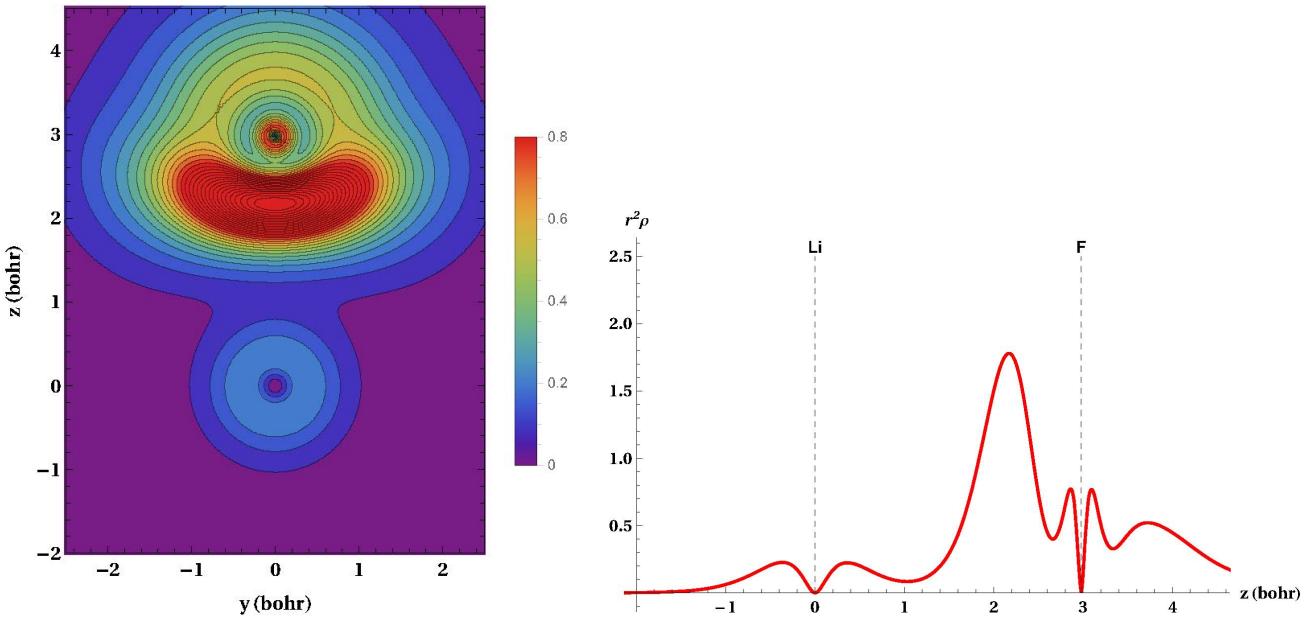
Figure 3.15: Bond electron density (BDEN) for LiH is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of Li at 0.0 bohr and H at 3.0390 bohr along z axis. The red, purple, and green disks are the core regions of r_{max}, r_{min}, and $r_{\langle r \rangle}$, respectively.

use $r_{\langle r \rangle}$ as a core size for calculations involving the Awad weight. From these figures, the Awad weight gives better bonding-region representation with both RDEN and BDEN than the Becke weight. As shown in Figure 3.16, it is clear from the RDEN in bonding region that LiF exhibits more ionic properties, in consistence with the chemical sense, when using the Awad weight. When using the Becke weight, the electron density is concentrated in the bonding region indicating more covalent-bond characteristics. Figure 3.17 shows that the electron density in the bonding region of LiF using Becke weight is contaminated from the electron density of the core. In contrast, using the Awad weight the core is assigned to the atom not to the bond or the cores of other atoms. The other studied molecules (H_2, N_2, Cl_2, CO, FCl, LiCl, HCl, HF, LiH, and NaH) have the same features for RDEN and BDEN. For brevity, the RDEN and BDEN plots of these molecules are moved to section A.3.
In general, the Awad weight has many favorable features over the Becke weight:

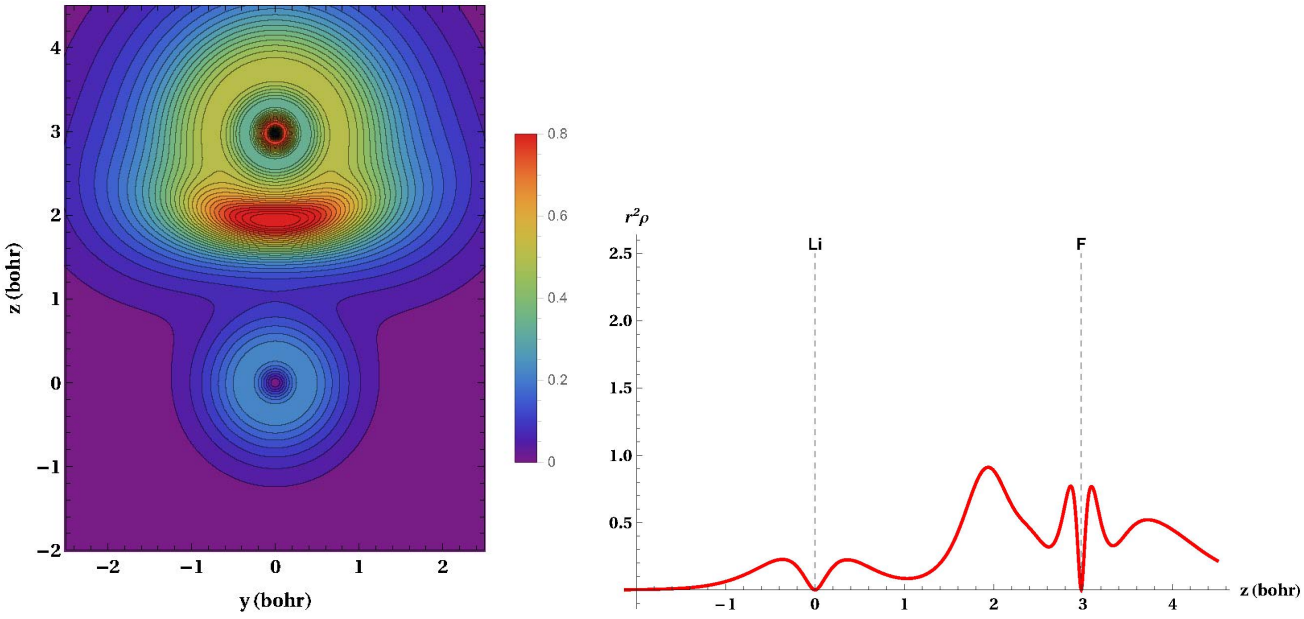
- The Awad weight has flexibility of choosing the core boundary of atoms within the molecule. For example the atomic core sizes can be stored in a database according to the geometry of the molecule.

- The ability to assign the core to the atom not to the bond or the cores of other atoms, giving us the ability to compute the molecular properties in the bonding region (such as the bond order).

- There is no special condition for the type of atom within the molecules. For example, in case of Becke the $\chi_{AB} = \frac{R_A}{R_B}$ should be adjusted between 0.200 and 0.417, but in our case the Awad weight is not restricted.

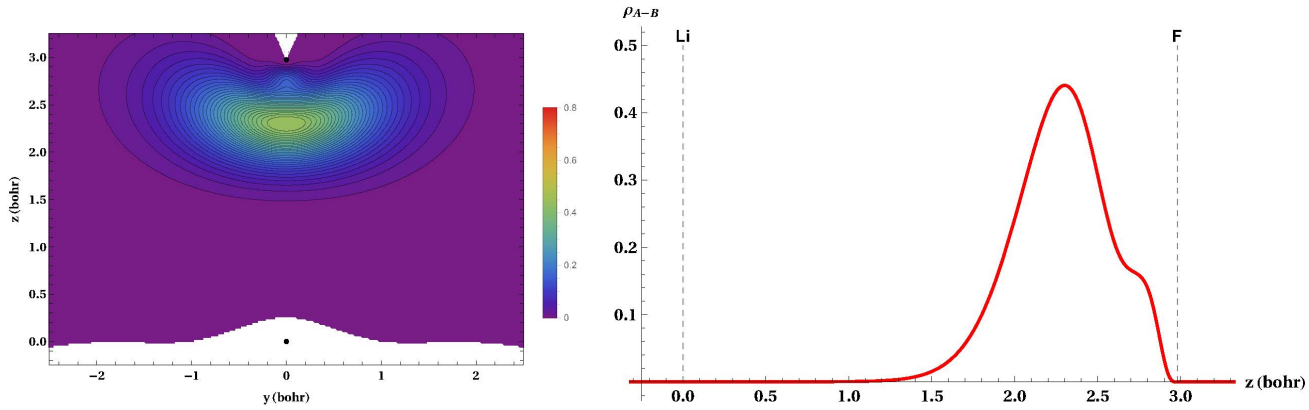


(a) The core size is defined as Becke weight.

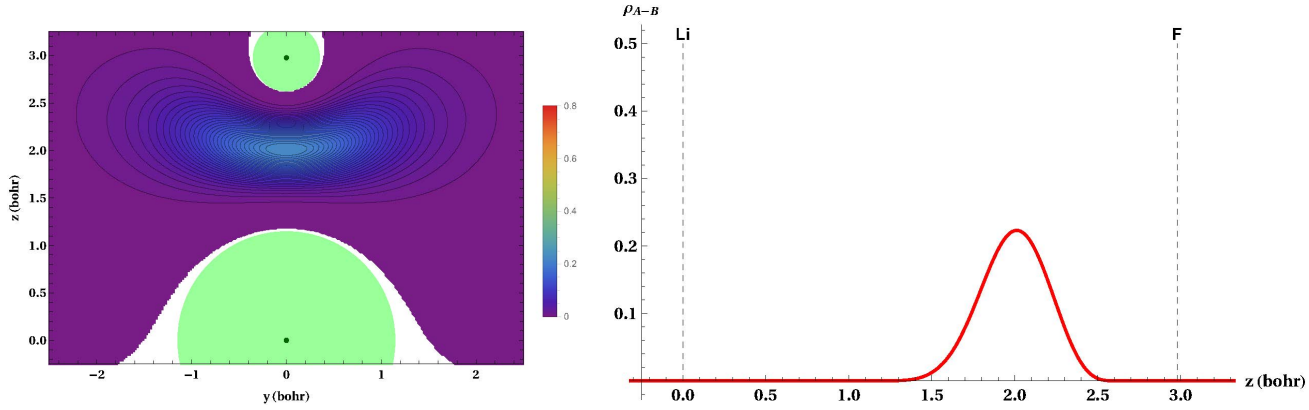


(b) Awad weight at $r_{\langle r \rangle}$.

Figure 3.16: Molecular radial electron density (RDEN) for LiF is obtained using the Becke and Awad weights. The atomic coordinates of Li at 0.0 bohr and F at 2.9777 bohr along z axis.



(a) The core size is defined as Becke weight.



(b) Awad weight at $r_{\langle r \rangle}$.

Figure 3.17: Bond electron density (BDEN) for LiF is obtained using the Becke and Awad weights. The atomic coordinates of Li at 0.0 bohr and F at 2.9777 bohr along z axis.

3.2.5 Numerical Integration Results

Using numerical integration techniques has become routine in quantum chemistry, especially in DFT. However, as mentioned in subsection 1.6.3, numerical integration is not straightforward in molecular calculations. The popular solution is by partitioning the molecular space into cells or discrete regions. One of the most common partitioning schemes is the Becke weight [4]. In this section the Awad weight is used in the numerical quadrature methods to integrate properties for which the exact analytical results are known, namely, the HF wavefunction. The calculations are performed using the different suggested core sizes r_{max}, r_{min}, and $r_{\langle r \rangle}$, then the results were compared with Becke weight. In general, the accuracy of numerical integration methods are evaluated by integrating the electron density for the total number of electrons, for the total DFT energy, or explicitly, for the exchange-correlation energy [5]. In this section the calculated molecular properties include the total number of electrons N_e, the electron-nuclear potential energy V_{ne}, and Coulomb potential energy V_{ee}. Therefore, the values of N_e, V_{ne}, and V_{ee} calculated using the numerical integration are compared with those calculated by HF wavefunction (the exact values).

The mean absolute error (MAE) of the integration is computed as,

$$\mathrm{MAE} = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}} \left| \mathrm{P}^{\mathrm{i}}_{\mathrm{NI}} - \mathrm{P}^{\mathrm{i}}_{\mathrm{HF}} \right| \tag{3.3}$$

where $\mathrm{P}^{\mathrm{i}}_{\mathrm{NI}}$ is the value of the molecular property, N_e, V_{ne}, or V_{ee} computed using numerical integration for a molecule i, and $\mathrm{P}^{\mathrm{i}}_{\mathrm{HF}}$ is the corresponding molecular property calculated using HF, and N is the number of molecules. In addition for each property the accuracy of

the integration is calculated using the formula given by Gill and Chien [6]

$$\text{accuracy} = -\log_{10}\left|\frac{\mathrm{P_{NI}}}{\mathrm{P_{HF}}} - 1\right| \tag{3.4}$$

where, $\mathrm{P_{NI}}$ is the value of a molecular property calculated using numerical integration, and $\mathrm{P_{HF}}$ is the HF value of the respective property. Table 3.4 shows the error $\mathrm{P^i_{NI}} - \mathrm{P^i_{HF}}$ and mean absolute error (MAE) values of the total number of electrons N_e (in μe), electron-nuclear potential energy V_{ne} (in μhartree), and Coulomb potential energy V_{ee} (in μhartree) using the Awad weight (with $r_{\langle r\rangle}$, r_{max}, and r_{min}) and the Becke weight.

Number of electrons N_e

The total number of electrons is calculated by integrating the HF electron density using,

$$N_e = \int \rho(\mathbf{r}) d\mathbf{r} \tag{3.5}$$

where N_e is number of electrons and $\rho(\mathbf{r})$ is the molecular electron density at position $\mathbf{r}$. The MAE for calculated total number of electrons using the Awad weight with $r_{\langle r\rangle}$ and r_{max} are 0.01524 and 0.003665 μe respectively, and it is 0.4916 μe using Becke weight. Clearly that the performance of Awad weight (with $r_{\langle r\rangle}$ and r_{max}) is much better than Becke weight. Using r_{min} with the Awad weight gives the worst result compared to other core sizes where MAE equals 0.2116 μe excluding the NaH molecule. Including NaH changes MAE significantly to 162.2 μe using r_{min}.

Figure 3.18 compares the accuracy (Equation 3.4) of numerical integration results for calculating number of electrons N_e using the Awad weight (with r_{max}, r_{min}, and $r_{\langle r\rangle}$) and Becke

weight. The accuracy results in Figure 3.18 are in agreement with the previous MAE results as the Awad weight (with $r_{\langle r \rangle}$ or r_{max}) gives better results than Becke weight especially for ionic compounds such as LiCl and LiF.

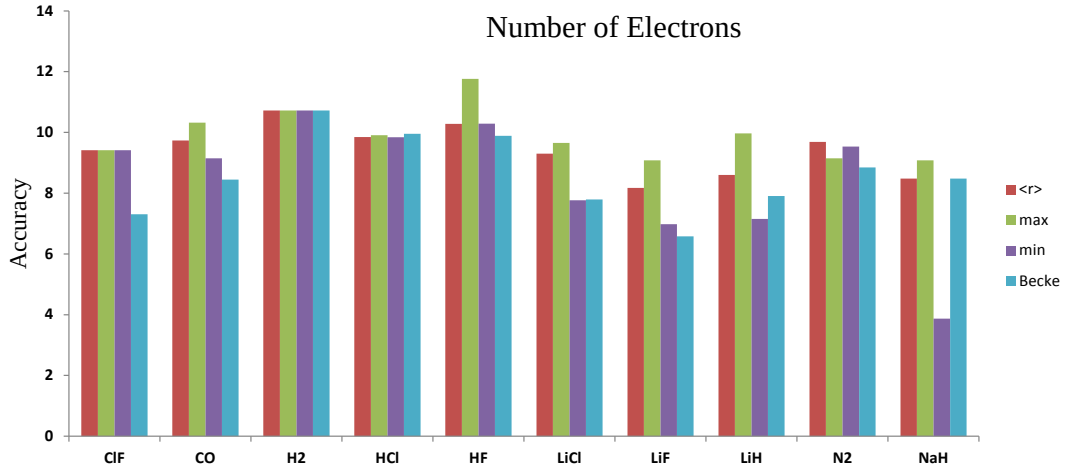


Figure 3.18: Comparing the accuracy (Equation 3.4) of numerical integration results for calculating number of electrons N_e using the Awad weight (with r_{max}, r_{min}, and $r_{\langle r \rangle}$) and Becke weight.

Potential Energy V_{ne}

The electron-nuclear potential energy V_{ne} is calculated using

$$V_{ne} = -\sum_A Z_A \int \frac{\rho(\mathbf{r})}{|\mathbf{r} - \mathbf{R}_A|} d\mathbf{r} \tag{3.6}$$

where $\mathbf{R}_A$ is the position of atom A with atomic charge Z_A and $\rho(\mathbf{r})$ is the molecular electron density at position $\mathbf{r}$. The errors and MAE values for potential energy are shown in Table 3.4 and the accuracy (Equation 3.4) of numerically calculated potential energy are shown in Figure 3.19. The MAE using the Awad weight with $r_{\langle r \rangle}$ and r_{max} are 102.45 and 102.48 μhartree respectively which are slightly better than MAE of 115.01 μhartree ob-

tained using Becke weight. Figure 3.19 shows that using the Awad weight (with $r_{\langle r \rangle}$ and r_{max}) and Becke weight almost have the same accuracy in computing the potential energy for all molecules except for LiF, in which the Becke weight is less accurate.

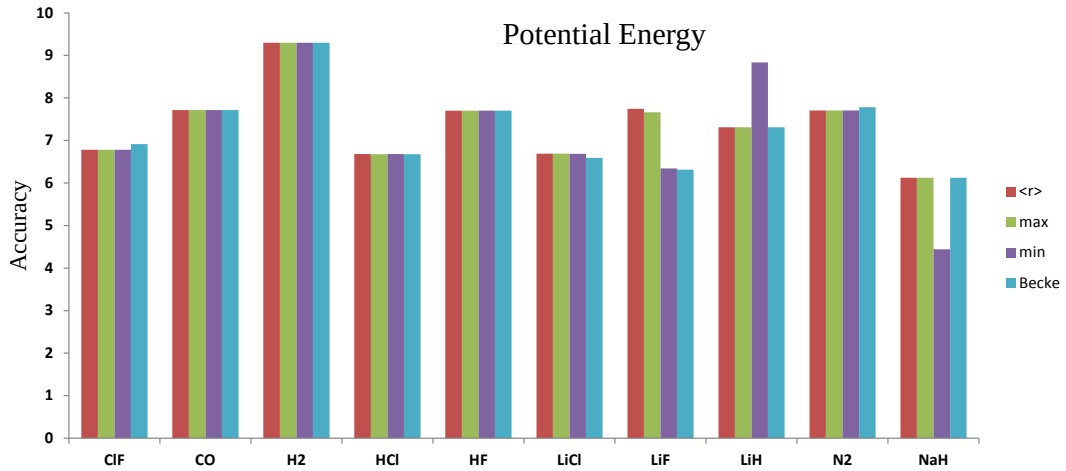


Figure 3.19: Comparing the accuracy (Equation 3.4) of numerical integration results for calculating the electron-nuclear potential energy V_{ne} using the Awad weight at different core sizes (r_{max}, r_{min}, and $r_{\langle r \rangle}$) and with Becke weight.

Coulomb energy V_{ee}

The Coulomb potential energy V_{ee} is calculated numerically as,

$$V_{ee} = \int \rho(\mathbf{r}_2) \left[\sum_{\mu\nu} P_{\mu\nu} \int \frac{\varphi_\mu^*(\mathbf{r}_1)\varphi_\nu(\mathbf{r}_1)}{|\mathbf{r}_1 - \mathbf{r}_2|} d\mathbf{r}_1 \right] d\mathbf{r}_2 \tag{3.7}$$

$P_{\mu\nu}$ is the density matrix element, $\rho(\mathbf{r}_2)$ is the molecular electron density at position $\mathbf{r}_2$, $\varphi_\mu(\mathbf{r}_1)$ and $\varphi_\nu(\mathbf{r}_1)$ are the μ^{th} and ν^{th} basis functions at position $\mathbf{r}_1$ respectively, and $|\mathbf{r}_1 - \mathbf{r}_2|$ is the distance between $\mathbf{r}_1$ and $\mathbf{r}_2$. The first integral ($d\mathbf{r}_1$) in the brackets is computed analytically, while the second integral ($d\mathbf{r}_2$) is computed numerically.

Again, the Coulomb potential energy results show that using the Awad weight with $r_{\langle r\rangle}$ and r_{max} gives better results than Becke weight where the MAE equal 0.09160, 0.08454, and 6.295 μhartree for $r_{\langle r\rangle}$, r_{max}, and Becke weight, respectively. In addition the accuracies of calculated Coulomb potential energies using the Awad weight (with $r_{\langle r\rangle}$ and r_{max}) are better than those obtained using Becke weight for all molecules except NaH as shown in Figure 3.20.

As a conclusion, even though Becke weight was originally designed to perform numerical integration efficiently for calculating molecular properties, the Awad weight gives better results in computing the molecular properties.

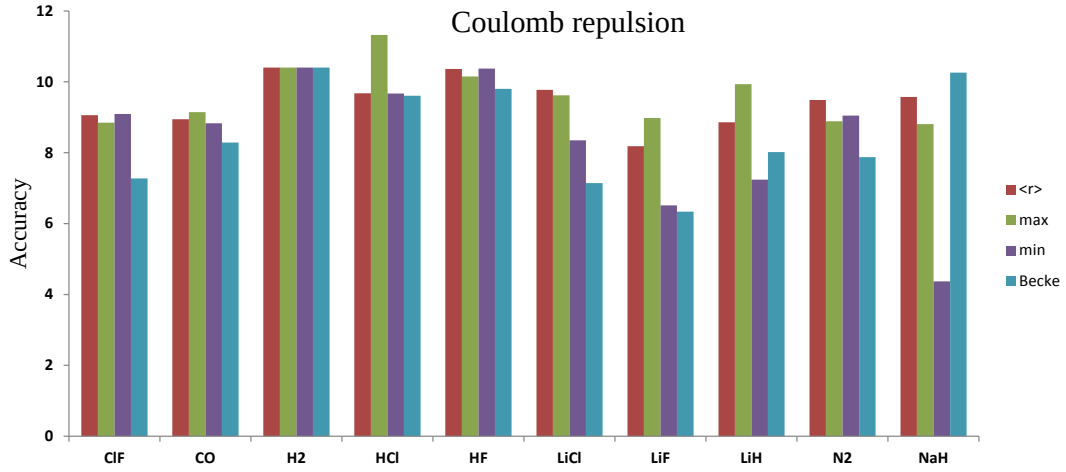


Figure 3.20: Comparing the accuracy (Equation 3.4) of numerical integration results for calculating the Coulomb potential energy V_{ee} using the Awad weight at different cores (r_{max}, r_{min}, and $r_{\langle r\rangle}$) and with Becke weight.

Table 3.4: The error $P^i_{NI} - P^i_{HF}$ and MAE values of the total number of electrons N_e in μe, the electron-nuclear potential energy V_{ne} in μhartree, and Coulomb potential energy V_{ee} in μhartree using the Awad weight (with $r_{\langle r\rangle}$, r_{max}, and r_{min}) and the Becke weight.

	Total number of electrons N_e				Coulomb potential energy V_{ee}				Electron-nuclear potential energy V_{ne}			
	$r_{\langle r\rangle}$	r_{max}	r_{min}	Becke	$r_{\langle r\rangle}$	r_{max}	r_{min}	Becke	$r_{\langle r\rangle}$	r_{max}	r_{min}	Becke
ClF	0.008130	-0.008290	0.010000	-1.290000	-0.262400	-0.432800	-0.245200	-16.10000	-236.9000	-236.6000	-236.9000	-174.2000
CO	-0.002590	0.000664	-0.005890	-0.049900	-0.086770	-0.055280	-0.113200	-0.395200	-5.850000	-5.977000	-5.762000	-5.846000
H2	-0.000038	-0.000038	-0.000038	-0.000038	-0.000052	-0.000052	-0.000052	-0.000052	-0.001845	-0.001845	-0.001845	-0.001845
HCl	-0.002540	0.002210	-0.002590	-0.001990	-0.044720	0.001009	-0.044930	-0.051700	-232.3000	-232.4000	-232.3000	-232.3000
HF	-0.000522	0.000017	-0.000516	-0.001290	-0.002420	0.003924	-0.002357	-0.008764	-4.557000	-4.602000	-4.557000	-4.536000
LiCl	0.008630	0.004440	0.339000	0.321000	-0.037450	0.053880	1.280000	15.79000	-233.1000	-233.4000	-235.0000	-293.5000
LiF	-0.084700	-0.005310	-1.260000	3.150000	-0.427500	-0.068940	19.59000	30.31000	-4.842000	-5.142000	-125.0000	-133.8000
LiH	-0.006580	0.000432	-0.282000	-0.046500	-0.007829	0.000659	-0.323000	-0.053670	-0.714700	-0.745700	0.029860	-0.636800
N2	0.002890	-0.007660	-0.004100	-0.015000	-0.024510	-0.097470	-0.067620	-0.230800	-5.835000	-5.611000	-5.723000	-4.992000
NaH	0.035800	-0.007590	-1620.000	0.040300	-0.022320	-0.131400	-3561.000	-0.004582	-300.4000	-300.3000	14370.00	-300.3000
MAE =	0.015242	0.003665	0.211570 [a]	0.491602	0.091597	0.084541	2.407373 [a]	6.294477	102.4501	102.4780	93.91930 [a]	115.0113

[a] excluding NaH

Bibliography

[1] R. A. Poirier, J. W. Hollett, and P. L. Warburton. MUNgauss, Memorial University, Chemistry Department, St. Johns, NL, A1B 3X7. With contributions from I. E. Awad, A. Alrawashdeh, J. P. Becker, J. Besaw, S. D. Bungay, F. Colonna, A.El-Sherbiny, T. Gosse, D. Keefe, A. Kelly, D. Nippard, C. C. Pye, D. Reid, K.Saputantri, M. Shaw, M. Staveley, O. Stueker, Y. Wang, and J. Xidos.

[2] Wolfram Research, Inc. Mathematica 11.2. URL `https://www.wolfram.com`.

[3] J. E. Besaw, P. L. Warburton, and R. A. Poirier. Atoms and Bonds in Molecules: Topology and Properties. *Theoretical Chemistry Accounts*, 134(10):1–15, 2015.

[4] A. D. Becke. A Multicenter Numerical Integration Scheme for Polyatomic Molecules. *The Journal of Chemical Physics*, 88(4):2547–2553, 1988.

[5] A. A. El-Sherbiny and R. A. Poirier. Comprehensive Study of Some Well-Known Molecular Numerical Integration Methods. *Canadian Journal of Chemistry*, 87(10): 1313–1321, 2009.

[6] P. M. W. Gill and S.-H. Chien. Radial Quadrature for Multiexponential Integrands. *Journal of Computational Chemistry*, 24(6):732–740, 2003.

Chapter 4

Rigid-body Transformations

"As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality."

— Albert Einstein

A rigid-body transformation preserves distances between every pair of points, in which the shape and size of the object do not change [1]. The rigid-body transformations include translations, rotations, reflections, or their combination. In this chapter we will give a brief introduction about rigid-body transformations.

4.1 Rotation and Translation Matrices

In this section, we will analyze the two dimensional (2D) and 3D transformations between two coordinate systems (one called global (fixed) and the other called local (movable)). We will also derive the translation and rotation matrices. Then we will use both of these matrices to translate, rotate and reflect a set of points.

Translation

In Figure 4.1 the translation of point P in 2D space between two coordinate systems (global and local) can be described as follow

$$\mathbf{P}_G = \mathbf{L}_G + \mathbf{P}_l \qquad ; \qquad \mathbf{P}_l = \mathbf{P}_G - \mathbf{L}_G \tag{4.1}$$

where $\mathbf{P}_G$ and $\mathbf{P}_l$ represent the vectors of point P in global and local coordinate systems respectively, and $\mathbf{L}_G$ is the vector representing the amount and direction to translate the local coordinate system to the global coordinate system. Each component of $\mathbf{P}_G$, $\mathbf{P}_l$, and

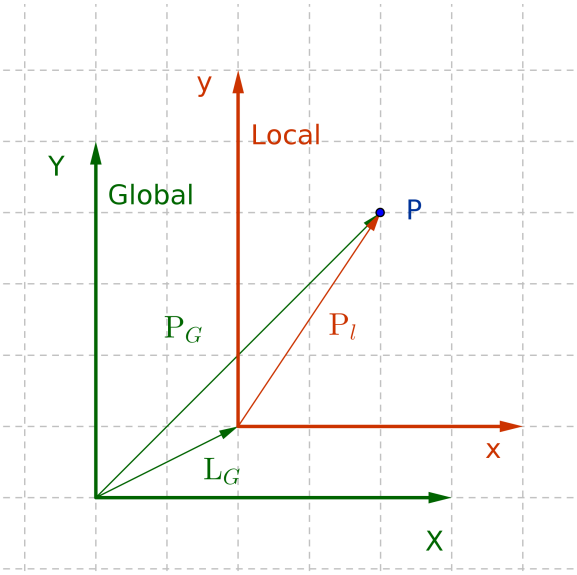


Figure 4.1: 2D translation of a coordinate system (local) to another coordinate system (global).

$\mathbf{L}_G$ can be written as the following 2D (Equation 4.2) and 3D (Equation 4.3) matrices,

$$\begin{bmatrix} \mathbf{P}_X \\ \mathbf{P}_Y \end{bmatrix} = \begin{bmatrix} \mathbf{L}_X \\ \mathbf{L}_Y \end{bmatrix} + \begin{bmatrix} \mathbf{P}_x \\ \mathbf{P}_y \end{bmatrix} \quad ; \quad \begin{bmatrix} \mathbf{P}_x \\ \mathbf{P}_y \end{bmatrix} = \begin{bmatrix} \mathbf{P}_X \\ \mathbf{P}_Y \end{bmatrix} - \begin{bmatrix} \mathbf{L}_X \\ \mathbf{L}_Y \end{bmatrix} \tag{4.2}$$

$$\begin{bmatrix} \mathbf{P}_X \\ \mathbf{P}_Y \\ \mathbf{P}_Z \end{bmatrix} = \begin{bmatrix} \mathbf{L}_X \\ \mathbf{L}_Y \\ \mathbf{L}_Z \end{bmatrix} + \begin{bmatrix} \mathbf{P}_x \\ \mathbf{P}_y \\ \mathbf{P}_z \end{bmatrix} \quad ; \quad \begin{bmatrix} \mathbf{P}_x \\ \mathbf{P}_y \\ \mathbf{P}_z \end{bmatrix} = \begin{bmatrix} \mathbf{P}_X \\ \mathbf{P}_Y \\ \mathbf{P}_Z \end{bmatrix} - \begin{bmatrix} \mathbf{L}_X \\ \mathbf{L}_Y \\ \mathbf{L}_Z \end{bmatrix} \tag{4.3}$$

where $\mathbf{P}_X$, $\mathbf{P}_Y$, and $\mathbf{P}_Z$ are the components of the vector $\mathbf{P}_G$; $\mathbf{P}_x$, $\mathbf{P}_y$, and $\mathbf{P}_z$ are the components of the vector $\mathbf{P}_l$; and $\mathbf{L}_X$, $\mathbf{L}_Y$, and $\mathbf{L}_Z$ are the components the vector of $\mathbf{L}_G$.

Rotation

The 2D rotation of a coordinate system (local) to another coordinate system (global) is shown in Figure 4.2. Our aim is to analyze the rotation of P from local to global coordi-

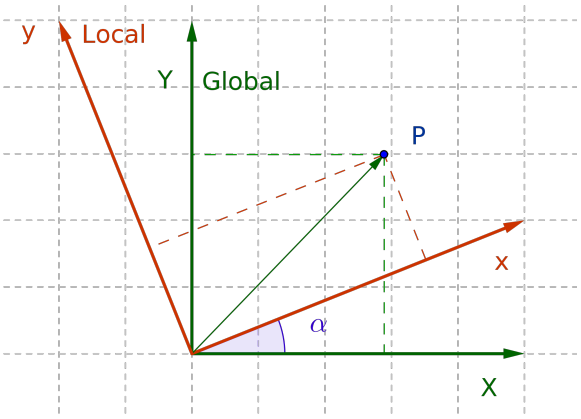


Figure 4.2: 2D rotation of a coordinate system (local) to other coordinate system (global).

nate systems and vice versa. To achieve this, a trigonometry method is used as shown in Figure 4.3. From Figure 4.3a, one can express the P components in the local coordinate system in terms of the global coordinate system as,

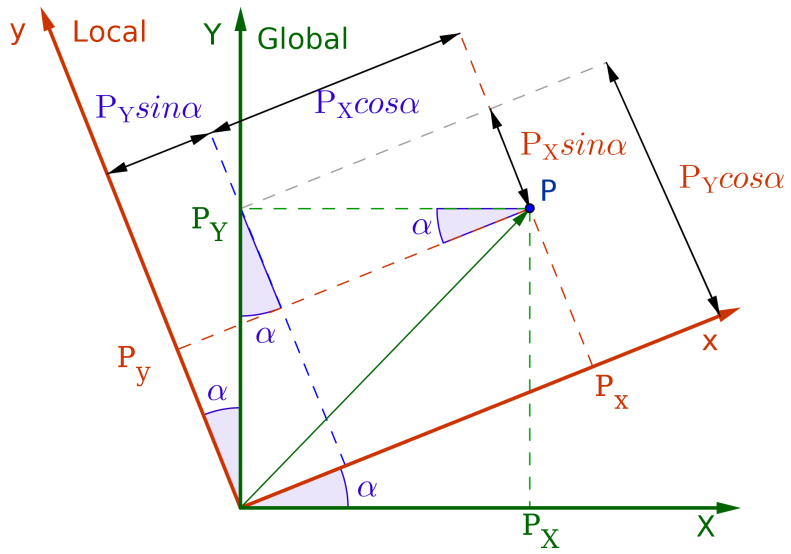
$$\begin{aligned} \mathbf{P}_x &= \mathbf{P}_X \cos\alpha + \mathbf{P}_Y \sin\alpha \\ \mathbf{P}_y &= -\mathbf{P}_X \sin\alpha + \mathbf{P}_Y \cos\alpha \end{aligned} \tag{4.4}$$

the above equations can be written in a matrix form as,

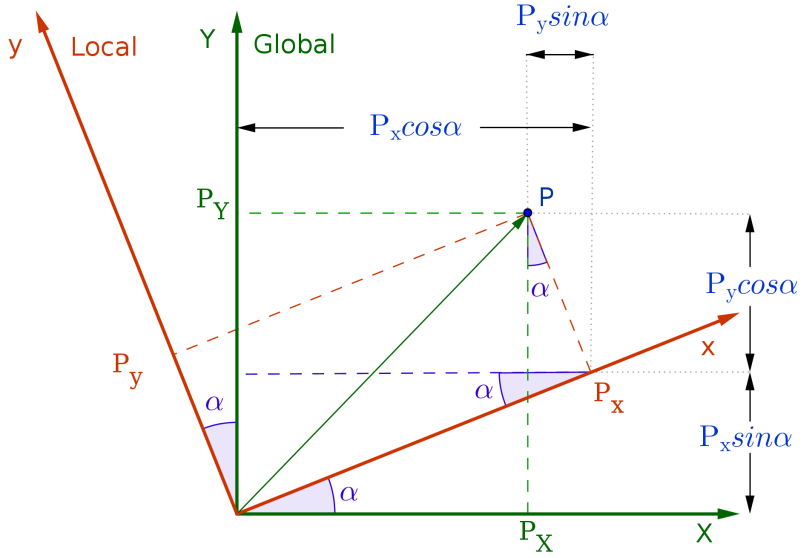
$$\begin{bmatrix} \mathbf{P}_x \\ \mathbf{P}_y \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \mathbf{P}_X \\ \mathbf{P}_Y \end{bmatrix} \tag{4.5}$$

or simply as,

$$\mathbf{P}_l = R_{lG}\mathbf{P}_G \qquad \text{where,} \qquad R_{lG} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \tag{4.6}$$



(a) Point **P** coordinate at the local coordinate system in terms of global coordinate system



(b) Point **P** coordinate at the global coordinate system in terms of local coordinate system

Figure 4.3: Express the point **P** in different coordinates (global and local) only knowing one coordinate system and the angle of rotation.

where, R_{lG} is a 2D rotation matrix to rotate P from global to local coordinate system. Likewise, using Figure 4.3b one can express the P components in the global coordinate system in terms of the local coordinate system as,

$$\begin{aligned} \mathbf{P}_X &= \mathbf{P}_x \cos\alpha - \mathbf{P}_y \sin\alpha \\ \mathbf{P}_Y &= \mathbf{P}_x \sin\alpha + \mathbf{P}_y \cos\alpha \end{aligned} \tag{4.7}$$

where the matrix form is,

$$\begin{bmatrix} \mathbf{P}_X \\ \mathbf{P}_Y \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \mathbf{P}_x \\ \mathbf{P}_y \end{bmatrix} \tag{4.8}$$

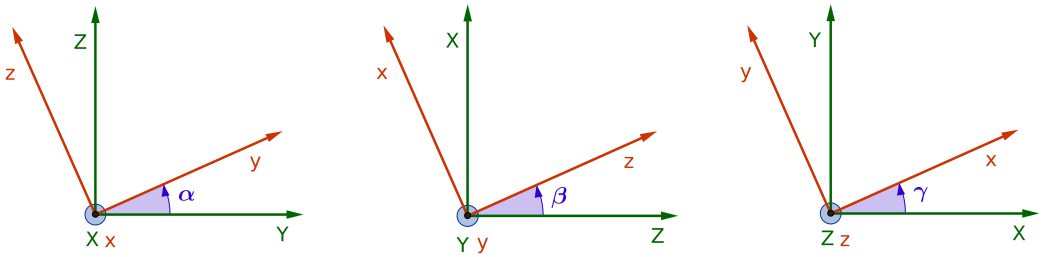
The last equation can be written as,

$$\mathbf{P}_G = R_{Gl}\mathbf{P}_l \qquad \text{where,} \qquad R_{Gl} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \tag{4.9}$$

where R_{Gl} is a 2D rotation matrix to rotate the point P from local to global coordinate systems. A nice property of the rotation matrix is that the products of R_{Gl} and R_{lG} gives the identity matrix.

$$R_{Gl}R_{lG} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I \tag{4.10}$$

To derive the rotation matrix in the 3D coordinate system, the rotation can be divided into rotations around X, Y, and Z axes as shown in Figure 4.4.



(a) Rotation around X axis **(b)** Rotation around Y axis **(c)** Rotation around Z axis

Figure 4.4: Rotation of 3D coordinate system around X, Y and Z axes

One can express the point P components in 3D coordinate system as follows,

$$\begin{aligned} \mathbf{P}_X &= R_{Gl}^X \mathbf{P}_x \\ \mathbf{P}_Y &= R_{Gl}^Y \mathbf{P}_y \\ \mathbf{P}_Z &= R_{Gl}^Z \mathbf{P}_z \end{aligned} \tag{4.11}$$

where R_{Gl}^X, R_{Gl}^Y, and R_{Gl}^Z are the rotation matrices around X, Y, and Z axes, respectively,

that are defined as,

$$R_{Gl}^{X} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{4.12}$$

$$R_{Gl}^{Y} = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \tag{4.13}$$

$$R_{Gl}^{Z} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

The 3D rotation matrix R_{Gl}^{XYZ} is the product of R_{Gl}^{X}, R_{Gl}^{Y}, and R_{Gl}^{Z}.

$$\begin{aligned} R_{Gl}^{XYZ} &= R_{Gl}^{X} R_{Gl}^{Y} R_{Gl}^{Z} \\ &= \begin{bmatrix} \cos\beta\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \cos\beta\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma \\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\beta \end{bmatrix} \end{aligned} \tag{4.15}$$

Rotation and Translation

The rotation and translation can be done with two steps: first rotating then translating the coordinate system. Figure 4.5 shows the rotation and translation of 2D coordinate system

from local to global coordinate system. Mathematically the rotation and translation are

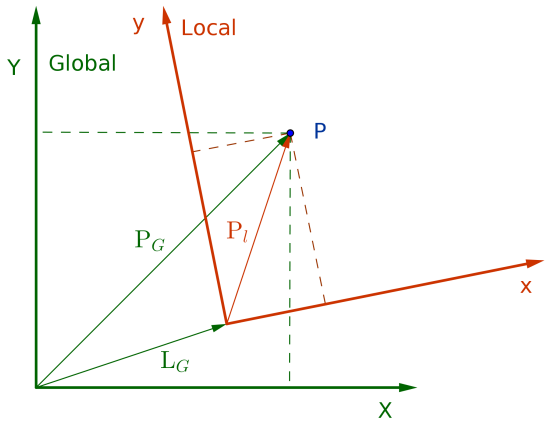


Figure 4.5: 2D translation and rotation of a coordinate system (local) to other coordinate system (global).

written as follows,

$$\mathbf{P}_G = R_{Gl}\mathbf{P}_l + \mathbf{L}_G \tag{4.16}$$

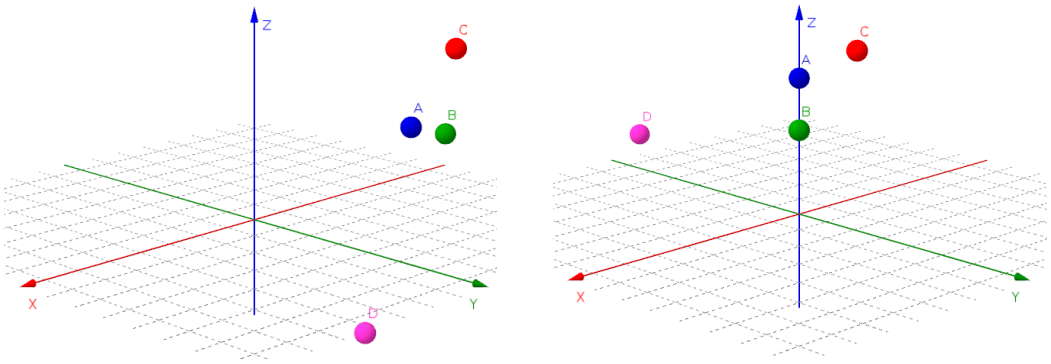
However, if the global coordinate system is rotated and translated to local coordinate system, then we should do the translation firstly and then do the rotation of the coordinate system.

$$\mathbf{P}_l = R_{lG}(\mathbf{P}_G - \mathbf{L}_G) \tag{4.17}$$

4.2 Optimal Rotation and Translation

Points matching, also known as a point set registration, is the process of finding the best transformation that aligns two sets of points. A large number of algorithms were developed to compute the rigid transformations between two sets of data [2–6]. These algorithms are trying to find rotation and translation matrices that align two sets of points. In this section,

we will find the best rotation and translation matrices to superimpose dataset A to dataset B (Figure 4.6) using the singular value decomposition (SVD) algorithm [4].



(a) Dataset A positions **(b)** Dataset B positions

Figure 4.6: Two data sets A and B with same number of points.

Assume that we have N points in a rigid body and let $\mathbf{A} = \{\mathbf{A}_1, \mathbf{A}_2, ..., \mathbf{A}_n\}$ be the 3D positions of these points before the movement and $\mathbf{B} = \{\mathbf{B}_1, \mathbf{B}_2, ..., \mathbf{B}_n\}$ be the positions after the movement. One can use the following equation to transform $\mathbf{A}$ into $\mathbf{B}$,

$$\mathbf{B} = \mathbf{L} + \mathbf{R}\mathbf{A} \tag{4.18}$$

where $\mathbf{R}$ and $\mathbf{L}$ are the rotation and translation matrices respectively. The steps to compute the optimal $\mathbf{R}$ and $\mathbf{L}$ matrices are:

- Computing the centered vectors of $\mathbf{A}$ and $\mathbf{B}$ datasets.
- Finding the optimal rotation matrix $\mathbf{R}$.
- Finding the translation matrix $\mathbf{L}$.

Computing the centered vectors

Calculating the centered vectors of **A** and **B** datasets is an easy task, and can be done by averaging over all points in the two datasets.

$$\mathbf{A}_c = \frac{1}{N}\sum_{i=1}^{N}\mathbf{A}_i \qquad ; \qquad \mathbf{B}_c = \frac{1}{N}\sum_{i=1}^{N}\mathbf{B}_i \tag{4.19}$$

Where $\mathbf{A}_c$ and $\mathbf{B}_c$ are the centered vectors of **A** and **B** datasets respectively, and N is number of points in the dataset.

Finding the optimal rotation matrix R

In order to find the optimal **R** matrix, we consider the SVD algorithm. SVD is the factorization or decomposition of matrix **M** into the product of three other matrices,

$$\mathbf{M} = \mathbf{USV}^T \tag{4.20}$$

where **M** is an $m \times n$ matrix, **U** is an $m \times m$ unitary matrix, **S** is a diagonal $m \times n$ matrix with non-negative real numbers on the diagonal, **V** is an $n \times n$ unitary matrix, and $\mathbf{V}^T$ is the transpose of matrix **V**. To find the optimal rotation we first need to set up a new matrix as follows,

$$\mathbf{H} = \sum_{i=1}^{N}\left(\mathbf{A}_i - \mathbf{A}_c\right)\left(\mathbf{B}_i - \mathbf{B}_c\right)^T \tag{4.21}$$

H is always a 3×3 matrix. The factorization of the **H** matrix using SVD method gives,

$$\mathbf{H} = \mathbf{USV}^T. \tag{4.22}$$

The rotation matrix $\mathbf{R}$ can be calculated as follows,

$$\mathbf{R} = \mathbf{V}\mathbf{U}^T \tag{4.23}$$

In some cases the reflection matrix is included within the $\mathbf{R}$ matrix.

Finding the translation matrix L

The translation matrix $\mathbf{L}$ can be computed as follows,

$$\mathbf{L} = -\mathbf{R}\mathbf{A}_c + \mathbf{B}_c \tag{4.24}$$

$\mathbf{L}$ is always a 3×1 matrix. In calculating $\mathbf{L}$, we first rotate $\mathbf{A}_c$ using $\mathbf{R}$ then translate by $\mathbf{B}_c$.

4.3 Standard Cartesian coordinates

In this section, we consider the transformation of the original Cartesian coordinates $\mathbf{P}^0 = \{\mathbf{P}_1^0, \mathbf{P}_2^0, \mathbf{P}_3^0, \mathbf{P}_4^0\}$ of a dataset into new suggested standard Cartesian coordinates $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4\}$. The new coordinates are explained in Table 4.1

Table 4.1: Suggested standard Cartesian coordinates

Point number	x	y	z
$\mathbf{P}_1$	0	0	0
$\mathbf{P}_2$	0	0	$+z_2$
$\mathbf{P}_3$	0	$+y_3$	z_3
$\mathbf{P}_4$	$+x_4$	y_4	z_4

In the new coordinates, the points $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ are located in the origin, on the positive z axis, and in yz plane with a positive y coordinate, respectively, and $\mathbf{P}_4$ is any point located in the xyz space with a positive x coordinate. Clearly, z_2 can be easily computed, which is the distance between the points: $\mathbf{P}_1^0 = (x_1^0, y_1^0, z_1^0)$ and $\mathbf{P}_2^0 = (x_2^0, y_2^0, z_2^0)$,

$$z_2 = \overline{P_1^0 P_2^0} = \sqrt{(x_2^0 - x_1^0)^2 + (y_2^0 - y_1^0)^2 + (z_2^0 - z_1^0)^2}. \tag{4.25}$$

The three non-collinear points P_1^0, P_2^0, and P_3^0 form the triangle $P_1^0 P_2^0 P_3^0$ as shown in Figure 4.7. In this figure, the altitude from side $\overline{P_1^0 P_2^0}$ represents the coordinate y_3. Since all

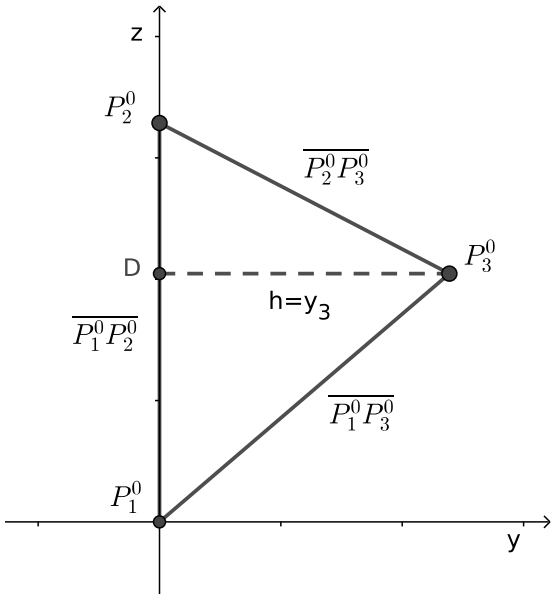


Figure 4.7: Triangle $P_1^0 P_2^0 P_3^0$, with altitude h from side $\overline{AB}$.

sides of this triangle (i.e., $\overline{P_1^0 P_2^0}$, $\overline{P_1^0 P_3^0}$, and $\overline{P_2^0 P_3^0}$) are known and its semiperimeter s is given by

$$s = \frac{\overline{P_1^0 P_2^0} + \overline{P_1^0 P_3^0} + \overline{P_2^0 P_3^0}}{2}, \tag{4.26}$$

the value of y_3 can be obtained using

$$y_3 = 2\frac{\sqrt{s(s-\overline{P_1^0P_2^0})(s-\overline{P_1^0P_3^0})(s-\overline{P_2^0P_3^0})}}{\overline{P_1^0P_2^0}}. \tag{4.27}$$

Because the values of y_3 and $\overline{P_1^0P_3^0}$ are known, z_3 can be calculated using Pythagorean triple rule, where

$$z_3 = \sqrt{\overline{P_1^0P_3^0}^2 - y_3^2} \tag{4.28}$$

Finding the Cartesian coordinates of $\mathbf{P}_4$ is more tricky. As shown in Figure 4.8, the non-

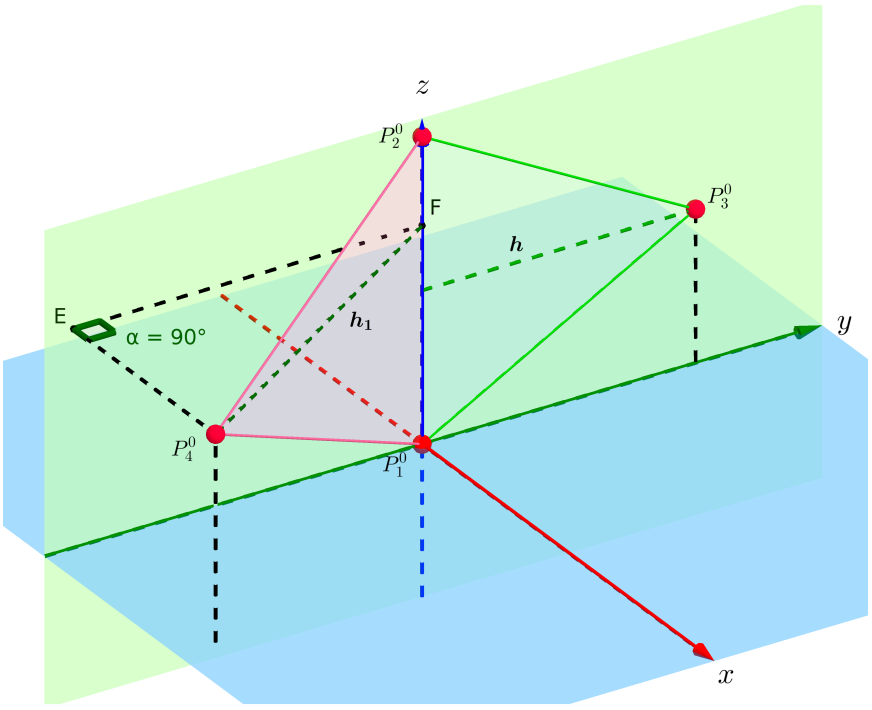


Figure 4.8: The relationship between the points P_1^0, P_2^0, P_3^0, and P_4^0 in the origin coordinates.

.

collinear points P_1^0, P_2^0, and P_3^0 are located on the yz plane of the origin coordinates. The plane can be described by a point and a normal (perpendicular) vector ($\overrightarrow{n}$). The unique

normal vector through P_1^0 and perpendicular to the plane yz can obtained by the cross product,

$$\begin{aligned}\overrightarrow{n} &= \overrightarrow{P_1^0 P_2^0} \times \overrightarrow{P_1^0 P_3^0} \\ &= \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_2^0 - x_1^0 & y_2^0 - y_1^0 & z_2^0 - z_1^0 \\ x_3^0 - x_1^0 & y_3^0 - y_1^0 & z_3^0 - z_1^0 \end{vmatrix}\end{aligned} \tag{4.29}$$

Because $\overrightarrow{n}$ is orthogonal to the plane, it is also orthogonal to any vector in the plane. Now, assume that $P^0 = (x^0, y^0, z^0)$ is any point in the plane, then the vector $\overrightarrow{P_1^0 P^0} = \langle x^0 - x_1^0, y^0 - y_1^0, z^0 - z_1^0 \rangle$ is located completely in the plane and it is perpendicular to $\overrightarrow{n}$. And because the dot product of two orthogonal vectors gives zero, the equation of the plane can be obtained as,

$$\overrightarrow{n} . \langle x^0 - x_1^0, y^0 - y_1^0, z^0 - z_1^0 \rangle = 0 \tag{4.30}$$

where Equation 4.30 is called the vector equation of the plane. By solving the above equation one obtains,

$$s_1 x^0 + s_2 y^0 + s_3 z^0 + d = 0 \tag{4.31}$$

where

$$s_1 = (y_2^0 - y_1^0)(z_3^0 - z_1^0) - (z_2^0 - z_1^0)(y_3^0 - y_1^0)$$

$$s_2 = (z_2^0 - z_1^0)(x_3^0 - x_1^0) - (x_2^0 - x_1^0)(z_3^0 - z_1^0)$$

$$s_3 = (x_2^0 - x_1^0)(y_3^0 - y_1^0) - (y_2^0 - y_1^0)(x_3^0 - x_1^0)$$

$$d = -(s_1 x_1^0 + s_2 y_1^0 + s_3 z_1^0)$$

In Figure 4.8, the shortest distance between $s_1x^0 + s_2y^0 + s_3z^0 + d = 0$ (the plane) and the point $P_4^0 = (x_4^0, y_4^0, z_4^0)$ can be computed using the law of distance from a point to a plane,

$$\left|\overline{P_4^0E}\right| = \frac{(s_1x_4^0 + s_2y_4^0 + s_3z_4^0 + d)}{\sqrt{s_1^2 + s_2^2 + s_3^2}} \tag{4.32}$$

It is clear as shown in Figure 4.8 that the value of x_4 equals the distance $\left|\overline{P_4^0E}\right|$. In Figure 4.8, h_1 is the altitude of the triangle $P_1^0P_2^0P_4^0$ from side $\overline{P_1^0P_2^0}$. Similar to the calculation of y_3 in Equation 4.27, one can calculate h_1. Using h_1, the values of y_4 and z_4 can be calculated using the Pythagorean triple rule as,

$$y_4 = \sqrt{h_1^2 - x_4^2} \tag{4.33}$$

$$z_4 = \sqrt{\left|\overline{P_1^0P_4^0}\right|^2 - h_1^2} \tag{4.34}$$

Bibliography

[1] W. B. Heard. *Rigid Body Mechanics, Mathematics, Physics and Applications*. WILEY-VCH Verlag GmbH & Co. KGaA, 2006. ISBN 9783527406203.

[2] A. Myronenko and X. Song. Point Set Registration: Coherent Point Drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, December 2010.

[3] J. W. M. Nissink, M. L. Verdonk, J. Kroon, T. Mietzner, and G. Klebe. Superposition of Molecules: Electron Density Fitting by Application of Fourier Transforms. *Journal of Computational Chemistry*, 18(5):638–645.

[4] P. J. Besl and N. D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.

[5] C. F. F. Karney. Quaternions in Molecular Modeling. *Journal of Molecular Graphics and Modelling*, 25(5):595–604, 2007.

[6] Finding Optimal Rotation and Translation Between Corresponding 3D Points — Nghia Ho. URL `http://nghiaho.com/?page_id=671`.

Chapter 5

Molecular Fragmentation

"It would be possible to describe everything scientifically, but it would make no sense; it would be without meaning, as if you described a Beethoven symphony as a variation of wave pressure."

— Albert Einstein

Our method of computing the molecular properties of the target molecules includes three general steps: (i) generating small molecules (fragments) from molecules of interest. Each fragment defined a specific atom in molecule type based on its neighbor (ii) storing some properties of the fragments in the database, (iii) using these properties to compute the molecular properties of the target molecule. Figure 5.1 illustrates the main steps used by code written within the MUNgauss package to generate the fragments, compute their properties, and store these values in the database:

- Generate unique fragments from the molecule of interest. The size of the fragments can be controlled through the code.
- Check the database for the availability of each fragment.
- Generating the missing fragments, and complete the valency with hydrogen atoms.
- Optimize the structure of these fragments (optional).
- Perform QM calculations on these fragments.
- Compute the atomic properties for the target atom in these fragments.
- Store these atomic properties in the database with all other necessary information, such as, basis set, grid type, partitioning weight name, and the coordinates of these fragments.
- Store the grid points with all other necessary information at each point, such as: the coordinates of these grids, electron density, partitioning weights, and angular weights.

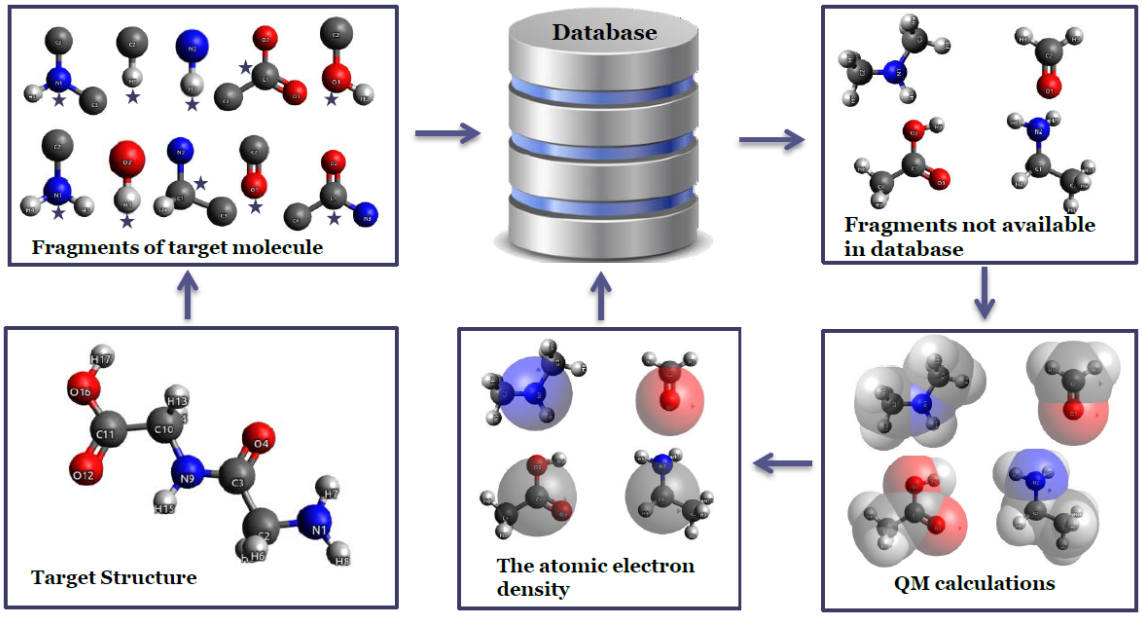


Figure 5.1: Describing the process of generating the fragments, computing the fragments properties, and storing these properties in the database.

In this chapter, we will use graph theory principles to generate fragments from molecules, illustrate our method to complete the valency of the fragment to build a real closed-shell molecule, explain the method for giving unique symbols and labelling the fragment atoms, and describe the structure of the database.

5.1 Fragment Generation Principle

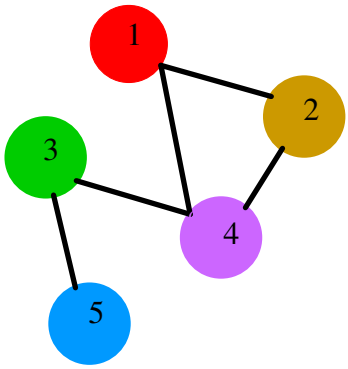
In this section, the principle of generating the fragment structures from the target molecule using graph theory will be discussed. An interesting property of adjacency matrices that allows us generate the fragment structures will be explained below.

5.1.1 Adjacency Matrix

In graph theory, an adjacency matrix is a square matrix, in which the elements indicate whether pairs of vertices are connected or not in a finite graph. Given a graph with n nodes, the adjacency matrix $A_{n\times n}$ has entries $A_{ij} = 1$, if i is adjacent to j, and $A_{ij} = 0$ otherwise. Because the labels of a graph may be selected in different ways, in general there are multiple possible adjacency matrices for a given graph [1].

Interesting things happen when the m power of the adjacency matrix is taken. The m power of an adjacency matrix has the property that represents the number of walks of length m of any pairs of vertices [2].

For example, Figure 5.2 shows a graph with five nodes. The adjacency matrix $A_{5\times5}$ for the graph is also shown in the figure. The matrix is symmetric with dimensions of 5×5, and its elements are binary numbers 0 or 1.



$$A_{5\times5} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Figure 5.2: Five nodes graph and its adjacency matrix ($A_{5\times5}$).

The square of adjacency matrix $A_{5\times5}$ gives a new matrix $A^2_{5\times5}$ with the same dimension (in

this case 5×5) but with edges connecting vertices with path length equal 2.

$$A^2_{5\times5} = \begin{pmatrix} 2 & 1 & 1 & 1 & 0 \\ 1 & 2 & 1 & 1 & 0 \\ 1 & 1 & 2 & 0 & 0 \\ 1 & 1 & 0 & 3 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{5.1}$$

Now by replacing the non-binary numbers with zeros, and removing all self-path connections by making the diagonal elements equal zeros, one obtains,

$$A^{2\prime}_{5\times5} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{5.2}$$

Because of the possibility of rings within the graph, there will be the possibility for different paths between pairs of vertices. To eliminate this possibility, the last matrix $A^{2\prime}_{5\times5}$ is modified to $A^{2\prime\prime}_{5\times5}$ by replacing all the ones that shared with $A_{5\times5}$ to zeros.

$$A^{2\prime\prime}_{5\times5} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{5.3}$$

The $A^{2\prime\prime}_{5\times5}$ matrix shows the pair of vertices that has a distance of exactly 2. The same principle can be done for higher order (i.e., distance of 3 and more).

The adjacency matrix is used to generate atomic fragments from a real molecule to any

desired m neighbours. For example, Figure 5.3 shows the structures of the unique first-neighbour fragments from the glycine-glycine (GlyGly) dimer. The valency of each fragment should be completed to build a real closed-shell fragment. The method of completing the valencies of the fragment will be discussed in the following section.

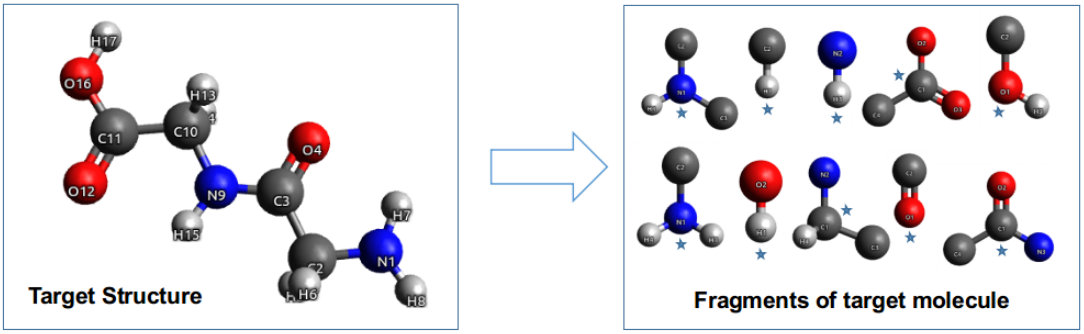


Figure 5.3: GlyGly dimer unique fragments for the first neighbor atom. The valency of the fragments still need to be completed.

5.1.2 Completing the Valency of the Fragment

To complete each missing valency of a fragment of interest, the fragment is translated and rotated until each of its terminal atoms (the atoms that need their valencies to be completed) is on the positive z axis, and the first neighbour atom has to be located at the origin of the coordinate system. If there is another atom connected to the first neighbour atom, it should be located in the yz plane. In order to determine the valency of the terminal atom, the order of the bond between the terminal atom and its first neighbour atom is calculated. The type of the bond (single, double, ...) is then determined from the bond order as illustrated in Table 5.1. The classifications shown in this table is used in our entire research, where the bond orders are calculated using the Mayer method [3]. For example, Figure 5.4 shows one

Table 5.1: The relation between the type of bond and the range of bond order

Bond Type	Bond order range
Partial single bond	[0, 0.5)
Single bond	[0.5, 1.13)
Aromatic double bond	[1.13, 1.6)
Double bond	[1.6, 2.2)
Aromatic triple bond	[2.2, 2.7)
Triple bond	[2.7, 3.2)

of the GlyGly fragment (HOC) in which the valency of the terminal (C) atom needs to be completed. Firstly, the initial coordinates of the HOC fragment are translated and rotated until the C atom is on the positive z axis and the first neighbour atom (O) is located at the origin and H atom is located at yz plane (Figure 5.4). Secondly, the CO bond is determined to be a "single bond" by calculating the CO bond order of molecule of interest. Thus, the carbon atom needs three hydrogen atoms to complete its valency. Because the locations and

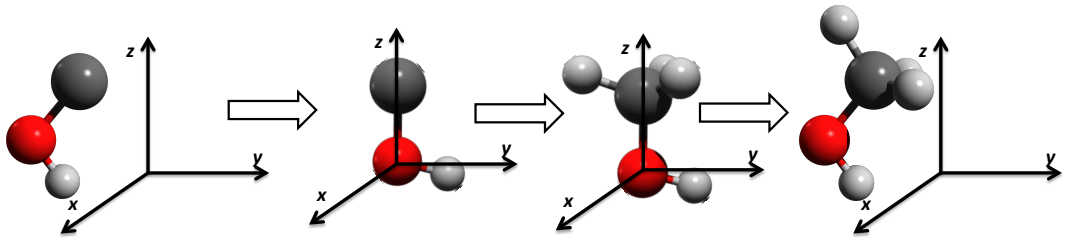


Figure 5.4: Completing the carbon atom valency by adding three hydrogen atoms. The oxygen is located at the origin, the carbon on the positive z axis, where CO has a single bond.

bond lengths of other atoms are known, the locations of these hydrogen atoms can easily be determined with minimum steric effect. Figure 5.5 shows another example to complete the valency for the terminal atoms. In this example, the bond between the target C atom and

the other C atom is a double bond, and because locations of the existing H atoms are in *yz* plane, the two new H atoms should be added in the same plane to form a planar structure.

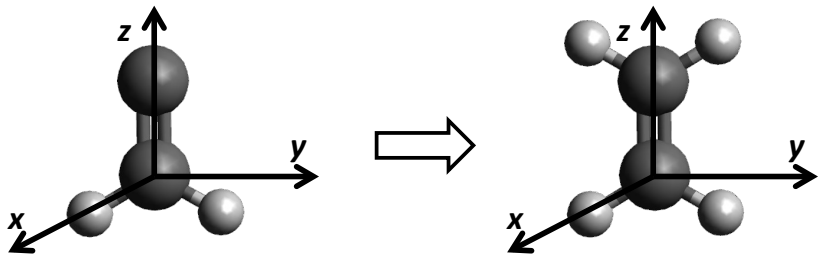


Figure 5.5: Completing the carbon atom valency for ethene by adding two hydrogen atoms, the target carbon is located at the positive *z* axis, the other carbon atom at the origin.

Table 5.2 shows the default Cartesian coordinates for the H atoms that have been used in our code to complete the valency of terminal atoms. Not all the bond types (such as aromatic bonds) are included within the code because they need special coding which hopefully will be done in the future. Optionally, the molecular coordinates can be stored in the database before optimization or after they are optimized. To store these molecules in the database, a special symbol (unique symbol for the target atom within the fragment) is needed.

5.2 Unique Symbol

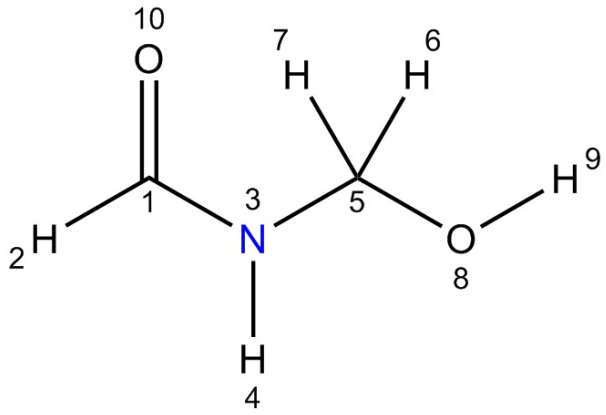
As mentioned in the previous section, a unique symbol (i.e., representation) for an atom within a fragment is very important in order to store the required information about that atom (such as its coordinates, connectivity, ...) in the database and to query this information when it is needed. Many representations of molecules exist in the literature. The Simplified Molecular-Input Line-Entry System (SMILES) [4] and SYBYL Line Notation (SLN) [5, 6]

Table 5.2: The default Cartesian coordinates for the hydrogen atoms to complete the valency of terminal atoms

Bond Type	Atom		Valance atoms coordinates (bohr)		
			x	y	z - BL[1]
Single bond	Carbon (-CH_3)	1H	0.00000000	-1.93061734	0.68284856
		2H	1.67199927	0.96543892	0.68284856
		3H	-1.67199914	0.96557695	0.68284856
	Nitrogen (-NH_2)	1H	0.00000000	-1.80992957	0.55955062
		2H	-1.64331915	0.75857027	0.55955062
	Oxygen (-OH)	1H	0.00000000	-1.81015167	0.46813705
double bond	Carbon (=CH_2)	1H	0.00000000	1.73196206	1.06759470
		2H	0.00000000	-1.73196206	1.06759470
	Nitrogen (=NH)	1H	0.00000000	0.00000000	1.89445030
Triple bond	Carbon (≡CH)	1H	0.00000000	0.00000000	2.03456410

[1] BL: bond length between terminal and first neighbour atoms.

are used for describing the structure of molecules in the form of line notations. Molfile [7] is a file format that is used for the storage of two- or three- dimensional coordinates for molecules. Structure-Data File (SDF) [7] contains a collection of properties related to the molecular structure. In addition, the International Chemical Identifier (InChI) representation, which was recommended by International Union of Pure and Applied Chemistry (IUPAC), provides a canonical representation of the molecule and offers different layers of description for the chemical structure [8]. Since we need a representation describing the target atom within a fragment rather than describing the entire molecule, all previous representations are not suitable for our research. The suggested symbol (representation) for



1	6(**8**()**7**(6,1)**1**())
2	1(**6**(8,7))
3	7(**6**(8,1,1)**6**(8,1)**1**())
4	1(**7**(6,6))
5	6(**8**(1)**7**(6,1)**1**()**1**())
6	1(**6**(8,7,1))
7	1(**6**(8,7,1))
8	8(**6**(7,1,1)**1**())
9	1(**8**(6))
10	8(**6**(7,1))

Figure 5.6: The second level fragment symbols within a molecule. The symbols show how the target atoms are connected for the second level, but does not show the coordinates of these atoms.

the atom of interest within its fragment is shown below,

$$Z(z_1, z_2, \dots)$$

where Z is the atomic number of atom of interest and z_i is the atomic number of atom i in the first neighbour level to the atom of interest, where the level is defined as how far through bonds the atom from the target atom. To extend the symbol for the second neighbour level, each atom in the first level is also written with its neighbour atoms. Therefore, the formula for the suggested symbol becomes,

$$Z(z_1(z_1^1, z_2^1, \dots)z_2(z_1^2, z_2^2, \dots)\dots)$$

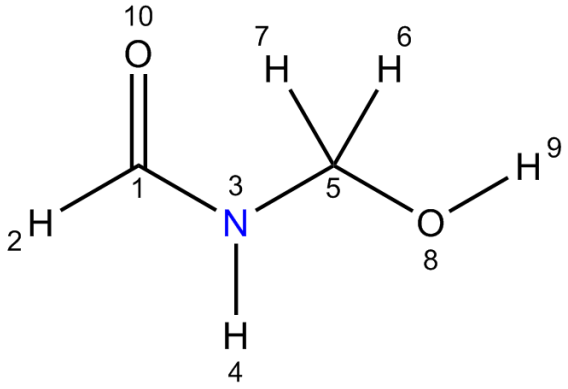
where z_i^1 is the atomic number of the i^{th} atom in the first neighbour level to the atom of z_1 and z_j^2 is the atomic number of the j^{th} atom in the first neighbour level to the atom of z_2. A similar method is applied for higher levels. It is good to mention that each atom should not be included more than once in the symbol and atomic numbers of the atoms in

each neighbour level should be in descending order. For example, Figure 5.6 shows the symbols for ten fragments of a molecule for the second level. As shown in Figure 5.6, two or more fragments can share the same symbol such as fragments 6 and 7. Such fragments are considered to be the same and only one of which is stored in the database.

5.3 Labelling Atoms in a Fragment

In order to build a molecule of interest using fragments from the database, the atoms in the molecule of interest and in the database fragment should have the same labelling. The rules used for labelling the atoms are best illustrated with the example that is given in Figure 5.7. Figure 5.7 shows a fragment with numbering of the atoms as given in the input file. Suppose that the target atom is the nitrogen atom (atom number 3):

- Initially, all atoms in the fragment are given the same index: "the number one".
- Sorting the atoms according to their levels (Criteria 1). The atoms are split into four different levels, two of which are already finalized because they consist of only one atom (atom 3 which is the target atom assigned to level 1, and atom number 9 which is the only atom in level 4).
- Fixing positions of atoms which are unique in a specific level, thus atoms 3 and 9 are fixed to positions 1 and 10, respectively (note that atom 9 takes the position 10 as it is the last atom in last level). So far, each of the second level atoms 1, 4, and 5 has given index 2 and each of the third level atoms 2, 6, 7 ,8 , and 10 has been given index 5 (note: they have been given index 5 because the four previous atoms have been indexed).



Atom number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Initiation	1	1	1	1	1	1	1	1	1	1
Criteria 1	2	5	1	2	2	5	5	5	10	5
Criteria 2	2	7	1	4	2	7	7	5	10	5
Criteria 3	3	7	1	4	2	7	7	5	10	6

Figure 5.7: An example for labeling the fragment's atoms using the suggested rules. In this example the nitrogen atom (atom number 3) is the target atom within the fragment. Not all atoms are sorted (in this case the hydrogen atoms 2, 7, and 8).

- Sorting the atoms according to their atomic number in each level (Criteria 2). The second level atoms are split into two different groups; a group of two C atoms (1 and 5) and a group of one H atom (4), the H group is already finalized and given the position 4. The third level atoms are also split into two different groups; a group consisting of three H atoms (2, 6, and 7), and a group consisting of two O atoms (8 and 10).

- Fixing positions of atoms which are unique in atomic number, thus atom 4 is fixed to position 4.

- Sorting the atoms according to the type of their first neighbour atoms and fixing their

positions (Criteria 3). Each of the two C atoms (1 and 5) has different neighbors; C (5) is connected to two H (6 and 7) and O (8), while C (1) is connected to one H (2) and one O (10), therefore the two carbon atoms are given positions 2 and 3 respectively. Similarly, O (8) is connected to C (5) and H (9), while O (10) is connected only to C (1), therefore they are given positions 5 and 6 respectively.

It is clear that not all terminal atoms are sorted; however, this is not a problem because a few sorted atoms are enough to reach the target of our research. Thus, the already sorted atoms are sufficient to compute the rotation and translation matrices needed to build the molecule of interest.

5.4 Storing in the Database

In this section, we consider the method for storing fragments in the database. In order to store a fragment in the database, Cartesian coordinates for the target atom and its neighbour atoms, as well as the radial grids of the target atom, should be transformed into the standard form (section 4.3). In section 4.3, the transformation of the original Cartesian coordinates with maximum of four points $\mathbf{P}^0 = \{\mathbf{P}_1^0, \mathbf{P}_2^0, \mathbf{P}_3^0, \mathbf{P}_4^0\}$ into the standard Cartesian coordinates $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4\}$ is discussed. As mentioned in the previous section (section 5.3), the target atom has a label = 1 and the other three atoms in the fragment have labels 2, 3, and 4. In the standard coordinates, the target atom ($\mathbf{P}_1$) is located at the origin, the second ($\mathbf{P}_2$), the third ($\mathbf{P}_3$), and the forth ($\mathbf{P}_4$) atoms are located on the positive z axis, in the yz plane with positive y axis, and in xyz space with positive x axis, respectively. In the origin fragment the four atoms have the same labelling (1, 2, 3, and 4) with coordinates of $\mathbf{P}_1^0$, $\mathbf{P}_2^0$, $\mathbf{P}_3^0$, and $\mathbf{P}_4^0$, respectively. Before starting the storing process, the existence of the target fragment in

the database is checked. If the fragment does not exist in the database, it can be stored as follows:

- Determine the optimal **R** and **L** matrices required to transform $\mathbf{P}^0$ into **P** (see section 4.2).
- Use **R** and **L** matrices to transfer all original coordinates of all atoms in the fragment to the standard coordinates and store these coordinates in the database.
- Use **R** and **L** matrices to transfer the radial grid coordinates of the target atom to the standard coordinates and store these coordinates in the database with all required information such as the electron density at that position.

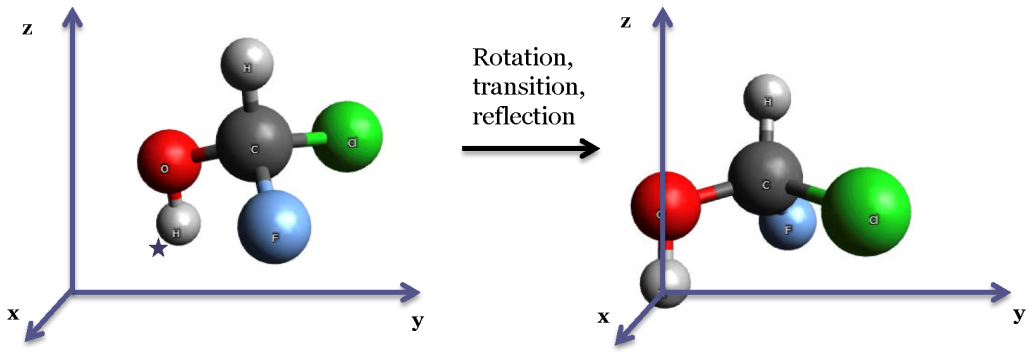


Figure 5.8: Rotation, translation and reflection of the target fragment atoms to obtain the new standard Cartesian coordinates.

Figure 5.8 shows an example for rotation, translation and reflection of a target fragment to obtain the standard coordinates.

5.5 Building the Atomic Electron Density for Molecule of Interest from the Database

In this section, the method that is used for building the target molecule from its atomic contributions in the database is illustrated (Figure 5.9). In order to build the molecular electron density, Cartesian coordinates for the radial grids of the target atom should be transformed from the database into the coordinates of molecule of interest. The following

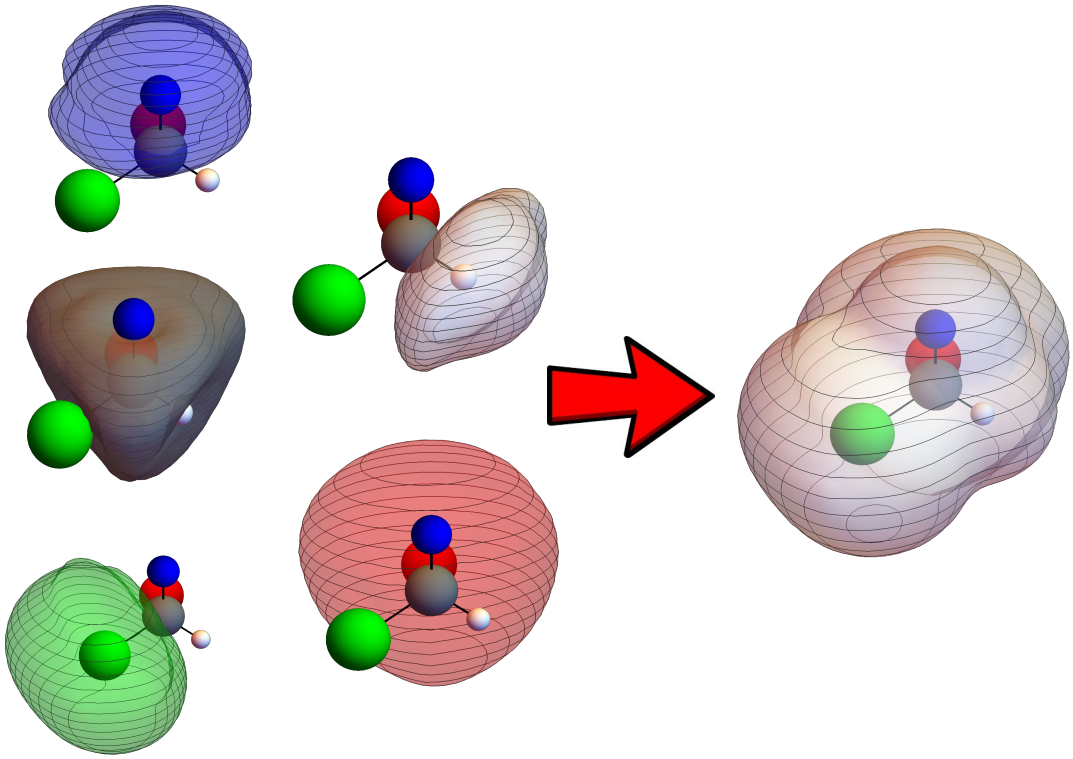


Figure 5.9: Building the molecular electron density for CClBrFH molecule from its atomic contributions.

steps have been used to achieve this goal:

- Search within the database for the availability of each atomic fragment that is needed

in building the molecule of interest. If one of the fragments is not available in the database, the program gives an error message and stops.

- Extract the coordinates of atoms for all fragments in the molecule (these are the original coordinates for the molecule that are available in the input file).

- Determine the optimal **R** and **L** matrices required to transform fragment coordinates in the database into coordinates for the molecule of interest (see section 4.2). Unfortunately, using **R** and **L** matrices gives approximate results in many cases. These approximations are due to the geometry of the fragment for the target atom in the molecule being different from its fragment in the database. In order to solve this problem, the following steps are used:

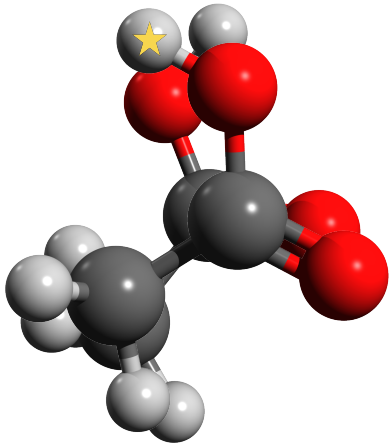
  - The target atom is given a large weight (e.g., 2000 times) and atoms located in level one and two are given weights equal to the cube and the square of their atomic number respectively.

  - Determine the optimal $\mathbf{R}_1$ and $\mathbf{L}_1$ matrices required to transform fragment coordinates in the database into coordinates for the molecule of interest.

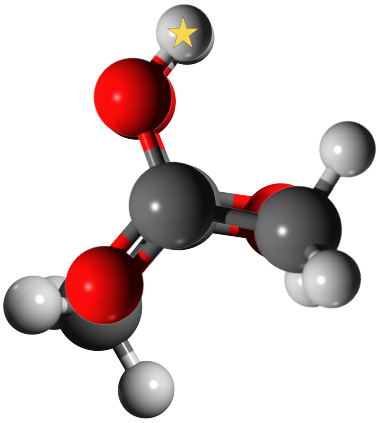
  - Another correction is done by translating $\mathbf{L}_2$ for the fragment atoms to make the target atoms exactly align (as shown in Figure 5.10).

- These $\mathbf{R}_1$, $\mathbf{L}_1$ and $\mathbf{L}_2$ matrices are also applied on the radial grid coordinates in the database.

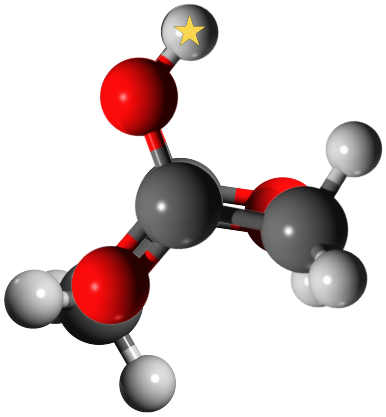
- All the new coordinates for the atoms and radial grids are stored in memory later to perform calculations of the molecular properties.



(a) Before applying the weight



(b) After applying the weight and before the second correction



(c) After applying the weight and the second correction

Figure 5.10: Superimpose the electron density for H atom in two diffrent fragments.

To make the process of extracting the information from the database easier to understand, the database structure will be briefly explained in the next section.

5.6 The Database Structure

Mainly, the database consists of four parts: the index table, the table of atomic coordinates in fragments, atomic properties table, and grid points tables.

5.6.1 Index Table

The index table can improve query performance by allowing applications to more quickly locate the data to retrieve from the database. The index fields contain the fragment information (i.e., index numbers, number of atoms, ...), the target atom, all neighbour atoms, unique symbols, and all the information that is required to build the database file such as (type of grid point (SG1), the number of grid points, basis sets, ...)

5.6.2 The Table of Atomic Coordinates in Fragments

The table of atomic coordinates in fragments contains the Cartesians of all atoms of the fragments (the target atom and its adjacent atoms) in standard form (see Figure 5.11).

index	Atom#	Symbol	Atoms coordinates x	y	z
1	1	N	-0.00000000	-0.00000000	-0.00000000
1	2	C	-0.00000000	0.00000000	2.75105551
1	3	H	-0.00000000	1.77530805	-0.66135862
1	4	H	1.55817246	-0.83684605	-0.67505690
1	5	C	-2.44253065	1.14781001	3.74911504
1	6	H	0.14082685	-1.94831120	3.37532447
1	7	H	1.61112811	1.00202446	3.56343617
1	8	H	-2.49533999	1.10132887	5.79954205
1	9	H	-4.05866791	0.11929414	3.02374509
1	10	H	-2.62572328	3.10852077	3.16403672
2	1	C	-0.00000000	0.00000000	0.00000000
2	2	N	-0.00000000	-0.00000000	2.72673483
2	3	C	0.00000000	2.67370798	-0.98681242
2	4	H	1.66412358	-0.95687825	-0.70841454
2	5	H	-1.63023945	-0.91857121	-0.87412323
2	6	O	0.73389535	2.78558377	-3.38519479
2	7	O	-0.65953675	4.47859262	0.17007957
2	8	H	-1.26242488	1.23880601	3.39503031
2	9	H	-0.42062851	-1.71298806	3.40485449
2	10	H	0.61017964	4.49551026	-3.93316048
3	1	C	0.00000000	-0.00000000	0.00000000
3	2	O	-0.00000000	0.00000000	2.51922813
3	3	O	0.00000000	1.91584741	-1.14523169
3	4	C	0.13718953	-2.58535095	-1.23415937

Figure 5.11: Example of the table of atomic coordinates in fragments.

5.6.3 Atomic Properties Table

The atomic properties table contains the atomic properties such as number of electrons, potential energy, exchange energy, and kinetic energy.

#	Electrons	Vne_Anal	Vne_Num	Vee	Exchange	Kinetic
1	7.17023098	-167.69150164	-168.94829210	44.38568770	-6.64508877	53.99759251
2	6.10412832	-147.49059857	-149.54784203	48.02137025	-5.27840352	37.98931031
3	6.12393590	-156.59233845	-159.43844553	48.70876440	-5.37452974	38.69595392
4	8.14254450	-232.27907844	-231.68841750	66.56720187	-8.20100789	74.21174504
5	0.93848127	-8.64949861	-7.87636617	4.95036302	-0.33245444	0.59184471
6	0.91258133	-7.27782717	-6.56666390	3.43856109	-0.33860744	0.68744836
7	8.01305440	-231.96480014	-231.44892209	60.98631847	-8.19022679	74.10709681
8	0.87780336	-7.75940322	-6.92628950	3.40707809	-0.34436075	0.79917164
9	7.16507596	-178.63941845	-179.96403976	49.35095418	-6.64341230	53.99839778
10	6.12406800	-161.16149114	-163.66636052	53.85300839	-5.28871878	38.02452901
11	6.12272372	-166.08348069	-169.03395019	52.26454970	-5.37412357	38.70963401
12	0.92491218	-10.45701213	-9.40375990	6.03059905	-0.32935389	0.58402255
13	6.12901262	-133.32490844	-135.08251948	39.60376836	-5.25774169	37.85059475
14	0.95506348	-6.71658982	-6.19719802	3.60926891	-0.33401225	0.59382795
15	6.11254657	-180.72207797	-183.47528396	64.67883359	-5.29140246	38.05579564

Figure 5.12: Example of the table of atomic properties.

5.6.4 Grid Points Tables

Grid points tables contain the Cartesian coordinates of the radial grid points, with the atomic properties at a specific grid point (such as, electron density, angular weight for numerical integration, partitioning weight, ...). For each atom there is a unique file name containing its properties.

X	Y	Z	Rho	BW	AW	Vpol
-0.01831060	0.01831060	0.00000000	0.42399444E+00	0.10000000E+01	0.26508891E-08	12.50446618
0.01831060	0.01831060	0.00000000	0.42405088E+00	0.10000000E+01	0.26508891E-08	12.51811693
-0.01831060	-0.01831060	0.00000000	0.42409066E+00	0.10000000E+01	0.26508891E-08	12.49843237
0.01831060	-0.01831060	0.00000000	0.42414790E+00	0.10000000E+01	0.26508891E-08	12.51231617
0.00000000	0.01831060	0.01831060	0.42798607E+00	0.10000000E+01	0.26508891E-08	12.55707167
0.00000000	0.01831060	-0.01831060	0.42019760E+00	0.10000000E+01	0.26508891E-08	12.46614573
0.00000000	-0.01831060	0.01831060	0.42808615E+00	0.10000000E+01	0.26508891E-08	12.55109776
0.00000000	-0.01831060	-0.01831060	0.42029089E+00	0.10000000E+01	0.26508891E-08	12.46028320
-0.01831060	-0.00000000	0.01831060	0.42800810E+00	0.10000000E+01	0.26508891E-08	12.54711734
-0.01831060	-0.00000000	-0.01831060	0.42021760E+00	0.10000000E+01	0.26508891E-08	12.45634761
0.01831060	-0.00000000	0.01831060	0.42806642E+00	0.10000000E+01	0.26508891E-08	12.56098496
0.01831060	-0.00000000	-0.01831060	0.42027305E+00	0.10000000E+01	0.26508891E-08	12.47001535
-0.01495054	0.01495054	0.01495054	0.42723408E+00	0.10000000E+01	0.26301338E-08	12.54244926
-0.01495054	0.01495054	-0.01495054	0.42087578E+00	0.10000000E+01	0.26301338E-08	12.46828486
0.01495054	0.01495054	0.01495054	0.42728120E+00	0.10000000E+01	0.26301338E-08	12.55367887
0.01495054	0.01495054	-0.01495054	0.42092100E+00	0.10000000E+01	0.26301338E-08	12.47938222
-0.01495054	-0.01495054	0.01495054	0.42731499E+00	0.10000000E+01	0.26301338E-08	12.53750221

Figure 5.13: Example of a table of grid points.

Bibliography

[1] B. Bollobas. *Graph Theory: An Introductory Course*, volume 173. 1979. ISBN 0387903992.

[2] D. Kranda. The Square of Adjacency Matrices. *ArXiv*, pages 1–8, 2012.

[3] A. J. Bridgeman, G. Cavigliasso, L. R. Ireland, and J. Rothery. The Mayer Bond Order as a Tool in Inorganic Chemistry. *Journal of the Chemical Society, Dalton Transactions*, pages 2095–2108, 2001.

[4] D. Weininger. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988.

[5] S. Ash, M. A. Cline, R. W. Homer, T. Hurst, and G. B. Smith. SYBYL Line Notation (SLN): A Versatile Language for Chemical Structure Representation. *Journal of Chemical Information and Computer Sciences*, 37(1):71–79, 1997.

[6] R. W. Homer, J. Swanson, R. J. Jilek, T. Hurst, and R. D. Clark. SYBYL Line Notation (SLN): A Single Notation To Represent Chemical Structures, Queries, Reactions, and Virtual Libraries. *Journal of Chemical Information and Modeling*, 48(12):2294–2307, 2008.

[7] A. Dalby, J. G. Nourse, W. D. Hounshell, A. K. I. Gushurst, D. L. Grier, B. A. Leland, and J. Laufer. Description of Several Chemical Structure File Formats Used by Computer Programs Developed at Molecular Design Limited. *Journal of Chemical Information and Computer Sciences*, 32(3):244–255, 1992.

[8] S. R. Heller, A. McNaught, I. Pletnev, S. Stein, and D. Tchekhovskoi. InChI, the IUPAC International Chemical Identifier. *Journal of Cheminformatics*, 7(1):1–34, 2015.

Chapter 6

Energy Components

"Not only is the Universe stranger than we think, it is stranger than we can think."

— Werner Heisenberg

6.1 Theoretical Background

Solving the Schrödinger equation or Kohn-Sham equations for large molecules consisting of several hundreds or thousands of atoms is not possible to deal with. Many strategies for dealing with large systems such as linear scaling methods [1–4] have attracted much attention and have become popular in the chemistry community. The reason why large systems containing many atoms are accessible with these algorithms is their linear scaling with respect to the number of atoms [5]. The divide-and-conquer method is an example of a linear scaling algorithm. This method was originally proposed for density functional theory [6]. The basic idea of the divide-and-conquer method is to break down a large system (entire system) into small subsystems. In this method, the effect of the surrounding environment is included, where a subsystem within the large system is different from the isolated ones. The higher the accuracy desired, the more correction terms are needed, which will greatly increase computational cost. One drawback of the divide-and-conquer method is that it does not provide information on the electronic structure of the entire system since it operates only on the energy of the subsystems [7].

6.2 Methodology of Our Theory

In this work, our intention is to obtain the electronic molecular energy components from the summation of their atomic contributions as,

$$T = \sum_{A=1}^{M} T^A, \qquad K = \sum_{A=1}^{M} K^A, \qquad V_{ne} = \sum_{A=1}^{M} V_{ne}^A, \qquad J = \sum_{A=1}^{M} J^A \tag{6.1}$$

where M is number of atoms within the molecule. T, K, V_{ne}, and J are the molecular kinetic energy, the molecular exchange energy, the molecular electron-nuclear attraction energy, and the molecular Coulomb repulsion energy, respectively. T^A, K^A, V_{ne}^A, and J^A are the kinetic energy, the exchange energy, the electron-nuclear attraction energy, and the Coulomb repulsion energy of atom A in the molecule, respectively.

In the next sections, we will discuss the derivation of T^A, K^A, V_{ne}^A, and J^A.

6.2.1 Kinetic Energy (T)

The molecular kinetic energy [8] can be defined as,

$$\begin{aligned} T &= \frac{1}{2}\sum_{a}^{N}\int |\nabla\psi_a(\mathbf{r})|^2\, d\mathbf{r} \\ &= \frac{1}{2}\sum_{a}^{N}\int \nabla\psi_a^*(\mathbf{r})\nabla\psi_a(\mathbf{r}) d\mathbf{r} \end{aligned} \tag{6.2}$$

where $\nabla\psi_a(\mathbf{r})$ is the gradient of molecular orbital a at position $\mathbf{r}$ and the summation runs over number of the electrons (N) of the system. The $\psi_a(\mathbf{r})$ can be expanded as a linear combination of basis functions $\{\varphi(\mathbf{r})\}$,

$$\psi_a(\mathbf{r}) = \sum_{\mu=1}^{k} C_{\mu a}\varphi_\mu(\mathbf{r}) \tag{6.3}$$

where k is the number of basis functions, and $C_{\mu a}$ are the expansion coefficients. Consequently, the kinetic energy (Equation 6.2) becomes,

$$T = \frac{1}{2}\sum_{a}^{N}\int \sum_{\mu} C_{\mu a}^*\nabla\varphi_\mu^*(\mathbf{r})\sum_{\nu} C_{\nu a}\nabla\varphi_\nu(\mathbf{r}) d\mathbf{r} \tag{6.4}$$

For RHF, Equation 6.4 can be rearranged into,

$$\begin{aligned} T &= \frac{1}{2}\sum_{\mu}\sum_{\nu}\left(2\sum_{a}^{N/2} C_{\mu a}^{*}C_{\nu a}\right)\int \nabla\varphi_{\mu}^{*}(\mathbf{r})\nabla\varphi_{\nu}(\mathbf{r})d\mathbf{r} \\ &= \frac{1}{2}\sum_{\mu}\sum_{\nu} P_{\nu\mu}\int dG_{\mu\nu}(\mathbf{r})d\mathbf{r} \end{aligned} \tag{6.5}$$

where,

$$P_{\nu\mu} = 2\sum_{a}^{N/2} C_{\mu a}^{*}C_{\nu a} \tag{6.6}$$

and

$$dG_{\mu\nu}(\mathbf{r}) = \nabla\varphi_{\mu}^{*}(\mathbf{r})\nabla\varphi_{\nu}(\mathbf{r}) \tag{6.7}$$

The integral term in Equation 6.5 can be computed as,

$$\int dG_{\mu\nu}(\mathbf{r})d\mathbf{r} = \sum_{A}\sum_{i} W_{A}(\mathbf{r}_{i})\omega(\mathbf{r}_{i})dG_{\mu\nu}(\mathbf{r}_{i}) \tag{6.8}$$

where $\mathbf{r}_i$ is the i^{th} grid point, $W_A(\mathbf{r}_i)$ is the partition function of atom A at $\mathbf{r}_i$, and $\omega(\mathbf{r}_i)$ is the product of angular and radial weights at $\mathbf{r}_i$.

By substituting Equation 6.8 into Equation 6.5 one obtains,

$$\begin{aligned} T &= \frac{1}{2}\sum_{A}\sum_{i} W_{A}(\mathbf{r}_{i})\omega(\mathbf{r}_{i})\left(\sum_{\mu}\sum_{\nu} P_{\nu\mu}dG_{\mu\nu}(\mathbf{r}_{i})\right) \\ &= \frac{1}{2}\sum_{A}\sum_{i} W_{A}(\mathbf{r}_{i})\omega(\mathbf{r}_{i})S(\mathbf{r}_{i}) \end{aligned} \tag{6.9}$$

where

$$S(\mathbf{r}_{i}) = \sum_{\mu}\sum_{\nu} P_{\nu\mu}dG_{\mu\nu}(\mathbf{r}_{i}) \tag{6.10}$$

Using Equation 6.9, the atomic kinetic energy can be defined as,

$$\begin{aligned} T^A &= \frac{1}{2}\sum_i W_A(\mathbf{r}_i)\omega(\mathbf{r}_i)S(\mathbf{r}_i) \\ &= \sum_i \omega(\mathbf{r}_i)\left(\frac{1}{2}W_A(\mathbf{r}_i)S(\mathbf{r}_i)\right) \\ &= \sum_i \omega(\mathbf{r}_i)T^A(\mathbf{r}_i) \end{aligned} \tag{6.11}$$

where

$$T^A(\mathbf{r}_i) = \frac{1}{2}W_A(\mathbf{r}_i)S(\mathbf{r}_i) \tag{6.12}$$

The molecular kinetic energy density at a given grid point $\mathbf{r}_i$ can be defined as,

$$T(\mathbf{r}_i) = \sum_A T^A(\mathbf{r}_i) \tag{6.13}$$

The radial molecular kinetic energy density at $\mathbf{r}_i$ is obtained by multiplying the last equation with $\mathbf{r}_{iA}^2$,

$$T_{rad}(\mathbf{r}_i) = \sum_A \mathbf{r}_{iA}^2 T^A(\mathbf{r}_i) \tag{6.14}$$

6.2.2 Exchange Energy (K)

The HF equation of ground state energy for RHF system, E_0, is defined as,

$$\begin{aligned} E_0 &= 2\sum_a^{N/2} h_{aa} + \sum_{ab}^{N/2} 2J_{ab} - K_{ab} \\ &= 2\sum_a^{N/2} h_{aa} + J_{HF} - K_{HF} \end{aligned} \tag{6.15}$$

where h_{aa} is the kinetic and attraction energy to all nuclei of an electron. J_{ab} and K_{ab} are called the Coulomb and exchange integrals respectively. J_{ab} and K_{ab} always have positive values. The term K_{HF} in Equation 6.15, which contains the Coulomb repulsion J_{aa} is defined as,

$$\begin{aligned} K_{HF} &= \sum_b \sum_a K_{ab} \\ &= 2\sum_b \sum_{b<a} K_{ab} + \sum_a J_{aa} \end{aligned} \tag{6.16}$$

The first term in the last equation is the exchange energy,

$$K = 2\sum_b \sum_{b<a} K_{ab} \tag{6.17}$$

which represents the interaction energies between electrons with parallel spins. The second term in Equation 6.16 is canceled by an equivalent term in the Coulomb energy in the HF equation (Equation 6.15).

The two-electron integral K_{ab} does not have a simple classical interpretation.

$$K_{ab} = \int \int \frac{\psi_a^*(\mathbf{r}_1)\psi_b^*(\mathbf{r}_2)\psi_b(\mathbf{r}_1)\psi_a(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_2 d\mathbf{r}_1 \tag{6.18}$$

For simplicity, K_{ab} can be rearranged and written as,

$$\begin{aligned} K_{ab} &= \int \psi_a^*(\mathbf{r}_1)\psi_b(\mathbf{r}_1) \left[\int \frac{\psi_b^*(\mathbf{r}_2)\psi_a(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_2 \right] d\mathbf{r}_1 \\ &= \int \psi_a^*(\mathbf{r}_1)\psi_b(\mathbf{r}_1) V^{ba}(\mathbf{r}_1) d\mathbf{r}_1 \end{aligned} \tag{6.19}$$

where

$$V^{ba}(\mathbf{r}_1) = \int \frac{\psi_b^*(\mathbf{r}_2)\psi_a(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_2 \tag{6.20}$$

Using Equation 6.3, the product $\psi_a^*(\mathbf{r}_1)\psi_b(\mathbf{r}_1)$ in Equation 6.19 becomes,

$$\begin{aligned}
\psi_a^*(\mathbf{r}_1)\psi_b(\mathbf{r}_1) &= \sum_\lambda C_{\lambda a}^* \varphi_\lambda^*(\mathbf{r}_1) \sum_\sigma C_{\sigma b}\varphi_\sigma(\mathbf{r}_1) \\
&= \sum_\lambda \sum_\sigma C_{\lambda a}^* C_{\sigma b} \varphi_\lambda^*(\mathbf{r}_1)\varphi_\sigma(\mathbf{r}_1) \\
&= \sum_\lambda \sum_\sigma P_{\sigma\lambda}^{ab} G_{\lambda\sigma}(\mathbf{r}_1)
\end{aligned} \tag{6.21}$$

where $P_{\sigma\lambda}^{ab}$ is the product of $C_{\lambda a}^*$ with the $C_{\sigma b}$, and $G_{\lambda\sigma}(\mathbf{r}_1)$ is the product of $\varphi_\lambda^*(\mathbf{r}_1)$ with $\varphi_\sigma(\mathbf{r}_1)$. Here we define $S^{ab}(\mathbf{r}_1)$ as follows,

$$S^{ab}(\mathbf{r}_1) = \sum_\lambda \sum_\sigma P_{\lambda\sigma}^{ab} G_{\sigma\lambda}(\mathbf{r}_1) \tag{6.22}$$

The substitution of Equation 6.3 into Equation 6.20 gives,

$$\begin{aligned}
V^{ba}(\mathbf{r}_1) &= \int \frac{\psi_b^*(\mathbf{r}_2)\psi_a(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_2 \\
&= \int \frac{\sum_\mu C_{\mu b}^* \varphi_\mu^*(\mathbf{r}_2) \sum_\nu C_{\nu a}\varphi_\nu(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_2
\end{aligned} \tag{6.23}$$

The last equation may be rearranged to obtain,

$$\begin{aligned}
V^{ba}(\mathbf{r}_1) &= \sum_\mu \sum_\nu C_{\mu b}^* C_{\nu a} \int \frac{\varphi_\mu^*(\mathbf{r}_2)\varphi_\nu(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_2 \\
&= \sum_\mu \sum_\nu P_{\nu\mu}^{ba} V_{\mu\nu}(\mathbf{r}_1)
\end{aligned} \tag{6.24}$$

where $P_{\nu\mu}^{ba}$ is the product of $C_{\mu b}^{*}$ with $C_{\nu a}$ and $V_{\mu\nu}(\mathbf{r}_1)$ is,

$$V_{\mu\nu}(\mathbf{r}_1) = \int \frac{\varphi_{\mu}^{*}(\mathbf{r}_2)\varphi_{\nu}(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_2 \tag{6.25}$$

Equation 6.19 can be rewritten as,

$$K_{ab} = \int S^{ab}(\mathbf{r}_1) V^{ba}(\mathbf{r}_1) d\mathbf{r}_1 \tag{6.26}$$

K_{ab} can be calculated numerically as,

$$\begin{aligned} K_{ab} &= \sum_A \sum_i W_A(\mathbf{r}_i)\omega(\mathbf{r}_i) S^{ab}(\mathbf{r}_i) V^{ba}(\mathbf{r}_i) \\ &= \sum_A \sum_i \omega(\mathbf{r}_i) \left(W_A(\mathbf{r}_i) S^{ab}(\mathbf{r}_i) V^{ba}(\mathbf{r}_i) \right) \\ &= \sum_A \sum_i \omega(\mathbf{r}_i) K_{ab}^{A}(\mathbf{r}_i) \end{aligned} \tag{6.27}$$

where,

$$K_{ab}^{A}(\mathbf{r}_i) = W_A(\mathbf{r}_i) S^{ab}(\mathbf{r}_i) V^{ba}(\mathbf{r}_i) \tag{6.28}$$

Using Equation 6.27, the atomic two-electron integral K_{ab}^{A} can be obtained as,

$$K_{ab}^{A} = \sum_i \omega(\mathbf{r}_i) K_{ab}^{A}(\mathbf{r}_i) \tag{6.29}$$

Using Equation 6.27, the exchange energy (K) and K_{HF} term can be calculated by summing over the molecular orbitals to get,

$$K = 2\sum_{A}\sum_{b}\sum_{b<a}\sum_{i}\omega(\mathbf{r}_i)K^A_{ab}(\mathbf{r}_i) \tag{6.30}$$

and

$$K_{HF} = 2\sum_{A}\sum_{b}\sum_{b<a}\sum_{i}\omega(\mathbf{r}_i)K^A_{ab}(\mathbf{r}_i) + \sum_{A}\sum_{a}\sum_{i}\omega(\mathbf{r}_i)K^A_{aa}(\mathbf{r}_i) \tag{6.31}$$

Using Equation 6.30, the atomic exchange energy (K^A) can be computed as,

$$\begin{aligned} K^A &= 2\sum_{b}\sum_{b<a}\sum_{i}\omega(\mathbf{r}_i)K^A_{ab}(\mathbf{r}_i) \\ &= \sum_{i}\omega(\mathbf{r}_i)\left(2\sum_{b}\sum_{b<a}K^A_{ab}(\mathbf{r}_i)\right) \\ &= \sum_{i}\omega(\mathbf{r}_i)K^A(\mathbf{r}_i) \end{aligned} \tag{6.32}$$

where,

$$K^A(\mathbf{r}_i) = 2\sum_{b}\sum_{b<a}K^A_{ab}(\mathbf{r}_i) \tag{6.33}$$

And by using Equation 6.31 the K^A_{HF} term can be computed as,

$$K^A_{HF} = \sum_{i}\omega(\mathbf{r}_i)K^A_{HF}(\mathbf{r}_i) \tag{6.34}$$

where

$$K^A_{HF}(\mathbf{r}_i) = 2\sum_{b}\sum_{b<a}K^A_{ab}(\mathbf{r}_i) + \sum_{a}K^A_{aa}(\mathbf{r}_i) \tag{6.35}$$

The molecular exchange energy density at the grid point $\mathbf{r}_i$ can be defined as,

$$K(\mathbf{r}_i) = \sum_A K^A(\mathbf{r}_i) \tag{6.36}$$

and by multiplying the last equation with $\mathbf{r}_{iA}^2$, the radial exchange energy density of a molecule can be obtained,

$$K_{rad}(\mathbf{r}_i) = \sum_A \mathbf{r}_{iA}^2 K^A(\mathbf{r}_i) \tag{6.37}$$

6.2.3 Coulomb Energy (J)

The term J_{HF} in Equation 6.15 is defined as,

$$\begin{aligned} J_{HF} &= 2\sum_b \sum_a J_{ab} \\ &= 2\left(2\sum_b \sum_{b<a} J_{ab} + \sum_a J_{aa}\right) \\ &= 4\sum_b \sum_{b<a} J_{ab} + 2\sum_a J_{aa} \end{aligned} \tag{6.38}$$

Whereas, the Coulomb energy (J) is,

$$J = 4\sum_b \sum_{b<a} J_{ab} + \sum_a J_{aa} \tag{6.39}$$

The two-electron integral J_{ab} represents the Coulomb repulsion between charge clouds $|\psi_a(\mathbf{r}_1)|^2$ and $\left|\psi_b(\mathbf{r}_2)\right|^2$.

$$J_{ab} = \int\int \frac{\psi_a^*(\mathbf{r}_1)\psi_b^*(\mathbf{r}_2)\psi_a(\mathbf{r}_1)\psi_b(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_1 d\mathbf{r}_2$$
$$= \int\int \frac{|\psi_a(\mathbf{r}_1)|^2\left|\psi_b(\mathbf{r}_2)\right|^2}{\mathbf{r}_{12}} d\mathbf{r}_1 d\mathbf{r}_2 \tag{6.40}$$

J_{ab} is called the Coulomb integral and always has positive values. In this section, we will derive the Coulomb energy using the molecular orbital expansion method and the electron density method.

6.2.3.1 Molecular Orbital Expansion Method

Using Equation 6.40, J_{ab} can be rewritten as,

$$J_{ab} = \int |\psi_a(\mathbf{r}_1)|^2 \left[\int \frac{\left|\psi_b(\mathbf{r}_2)\right|^2}{\mathbf{r}_{12}} d\mathbf{r}_2\right] d\mathbf{r}_1$$
$$= \int |\psi_a(\mathbf{r}_1)|^2 V^{bb}(\mathbf{r}_1) d\mathbf{r}_1 \tag{6.41}$$

where $V^{bb}(\mathbf{r}_1)$ is defined as,

$$V^{bb}(\mathbf{r}_1) = \int \frac{\left|\psi_b(\mathbf{r}_2)\right|^2}{\mathbf{r}_{12}} d\mathbf{r}_2 \tag{6.42}$$

Using Equation 6.3 the $|\psi_a(\mathbf{r}_1)|^2$ term in Equation 6.41 becomes,

$$\begin{aligned}
|\psi_a(\mathbf{r}_1)|^2 &= \psi_a^*(\mathbf{r}_1)\psi_a(\mathbf{r}_1) \\
&= \sum_\lambda C_{\lambda a}^* \varphi_\lambda^*(\mathbf{r}_1) \sum_\sigma C_{\sigma a} \varphi_\sigma(\mathbf{r}_1) \\
&= \sum_\lambda \sum_\sigma C_{\lambda a}^* C_{\sigma a} \varphi_\lambda^*(\mathbf{r}_1) \varphi_\sigma(\mathbf{r}_1) \\
&= \sum_\lambda \sum_\sigma P_{\sigma\lambda}^{aa} G_{\lambda\sigma}(\mathbf{r}_1)
\end{aligned} \tag{6.43}$$

where $P_{\sigma\lambda}^{aa}$ is the product of $C_{\lambda a}^*$ with $C_{\sigma a}$ and $G_{\lambda\sigma}(\mathbf{r}_1)$ is the product of $\varphi_\lambda^*(\mathbf{r}_1)$ with $\varphi_\sigma(\mathbf{r}_1)$. Furthermore, the substitution of the molecular orbital expansion 6.3 into Equation 6.42 gives,

$$\begin{aligned}
V^{bb}(\mathbf{r}_1) &= \int \frac{\left|\psi_b(\mathbf{r}_2)\right|^2}{\mathbf{r}_{12}} d\mathbf{r}_2 \\
&= \int \frac{\sum_\mu C_{\mu b}^* \varphi_\mu^*(\mathbf{r}_2) \sum_\nu C_{\nu b} \varphi_\nu(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_2 \\
&= \sum_\mu \sum_\nu C_{\mu b}^* C_{\nu b} \int \frac{\varphi_\mu^*(\mathbf{r}_2)\varphi_\nu(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_2 \\
&= \sum_\mu \sum_\nu P_{\nu\mu}^{bb} V_{\mu\nu}(\mathbf{r}_1)
\end{aligned} \tag{6.44}$$

where $P_{\nu\mu}^{bb}$ is the product of $C_{\mu b}^*$ with $C_{\nu b}$ and $V_{\mu\nu}(\mathbf{r}_1)$ is defined in Equation 6.25. Equation 6.41 can be written as,

$$\begin{aligned}
J_{ab} &= \int \sum_\lambda \sum_\sigma P_{\sigma\lambda}^{aa} G_{\lambda\sigma}(\mathbf{r}_1) \sum_\mu \sum_\nu P_{\nu\mu}^{bb} V_{\mu\nu}(\mathbf{r}_1) d\mathbf{r}_1 \\
&= \int S^{aa}(\mathbf{r}_1) V^{bb}(\mathbf{r}_1) d\mathbf{r}_1
\end{aligned} \tag{6.45}$$

where $S^{aa}(\mathbf{r}_1)$ is defined as,

$$S^{aa}(\mathbf{r}_1) = \sum_{\lambda} \sum_{\sigma} P^{aa}_{\sigma\lambda} G_{\lambda\sigma}(\mathbf{r}_1) \tag{6.46}$$

J_{ab} can be calculated numerically as follows,

$$\begin{aligned} J_{ab} &= \sum_{A} \sum_{i} W_A(\mathbf{r}_i)\omega(\mathbf{r}_i) S^{aa}(\mathbf{r}_i) V^{bb}(\mathbf{r}_i) \\ &= \sum_{A} \sum_{i} \omega(\mathbf{r}_i) \left(W_A(\mathbf{r}_i) S^{aa}(\mathbf{r}_i) V^{bb}(\mathbf{r}_i) \right) \\ &= \sum_{A} \sum_{i} \omega(\mathbf{r}_i) J^A_{ab}(\mathbf{r}_i) \end{aligned} \tag{6.47}$$

where,

$$J^A_{ab}(\mathbf{r}_i) = W_A(\mathbf{r}_i) S^{aa}(\mathbf{r}_i) V^{bb}(\mathbf{r}_i) \tag{6.48}$$

Using Equation 6.47, the atomic two-electron integral J^A_{ab} can be obtained as,

$$J^A_{ab} = \sum_{i} \omega(\mathbf{r}_i) J^A_{ab}(\mathbf{r}_i) \tag{6.49}$$

Using Equation 6.47, the Coulomb energy (J) and J_{HF} term can be computed by summing over molecular orbitals to get,

$$\begin{aligned} J &= 4 \sum_{A} \sum_{b} \sum_{b<a} \sum_{i} \omega(\mathbf{r}_i) J^A_{ab}(\mathbf{r}_i) + \sum_{A} \sum_{a} \sum_{i} \omega(\mathbf{r}_i) J^A_{aa}(\mathbf{r}_i) \\ &= \sum_{A} \sum_{i} \omega(\mathbf{r}_i) \left(4 \sum_{b} \sum_{b<a} J^A_{ab}(\mathbf{r}_i) + \sum_{a} J^A_{aa}(\mathbf{r}_i) \right) \end{aligned} \tag{6.50}$$

and

$$J_{HF} = \sum_{A}\sum_{i} \omega(\mathbf{r}_i) \left(4\sum_{b}\sum_{b<a} J^{A}_{ab}(\mathbf{r}_i) + 2\sum_{a} J^{A}_{aa}(\mathbf{r}_i) \right) \tag{6.51}$$

Using Equation 6.50, the atomic Coulomb energy J^A can be computed as,

$$\begin{aligned} J^A &= \sum_{i} \omega(\mathbf{r}_i) \left(4\sum_{b}\sum_{b<a} J^{A}_{ab}(\mathbf{r}_i) + \sum_{a} J^{A}_{aa}(\mathbf{r}_i) \right) \\ &= \sum_{i} \omega(\mathbf{r}_i) J^A(\mathbf{r}_i) \end{aligned} \tag{6.52}$$

where,

$$J^A(\mathbf{r}_i) = 4\sum_{b}\sum_{b<a} J^{A}_{ab}(\mathbf{r}_i) + \sum_{a} J^{A}_{aa}(\mathbf{r}_i) \tag{6.53}$$

Using Equation 6.51, the J^A_{HF} term can be computed as,

$$J^A_{HF} = \sum_{i} \omega(\mathbf{r}_i) J^A_{HF}(\mathbf{r}_i) \tag{6.54}$$

where,

$$J^A_{HF}(\mathbf{r}_i) = 4\sum_{b}\sum_{b<a} J^{A}_{ab}(\mathbf{r}_i) + 2\sum_{a} J^{A}_{aa}(\mathbf{r}_i) \tag{6.55}$$

The molecular Coulomb energy density at the grid point $\mathbf{r}_i$ can be defined as,

$$J(\mathbf{r}_i) = \sum_{A} J^A(\mathbf{r}_i) \tag{6.56}$$

and by multiplying the last equation with $\mathbf{r}_{iA}^2$, the radial Coulomb energy of molecule can be obtained,

$$J_{rad}(\mathbf{r}_i) = \sum_A \mathbf{r}_{iA}^2 J^A(\mathbf{r}_i) \tag{6.57}$$

6.2.3.2 Electron Density Method

Starting from molecular electron density $\rho(\mathbf{r})$, the classical Coulomb energy can be written as,

$$J[\rho] = \frac{1}{2}\int\int \frac{\rho(\mathbf{r}_1)\rho(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_1 d\mathbf{r}_2 \tag{6.58}$$

The above equation can be rearranged as below,

$$\begin{aligned} J[\rho] &= \frac{1}{2}\int \rho(\mathbf{r}_1)\left[\int \frac{\rho(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_2\right] d\mathbf{r}_1 \\ &= \frac{1}{2}\int \rho(\mathbf{r}_1) V(\mathbf{r}_1) d\mathbf{r}_1 \\ &= \frac{1}{2}\int \sum_A W_A(\mathbf{r}_1)\rho(\mathbf{r}_1) V(\mathbf{r}_1) d\mathbf{r}_1 \\ &= \frac{1}{2}\int \sum_A \rho_A(\mathbf{r}_1) V(\mathbf{r}_1) d\mathbf{r}_1 \end{aligned} \tag{6.59}$$

where $\rho_A(\mathbf{r})$ is the atomic electron density of atom A and

$$V(\mathbf{r}_1) = \int \frac{\rho(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_2 \tag{6.60}$$

$V(\mathbf{r}_1)$ can be written as,

$$\begin{aligned} V(\mathbf{r}_1) &= \int \frac{\rho(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_2 \\ &= \sum_B \int \frac{W_B(\mathbf{r}_2)\rho(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_2 \\ &= \sum_B \int \frac{\rho_B(\mathbf{r}_2)}{\mathbf{r}_{12}} d\mathbf{r}_2 \end{aligned} \tag{6.61}$$

$V(\mathbf{r}_1)$ can be computed numerically as follows,

$$V(\mathbf{r}_1) = \sum_B \sum_j \frac{W_B(\mathbf{r}_j)\omega(\mathbf{r}_j)\rho(\mathbf{r}_j)}{\mathbf{r}_{1j}} \tag{6.62}$$

At grid $\mathbf{r}_i$ the equation becomes,

$$V(\mathbf{r}_i) = \sum_B \sum_j \frac{W_B(\mathbf{r}_j)\omega(\mathbf{r}_j)\rho(\mathbf{r}_j)}{\mathbf{r}_{ij}} \tag{6.63}$$

Starting from Equation 6.59, the Coulomb energy can be calculated numerically as follows,

$$J[\rho] = \frac{1}{2} \sum_A \sum_i W_A(\mathbf{r}_i)\omega(\mathbf{r}_i)\rho(\mathbf{r}_i)V(\mathbf{r}_i) \tag{6.64}$$

and the atomic Coulomb energy density $J^A[\rho]$ can be computed as,

$$\begin{aligned} J^A[\rho] &= \frac{1}{2} \sum_i W_A(\mathbf{r}_i)\omega(\mathbf{r}_i)\rho(\mathbf{r}_i)V(\mathbf{r}_i) \\ &= \sum_i \omega(\mathbf{r}_i) \left(\frac{1}{2} W_A(\mathbf{r}_i)\rho(\mathbf{r}_i)V(\mathbf{r}_i) \right) \\ &= \sum_i \omega(\mathbf{r}_i) J^A[\rho(\mathbf{r}_i)] \end{aligned} \tag{6.65}$$

where

$$J^A[\rho(\mathbf{r}_i)] = \frac{1}{2} W_A(\mathbf{r}_i)\rho(\mathbf{r}_i)V(\mathbf{r}_i) \tag{6.66}$$

The molecular Coulomb energy density at the grid point $\mathbf{r}_i$ can be defined as,

$$J[\rho(\mathbf{r}_i)] = \sum_A J^A[\rho(\mathbf{r}_i)] \tag{6.67}$$

and by multiplying the last equation with $\mathbf{r}_{iA}^2$, the radial Coulomb energy density of a molecule can be obtained,

$$J_{rad}[\rho(\mathbf{r}_i)] = \sum_A \mathbf{r}_{iA}^2 J^A[\rho(\mathbf{r}_i)] \tag{6.68}$$

With substitution of Equation 6.63 in Equation 6.64, J becomes,

$$\begin{aligned} J[\rho] &= \frac{1}{2}\sum_A\sum_i W_A(\mathbf{r}_i)\omega(\mathbf{r}_i)\rho(\mathbf{r}_i)\left[\sum_B\sum_j \frac{W_B(\mathbf{r}_j)\omega(\mathbf{r}_j)\rho(\mathbf{r}_j)}{\mathbf{r}_{ij}}\right] \\ &= \frac{1}{2}\sum_A\sum_i \omega(\mathbf{r}_i)\rho_A(\mathbf{r}_i)\left[\sum_B\sum_j \frac{\omega(\mathbf{r}_j)\rho_B(\mathbf{r}_j)}{\mathbf{r}_{ij}}\right] \\ &= \frac{1}{2}\sum_A\sum_B\sum_i\sum_j \frac{Q_A(\mathbf{r}_i)Q_B(\mathbf{r}_j)}{\mathbf{r}_{ij}} \end{aligned} \tag{6.69}$$

where

$$Q_A(\mathbf{r}_k) = W_A(\mathbf{r}_k)\omega(\mathbf{r}_k)\rho(\mathbf{r}_k) \tag{6.70}$$

Note that the values $W_A(\mathbf{r}_k)$, $\omega(\mathbf{r}_k)$, and $\rho(\mathbf{r}_k)$ can be stored in the database, or for simplicity $Q_A(\mathbf{r}_k)$ can be stored instead of these values. Later these values can be used for calculating J of the target molecule.

6.2.4 Potential Energy, Nuclear-Electron Attraction Energy (V_{ne})

The electron-nuclear potential energy is defined as,

$$V_{ne}[\rho] = -\sum_{A} Z_A \int \frac{\rho(\mathbf{r})}{|\mathbf{r}-\mathbf{R}_A|} d\mathbf{r} \tag{6.71}$$

where, $\mathbf{R}_A$ is the position vector of the atom A with atomic charge Z_A. The above equation can be written as follows,

$$V_{ne}[\rho] = -\sum_{A} Z_A \sum_{B} \int \frac{W_B(\mathbf{r})\rho(\mathbf{r})}{|\mathbf{r}-\mathbf{R}_A|} d\mathbf{r} \tag{6.72}$$

$V_{ne}[\rho]$ can be computed numerically as,

$$\begin{aligned} V_{ne}[\rho] &= -\sum_{A} Z_A \sum_{B} \sum_{j} \frac{W_B(\mathbf{r}_j)\omega(\mathbf{r}_j)\rho(\mathbf{r}_j)}{\mathbf{r}_{jA}} \\ &= -\sum_{A} Z_A \sum_{B} \sum_{j} \frac{\omega(\mathbf{r}_j)\rho_B(\mathbf{r}_j)}{\mathbf{r}_{jA}} \\ &= -\sum_{A} Z_A \sum_{B} \sum_{j} \frac{Q_B(\mathbf{r}_j)}{\mathbf{r}_{jA}} \end{aligned} \tag{6.73}$$

Using the above equation, the atomic electron-nuclear potential energy V_{ne}^{A} can be obtained as,

$$V_{ne}^{A}[\rho] = -Z_A \sum_{B} \sum_{j} \frac{Q_B(\mathbf{r}_j)}{\mathbf{r}_{jA}} \tag{6.74}$$

The molecular potential energy density at the grid point $\mathbf{r}_j$ can be defined as,

$$V_{ne}[\rho(\mathbf{r}_j)] = -\sum_A Z_A \sum_B \frac{W_B(\mathbf{r}_j)\rho(\mathbf{r}_j)}{\mathbf{r}_{jA}}$$
$$= -\sum_A Z_A \sum_B \frac{\rho_B(\mathbf{r}_j)}{\mathbf{r}_{jA}} \tag{6.75}$$

and by multiplying the last equation with $\mathbf{r}_{jB}^2$, the radial molecular potential energy can be obtained,

$$V_{ne,rad}[\rho(\mathbf{r}_j)] = -\sum_A Z_A \sum_B \frac{\mathbf{r}_{jB}^2 \rho_B(\mathbf{r}_j)}{\mathbf{r}_{jA}} \tag{6.76}$$

Bibliography

[1] G. Galli. Linear Scaling Methods for Electronic Structure Calculations and Quantum Molecular Dynamics Simulations. *Current Opinion in Solid State and Materials Science*, 1(6):864 – 874, 1996.

[2] D. R. Bowler and T. Miyazaki. Calculations for Millions of Atoms With Density Functional Theory: Linear Scaling Shows its Potential. *Journal of Physics: Condensed Matter*, 22(7):074207, 2010.

[3] T. Ozaki. Efficient low-order Scaling Method for Large-Scale Electronic Structure Calculations With Localized Basis Functions. *Physical Review B*, 82:075131, Aug 2010.

[4] Y. Tsukamoto, Y. Ikabata, J. Romero, A. Reyes, and H. Nakai. The Divide-and-Conquer Second-Order Proton Propagator Method Based on Nuclear Orbital Plus Molecular Orbital Theory for the Efficient Computation of Proton Binding Energies. *Physical Chemistry Chemical Physics*, 18:27422–27431, 2016.

[5] S. Goedecker. Linear Scaling Electronic Structure Methods. *Reviews of Modern Physics*, 71:1085–1123, Jul 1999.

[6] W. Yang. Direct Calculation of Electron Density in Density-Functional theory. *Physical Review Letters*, 66:1438–1441, Mar 1991.

[7] G.-L. Song, Z. H. Li, Z.-P. Liu, X.-M. Cao, W. Wang, K.-N. Fan, Y. Xie, and H. F. Schaefer. Local Hybrid Divide-and-Conquer Method for the Computation of Medium and Large Molecules. *Journal of Chemical Theory and Computation*, 4(12):2049–2056, 2008.

[8] E. Sim, J. Larkin, K. Burke, and C. W. Bock. Testing the Kinetic Energy Functional: Kinetic Energy Density as a Density Functional. *The Journal of Chemical Physics*, 118 (18):8140–8148, 2003.

Chapter 7

AIMD Calculations and Results

"Everything we call real is made of things that cannot be regarded as real."

— Niels Bohr

7.1 Computational Method

All the calculations were performed in MUNgauss package [1]. The molecular radial kinetic, exchange, Coulomb, and potenial energy densities are computed at HF/6-31G(d) and employing a mesh size of 0.05 bohr. The results of numerical integration were calculated using the 6-31G(d) basis sets on HF/6-31G(d) optimized structures, and the SG1 grid is used in the calculations. All of the visual aids, including contour, relief, and the 3D plots were created using Mathematica Version 11.2 graphing package [2].

7.2 Molecular Properties Results and Discussions

In this chapter, we will discuss the results of computing the molecular properties from the corresponding fragment (atomic) properties. As mentioned before, each fragment represents an atom in a specific molecular environment. The calculated molecular properties are compared with those calculated by the HF wavefunction (the exact values). The "%error" is computed as,

$$\%\text{error} = \frac{P_{HF} - P_{AIM}}{P_{HF}} \times 100\% \tag{7.1}$$

where P_{AIM} and P_{HF} are the values of a molecular property computed using atoms in molecules density (AIMD) method and HF, respectively. In addition for each molecular property the mean absolute percentage error (MAPE) is computed as,

$$\text{MAPE} = \frac{100\%}{M} \sum_{i} \left| \frac{P^{i}_{HF} - P^{i}_{AIM}}{P^{i}_{HF}} \right| \tag{7.2}$$

where P^{i}_{AIM} is the value of the molecular property computed using AIMD for a molecule i, and P^{i}_{HF} is the corresponding molecular property calculated using HF, and M is the number of molecules.

Table 7.1 shows some examples of atomic properties that have been stored in the database. In these examples the atomic properties are computed using Awad weight, and second neighbour of non-optimized fragments.

In Figure 7.1 we introduce a notation to represent the fragment for the target atom in the molecule of interest (i.e., the target molecule). To write the notation, the size of fragment (i.e., the number of neighbour level) goes above the symbol whereas the atomic symbol and the atom number of the atom in the molecule of interest are written as a subscript. For

Table 7.1: Some of the atomic properties N_e^A, K^A, K_{HF}^A, T^A, V_{ne}^A, J^A, and J_{HF}^A, calculated using SG1 grid, Awad weight, second neighbour of non-optimized fragments at HF/6-31G(d).

Symbol	N_e^A	K^A	K_{HF}^A	T^A	V_{ne}^A	J^A	J_{HF}^A
1(6(6,1,1))	1.071925	-0.188903	-0.374779	0.594538	-6.729386	4.075700	4.261576
1(6(7,6,1))	1.069721	-0.195488	-0.377186	0.590994	-8.639958	5.456195	5.637894
1(6(7,6,6))	1.074858	-0.201258	-0.379658	0.582953	-10.438871	6.766430	6.944830
1(6(8,6,6))	1.065122	-0.193933	-0.379249	0.588868	-10.564649	6.731344	6.916660
1(7(6,1))	1.020490	-0.175350	-0.379122	0.687671	-7.302757	3.710862	3.914635
1(7(6,6))	1.004696	-0.189775	-0.373623	0.663992	-9.078792	4.777218	4.961065
1(8(6))	0.957115	-0.162286	-0.375158	0.792281	-7.781516	3.554150	3.767022
6(6(7,6,1)1()1()1())	5.797046	-0.946114	-5.140066	37.832324	-133.014107	33.046856	37.240808
6(6(8,6,1)1()1()1())	5.793904	-0.938579	-5.142095	37.834155	-133.263508	34.221182	38.424698
6(7(1,1)6(8,6,1)6(8,7)1())	5.874279	-1.058858	-5.196807	38.012005	-179.724239	57.117083	61.255032
6(7(1,1)6(8,6,1)6(8,8)1())	5.865734	-1.035455	-5.195416	38.018367	-180.023043	57.457843	61.617804
6(7(1,1)6(8,7)1()1())	5.820390	-0.941964	-5.171103	37.979300	-146.945699	41.124838	45.353977
6(7(1,1)6(8,7)6(1,1,1)1())	5.867287	-0.967023	-5.193536	38.022886	-160.737219	46.995839	51.222352
6(7(1,1)6(8,8)1()1())	5.810699	-0.879369	-5.167286	37.968746	-147.175448	41.347383	45.635301
6(7(1,1)6(8,8)6(1,1,1)1())	5.858824	-0.947523	-5.189427	38.003355	-160.849703	47.117248	51.359152
6(7(6,1)6(8,6,1)6(8,8)1())	5.856394	-1.086767	-5.186843	37.956271	-189.041650	60.672298	64.772374
6(7(6,1)6(8,7)1()1())	5.808371	-0.992022	-5.160104	37.919678	-155.923481	44.174403	48.342486
6(7(6,1)6(8,8)1()1())	5.804767	-0.969884	-5.168492	37.954400	-156.995171	45.013871	49.212480
6(7(6,1)6(8,8)6(1,1,1)1())	5.852320	-1.014654	-5.188123	37.976373	-170.491347	50.688196	54.861665
6(8()7(6,1)6(7,1,1))	5.876251	-0.957532	-5.245189	38.500984	-164.810017	45.769518	50.057175
6(8()7(6,1)6(7,6,1))	5.872220	-0.971928	-5.243648	38.509980	-174.386728	49.133169	53.404889
6(8(1)6(7,6,1)6(1,1,1)1())	5.833517	-0.985297	-5.184959	38.166212	-163.262139	44.362219	48.561881
6(8(1)8()6(7,1,1))	5.850333	-0.896454	-5.248576	38.697097	-156.560471	42.169020	46.521142
6(8(1)8()6(7,6,1))	5.854054	-0.920244	-5.250513	38.710557	-166.075004	45.504417	49.834686
7(6(6,1,1)1()1())	6.995128	-1.207685	-6.588524	54.009604	-167.953973	38.223335	43.604174
7(6(6,6,1)1()1())	6.997995	-1.296852	-6.589653	54.011239	-178.900238	43.216108	48.508909
7(6(6,6,1)6(8,6)1())	6.987776	-1.439793	-6.617214	54.047767	-217.879964	60.669501	65.846921
7(6(8,6)6(6,1,1)1())	6.977344	-1.404146	-6.614876	54.045749	-206.902577	55.784664	60.995394
8(6(6,6,1)1())	8.137391	-1.986194	-8.244030	74.165832	-233.187879	56.538266	62.796103
8(6(7,6))	8.275259	-1.942467	-8.258353	74.218555	-231.329357	59.417003	65.732889
8(6(8,6))	8.237650	-1.888308	-8.252445	74.225501	-232.183586	61.040105	67.404242
8(6(8,6)1())	8.044935	-2.036277	-8.213370	74.121111	-231.986033	55.098164	61.275256

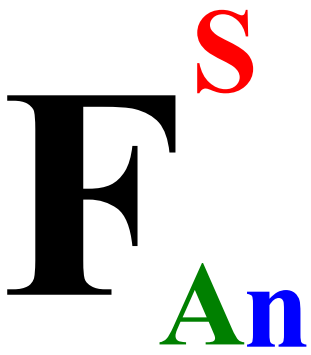


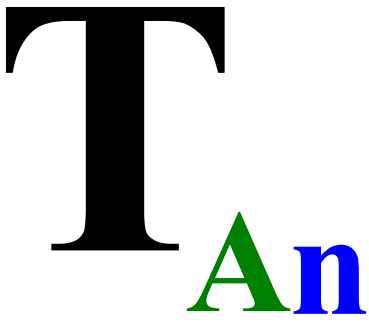
Figure 7.1: The suggested notation to represent the fragment for the target atom in the molecule of interest. **S**, **A**, and **n** are the size of fragment, the atomic symbol of target atom, and the atom number in the molecule of interest, respectively.

example, in Figure 7.3 the $\mathbf{F^1_{C11}}$ (molecule A), $\mathbf{F^2_{C11}}$ (molecule B), and $\mathbf{F^3_{C11}}$ (molecule C) are three different carbon atom fragments obtained from the molecule of interest (molecule D) as first, second, and third neighbour atoms, respectively.

Also, we suggest notations to represent the target atom within its fragment and within the molecule of interest. These notations are shown in Figures 7.2a and 7.2b respectively. For example, in Figure 7.3 the $^{\mathbf{C1}}\mathbf{F^1_{C11}}$, $^{\mathbf{C1}}\mathbf{F^2_{C11}}$, $^{\mathbf{C1}}\mathbf{F^3_{C11}}$, and $\mathbf{T_{C11}}$ are the target carbon atoms in $\mathbf{F^1_{C11}}$, $\mathbf{F^2_{C11}}$, $\mathbf{F^3_{C11}}$, and the molecule of interest, respectively.

The effect of geometry on the atomic properties was studied by calculating some of atomic properties (e.g., T^A, K^A, J^A, and V^A_{ne}) of the carbon atom within four different geometries: $\mathbf{F^1_{C11}}$, $\mathbf{F^2_{C11}}$, $\mathbf{F^3_{C11}}$, and the molecule of interest (D molecule). As can be seen from Figure 7.3, the fragments $\mathbf{F^1_{C11}}$, $\mathbf{F^2_{C11}}$, and $\mathbf{F^3_{C11}}$ are completed with hydrogen atoms to get real molecules. In addition to the properties of the carbon atoms, $^{\mathbf{C1}}\mathbf{F^1_{C11}}$, $^{\mathbf{C1}}\mathbf{F^2_{C11}}$, $^{\mathbf{C1}}\mathbf{F^3_{C11}}$, and $\mathbf{T_{C11}}$, their molecular properties (e.g., T, K, J, and V_{ne}) are also studied. In the following sections, each of the studied properties will be discussed separately.

(a) The target atom within the fragment. **S**, **A**, **n**, **T**, and **m** are the size of the fragment, the atomic symbol of the target atom in the molecule of interest, the atom number in the molecule of interest, the atomic symbol of target atom within the fragment, and the atom number within the fragment, respectively.



(b) The target atom within the molecule of interest (target molecule). **A** and **n** are the atomic symbol and its atom number of the target atom in the molecule of interest respectively.

Figure 7.2: Suggested notations to represent the target atom within its fragment and the molecule of the interest.

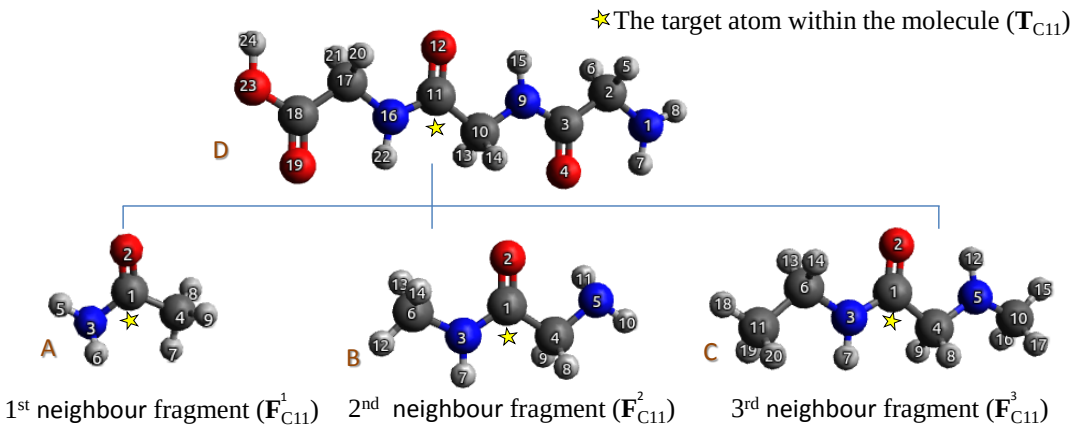


Figure 7.3: The target carbon atom within the four structures $\mathbf{F}^1_{\mathbf{C11}}$, $\mathbf{F}^2_{\mathbf{C11}}$, $\mathbf{F}^3_{\mathbf{C11}}$, and $\mathbf{T}_{\mathbf{C11}}$, to study the effect of geometry on the atomic energies components.

7.3 Kinetic Energy (T)

Figure 7.4 shows a 3D-contour plot of the molecular radial kinetic energy density (Equation 6.14) of the target carbon atom ($\mathbf{T}_{\mathbf{C11}}$) in molecule of interest (D molecule in Figure 7.3). As shown in Figure 7.4, the kinetic energies of electrons in the bonding region are very small compared to the non-bonding region, in which the electrons have more kinetic energy when they are close to the nuclei. This is expected because the velocity is inversely proportional to the distance from the nucleus, thus electrons have greater velocities near the nucleus, therefore more kinetic energy. As shown in Figure 7.4, we can visually distinguish the radial kinetic energy density for each atom in the molecule, where the atomic radial kinetic energy densities appear as cone shapes. The atomic radial kinetic energy is proportional to the atomic number of the atom, where oxygen has the largest kinetic energy followed by nitrogen, then carbon.

In Figure 7.5, we compare the radial kinetic energy density (Equation 6.14) along the bond between the carbon atom and oxygen atom for $\mathbf{F}^1_{\mathbf{C11}}$ (red dash line), $\mathbf{F}^2_{\mathbf{C11}}$ (green dash line),

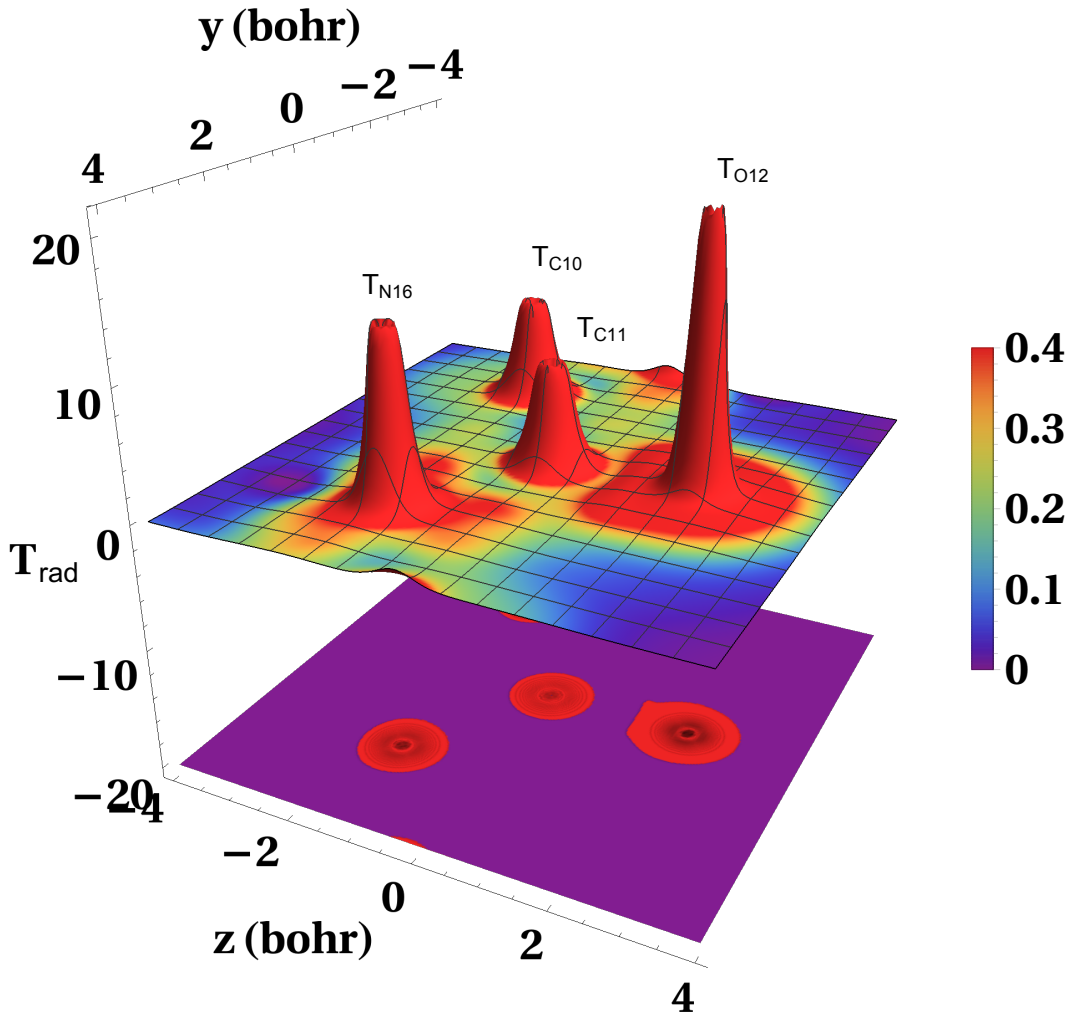


Figure 7.4: The radial kinetic energy density for the molecule of interest in Figure 7.3, $\mathbf{T_{C11}}$ is the target carbon atom.

$\mathbf{F^3_{C11}}$ (purple dash line), and the D molecule (blue solid line). As shown in this figure, the radial kinetic energy densities for $\mathbf{F^1_{C11}}$, $\mathbf{F^2_{C11}}$, $\mathbf{F^3_{C11}}$, and D molecule are almost identical. Maximum peaks of the radial kinetic energy densities adjacent to the carbon nucleus are located at -0.1586 and 0.1587 bohr with 8.335 and 8.344 radial kinetic energy densities respectively, whereas the maximum peaks adjacent to the oxygen nucleus are located at -0.1205 and 0.1208 bohr (relative to oxygen nucleus) with 20.909 and 21.128 radial kinetic

energy densities respectively. These results show that positions of the peaks in the oxygen atom are closer to its nucleus than the peaks in the carbon atom to its nucleus due to the difference in effective nuclear charges.

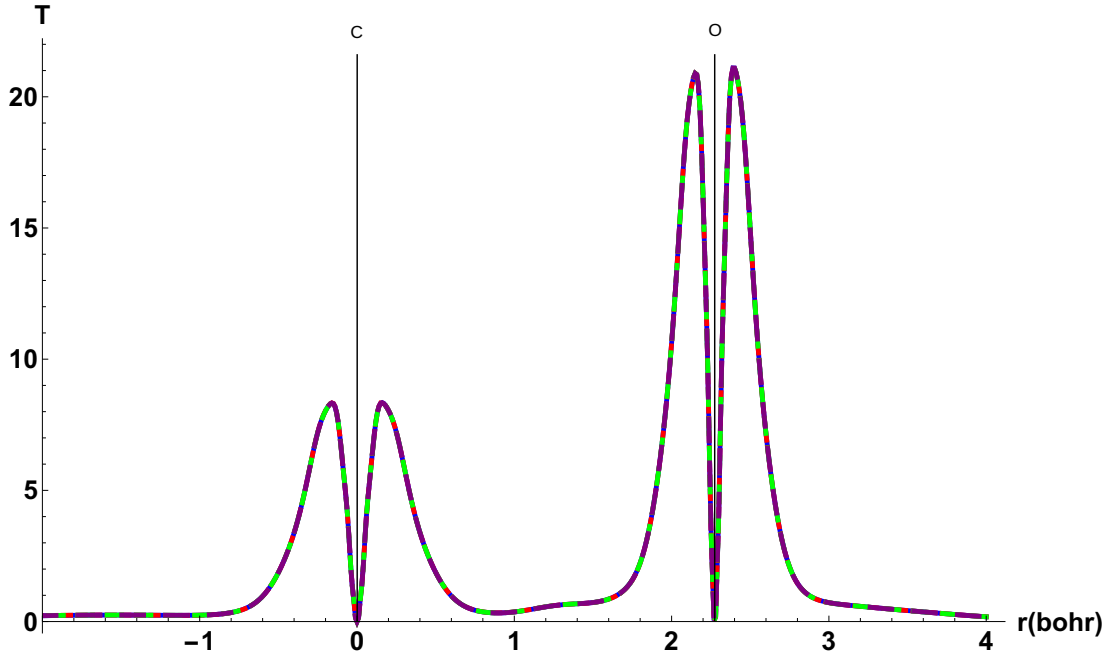


Figure 7.5: The radial kinetic energy density along the bond between the target carbon atom and the oxygen atom for $\mathbf{F}^{1}_{C11}$ (red dash line), $\mathbf{F}^{2}_{C11}$ (green dash line), $\mathbf{F}^{3}_{C11}$ (purple dash line), and the molecule of interest, D molecule, (blue solid line). The carbon is located at 0.0 bohr and oxygen is located at 2.2739 bohr.

The computed kinetic energy (Equation 6.11) of the target carbon atom for the four molecules $\mathbf{F}^{1}_{C11}$, $\mathbf{F}^{2}_{C11}$, $\mathbf{F}^{3}_{C11}$, and D molecule are displayed in Figure 7.6. The results show that the kinetic energies of the carbon atom are inversely proportional to the size of the molecule. As shown in Figure 7.6, there is no significant change in the carbon kinetic energy of $^{C1}\mathbf{F}^{3}_{C11}$ and $\mathbf{T}_{C11}$. It is clear from these results that the first and second neighbors of atoms have the largest effect on the computed kinetic energies. Therefore, the kinetic energy is not an atomic property as it is affected by surrounding atoms. For instance, the atomic kinetic

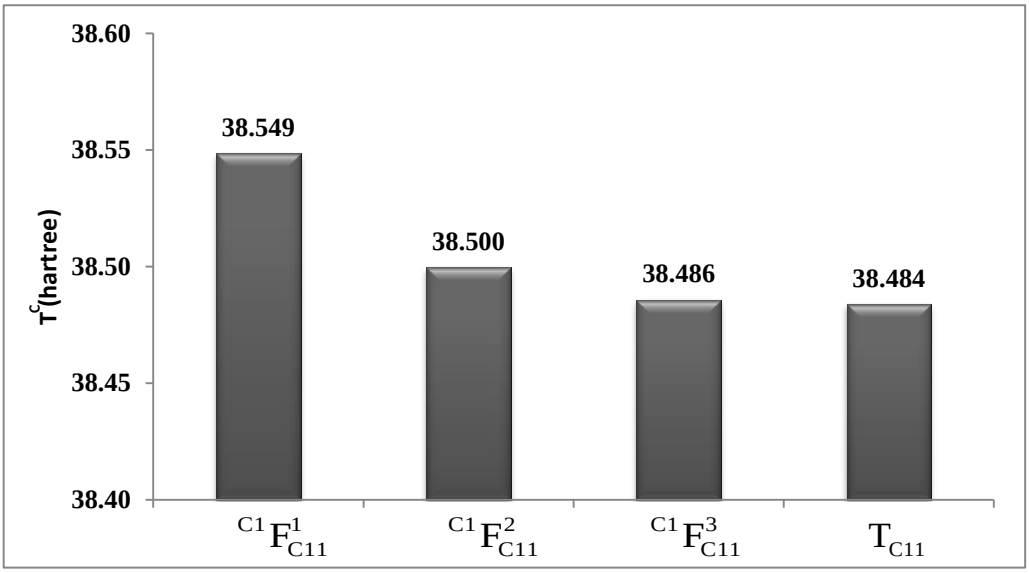


Figure 7.6: Effect of size of the fragment on the atomic kinetic energy (T^C) of the target carbon atom.

energy of the carbon atom is computed in the following environments: isolated atom, CH_4, HCOH, $HCONH_2$, and CF_4, and found to be 37.945, 37.706, 38.153, 38.363, and 39.948 hartree, respectively. These results clearly show that the atomic kinetic energy depends on the surrounding environment.

The molecular kinetic energy is calculated by summing over atomic kinetic energies,

$$T^{sum} = \sum_{A=1}^{M} T^A \tag{7.3}$$

where T^A is the kinetic energy of atom A computed using Equation 6.11 and M is the number of atoms in the target molecule. The atomic kinetic energies are obtained two different ways: from that stored in the database ($T^A_{database}$) and from using the direct method (T^A_{direct}). In the direct method, the kinetic energies for each atom are calculated in a specific environment (fragment) by keeping the same Cartesian coordinates of the atoms in the target

Table 7.2: Comparing the molecular kinetic energies calculated using the sum over atomic kinetic energies that are stored in the database ($T^{sum}_{database} = \sum_{A=1} T^{A}_{database}$) and by using the direct method ($T^{sum}_{direct} = \sum_{A=1} T^{A}_{direct}$) with those calculated numerically (Equation 6.9) for the target molecule (the exact values). $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ are the size of fragments as first, second, and third neighbor atoms, respectively.

	Target molecule T		Fragments (Sum over atoms) $T^{sum}_{database} = \sum_{A} T^{A}_{database}$ (%error$_{numerical}$)			Fragments (Sum over atoms) $T^{sum}_{direct} = \sum_{A} T^{A}_{direct}$ (%error$_{numerical}$)		
Molecule	**(Analytical)**	**(Numerical)**	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$
Gly	282.300	282.295	282.588 (-0.104%)	282.372 (-0.027%)	282.310 (-0.005%)	282.578 (-0.100%)	282.362 (-0.024%)	282.310 (-0.005%)
Ala	321.328	321.312	321.634 (-0.100%)	321.438 (-0.039%)	321.364 (-0.016%)	321.606 (-0.091%)	321.421 (-0.034%)	321.346 (-0.010%)
Thr	435.039	435.008	435.448 (-0.101%)	435.168 (-0.037%)	435.113 (-0.024%)	435.361 (-0.081%)	435.153 (-0.033%)	435.070 (-0.014%)
GlyGly	488.763	488.760	489.388 (-0.128%)	488.948 (-0.038%)	488.825 (-0.013%)	489.375 (-0.126%)	488.935 (-0.036%)	488.836 (-0.016%)
AlaAla	566.816	566.776	567.481 (-0.125%)	567.073 (-0.053%)	566.942 (-0.029%)	567.409 (-0.112%)	567.039 (-0.046%)	566.890 (-0.020%)
ThrThr	794.214	794.193	795.108 (-0.115%)	794.472 (-0.035%)	794.399 (-0.026%)	794.892 (-0.088%)	794.422 (-0.029%)	794.331 (-0.017%)
GlyAla	527.791	527.748	528.434 (-0.130%)	528.019 (-0.051%)	527.886 (-0.026%)	528.379 (-0.119%)	527.987 (-0.045%)	527.842 (-0.018%)
GlyThr	641.493	641.463	642.248 (-0.122%)	641.697 (-0.037%)	641.593 (-0.020%)	642.101 (-0.100%)	641.673 (-0.033%)	641.568 (-0.016%)
AlaThr	680.521	680.487	681.295 (-0.119%)	680.768 (-0.041%)	680.685 (-0.029%)	681.141 (-0.096%)	680.741 (-0.037%)	680.615 (-0.019%)
GlyGlyGly1	695.223	695.181	696.188 (-0.145%)	695.479 (-0.043%)	695.103 (0.011%)	696.119 (-0.135%)	695.490 (-0.044%)	695.307 (-0.018%)
GlyGlyGly2	695.226	695.191	696.188 (-0.143%)	695.479 (-0.041%)	695.103 (0.013%)	696.080 (-0.128%)	695.395 (-0.029%)	695.295 (-0.015%)
		MAPE =	0.121%	0.040%	0.019%	0.107%	0.036%	0.015%

molecule.

To investigate the validity of our results, we compare the kinetic energies of three mono-amino acids (glycine Gly, alanine Ala, and threonine Thr), six di-amino acids (GlyGly, AlaAla, ThrThr, GlyAla, GlyThr, and AlaThr), and two isomers of tri-amino acid (GlyGlyGly1, GlyGlyGly2) calculated using the sum over atomic kinetic energies stored in the database ($T^{sum}_{database} = \sum_{A=1} T^{A}_{database}$) and by using the direct method ($T^{sum}_{direct} = \sum_{A=1} T^{A}_{direct}$) with those calculated numerically (Equation 6.9) for the target molecule (the exact values).

Table 7.2 shows the results of $T^{sum}_{database}$ and T^{sum}_{direct} calculated using different size of fragments, $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ as first, second, and third neighbor atoms, respectively. The MAPE of $T^{sum}_{database}$ are about 0.121%, 0.040%, and 0.019% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively. Whereas, The MAPE of T^{sum}_{direct} are about 0.107%, 0.036%, and 0.015% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively. These results as tabulated in Table 7.2 are encouraging, where %errors obtained for T^{sum}_{direct} are slightly better than %errors obtained for $T^{sum}_{database}$ and the %errors have always negative values (two exceptions are GlyGlyGly1 and GlyGlyGly2 for $T^{sum}_{database}$ using $\mathbf{F^3}$).

7.4 Exchange Energy (K)

Figure 7.7 shows a 3D-contour plot of the molecular radial exchange energy density (Equation 6.37) of target carbon atom $\mathbf{T_{C11}}$ in the molecule of interest (D molecule in Figure 7.3). As shown in Figure 7.7, we cannot visually distinguish the exchange energy density for each atom in the molecule, where the atomic radial exchange energy densities are shared in the bonding regions.

The radial exchange energy density along the bond between the target carbon atom and oxygen atom for $\mathbf{F^1_{C11}}$ (red dash line), $\mathbf{F^2_{C11}}$ (green dash line), $\mathbf{F^3_{C11}}$ (purple dash line), and

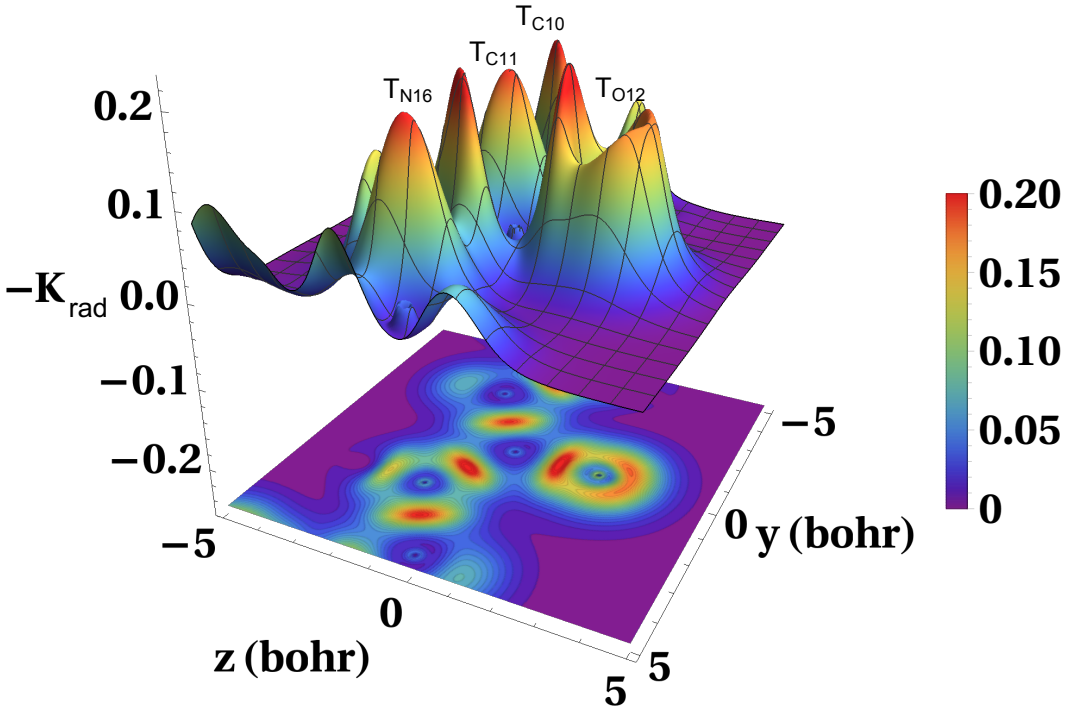


Figure 7.7: The radial exchange energy density for D molecule (Figure 7.3), $\mathbf{T_{C11}}$ is the target carbon atom.

the D molecule (blue solid line) are compared in Figure 7.8. As shown in Figure 7.8, the exchange energy density is dependent on the size of the molecule. For example, the molecular radial exchange energy density for the D molecule is larger than $\mathbf{F^1_{C11}}$.

The computed exchange energies (Equation 6.32) of the target carbon atom for the four molecules $\mathbf{F^1_{C11}}$, $\mathbf{F^2_{C11}}$, $\mathbf{F^3_{C11}}$, and the molecule of interest (D molecule) are displayed in Figure 7.9. The results show that the atomic exchange energy of the carbon atom is proportional to the size of the molecule.

The molecular exchange energy is calculated by summing over atomic exchange energies,

$$K^{sum} = \sum_{A=1}^{M} K^A \tag{7.4}$$

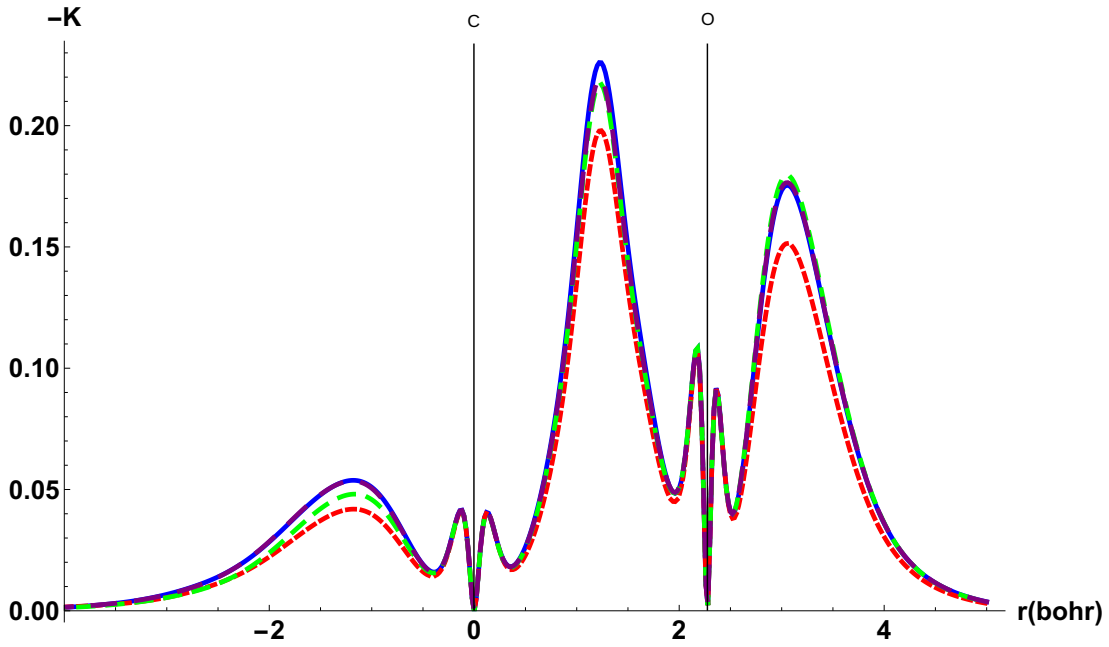


Figure 7.8: The radial exchange energy density along the bond between the target carbon atom and the oxygen atom for $\mathbf{F}^{1}_{C11}$ (red dash line), $\mathbf{F}^{2}_{C11}$ (green dash line), $\mathbf{F}^{3}_{C11}$ (purple dash line), and the molecule of interest, D molecule, (blue solid line). The carbon is located at 0.0 bohr and oxygen is located at 2.2739 bohr.

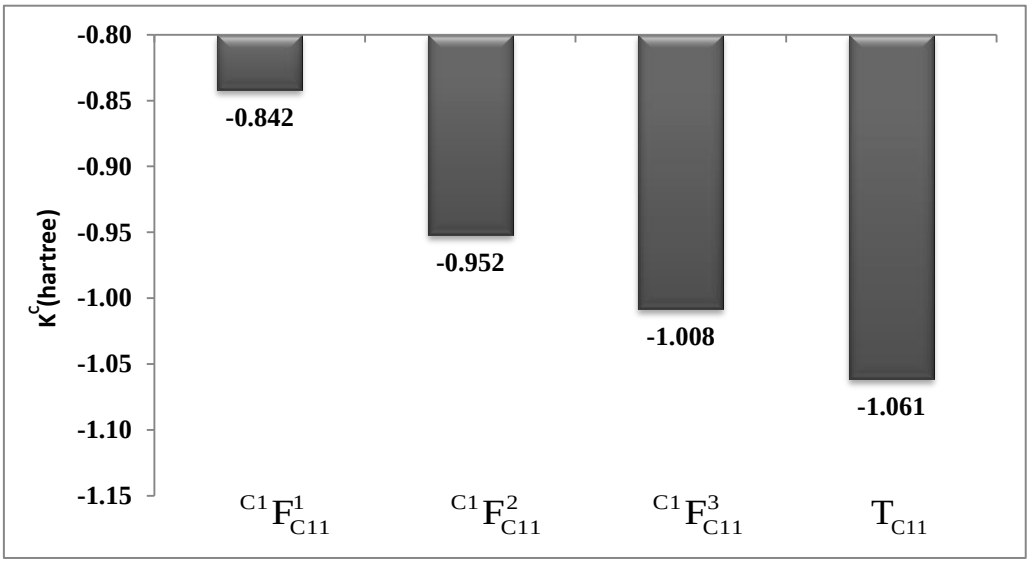


Figure 7.9: Effect of size of the fragment on the atomic exchange energy density (K) of target carbon atom.

where K^A is the exchange energy of atom A computed using Equation 6.32 and M is the number of atoms in the target molecule. Table 7.3 compares the exchange energies of three mono-amino acids (glycine Gly, alanine Ala, and threonine Thr), six di-amino acids (GlyGly, AlaAla, ThrThr, GlyAla, GlyThr, and AlaThr), and two isomers of tri-amino acid (GlyGlyGly1 and GlyGlyGly2) calculated using the sum over atomic exchange energies stored in the database ($K^{sum}_{database} = \sum_{A=1} K^A_{database}$) and by using the direct method ($K^{sum}_{direct} = \sum_{A=1} K^A_{direct}$) of $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ fragments with those calculated numerically (Equation 6.30) for the target molecule. Unfortunately, the MAPE for these molecules is very large. The MAPE of $K^{sum}_{database}$ are about 26.72%, 12.15%, and 5.86% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively, and the MAPE of K^{sum}_{direct} are about 27.03%, 12.32%, and 6.23% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively. Thus we cannot consider the exchange as an atomic property as its value depends on the geometry and other atoms of the molecule.

Figure 7.10 compares the $K_{HF,rad}$ term (i.e., the radial exhange energy containing J_{aa} term) along the bond between the target carbon atom and the oxygen atom for $\mathbf{F^1_{C11}}$ (red dash line), $\mathbf{F^2_{C11}}$ (green dash line), $\mathbf{F^3_{C11}}$ (purple dash line), and the D molecule (blue solid line). As shown in Figure 7.10, the $K_{HF,rad}$ for $\mathbf{F^1_{C11}}$, $\mathbf{F^2_{C11}}$, $\mathbf{F^3_{C11}}$, and D molecule are almost identical.

In Figure 7.11, the K^A_{HF} term of the carbon atoms, $\mathbf{^{C1}F^1_{C11}}$, $\mathbf{^{C1}F^2_{C11}}$, $\mathbf{^{C1}F^3_{C11}}$, and $\mathbf{T_{C11}}$, are calculated by using Equation 6.34. As shown in Figure 7.11, there is no significant change in K^A_{HF} of the carbon atoms in these molecules. Where the K^A_{HF} are -5.241, -5.244, -5.240, and -5.240 hartree for the target carbon atom of $\mathbf{F^1_{C11}}$, $\mathbf{F^2_{C11}}$, $\mathbf{F^3_{C11}}$ and D molecule, respectively. However, the K_{HF} is not an atomic property as it is affected by surrounding atoms. For example, the K^A_{HF} of the carbon atoms are computed in the following environments: isolated atom, CH_4, HCOH, $HCONH_2$, and CF_4 as -5.564, -5.093, -5.138, -5.195,

Table 7.3: Comparing the molecular exchange energy densities calculated using the sum over atomic exchange energy densities that are stored in the database ($K^{sum}_{database} = \sum_{A=1} K^{A}_{database}$) and by using the direct method ($K^{sum}_{direct} = \sum_{A=1} K^{A}_{direct}$) with those calculated numerically (Equation 6.30) for the target molecule. $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ are the size of fragments as first, second, and third neighbor atoms, respectively.

	Target molecule	**Fragments (Sum over atoms)** $\mathbf{K^{sum}_{database} = \sum_{A} K^{A}_{database}}$ **(%error)**			**Fragments (Sum over atoms)** $\mathbf{K^{sum}_{direct} = \sum_{A} K^{A}_{direct}}$ **(%error)**		
Molecule	**K (Numerical)**	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$
Gly	-8.091	-6.316 (21.94%)	-7.812 (3.45%)	-8.066 (0.32%)	-6.315 (21.95%)	-7.811 (3.47%)	-8.051 (0.51%)
Ala	-9.805	-8.129 (17.09%)	-9.316 (4.99%)	-9.734 (0.72%)	-8.128 (17.11%)	-9.299 (5.17%)	-9.718 (0.89%)
Thr	-14.275	-10.856 (23.95%)	-12.724 (10.86%)	-13.758 (3.62%)	-10.856 (23.95%)	-12.711 (10.96%)	-13.708 (3.97%)
GlyGly	-15.582	-10.912 (29.97%)	-13.729 (11.89%)	-14.773 (5.19%)	-10.907 (30.00%)	-13.724 (11.92%)	-14.783 (5.13%)
AlaAla	-18.960	-14.538 (23.32%)	-16.608 (12.40%)	-17.705 (6.62%)	-14.435 (23.87%)	-16.582 (12.54%)	-17.662 (6.85%)
ThrThr	-27.994	-19.992 (28.58%)	-23.413 (16.36%)	-25.635 (8.43%)	-19.596 (30.00%)	-23.380 (16.48%)	-25.529 (8.81%)
GlyAla	-17.104	-12.725 (25.60%)	-15.181 (11.24%)	-16.089 (5.94%)	-12.943 (24.33%)	-15.162 (11.35%)	-16.052 (6.15%)
GlyThr	-21.855	-15.452 (29.30%)	-18.549 (15.13%)	-20.364 (6.82%)	-15.095 (30.93%)	-18.447 (15.60%)	-19.943 (8.75%)
AlaThr	-23.819	-17.265 (27.52%)	-20.001 (16.03%)	-21.693 (8.93%)	-17.017 (28.56%)	-19.938 (16.29%)	-21.502 (9.73%)
GlyGlyGly1	-22.354	-15.508 (30.63%)	-19.606 (12.29%)	-21.173 (5.28%)	-15.504 (30.65%)	-19.549 (12.55%)	-20.865 (6.66%)
GlyGlyGly2	-24.218	-15.508 (35.97%)	-19.606 (19.04%)	-21.173 (12.57%)	-15.493 (36.03%)	-19.558 (19.24%)	-21.527 (11.11%)
	MAPE =	26.715%	12.154%	5.858%	27.033%	12.325%	6.231%

and -5.142 hartree, respectively. These results clearly show that the K_{HF}^{A} depends on the surrounding environment and the first neighbour of atoms is enough to compute the K_{HF}^{A}. The K_{HF} is computed by summing over K_{HF}^{A},

$$K_{HF}^{sum} = \sum_{A=1}^{M} K_{HF}^{A} \tag{7.5}$$

where M is the number of atoms in the target molecule.

To investigate the validity of our results, we compare the K_{HF} of three mono-amino acids (glycine Gly, alanine Ala, and threonine Thr), six di-amino acids (GlyGly, AlaAla, ThrThr, GlyAla, GlyThr, and AlaThr), and two isomers of tri-amino acid (GlyGlyGly1, GlyGlyGly2) computed using the sum over K_{HF}^{A} that stored in the database ($K_{HF,database}^{sum} = \sum_{A=1} K_{HF,database}^{A}$) and by using the direct method ($K_{HF,direct}^{sum} = \sum_{A=1} K_{HF,direct}^{A}$) of $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ fragments size with those computed numerically (Equation 6.31) for the target molecule. These results as tabulated in Table 7.4 are encouraging. As shown in Table 7.4, the MAPE of $K_{HF,database}^{sum}$ are about 0.053%, 0.058%, and 0.035% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively, and the MAPE of $K_{HF,direct}^{sum}$ are about 0.067%, 0.058%, and 0.032% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively.

7.5 Potential Energy (V_{ne})

3D-contour plot of the molecular radial potential energy density (Equation 6.76) of the target carbon atom ($\mathbf{T_{C11}}$) in the D molecule is shown in Figure 7.12. As shown in Figure 7.12, the electrons have more radial potential energy densities when they are close to the nuclei. This is expected because the potential energy density at specific coordinate

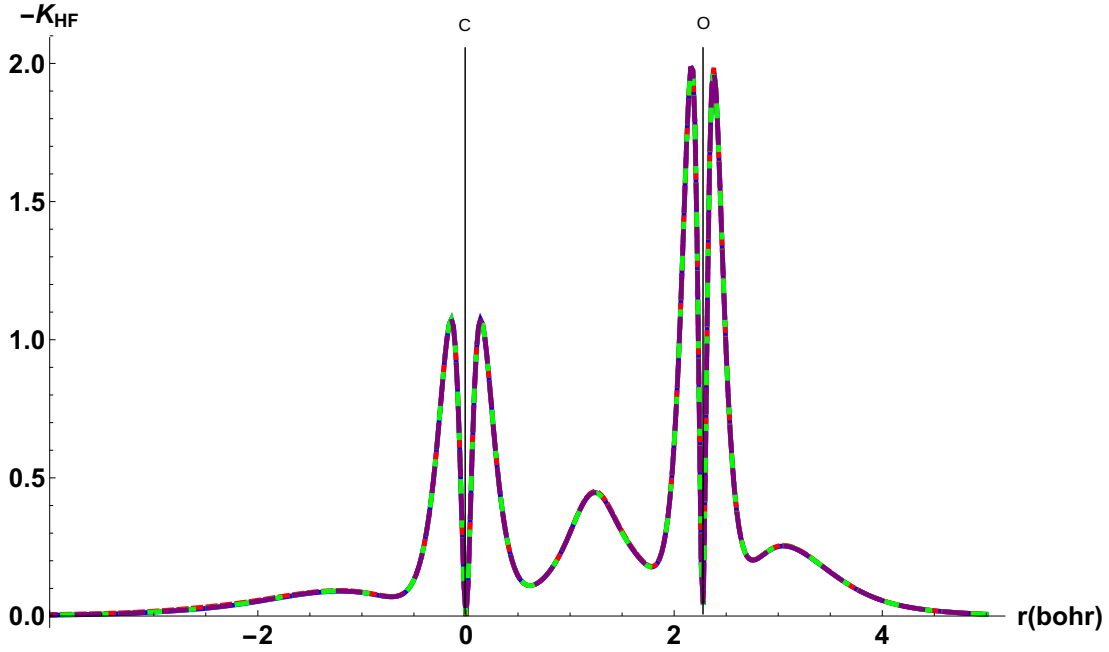


Figure 7.10: The $K^{A}_{HF,rad}$ along the bond between the target carbon atom and the oxygen atom for $\mathbf{F}^{1}_{\mathbf{C11}}$ (red dash line), $\mathbf{F}^{2}_{\mathbf{C11}}$ (green dash line), $\mathbf{F}^{3}_{\mathbf{C11}}$ (purple dash line), and the molecule of interest, D molecule, (blue solid line). The carbon is located at 0.0 bohr and oxygen is located at 2.2739 bohr.

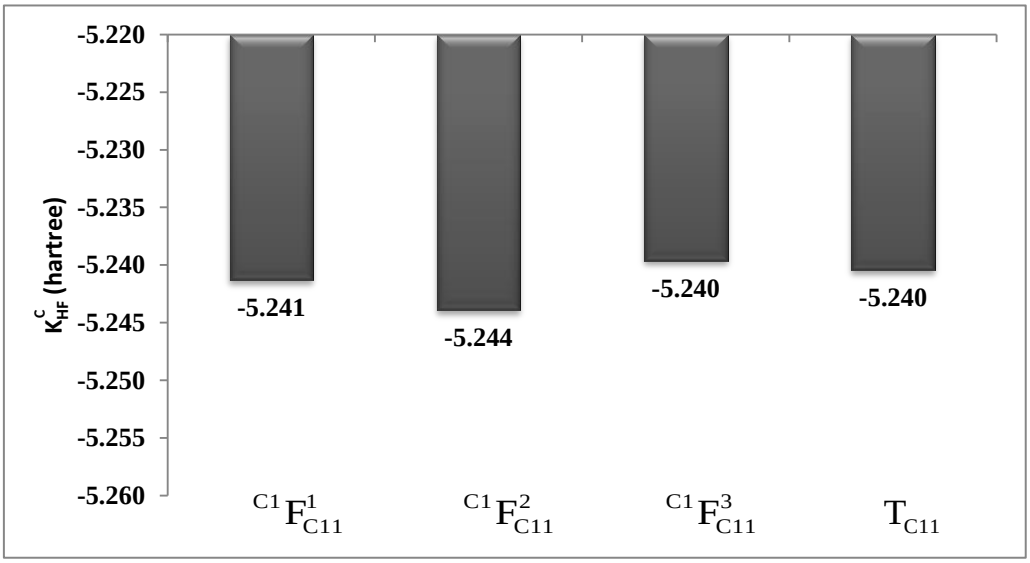


Figure 7.11: Effect of size of the fragment on the K^{A}_{HF} (Equation 6.34) of target carbon atom.

Table 7.4: Comparing the molecular HF exchange energy densities calculated using the sum over atomic HF exchange energy densities that are stored in the database ($K^{sum}_{HF,database} = \sum_{A=1} K^{A}_{HF,database}$) and by using the direct method ($K^{sum}_{HF,direct} = \sum_{A=1} K^{A}_{HF,direct}$) with those calculated numerically (Equation 6.31) for the target molecule. $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ are the size of fragments as first, second, and third neighbor atoms, respectively.

	Target molecule K_{HF}		Fragments (Sum over atoms) $K^{sum}_{HF,database} = \sum_A K^{A}_{HF,database}$ (%error$_{numerical}$)			$K^{sum}_{HF,direct} = \sum_A K^{A}_{HF,direct}$ (%error$_{numerical}$)		
Molecule	**(Analytical)**	**(Numerical)**	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$
Gly	-35.32928	-35.32928	-35.334 (-0.012%)	-35.358 (-0.081%)	-35.334 (-0.013%)	-35.332 (-0.008%)	-35.355 (-0.074%)	-35.334 (-0.013%)
Ala	-41.24681	-41.24680	-41.229 (0.044%)	-41.273 (-0.063%)	-41.266 (-0.047%)	-41.226 (0.051%)	-41.271 (-0.058%)	-41.262 (-0.036%)
Thr	-55.43881	-55.43875	-55.406 (0.059%)	-55.464 (-0.046%)	-55.460 (-0.038%)	-55.391 (0.086%)	-55.468 (-0.053%)	-55.448 (-0.017%)
GlyGly	-61.72763	-61.72759	-61.712 (0.026%)	-61.777 (-0.080%)	-61.746 (-0.030%)	-61.711 (0.027%)	-61.776 (-0.079%)	-61.748 (-0.033%)
AlaAla	-73.56106	-73.56094	-73.502 (0.080%)	-73.602 (-0.056%)	-73.610 (-0.067%)	-73.497 (0.086%)	-73.603 (-0.058%)	-73.600 (-0.053%)
ThrThr	-101.94512	-101.94481	-101.857 (0.086%)	-101.975 (-0.029%)	-101.979 (-0.033%)	-101.811 (0.131%)	-101.973 (-0.028%)	-101.961 (-0.016%)
GlyAla	-67.64293	-67.64281	-67.607 (0.053%)	-67.688 (-0.067%)	-67.682 (-0.058%)	-67.605 (0.056%)	-67.690 (-0.070%)	-67.676 (-0.048%)
GlyThr	-81.83706	-81.83680	-81.784 (0.064%)	-81.874 (-0.046%)	-81.854 (-0.022%)	-81.761 (0.093%)	-81.871 (-0.042%)	-81.857 (-0.025%)
AlaThr	-87.75464	-87.75452	-87.679 (0.086%)	-87.786 (-0.036%)	-87.800 (-0.052%)	-87.655 (0.114%)	-87.788 (-0.039%)	-87.785 (-0.034%)
GlyGlyGly1	-88.12133	-88.12117	-88.090 (0.036%)	-88.183 (-0.070%)	-88.112 (0.010%)	-88.088 (0.037%)	-88.189 (-0.077%)	-88.161 (-0.045%)
GlyGlyGly2	-88.12523	-88.12508	-88.090 (0.040%)	-88.183 (-0.066%)	-88.112 (0.015%)	-88.079 (0.052%)	-88.181 (-0.063%)	-88.153 (-0.032%)
		MAPE =	0.053%	0.058%	0.035%	0.067%	0.058%	0.032%

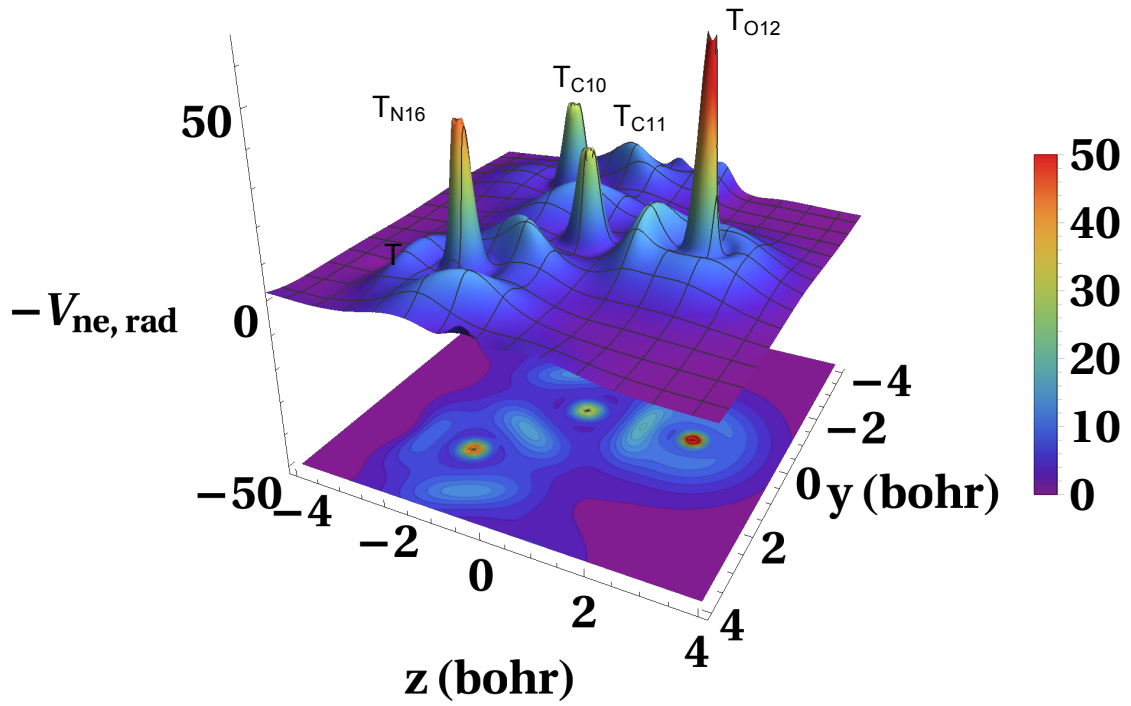


Figure 7.12: The molecular radial potential energy density of molecule D (Figure 7.3), $\mathbf{T_{C11}}$ is the target carbon atom.

represents the interaction between an electron at those coordinates with all nuclei in the molecule. This interaction increases as the electron becomes closer to a nucleus in the molecule. Moreover, the potential energy density is proportional to the nuclear charge of the atom, where the potential energy density near the oxygen atom has the largest values followed by nitrogen, then carbon (Figure 7.12).

The radial potential energy density (Equation 6.76) along the bond between the target carbon atom and oxygen atom for $\mathbf{F^1_{C11}}$ (red dash line), $\mathbf{F^2_{C11}}$ (green dash line), $\mathbf{F^3_{C11}}$ (purple dash line), and the D molecule (blue solid line) are compared in Figure 7.13. As shown in Figure 7.13 the potential energy density is proportional to the size of molecules. For example, the molecular radial potential energy density for D molecule is larger than the

fragment $\mathbf{F}^{\mathbf{1}}_{\mathbf{C11}}$.

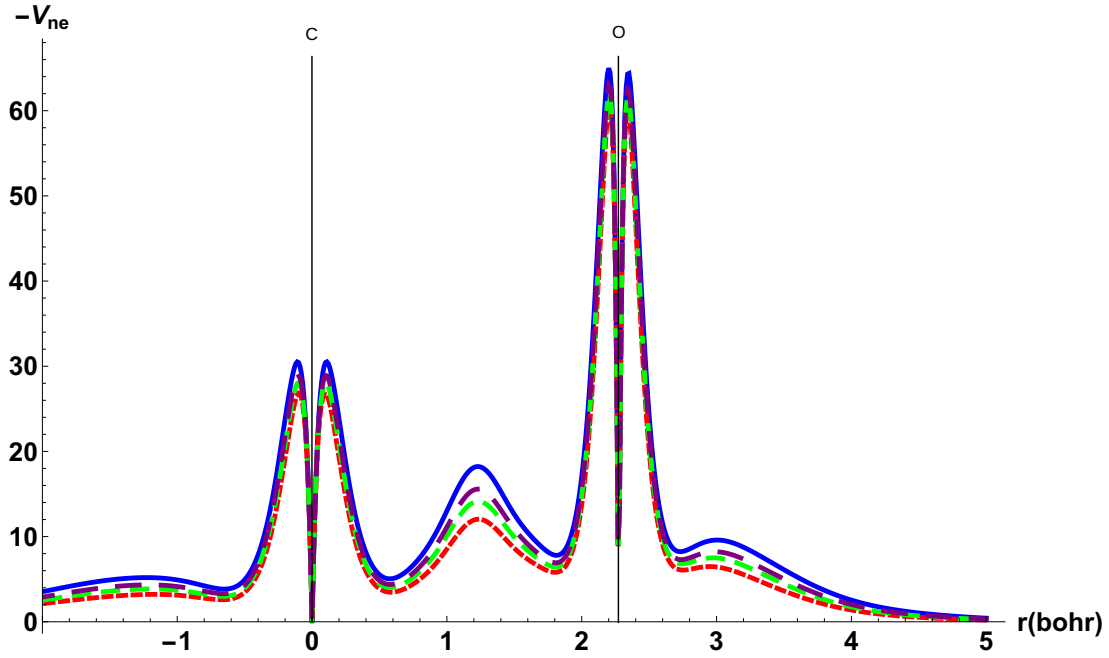


Figure 7.13: The radial potential energy density along the bond between the target carbon atom and the oxygen atom for $\mathbf{F}^{\mathbf{1}}_{\mathbf{C11}}$ (red dash line), $\mathbf{F}^{\mathbf{2}}_{\mathbf{C11}}$ (green dash line), $\mathbf{F}^{\mathbf{3}}_{\mathbf{C11}}$ (purple dash line), and the molecule of interest, D molecule, (blue solid line). The carbon is located at 0.0 bohr and oxygen is located at 2.2739 bohr.

The potential energy of the target carbon atoms $^{\mathbf{C1}}\mathbf{F}^{\mathbf{1}}_{\mathbf{C11}}$, $^{\mathbf{C1}}\mathbf{F}^{\mathbf{2}}_{\mathbf{C11}}$, $^{\mathbf{C1}}\mathbf{F}^{\mathbf{3}}_{\mathbf{C11}}$, and $\mathbf{T}_{\mathbf{C11}}$ are displayed in Figure 7.14. The results show that the atomic potential energy of the carbon atom is proportional to the size of the molecule.

The molecular potential energy is calculated by summing over atomic potential energies,

$$V_{ne}^{sum}[\rho] = \sum_{A=1}^{M} V_{ne}^{A}[\rho] \tag{7.6}$$

where V_{ne}^{A} is the potential energy density of atom A computing using Equation 6.74 and M is the number of atoms in the target molecule. Table 7.5 compares the potential energy den-

Table 7.5: Comparing the molecular potential energy densities calculated using the sum over atomic potential energy densities that are stored in the database ($V_{ne,database}^{sum}[\rho] = \sum_{A=1} V_{ne,database}^{A}[\rho]$) and by using the direct method ($V_{ne,direct}^{sum}[\rho] = \sum_{A=1} V_{ne,direct}^{A}[\rho]$) with those calculated numerically (Equation 6.73) for the target molecule. $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ are the size of fragments as first, second, and third neighbor atoms, respectively.

	Target molecule V_{ne}		Fragments (Sum over atoms) $V_{ne,database}^{sum}[\rho] = \sum_A V_{ne,database}^{A}[\rho]$ (%error$_{numerical}$)			$V_{ne,direct}^{sum}[\rho] = \sum_A V_{ne,direct}^{A}[\rho]$ (%error$_{numerical}$)		
Molecule	**(Analytical)**	**(Numerical)**	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$
Gly	-1028.734	-1028.736	-872.49 (15.19%)	-975.53 (5.17%)	-1023.52 (0.51%)	-872.47 (15.19%)	-975.47 (5.18%)	-1023.09 (0.55%)
Ala	-1259.491	-1259.493	-1010.16 (19.80%)	-1156.02 (8.22%)	-1246.36 (1.04%)	-1010.01 (19.81%)	-1155.89 (8.23%)	-1246.33 (1.04%)
Thr	-1847.851	-1847.852	-1373.81 (25.65%)	-1590.24 (13.94%)	-1756.55 (4.94%)	-1372.50 (25.72%)	-1590.13 (13.95%)	-1755.89 (4.98%)
GlyGly	-2067.008	-2067.014	-1542.80 (25.36%)	-1761.69 (14.77%)	-1901.82 (7.99%)	-1542.39 (25.38%)	-1761.44 (14.78%)	-1901.32 (8.02%)
AlaAla	-2637.297	-2637.298	-1818.15 (31.06%)	-2122.72 (19.51%)	-2358.44 (10.57%)	-1817.46 (31.09%)	-2122.72 (19.51%)	-2358.90 (10.56%)
ThrThr	-4080.834	-4080.823	-2545.44 (37.62%)	-2990.35 (26.72%)	-3372.24 (17.36%)	-2541.73 (37.72%)	-2989.29 (26.75%)	-3374.76 (17.30%)
GlyAla	-2332.480	-2332.480	-1680.48 (27.95%)	-1942.37 (16.73%)	-2123.56 (8.96%)	-1680.06 (27.97%)	-1942.61 (16.71%)	-2124.03 (8.94%)
GlyThr	-3048.499	-3048.490	-2044.12 (32.95%)	-2375.64 (22.07%)	-2642.65 (13.31%)	-2042.09 (33.01%)	-2375.79 (22.07%)	-2643.80 (13.28%)
AlaThr	-3357.328	-3357.332	-2181.79 (35.01%)	-2556.31 (23.86%)	-2868.63 (14.56%)	-2179.67 (35.08%)	-2556.24 (23.86%)	-2865.50 (14.65%)
GlyGlyGly1	-3201.414	-3201.417	-2213.12 (30.87%)	-2547.02 (20.44%)	-2776.16 (13.28%)	-2212.37 (30.89%)	-2548.45 (20.40%)	-2781.38 (13.12%)
GlyGlyGly2	-3330.655	-3330.660	-2213.12 (33.55%)	-2547.02 (23.53%)	-2776.16 (16.65%)	-2211.48 (33.60%)	-2544.68 (23.60%)	-2777.47 (16.61%)
		MAPE =	28.64%	17.72%	9.93%	28.68%	17.73%	9.91%

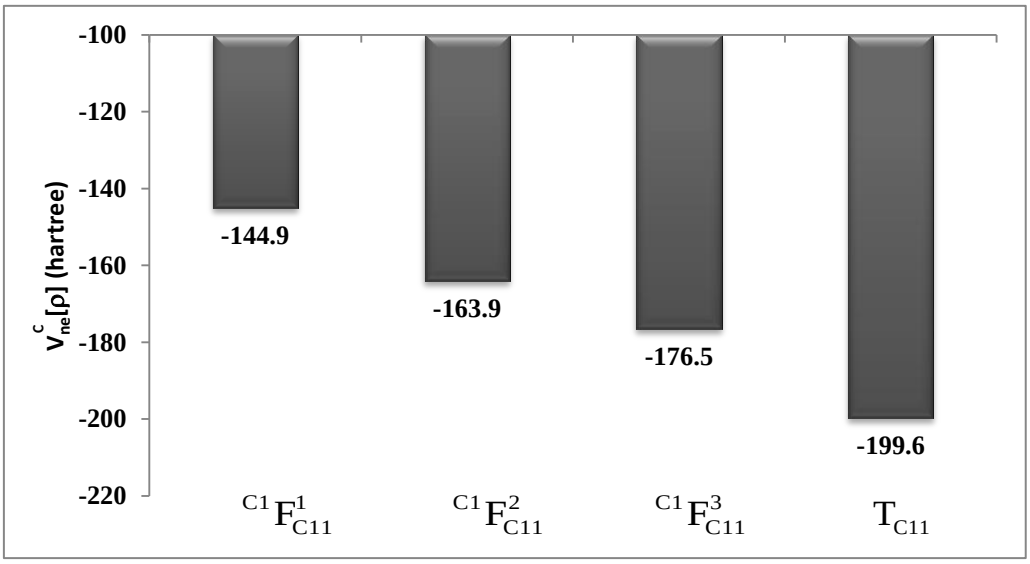


Figure 7.14: Effect of size of the fragment on the atomic potential energy density ($V_{ne}[\rho]$) of target carbon atom.

sity of three mono-amino acids (glycine Gly, alanine Ala, and threonine Thr), six di-amino acids (GlyGly, AlaAla, ThrThr, GlyAla, GlyThr, and AlaThr), and two isomers of tri-amino acid (GlyGlyGly1 and GlyGlyGly2) calculated using the sum over atomic potential energy densities that stored in the database ($V^{sum}_{ne,database}[\rho] = \sum_{A=1} V^{A}_{ne,database}[\rho]$) and by using the direct method ($V^{sum}_{ne,direct}[\rho] = \sum_{A=1} V^{A}_{ne,direct}[\rho]$) of $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ fragments with those calculated numerically using Equation 6.73 for the target molecule. Unfortunately, the MAPE for these molecules is very large. The MAPE of $V^{sum}_{ne,database}[\rho]$ are about 28.64%, 17.72%, and 9.93% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively, and the MAPE of $V^{sum}_{ne,direct}[\rho]$ are about 28.68%, 17.73%, and 9.91% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively. Thus, we cannot consider the potential energy as an atomic property as its value depends very much on the geometry and other atoms of the molecule.

In order to compute the molecular potential energy correctly for the molecule of interest,

one can calculate the molecular potential energy from its atomic electron densities ($V_{ne}[\rho]$) by using Equation 6.73. The atomic electron densities can be built using two different ways: from the database or by using the direct way. In the first way the Cartesian coordinates for the radial grids of the target atom are transformed from the database into the coordinates of the molecule of interest as discussed in section 5.5. In the direct way, the electron densities for each atom are computed in a specific environment (fragment) by keeping the same Cartesian coordinates of the atoms and their radial grids in the molecule of interest. Computing the molecular potential energy for the molecule from its atomic electron densities involves performing numerical integration calculations. Thus, these calculations need computer time and may produce large errors.

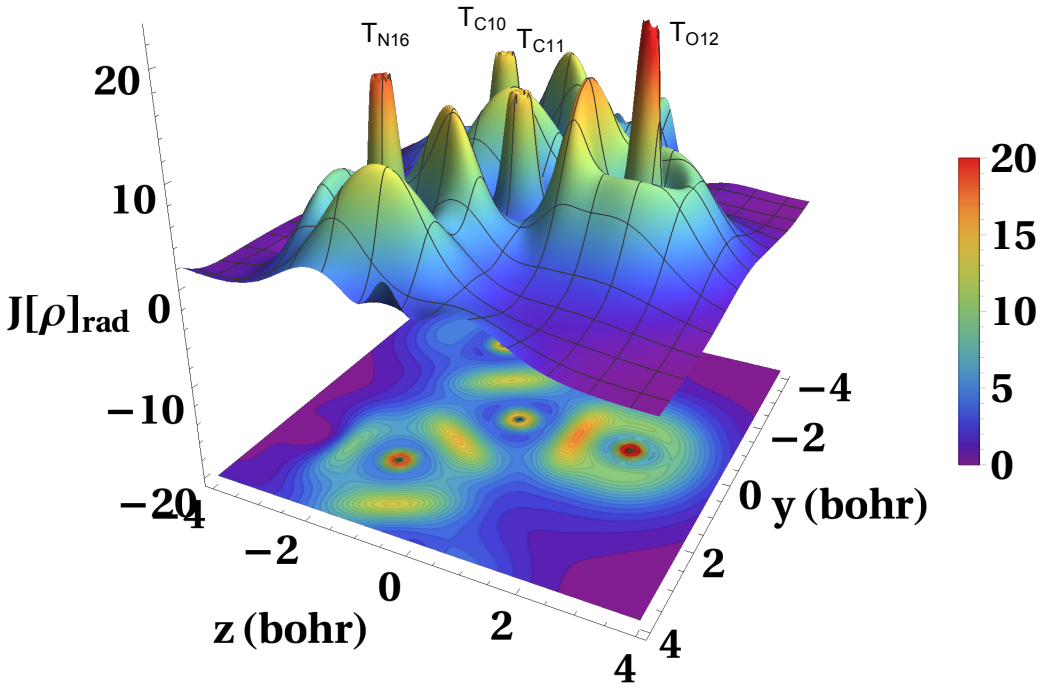
To investigate the validity of our results, we compare the calculated potential energies using the atomic electron densities stored in the database ($V_{ne}[\rho_{database}]$) and using the direct method ($V_{ne}[\rho_{direct}]$) for the $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ fragments with those computed numerically ($V_{ne}[\rho_{target}]$) for the target molecule (Equation 6.31). The results are tabulated in Table 7.6, The MAPE of $V_{ne}[\rho_{database}]$ are about 0.046%, 0.087%, and 0.066% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively, and the MAPE of $V_{ne}[\rho_{direct}]$ are about 0.034%, 0.085%, and 0.051% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively. The results are encouraging, in which the %errors are small and the %errors of $V_{ne}[\rho_{database}]$ and $V_{ne}[\rho_{direct}]$ are almost same.

7.6 Coulomb Energy (J)

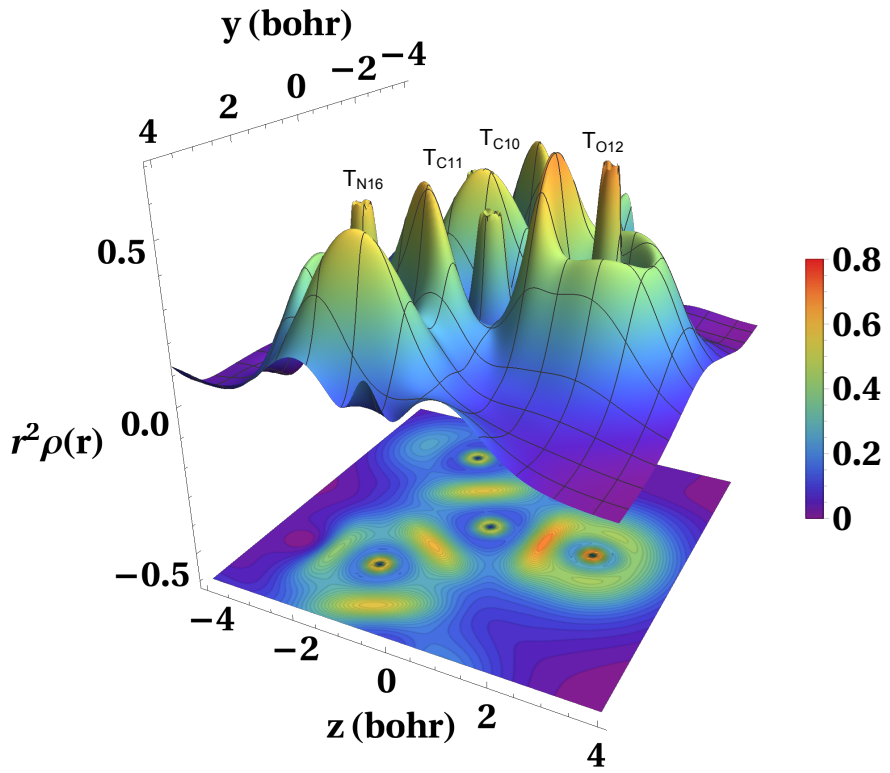
3D-contour plots of the molecular radial Coulomb energy density (Equation 6.68) and molecular radial electron density (RDEN) of the target carbon ($\mathbf{T_{C11}}$) atom in the D molecule are shown in Figures 7.15a and b, respectively. As shown in Figures 7.15a and b, the ra-

Table 7.6: Comparing the molecular potential energy densities using the atomic electron densities that are stored in the database ($V_{ne}[\rho_{database}]$) and by using the direct method ($V_{ne}[\rho_{direct}]$) with those calculated numerically (Equation 6.73) for the target molecule. $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ are the size of fragments as first, second, and third neighbor atoms, respectively.

	Target molecule $\mathbf{V_{ne}}$		**Fragments (Using the electron densities and the fragments partitioning weights)** $\mathbf{V_{ne}[\rho_{database}]}$ **(%error$_{numerical}$)**			$\mathbf{V_{ne}[\rho_{direct}]}$ **(%error$_{numerical}$)**		
Molecule	**(Analytical)**	**(Numerical)**	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$
Gly	-1028.734	-1028.736	-1028.755 (-0.002%)	-1029.468 (-0.071%)	-1028.834 (-0.009%)	-1028.754 (-0.002%)	-1029.436 (-0.068%)	-1028.883 (-0.014%)
Ala	-1259.491	-1259.493	-1258.967 (0.042%)	-1260.251 (-0.060%)	-1260.105 (-0.049%)	-1258.949 (0.043%)	-1260.214 (-0.057%)	-1259.963 (-0.037%)
Thr	-1847.851	-1847.852	-1846.398 (0.079%)	-1848.621 (-0.042%)	-1848.478 (-0.034%)	-1846.642 (0.065%)	-1848.838 (-0.053%)	-1848.201 (-0.019%)
GlyGly	-2067.008	-2067.014	-2066.854 (0.008%)	-2068.954 (-0.094%)	-2067.938 (-0.045%)	-2067.155 (-0.007%)	-2069.214 (-0.106%)	-2067.985 (-0.047%)
AlaAla	-2637.297	-2637.298	-2635.765 (0.058%)	-2639.641 (-0.089%)	-2639.792 (-0.095%)	-2636.073 (0.046%)	-2639.654 (-0.089%)	-2639.311 (-0.076%)
ThrThr	-4080.834	-4080.823	-4076.722 (0.100%)	-4084.093 (-0.080%)	-4084.009 (-0.078%)	-4077.939 (0.071%)	-4083.728 (-0.071%)	-4082.602 (-0.044%)
GlyAla	-2332.480	-2332.480	-2331.687 (0.034%)	-2334.602 (-0.091%)	-2334.339 (-0.080%)	-2331.956 (0.022%)	-2334.717 (-0.096%)	-2334.037 (-0.067%)
GlyThr	-3048.499	-3048.490	-3046.485 (0.066%)	-3051.753 (-0.107%)	-3051.035 (-0.083%)	-3047.279 (0.040%)	-3050.979 (-0.082%)	-3050.179 (-0.055%)
AlaThr	-3357.328	-3357.332	-3354.497 (0.084%)	-3360.457 (-0.093%)	-3360.834 (-0.104%)	-3355.338 (0.059%)	-3359.983 (-0.079%)	-3359.568 (-0.067%)
GlyGlyGly1	-3201.414	-3201.417	-3201.157 (0.008%)	-3205.321 (-0.122%)	-3203.989 (-0.080%)	-3201.762 (-0.011%)	-3205.277 (-0.121%)	-3203.830 (-0.075%)
GlyGlyGly2	-3330.655	-3330.660	-3329.984 (0.020%)	-3334.231 (-0.107%)	-3332.867 (-0.066%)	-3330.791 (-0.004%)	-3334.269 (-0.108%)	-3332.496 (-0.055%)
		MAPE =	0.046%	0.087%	0.066%	0.034%	0.085%	0.051%



(a) The molecular radial Coulomb energy density using Equation 6.68.



(b) The molecular radial electron density (RDEN).

Figure 7.15: Comparing the molecular radial electron density (RDEN) and the molecular radial Coulomb energy density $J[\rho]_{rad}$ (Equation 6.68) of molecule D (Figure 7.3). $\mathbf{T_{C11}}$ is the target carbon atom.

dial Coulomb energy density $J[\rho]_{rad}$ and RDEN have a similar shape. The similarity of Coulomb energy with RDEN may be due to the proportionality between the radial Coulomb energy and RDEN.

The radial Coulomb energy densities (Equation 6.68) along the bond between the target carbon atom and oxygen atom for: $\mathbf{F}^{1}_{C11}$ (red dash line), $\mathbf{F}^{2}_{C11}$ (green dash line), $\mathbf{F}^{3}_{C11}$ (purple dash line), and the D molecule (blue solid line) are compared in Figure 7.16. As shown in Figure 7.16, the Coulomb energy density is proportional to the size of molecules (e.g., the molecular radial Coulomb energy density for D molecule is larger than the fragment $\mathbf{F}^{3}_{C11}$).

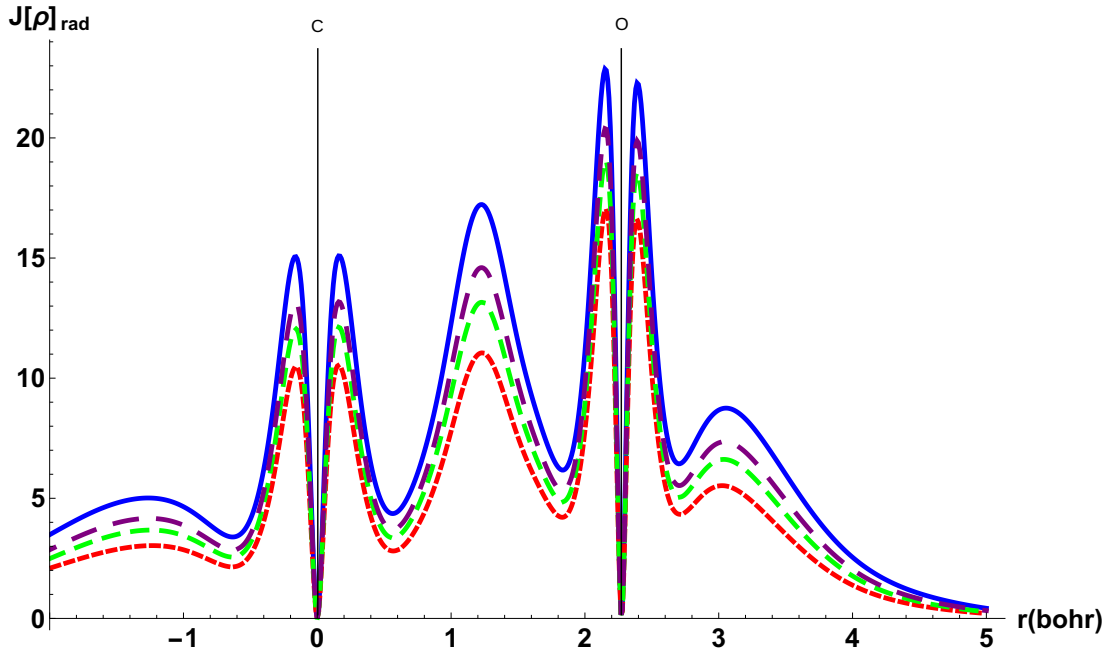


Figure 7.16: The radial Coulomb energy density (Equation 6.68) along the bond between the target carbon atom and the oxygen atom for $\mathbf{F}^{1}_{C11}$ (red dash line), $\mathbf{F}^{2}_{C11}$ (green dash line), $\mathbf{F}^{3}_{C11}$ (purple dash line), and the molecule of interest, D molecule, (blue solid line). The carbon is located at 0.0 bohr and oxygen is located at 2.2739 bohr.

Figure 7.17 shows the Coulomb energies of the target carbon atoms: $^{C1}\mathbf{F}^{1}_{C11}$, $^{C1}\mathbf{F}^{2}_{C11}$,

$^{\mathbf{C1}}\mathbf{F}^{\mathbf{3}}_{\mathbf{C11}}$, and $\mathbf{T_{C11}}$ obtained using Equation 6.65 ($V(\mathbf{r})$ is computed analytically). The results show that the atomic Coulomb energies of the target carbon atom are proportional to the size of the molecule.

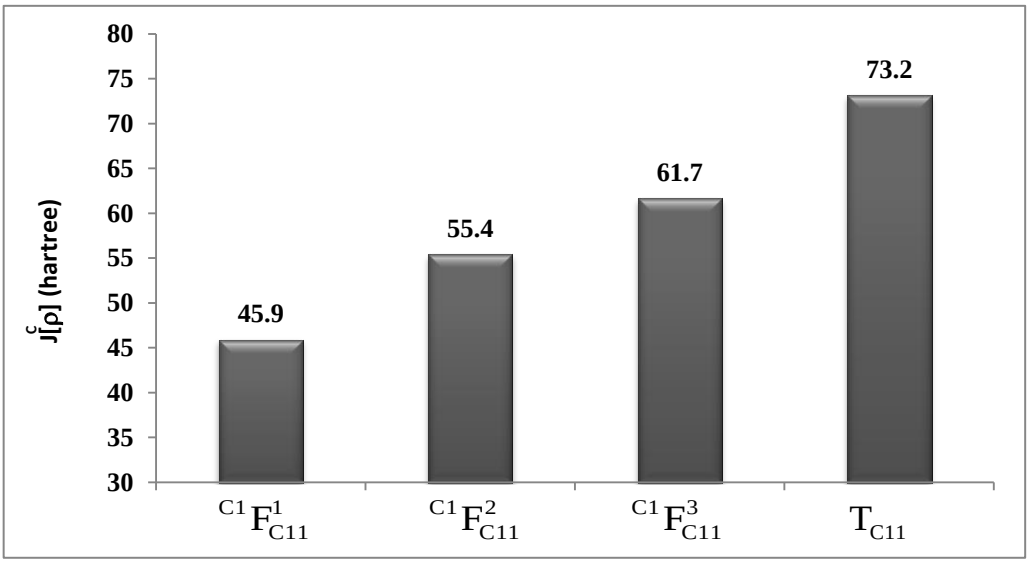


Figure 7.17: Effect of size of the fragment on the atomic Coulomb energy density ($J[\rho]$) of target carbon atom (Equation 6.65).

The molecular Coulomb energy is calculated by summing over atomic Coulomb energies,

$$J^{sum}[\rho] = \sum_{A=1}^{M} J^{A}[\rho] \tag{7.7}$$

where M is the number of atoms in the target molecule and $J^{A}[\rho]$ is the Coulomb energy of atom A computed using Equation 6.65. Table 7.7 compares the Coulomb energy of three mono-amino acids (glycine Gly, alanine Ala, and threonine Thr), six di-amino acids (GlyGly, AlaAla, ThrThr, GlyAla, GlyThr, and AlaThr), and two isomers of tri-amino acid (GlyGlyGly1 and GlyGlyGly2) calculated using the sum over atomic Coulomb ener-

Table 7.7: Comparing the molecular Coulomb energies calculated using the sum over atomic Coulomb energies that are stored in the database ($J^{sum}_{database}[\rho] = \sum_{A=1} J^{A}_{database}[\rho]$) and by using the direct method ($J^{sum}_{direct}[\rho] = \sum_{A=1} J^{A}_{direct}[\rho]$) with those calculated analytically/numerically (Equation 6.65) for the target molecule. $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ are the size of fragments as first, second, and third neighbor atoms, respectively.

	Target molecule J[ρ]		Fragments (Sum over atoms)					
			$J^{sum}_{database}[\rho] = \sum_{A} J^{A}_{database}[\rho]$ (%error$_{analytical/numerical}$)			$J^{sum}_{direct}[\rho] = \sum_{A} J^{A}_{direct}[\rho]$ (%error$_{analytical/numerical}$)		
Molecule	**(Analytical)**	**(Analytical/ Numerical)**	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$
Gly	317.4368	317.4378	239.44 (24.57%)	290.93 (8.35%)	314.85 (0.81%)	239.44 (24.57%)	290.90 (8.36%)	314.66 (0.88%)
Ala	405.9641	405.9650	281.36 (30.69%)	354.08 (12.78%)	399.47 (1.60%)	281.29 (30.71%)	354.00 (12.80%)	399.41 (1.61%)
Thr	620.8853	620.8858	383.86 (38.17%)	491.83 (20.79%)	575.02 (7.39%)	383.33 (38.26%)	491.85 (20.78%)	574.60 (7.45%)
GlyGly	692.5468	692.5492	430.61 (37.82%)	539.98 (22.03%)	609.96 (11.93%)	430.49 (37.84%)	539.92 (22.04%)	609.73 (11.96%)
AlaAla	923.9177	923.9179	514.45 (44.32%)	666.40 (27.87%)	784.59 (15.08%)	514.20 (44.35%)	666.40 (27.87%)	784.70 (15.07%)
ThrThr	1487.2822	1487.2767	719.45 (51.63%)	941.49 (36.70%)	1135.34 (23.66%)	717.96 (51.73%)	941.12 (36.72%)	1133.59 (23.78%)
GlyAla	798.3259	798.3258	472.53 (40.81%)	603.19 (24.44%)	694.14 (13.05%)	472.40 (40.83%)	603.34 (24.42%)	694.29 (13.03%)
GlyThr	1076.9553	1076.9508	575.03 (46.61%)	740.60 (31.23%)	873.89 (18.85%)	574.27 (46.68%)	740.76 (31.22%)	874.50 (18.80%)
AlaThr	1204.5755	1204.5772	616.95 (48.78%)	803.81 (33.27%)	960.28 (20.28%)	616.14 (48.85%)	803.92 (33.26%)	958.60 (20.42%)
GlyGlyGly1	1115.4045	1115.4058	621.77 (44.26%)	788.57 (29.30%)	903.13 (19.03%)	621.57 (44.27%)	789.29 (29.24%)	905.67 (18.80%)
GlyGlyGly2	1180.0865	1180.0886	621.77 (47.31%)	788.57 (33.18%)	903.13 (23.47%)	621.17 (47.36%)	787.56 (33.26%)	903.72 (23.42%)
		MAPE =	41.36%	25.45%	14.11%	41.40%	25.45%	14.11%

gies stored in the database ($J^{sum}_{database}[\rho] = \sum_{A=1} J^{A}_{database}[\rho]$) and by using the direct method ($J^{sum}_{direct}[\rho] = \sum_{A=1} J^{A}_{direct}[\rho]$) of $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ fragments size with those calculated analytically/numerically using Equation 6.65 for the target molecule. Unfortunately, the MAPEs for these molecules is very large. The MAPE of $J^{sum}_{database}[\rho]$ are about 41.36%, 25.45%, and 14.11% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively, and the MAPE of $J^{sum}_{direct}[\rho]$ are about 41.40%, 25.45%, and 14.11% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively. Thus, we cannot consider the Coulomb energy, $J[\rho]$, as an atomic property as its value depends on the geometry and other atoms of the molecule.

Figure 7.18 compares the molecular radial Coulomb energy density obtained by using the molecular orbital expansion (Equation 6.39) along the bond between the target carbon atom and the oxygen atom for $\mathbf{F^1_{C11}}$ (red dash line), $\mathbf{F^2_{C11}}$ (green dash line), $\mathbf{F^3_{C11}}$ (purple dash line), and the D molecule (blue solid line). As shown in Figure 7.18, the molecular radial Coulomb energy density is proportional to the size of molecules (e.g., the Coulomb energy density for the D molecule is larger than the fragment $\mathbf{F^3_{C11}}$).

The computed Coulomb energies obtained by using the molecular orbital expansion (Equation 6.52) of the target carbon atom for the four molecules $\mathbf{F^1_{C11}}$, $\mathbf{F^2_{C11}}$, $\mathbf{F^3_{C11}}$, and the D molecule are displayed in Figure 7.19. The results as displayed in Figure 7.19 show that the atomic Coulomb energies, J, (Equation 6.52) of the carbon atom are proportional to the size of the molecule.

The molecular Coulomb energy is calculated by summing over atomic Coulomb energies,

$$J^{sum} = \sum_{A=1}^{M} J^{A} \tag{7.8}$$

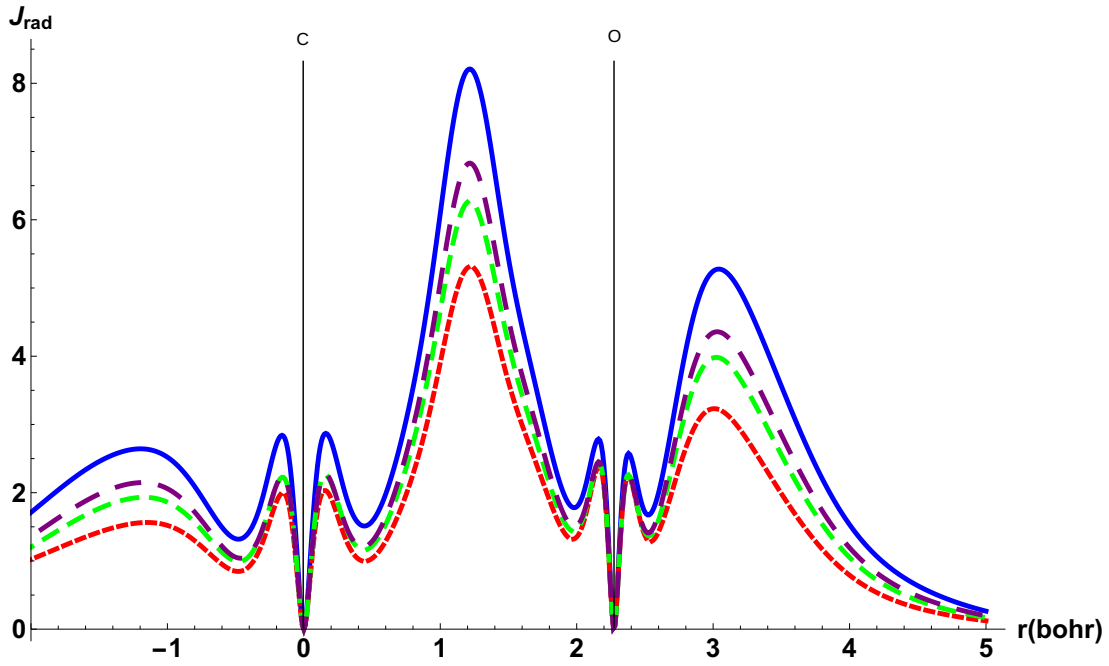


Figure 7.18: The radial Coulomb energy density obtained by using the molecular orbital expansion (Equation 6.39) along the bond between the target carbon atom and the oxygen atom for $\mathbf{F}^{1}_{C11}$ (red dash line), $\mathbf{F}^{2}_{C11}$ (green dash line), $\mathbf{F}^{3}_{C11}$ (purple dash line), and the molecule of interest, D molecule, (blue solid line). The carbon is located at 0.0 bohr and oxygen is located at 2.2739 bohr.

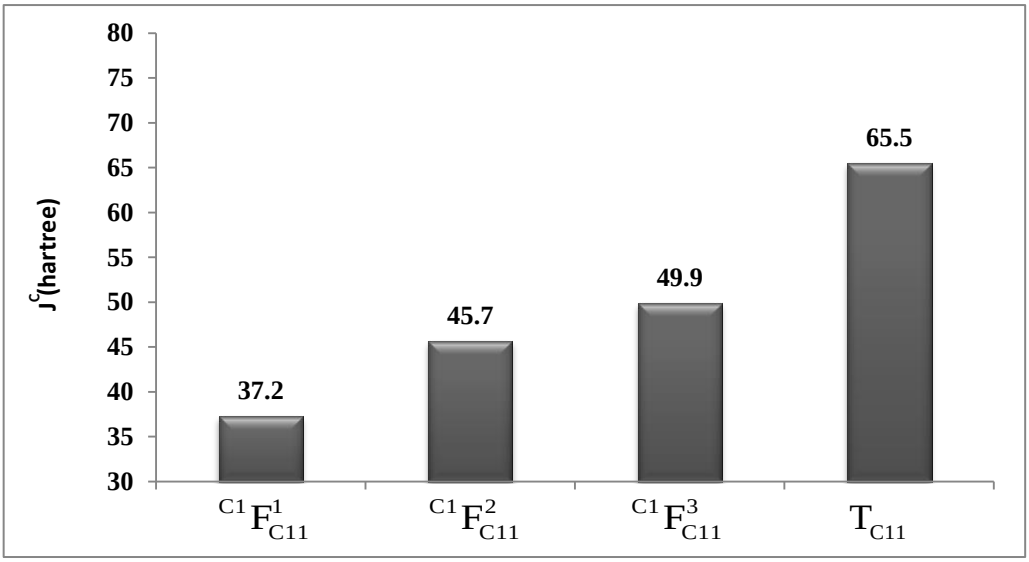


Figure 7.19: Effect of size of the fragment on the atomic Coulomb energy (J) of target carbon atom.

where J^A is the Coulomb energy of atom A computed using Equation 6.52 and M is the number of atoms in the target molecule. Table 7.8 compares the Coulomb energies, J, computed using the sum over atomic Coulomb energies stored in the database ($J^{sum}_{database} = \sum_{A=1} J^A_{database}$) and by using the direct method ($J^{sum}_{direct} = \sum_{A=1} J^A_{direct}$) of $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ fragments size with those calculated using Equation 6.52 for the target molecule. Unfortunately, the MAPE for these molecules is very large. The MAPE of $J^{sum}_{database}$ are about 46.95%, 29.12%, and 16.48% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively, and the MAPE of J^{sum}_{direct} are about 47.08%, 29.14%, and 16.48% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively. Thus we cannot consider the Coulomb energy obtained by using the molecular orbital expansion as an atomic property as its value depends on the geometry and other atoms of the molecule.

Figure 7.20 compares the $J_{HF,rad}$ (i.e., the radial Coulomb energy contain J_{aa} term) along the bond between the target carbon atom and the oxygen atom for $\mathbf{F^1_{C11}}$ (red dash line), $\mathbf{F^2_{C11}}$ (green dash line), $\mathbf{F^3_{C11}}$ (purple dash line), and the D molecule (blue solid line). As shown in Figure 7.20, the $J_{HF,rad}$ is depending on the size of the molecule (e.g., the $J_{HF,rad}$ for the D molecule is larger than the fragment $\mathbf{F^1_{C11}}$).

The computed J^A_{HF} (Equation 6.54) of the target carbon atom for the four molecules $\mathbf{F^1_{C11}}$, $\mathbf{F^2_{C11}}$, $\mathbf{F^3_{C11}}$, and the D molecule are displayed in Figure 7.21. The results as displayed in Figure 7.21 show that the J^A_{HF} (Equation 6.54) of the carbon atom are directly proportional to the size of the molecules.

The J_{HF} is calculated by summing over J^A_{HF},

$$J^{sum}_{HF} = \sum_{A=1}^{M} J^A_{HF} \tag{7.9}$$

where M is the number of atoms in the target molecule.

Table 7.8: Comparing the molecular Coulomb energy densities calculated using the sum over atomic Coulomb energy densities that are stored in the database ($J^{sum}_{database} = \sum_{A=1} J^{A}_{database}$) and by using the direct method ($J^{sum}_{direct} = \sum_{A=1} J^{A}_{direct}$) with those calculated numerically (Equation 6.30) for the target molecule. $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ are the size of fragments as first, second, and third neighbor atoms, respectively.

	Target molecule	**Fragments (Sum over atoms)**					
		$\mathbf{J^{sum}_{database} = \sum_{A} J^{A}_{database}}$ **(%error)**			$\mathbf{J^{sum}_{direct} = \sum_{A} J^{A}_{direct}}$ **(%error)**		
Molecule	**J (Numerical)**	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$
Gly	290.199002	202.42 (30.25%)	259.77 (10.49%)	286.88 (1.14%)	202.42 (30.25%)	259.71 (10.51%)	286.45 (1.29%)
Ala	374.521914	238.90 (36.21%)	314.99 (15.89%)	365.19 (2.49%)	238.88 (36.22%)	314.58 (16.00%)	365.05 (2.53%)
Thr	579.717747	325.06 (43.93%)	437.69 (24.50%)	524.28 (9.56%)	323.13 (44.26%)	437.48 (24.54%)	524.05 (9.60%)
GlyGly	646.396794	363.78 (43.72%)	481.22 (25.55%)	555.35 (14.09%)	363.70 (43.73%)	481.18 (25.56%)	554.99 (14.14%)
AlaAla	869.305727	436.75 (49.76%)	591.60 (31.95%)	715.80 (17.66%)	435.55 (49.90%)	591.05 (32.01%)	715.75 (17.66%)
ThrThr	1413.300372	609.06 (56.91%)	836.42 (40.82%)	1028.11 (27.25%)	605.89 (57.13%)	836.22 (40.83%)	1029.45 (27.16%)
GlyAla	747.778136	400.26 (46.47%)	536.57 (28.24%)	633.28 (15.31%)	400.30 (46.47%)	536.55 (28.25%)	633.27 (15.31%)
GlyThr	1016.955671	486.42 (52.17%)	658.59 (35.24%)	791.25 (22.19%)	484.44 (52.36%)	658.77 (35.22%)	793.13 (22.01%)
AlaThr	1140.623478	522.90 (54.16%)	713.94 (37.41%)	871.94 (23.56%)	521.24 (54.30%)	713.98 (37.40%)	871.56 (23.59%)
GlyGlyGly1	1049.620738	525.15 (49.97%)	702.05 (33.11%)	822.20 (21.67%)	524.98 (49.98%)	702.77 (33.05%)	823.36 (21.56%)
GlyGlyGly2	1116.168212	525.15 (52.95%)	702.05 (37.10%)	822.20 (26.34%)	521.08 (53.32%)	701.23 (37.18%)	821.00 (26.44%)
	MAPE =	46.95%	29.12%	16.48%	47.08%	29.14%	16.48%

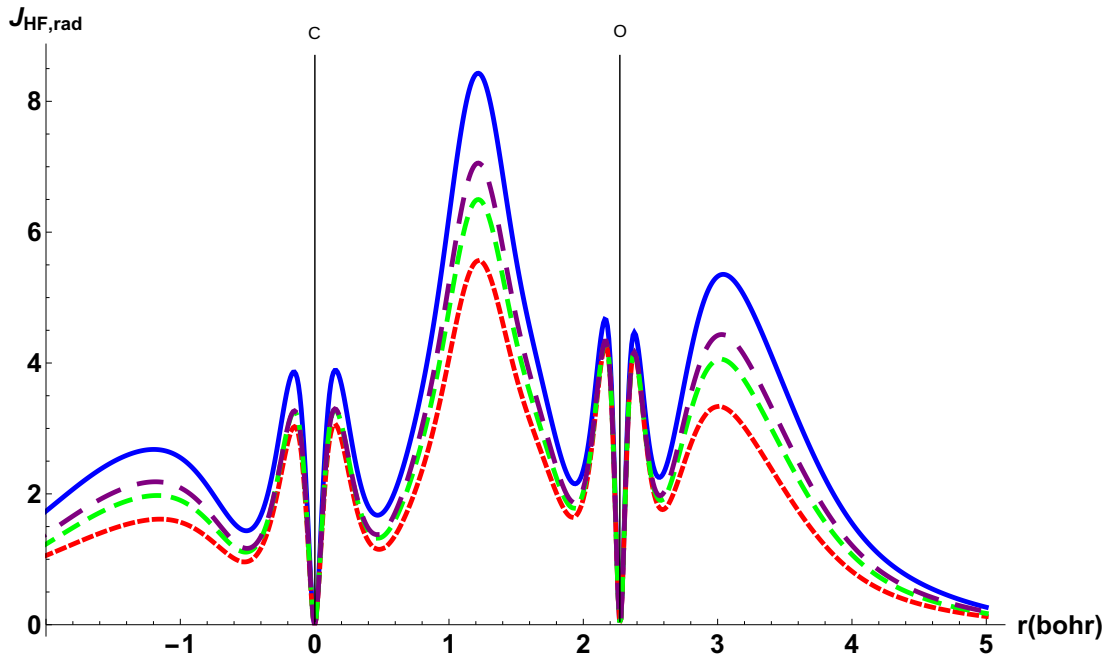


Figure 7.20: The $J_{HF,rad}$ using the molecular orbital expansion (Equation 6.38) along the bond between the target carbon atom and the oxygen atom for $\mathbf{F}^{1}_{C11}$ (red dash line), $\mathbf{F}^{2}_{C11}$ (green dash line), $\mathbf{F}^{3}_{C11}$ (purple dash line), and the molecule of interest, D molecule, (blue solid line). The carbon is located at 0.0 bohr and oxygen is located at 2.2739 bohr.

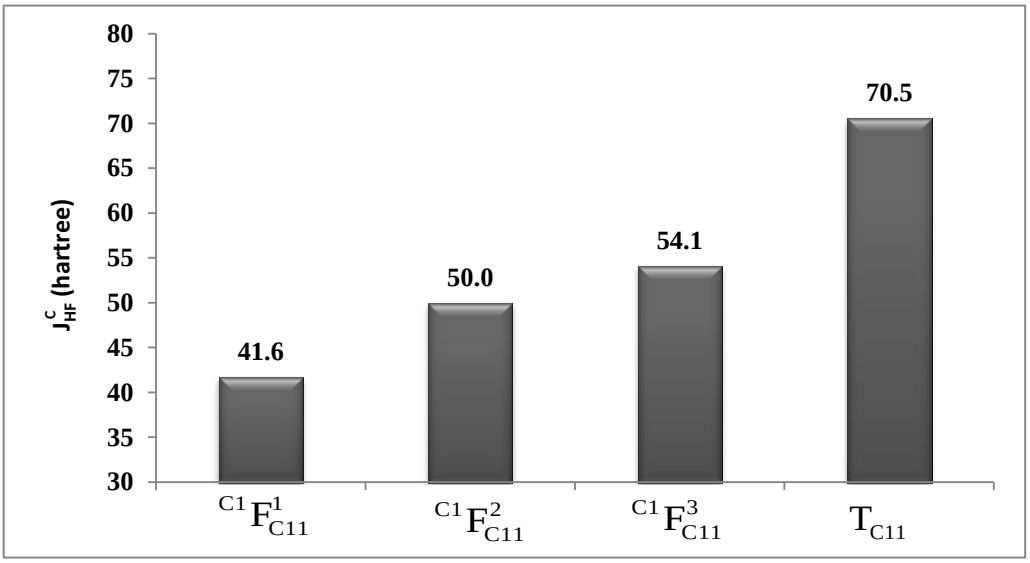


Figure 7.21: Effect of size of the fragment on the J^{A}_{HF} of target carbon atom.

Table 7.9: Comparing J_{HF} calculated using the sum over J^{A}_{HF} that are stored in the database ($J^{sum}_{HF,database} = \sum_{A=1} J^{A}_{HF,database}$) and by using the direct method ($J^{sum}_{HF,direct} = \sum_{A=1} J^{A}_{HF,direct}$) with those calculated using Equation 6.54 for the target molecule. $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ are the size of fragments as first, second, and third neighbor atoms, respectively.

	Target molecule	Fragments (Sum over atoms) $J^{sum}_{HF,database} = \sum_{A} J^{A}_{HF,database}$ (%error)			$J^{sum}_{HF,direct} = \sum_{A} J^{A}_{HF,direct}$ (%error)		
Molecule	**J_{HF} (Numerical)**	**F^1**	**F^2**	**F^3**	**F^1**	**F^2**	**F^3**
Gly	317.4368	231.44 (27.09%)	287.31 (9.49%)	314.15 (1.04%)	231.43 (27.09%)	287.25 (9.51%)	313.74 (1.17%)
Ala	405.9635	272.00 (33.00%)	346.95 (14.54%)	396.72 (2.28%)	271.98 (33.00%)	346.56 (14.63%)	396.60 (2.31%)
Thr	620.8816	369.61 (40.47%)	480.43 (22.62%)	565.98 (8.84%)	367.66 (40.78%)	480.24 (22.65%)	565.79 (8.87%)
GlyGly	692.5425	414.58 (40.14%)	529.27 (23.58%)	602.32 (13.03%)	414.50 (40.15%)	529.23 (23.58%)	601.96 (13.08%)
AlaAla	923.9067	495.71 (46.35%)	648.59 (29.80%)	771.71 (16.47%)	494.62 (46.46%)	648.08 (29.85%)	771.69 (16.48%)
ThrThr	1487.2513	690.93 (53.54%)	914.98 (38.48%)	1104.45 (25.74%)	688.11 (53.73%)	914.81 (38.49%)	1105.88 (25.64%)
GlyAla	798.3170	455.15 (42.99%)	589.08 (26.21%)	684.87 (14.21%)	454.96 (43.01%)	589.08 (26.21%)	684.90 (14.21%)
GlyThr	1076.9370	552.75 (48.67%)	721.91 (32.97%)	852.74 (20.82%)	551.11 (48.83%)	722.19 (32.94%)	855.05 (20.60%)
AlaThr	1204.5590	593.32 (50.74%)	781.73 (35.10%)	938.05 (22.12%)	591.87 (50.86%)	781.83 (35.09%)	937.84 (22.14%)
GlyGlyGly1	1115.3876	597.73 (46.41%)	770.63 (30.91%)	889.14 (20.28%)	597.57 (46.42%)	771.41 (30.84%)	890.65 (20.15%)
GlyGlyGly2	1180.0750	597.73 (49.35%)	770.63 (34.70%)	889.14 (24.65%)	593.66 (49.69%)	769.85 (34.76%)	887.63 (24.78%)
	MAPE =	43.52%	27.13%	15.41%	43.64%	27.14%	15.40%

Table 7.9 compares the J_{HF} computed using the sum over J^{A}_{HF} stored in the database ($J^{sum}_{HF,database} = \sum_{A=1} J^{A}_{HF,database}$) and by using the direct method ($J^{sum}_{HF,direct} = \sum_{A=1} J^{A}_{HF,direct}$) of $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ fragments size with those calculated for the target molecule (Equation 6.54). Unfortunately, the MAPE for these molecules is very large. The MAPE of $J^{sum}_{HF,database}$ are about 43.52%, 27.13%, and 15.41% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively, and the MAPE of $J^{sum}_{HF,direct}$ are about 43.64%, 27.14%, and 15.40% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively. Thus we cannot consider the J_{HF} as atomic property as its value depends on the geometry and other atoms of the molecule.

In order to compute the molecular Coulomb energy correctly for the molecule of interest, one can calculate the molecular Coulomb energy from its atomic electron densities ($J[\rho]$) by using Equation 6.69. The atomic electron densities can be built using two different ways: from the database or by using the direct way. In the first way the Cartesian coordinates for the radial grids of the target atom are transformed from the database into the coordinates of the molecule of interest as discussed in section 5.5. In the direct way, the electron densities for each atom are computed in a specific environment (fragment) by keeping the same Cartesian coordinates of the atoms and their radial grids in the molecule of interest. Computing the molecular potential energy for the molecule from its atomic electron densities involves performing full numerical integration calculations (i.e., the two integrals in Equation 6.58). Thus, these calculations need computer time and may produce large errors.

To investigate the validity of our results, we compare the calculated Coulomb energies using the atomic electron densities stored in the database ($J[\rho_{database}]$) and using the direct method ($J[\rho_{direct}]$) for the $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ fragments with those computed numerically (i.e., the two integrals in Equation 6.58) for the target molecule ($J[\rho_{target}]$). The results are tab-

Table 7.10: Comparing the molecular Coulomb energy densities using the atomic electron densities that are stored in the database ($J[\rho_{database}]$) and by using the direct method ($J[\rho_{direct}]$) with those calculated numerically (Equation 6.58) for the target molecule. $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ are the size of fragments as first, second, and third neighbor atoms, respectively.

	Target molecule		**Fragments (Using the electron densities and the fragments partitioning weights)**					
	J[ρ]		**$J[\rho_{database}]$ (%error$_{numerical/numerical}$)**			**$J[\rho_{direct}]$ (%error$_{numerical/numerical}$)**		
Molecule	**(Analytical)**	**(Numerical/ Numerical)**	**F^1**	**F^2**	**F^3**	**F^1**	**F^2**	**F^3**
Gly	317.437	317.773	317.880 (-0.034%)	318.472 (-0.220%)	317.864 (-0.029%)	317.882 (-0.034%)	318.443 (-0.211%)	317.912 (-0.044%)
Ala	405.964	406.382	406.010 (0.092%)	407.122 (-0.182%)	406.976 (-0.146%)	406.001 (0.094%)	407.086 (-0.173%)	406.835 (-0.111%)
Thr	620.885	621.436	620.181 (0.202%)	622.198 (-0.123%)	622.038 (-0.097%)	620.503 (0.150%)	622.408 (-0.157%)	621.774 (-0.054%)
GlyGly	692.547	693.175	693.155 (0.003%)	694.961 (-0.258%)	694.074 (-0.130%)	693.481 (-0.044%)	695.316 (-0.309%)	694.112 (-0.135%)
AlaAla	923.918	924.712	923.436 (0.138%)	927.020 (-0.250%)	927.156 (-0.264%)	923.789 (0.100%)	927.030 (-0.251%)	926.679 (-0.213%)
ThrThr	1487.282	1488.316	1484.543 (0.254%)	1491.428 (-0.209%)	1491.340 (-0.203%)	1485.992 (0.156%)	1491.237 (-0.196%)	1490.073 (-0.118%)
GlyAla	798.326	799.034	798.437 (0.075%)	801.100 (-0.258%)	800.850 (-0.227%)	798.738 (0.037%)	801.213 (-0.273%)	800.549 (-0.190%)
GlyThr	1076.955	1077.785	1076.010 (0.165%)	1080.876 (-0.287%)	1080.126 (-0.217%)	1076.943 (0.078%)	1080.255 (-0.229%)	1079.446 (-0.154%)
AlaThr	1204.576	1205.498	1202.960 (0.211%)	1208.497 (-0.249%)	1208.840 (-0.277%)	1203.940 (0.129%)	1208.138 (-0.219%)	1207.698 (-0.183%)
GlyGlyGly1	1115.404	1116.319	1116.234 (0.008%)	1120.124 (-0.341%)	1118.910 (-0.232%)	1116.892 (-0.051%)	1120.079 (-0.337%)	1118.669 (-0.211%)
GlyGlyGly2	1180.087	1180.997	1180.492 (0.043%)	1184.355 (-0.284%)	1183.128 (-0.180%)	1181.402 (-0.034%)	1184.571 (-0.303%)	1182.794 (-0.152%)
		MAPE =	0.111%	0.242%	0.182%	0.083%	0.242%	0.142%

ulated in Table 7.10, the MAPE of $J[\rho_{database}]$ are about 0.111%, 0.242%, and 0.182% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively, and the MAPE of $J[\rho_{direct}]$ are about 0.083%, 0.242%, and 0.142% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively. The results are encouraging, the %errors are small and computing of $J[\rho_{database}]$ and $J[\rho_{direct}]$ have almost the same %errors.

7.7 Total Energy

In this section, we obtain the total molecular energy as follows,

$$E = T^{sum} + V_{ne}[\rho] + J[\rho] - K_{HF}^{sum} + V_{nn} \tag{7.10}$$

where, T^{sum} is the molecular kinetic energy density calculated by summing over the atomic kinetic energy densities (Equation 7.3), K_{HF}^{sum} is the molecular exchange energy contain J_{aa} calculated by summing over the K_{HF}^{A} (Equation 7.5), $V_{ne}[\rho]$ is the molecular potential energy calculated numerically using its atomic electron densities (Equation 6.73), $J[\rho]$ is the molecular Coulomb energy calculated numerically using its atomic electron densities (Equation 6.69), and V_{nn} is the molecular nuclear-nuclear repulsion energy calculated directly for the entire molecule as,

$$V_{nn} = \sum_{A} \sum_{B>A} \frac{Z_A Z_B}{R_{AB}} \tag{7.11}$$

where Z_A and Z_B are the atomic numbers of atoms A and B respectively and R_{AB} is the distance between atoms A and B.

Table 7.11 compares the total energy of three mono-amino acids (glycine Gly, alanine Ala, and threonine Thr), six di-amino acids (GlyGly, AlaAla, ThrThr, GlyAla, GlyThr,

and AlaThr), and two isomers of tri-amino acid (GlyGlyGly1 and GlyGlyGly2) calculated using atomic properties and the electron densities that stored in the database $E_{database}$ and that obtained using the direct methods E_{direct} of $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ fragments size with those calculated numerically for the target molecule. The MAPE for these molecules as tabulated in Table 7.11 are relatively small. The MAPE of $E_{database}$ are about 0.165%, 0.017%, and 0.008% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively, and the MAPE of E_{direct} are about 0.164%, 0.022%, and 0.006% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively. Further work is needed to improve our results.

Table 7.11: Comparing the calculated total energy (E) using the atomic properties and the electron densities that are stored in the database $E_{database}$ and that obtained using the direct methods E_{direct} of $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$ fragments size with those calculated numerically for the target molecule.

	Target molecule E		Fragments (Using fragments partitioning weights) $E_{database}(\%error_{numerical})$			$E_{direct}(\%error_{numerical})$		
Molecule	**(Analytical)**	**(Numerical)**	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$	$\mathbf{F^1}$	$\mathbf{F^2}$	$\mathbf{F^3}$
Gly	-282.816	-282.487	-282.110 (0.134%)	-282.472 (0.005%)	-282.484 (0.001%)	-282.116 (0.132%)	-282.476 (0.004%)	-282.485 (0.001%)
Ala	-321.854	-321.453	-320.959 (0.154%)	-321.371 (0.025%)	-321.438 (0.004%)	-320.976 (0.148%)	-321.386 (0.021%)	-321.452 (0.000%)
Thr	-435.739	-435.221	-434.550 (0.154%)	-435.092 (0.030%)	-435.161 (0.014%)	-434.543 (0.156%)	-435.119 (0.023%)	-435.179 (0.010%)
GlyGly	-489.637	-489.017	-488.234 (0.160%)	-489.032 (-0.003%)	-488.996 (0.004%)	-488.222 (0.163%)	-488.950 (0.014%)	-488.996 (0.004%)
AlaAla	-567.708	-566.955	-565.934 (0.180%)	-566.733 (0.039%)	-566.888 (0.012%)	-565.957 (0.176%)	-566.772 (0.032%)	-566.925 (0.005%)
ThrThr	-795.480	-794.457	-793.127 (0.167%)	-794.365 (0.012%)	-794.447 (0.001%)	-793.064 (0.175%)	-794.241 (0.027%)	-794.356 (0.013%)
GlyAla	-528.671	-528.005	-527.087 (0.174%)	-527.836 (0.032%)	-527.949 (0.011%)	-527.109 (0.170%)	-527.872 (0.025%)	-527.986 (0.004%)
GlyThr	-642.556	-641.747	-640.679 (0.166%)	-641.723 (0.004%)	-641.838 (-0.014%)	-640.664 (0.169%)	-641.590 (0.025%)	-641.690 (0.009%)
AlaThr	-681.593	-680.709	-679.529 (0.173%)	-680.585 (0.018%)	-680.717 (-0.001%)	-679.520 (0.175%)	-680.500 (0.031%)	-680.647 (0.009%)
GlyGlyGly1	-696.450	-695.580	-694.367 (0.174%)	-695.443 (0.020%)	-695.630 (-0.007%)	-694.381 (0.172%)	-695.439 (0.020%)	-695.556 (0.003%)
GlyGlyGly2	-696.457	-695.585	-694.384 (0.173%)	-695.570 (0.002%)	-695.736 (-0.022%)	-694.377 (0.174%)	-695.473 (0.016%)	-695.550 (0.005%)
		MAPE =	0.165%	0.017%	0.008%	0.164%	0.022%	0.006%

Bibliography

[1] R. A. Poirier, J. W. Hollett, and P. L. Warburton. MUNgauss, Memorial University, Chemistry Department, St. Johns, NL, A1B 3X7. With contributions from I. E. Awad, A. Alrawashdeh, J. P. Becker, J. Besaw, S. D. Bungay, F. Colonna, A.El-Sherbiny, T. Gosse, D. Keefe, A. Kelly, D. Nippard, C. C. Pye, D. Reid, K.Saputantri, M. Shaw, M. Staveley, O. Stueker, Y. Wang, and J. Xidos.

[2] Wolfram Research, Inc. Mathematica 11.2. URL `https://www.wolfram.com`.

Chapter 8

Conclusions and Future Work

"The truth is, everyone is confused by quantum physics."

— David Walton

8.1 Conclusions and Future Work

8.1.1 Conclusions

In the first part of this thesis, we have successfully developed a new partitioning weight (Awad weight). The Awad weight has flexibility of choosing the core boundary of atoms within the molecule, and has the ability to assign the core to the target atom not to the core or the bond of other atoms. The computer time using the Awad weight is very close to the Becke weight. The effect of choosing the core size on Awad weight is studied visually and numerically. The visualization study includes molecular radial electron density (RDEN) and bond electron density (BDEN). The results show that the topology of RDEN and BDEN are affected by changing the core size. These changes are more pronounced in molecules containing metal atoms such as Li and Na than molecules contain only non-metallic atoms.

The numerical study includes comparing the following molecular properties; number of electrons, potential energy, and Coulomb energy with those calculated by the HF wavefunction (the exact values). The mean absolute error (MAE) for calculated total number of electrons using the Awad weight with $r_{\langle r\rangle}$ and r_{max} are 0.01524 and 0.003665 μe respectively, and it is 0.4916 μe using Becke weight. The MAEs for calculating the potential energy numerically using the Awad weight with $r_{\langle r\rangle}$ and r_{max} are 102.45 and 102.48 μhartree respectively which are slightly better than MAE of 115.01 μhartree obtained using Becke weight. The Coulomb potential energy results show that the Awad weight with $r_{\langle r\rangle}$ and r_{max} gives better results than Becke weight where the MAEs equal 0.09160, 0.08454, and 6.295 μhartree for $r_{\langle r\rangle}$, r_{max}, and Becke weight, respectively.

In the second part of this thesis, a new approach for calculating the total energy of molecules called atoms in molecules density (AIMD) is proposed. The AIMD includes three general steps for computing the molecular properties of a target molecule: (i) generating small molecules (fragments) from molecules of interest (ii) storing some properties of the fragments in the database, (iii) using these properties to compute the molecular properties of the target molecule. The studied molecular properties include: kinetic energy, exchange energy, Coulomb energy, and potential energy. The atomic properties are obtained using two different ways: using the data stored in the database and by using the direct method. The MAPE of the molecular kinetic energy calculated using the sum over atomic kinetic energies stored in the database $T^{sum}_{database}$ are about 0.121%, 0.040%, and 0.019% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively. Whereas, The MAPE of direct method T^{sum}_{direct} are about 0.107%, 0.036%, and 0.015% for $\mathbf{F^1}$, $\mathbf{F^2}$, and $\mathbf{F^3}$, respectively. For the exchange energy contains J_{aa} term the MAPE of $K^{sum}_{HF,database}$ are about 0.053%, 0.058%, and 0.035% for $\mathbf{F^1}$, $\mathbf{F^2}$, $\mathbf{F^3}$, respectively, and the MAPE of $K^{sum}_{HF,direct}$ are about 0.067%, 0.058%, and 0.032% for $\mathbf{F^1}$, $\mathbf{F^2}$, $\mathbf{F^3}$, respectively. The MAPE of the calculated potential energies using the atomic electron densities stored in the database $V_{ne}[\rho_{database}]$ are about 0.046%, 0.087%, and 0.066% for $\mathbf{F^1}$, $\mathbf{F^2}$, $\mathbf{F^3}$, respectively, and the MAPE of $V_{ne}[\rho_{direct}]$ are about 0.034%, 0.085%, and 0.051% for $\mathbf{F^1}$, $\mathbf{F^2}$, $\mathbf{F^3}$, respectively. The MAPE of the calculated Coulomb energies using the atomic electron densities stored in the database $J[\rho_{database}]$ are about 0.111%, 0.242%, and 0.182% for $\mathbf{F^1}$, $\mathbf{F^2}$, $\mathbf{F^3}$, respectively, and the MAPE of $J[\rho_{direct}]$ are about 0.083%, 0.242%, and 0.142% for $\mathbf{F^1}$, $\mathbf{F^2}$, $\mathbf{F^3}$, respectively. Also, the calculated total molecular energy using AIMD is compared with with those calculated numerically for the target molecule. The MAPEs are relatively small. The MAPE of $E_{database}$ are about 0.165%, 0.017%, and 0.008% for $\mathbf{F^1}$, $\mathbf{F^2}$, $\mathbf{F^3}$, respectively, and the MAPE of E_{direct} are about 0.164%, 0.022%,

and 0.006% for $\mathbf{F^1}$, $\mathbf{F^2}$, $\mathbf{F^3}$, respectively.

8.1.2 Future Work

Further work for the development of AIMD is required. Some future goals include:

- Optimizing the core sizes of Awad weight based on some molecular properties such as: bond order, atomic charges, dipole moments, etc.
- Storing the core sizes of Awad weight in the database according to the environment of the target atom in the molecule of interest (i.e., the neighbour atoms, basis set, level of theory, etc).
- Developing a new numerical method for computing the molecular properties in different regions (e.g., the number of electrons in the core region for a specific atom in a molecule).
- Improving the numerical integration techniques for computing the Coulomb and potential energies to increase the accuracy of the results and reduce the computer time.
- Developing a hybrid QM/AIMD method which can describe large molecules.
- Storing the Coulomb and potential energy densities of the target atom in the core region in the database that can be used as pseudopotential to reduce the computer time.
- Fitting the AIM densities.
- Considering cut-off distances for long range interactions to reduce the computer time.

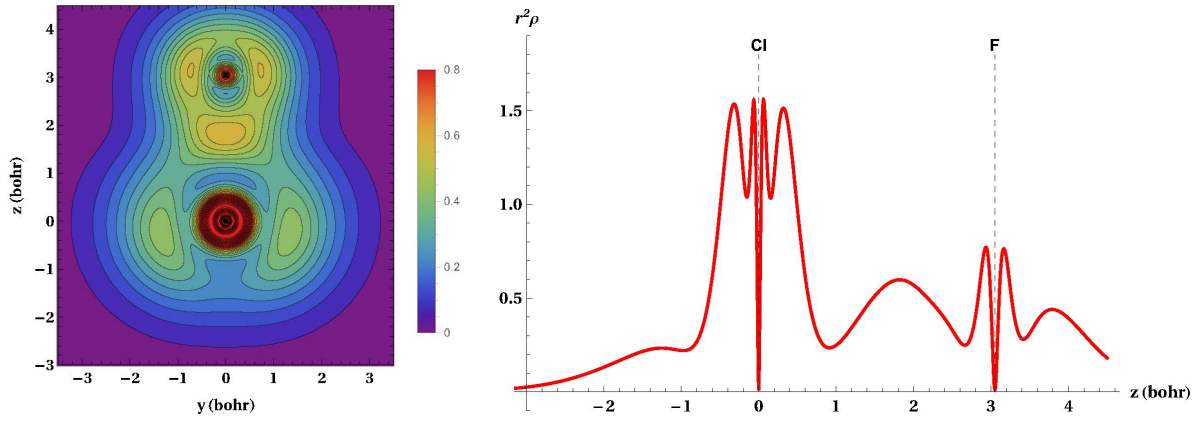
- Improving the descriptions for AIM.
- Improving the visualization tools for molecular properties of AIMD.
- Developing a density functional version of AIMD approach.
- Reducing the computer time by modifying the code and using parallel processing.

Appendix A

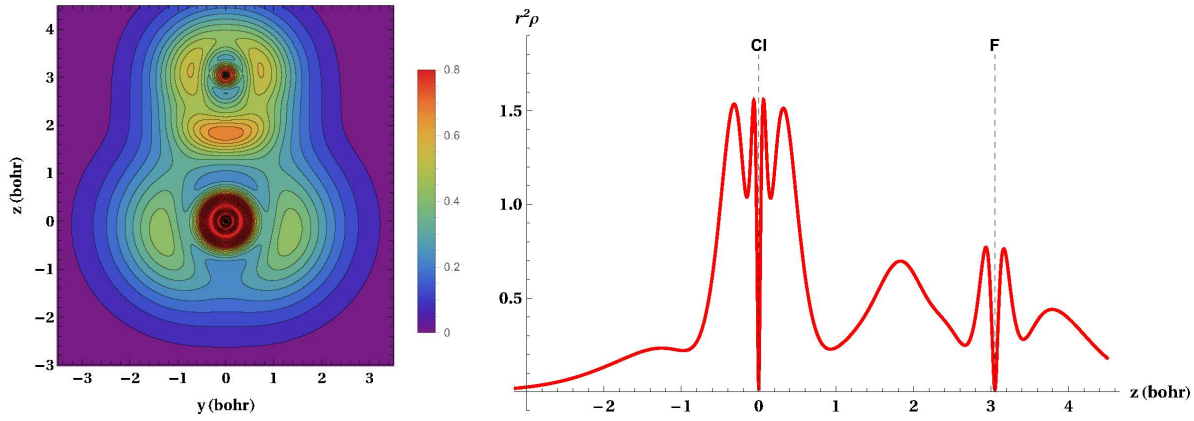
Figures and Charts

A.1 Radial Electron Densities

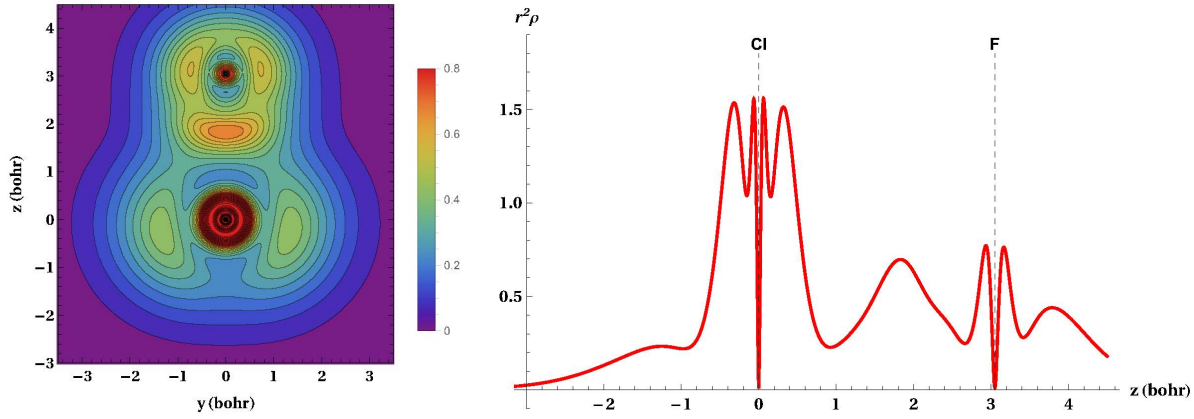
In this section, we give some examples of the molecular radial electron density (RDEN) using Awad weight at: r_{max}, r_{min}, and $r_{\langle r \rangle}$.



(a) r_{max}



(b) r_{min}



(c) $r_{\langle r \rangle}$

Figure A.1: Molecular radial electron density (RDEN) for FCl is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of Cl at 0.0 bohr and F at 3.0504 bohr along z axis.

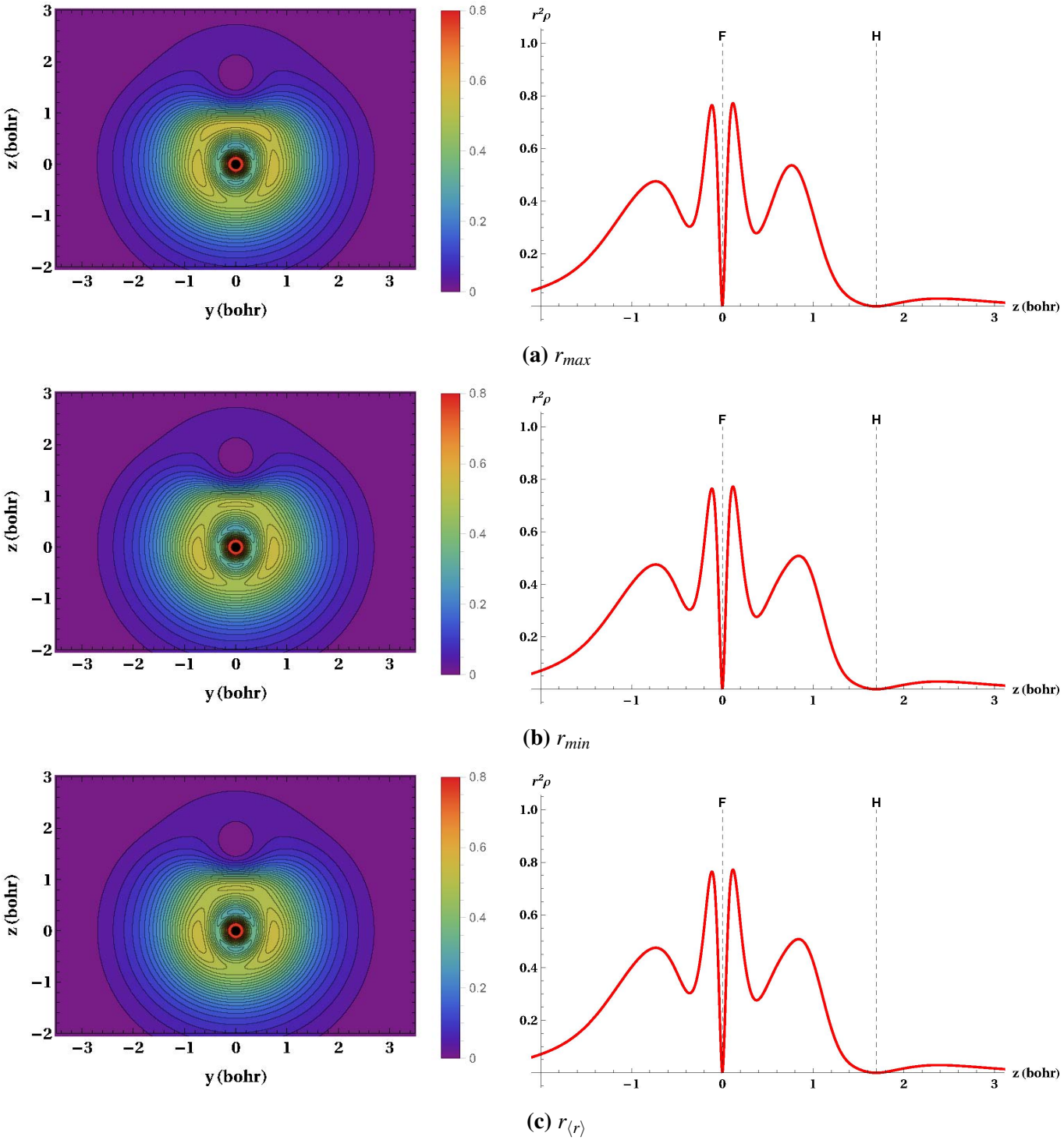
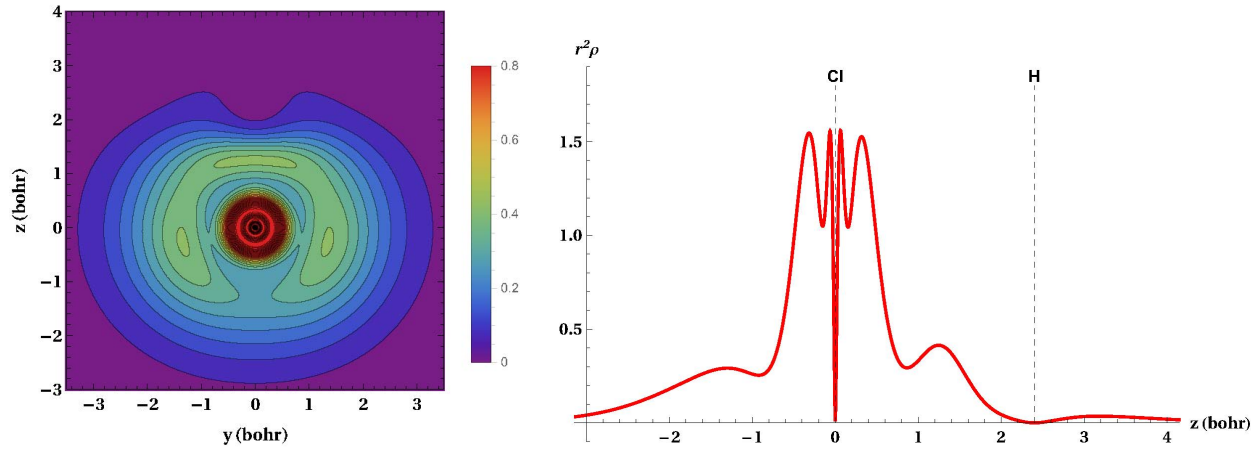
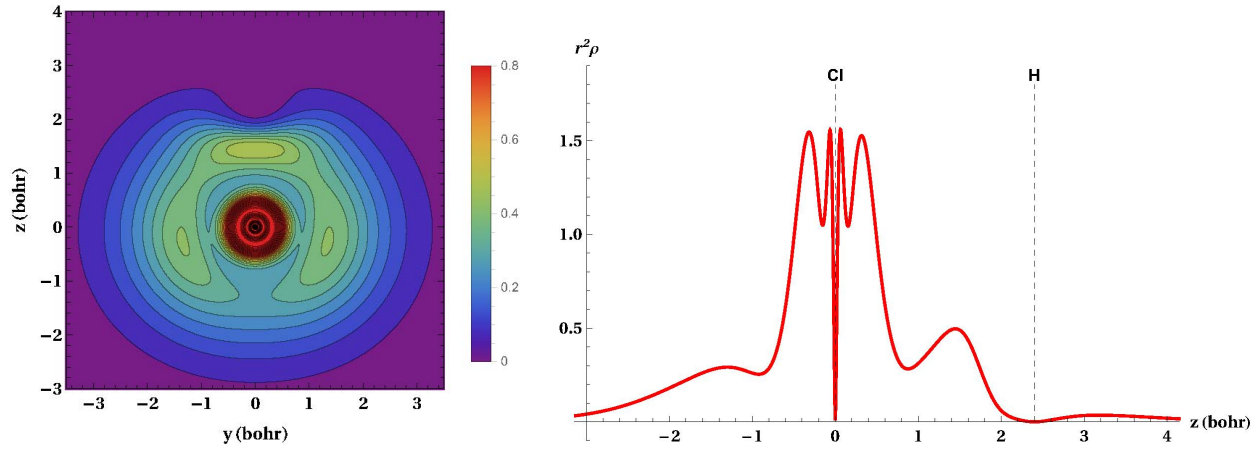


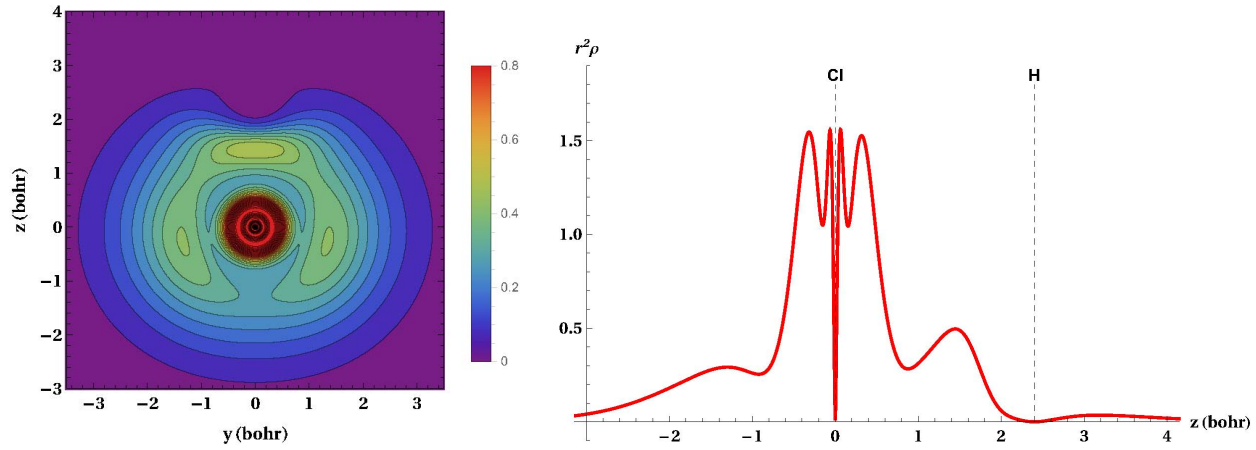
Figure A.2: Molecular radial electron density (RDEN) for HF is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of F at 0.0 bohr and H at 1.6960 bohr along z axis.



(a) r_{max}

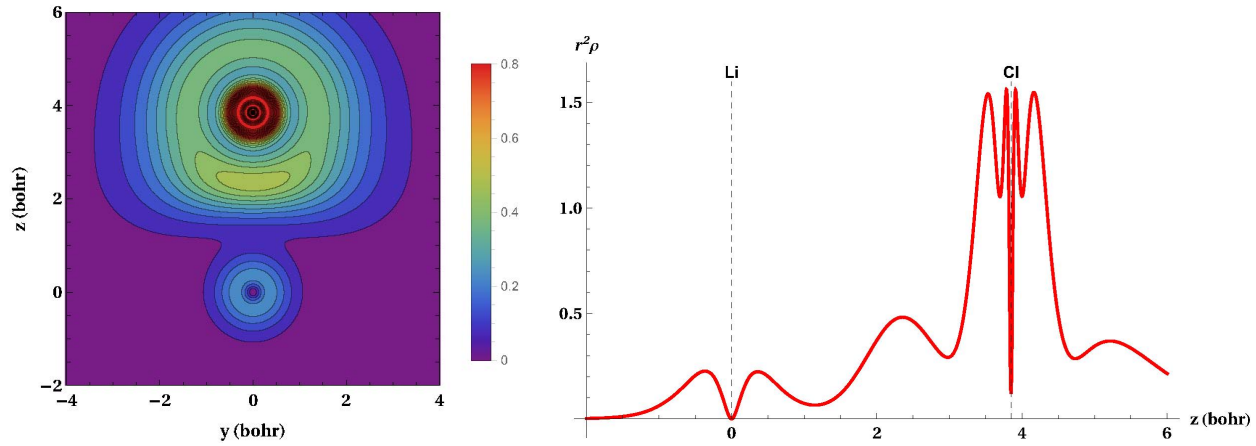


(b) r_{min}

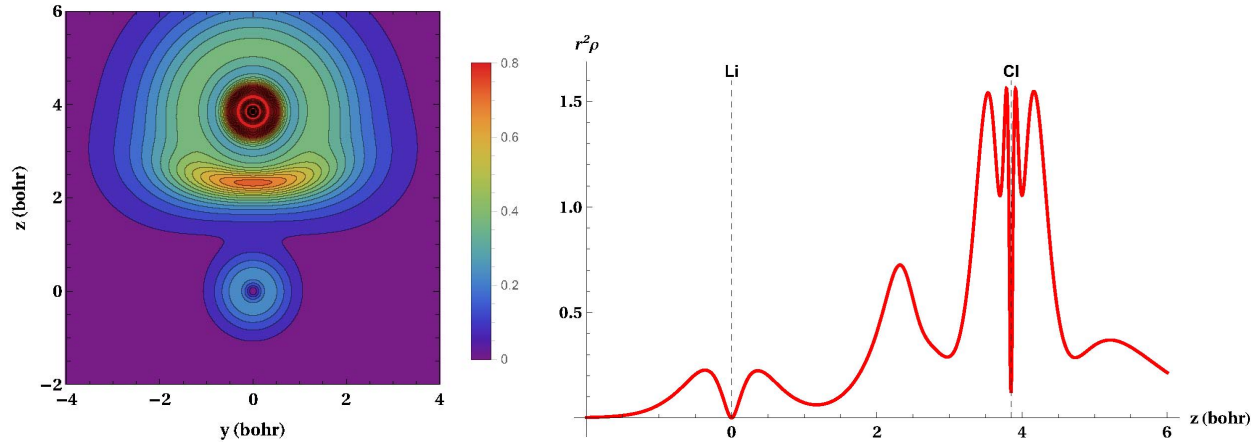


(c) $r_{\langle r \rangle}$

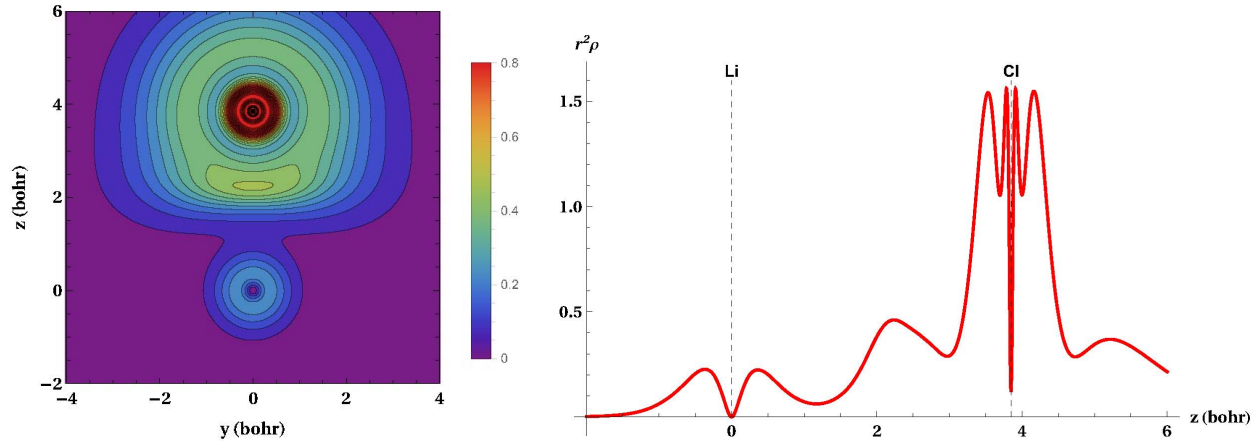
Figure A.3: Molecular radial electron density (RDEN) for HCl is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of Cl at 0.0 bohr and H at 2.4000 bohr along z axis.



(a) r_{max}

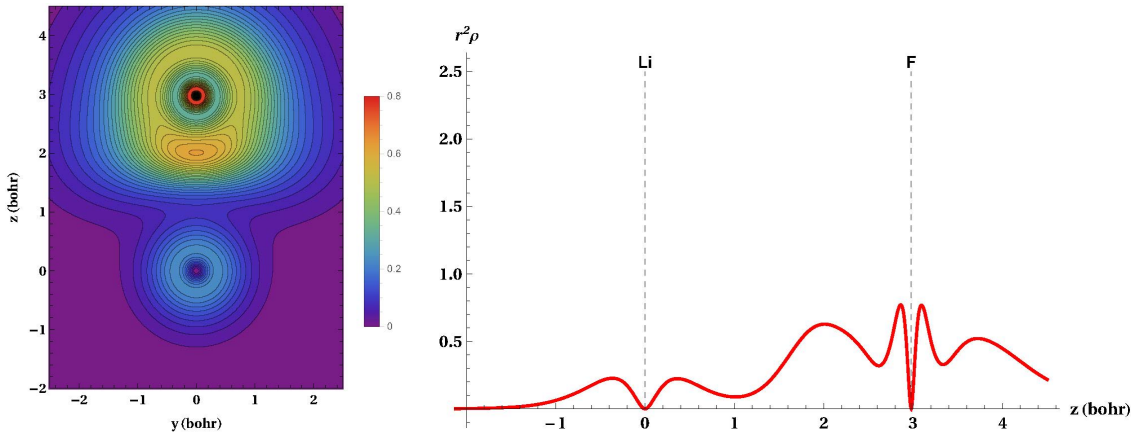


(b) r_{min}

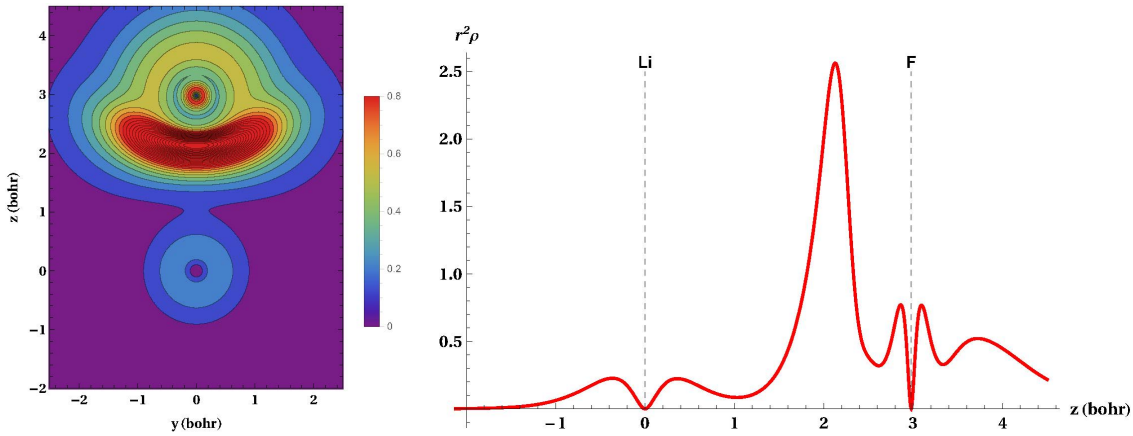


(c) $r_{\langle r \rangle}$

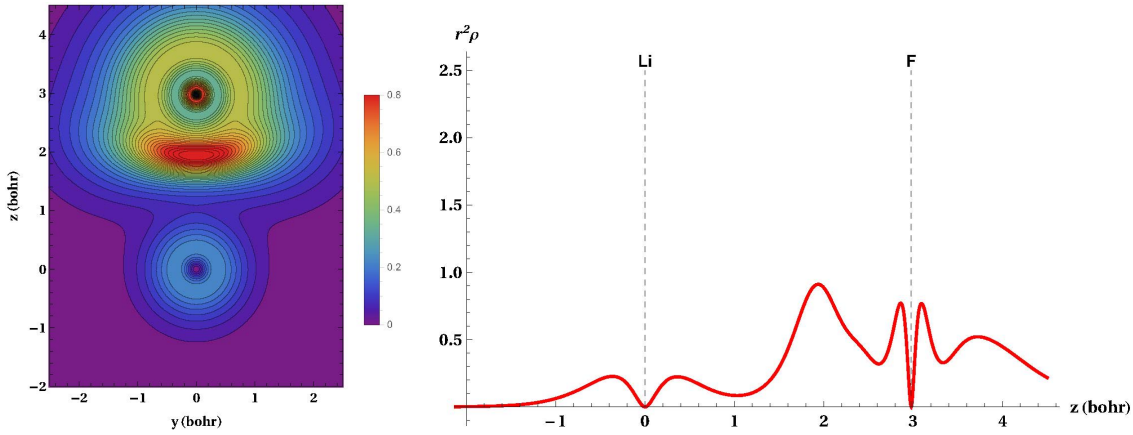
Figure A.4: Molecular radial electron density (RDEN) for LiCl is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of Li at 0.0 bohr and Cl at 3.8492 bohr along z axis.



(a) r_{max}



(b) r_{min}

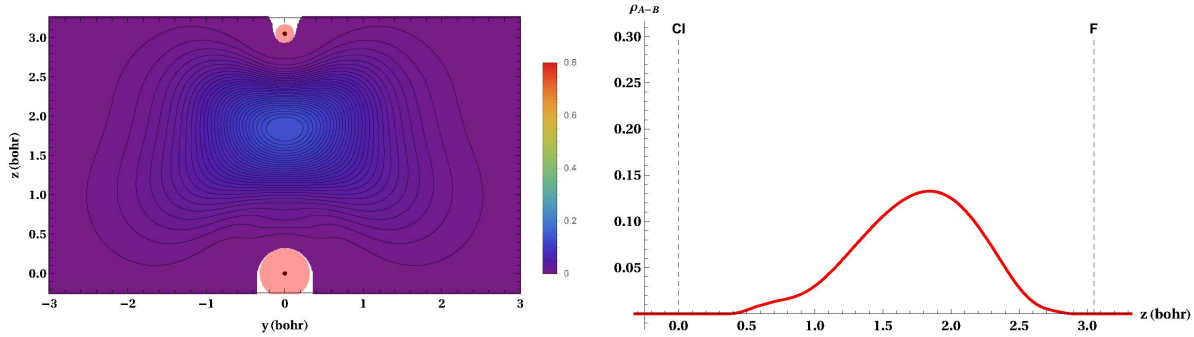


(c) $r_{\langle r \rangle}$

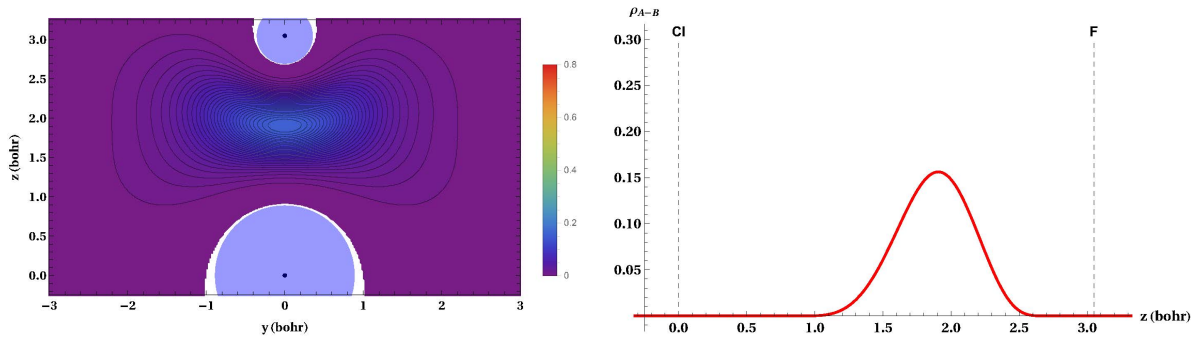
Figure A.5: Molecular radial electron density (RDEN) for LiF is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of Li at 0.0 bohr and F at 2.9777 bohr along z axis.

A.2 Bond Electron Densities

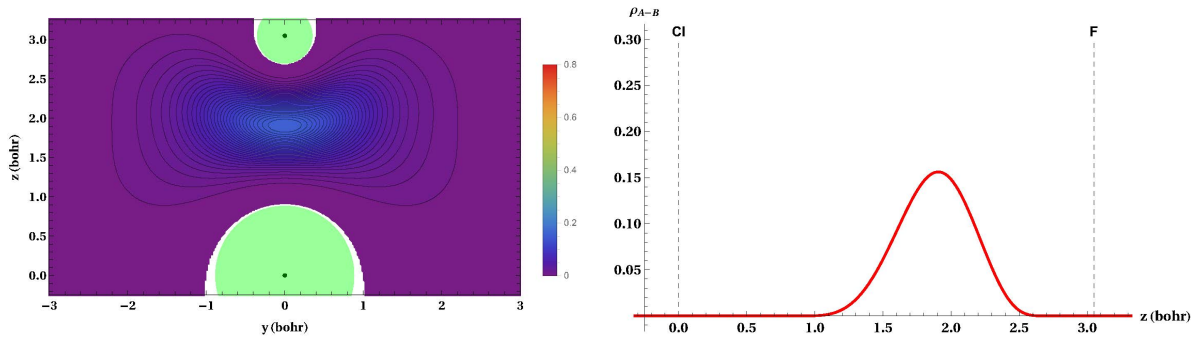
In this section, we give some examples of the bond electron density (BDEN) using Awad weight at: r_{max}, r_{min}, and $r_{\langle r \rangle}$.



(a) r_{max}

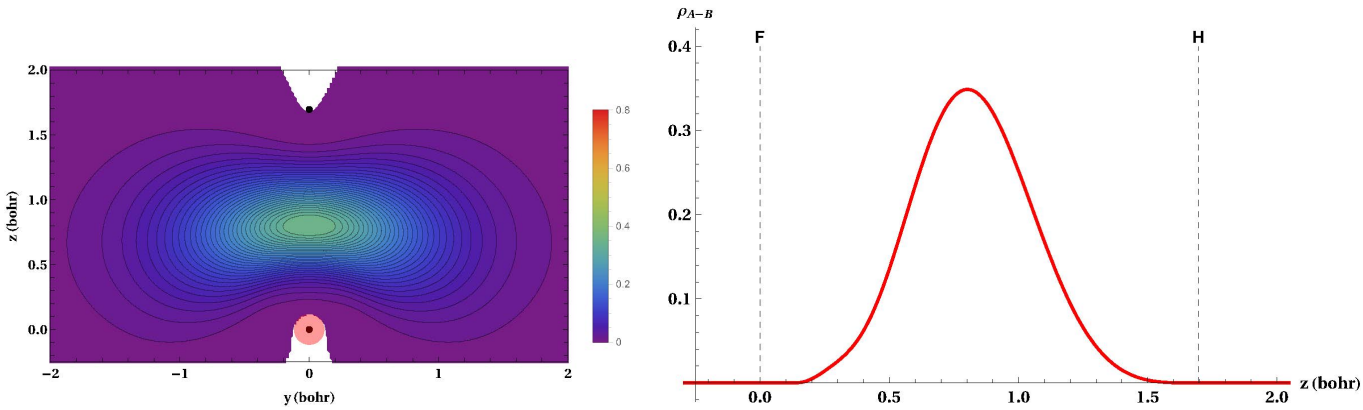


(b) r_{min}

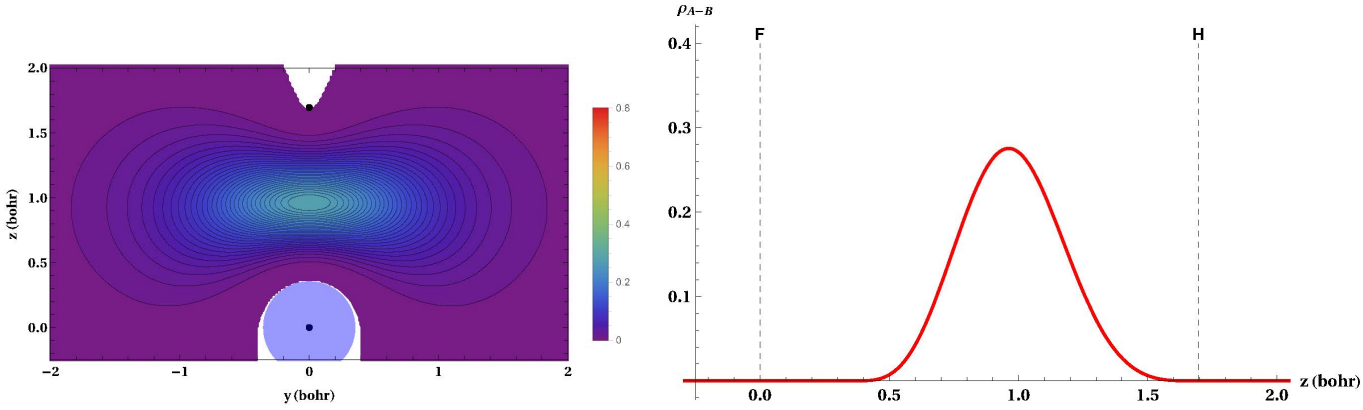


(c) $r_{\langle r \rangle}$

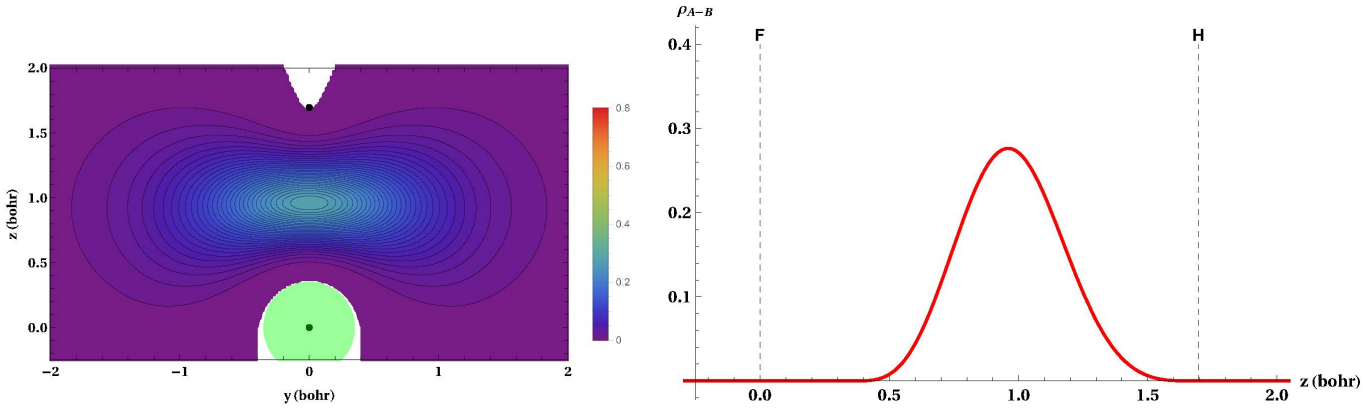
Figure A.6: Bond electron density (BDEN) for FCl is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of Cl at 0.0 bohr and F at 3.0504 bohr along z axis. The red, purple, and green disks are the core regions of r_{max}, r_{min}, and $r_{\langle r \rangle}$, respectively.



(a) r_{max}

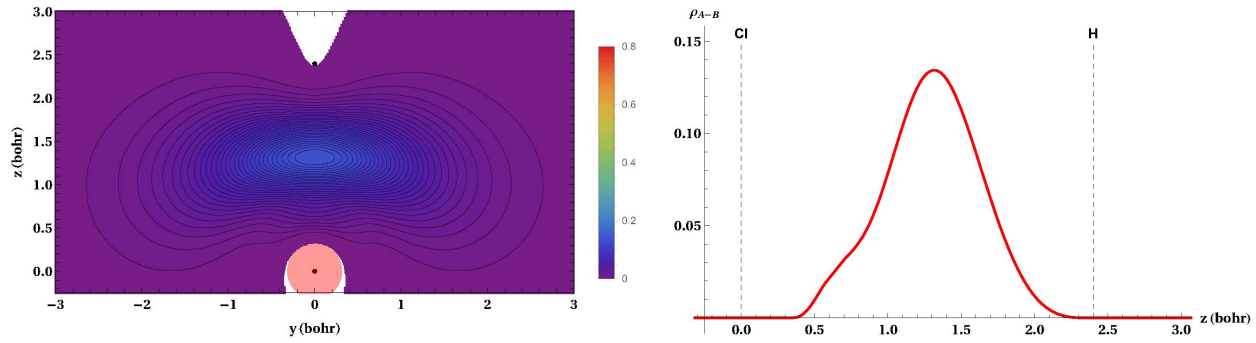


(b) r_{min}

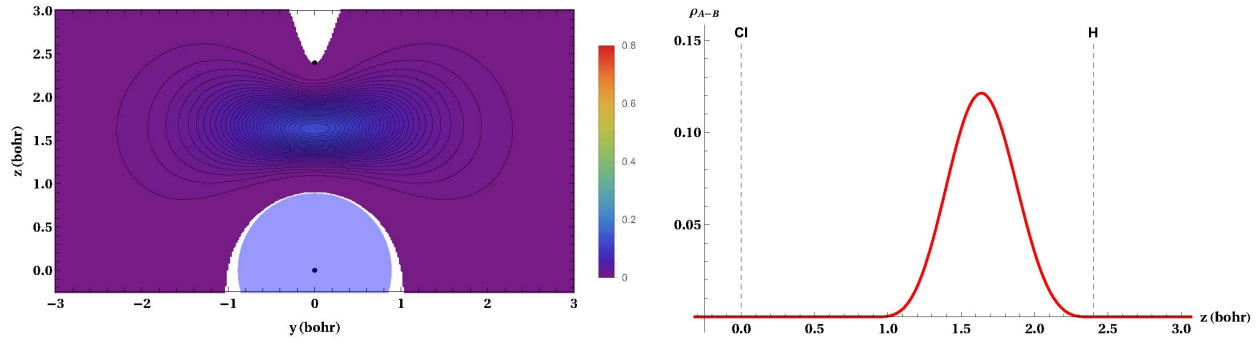


(c) $r_{\langle r \rangle}$

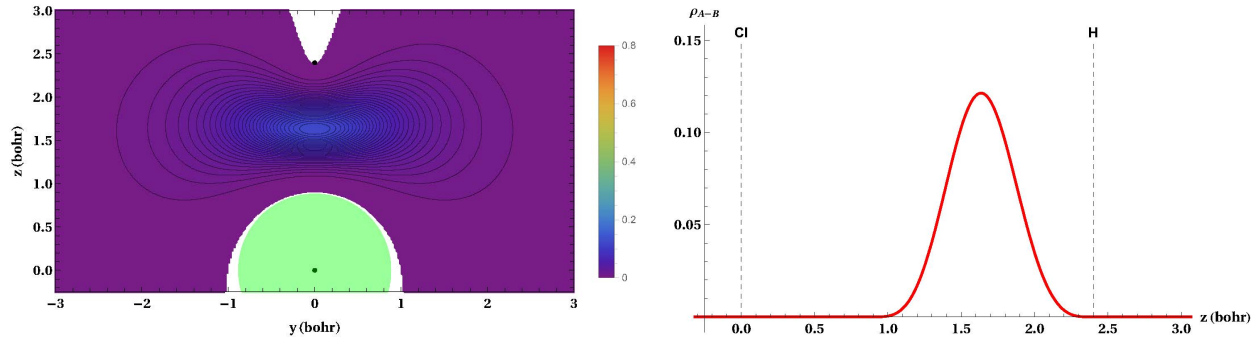
Figure A.7: Bond electron density (BDEN) for HF is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of F at 0.0 bohr and H at 1.6960 bohr along z axis. The red, purple, and green disks are the core regions of r_{max}, r_{min}, and $r_{\langle r \rangle}$, respectively.



(a) r_{max}



(b) r_{min}



(c) $r_{\langle r \rangle}$

Figure A.8: Bond electron density (BDEN) for HCl is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of Cl at 0.0 bohr and H at 2.4000 bohr along z axis. The red, purple, and green disks are the core regions of r_{max}, r_{min}, and $r_{\langle r \rangle}$, respectively.

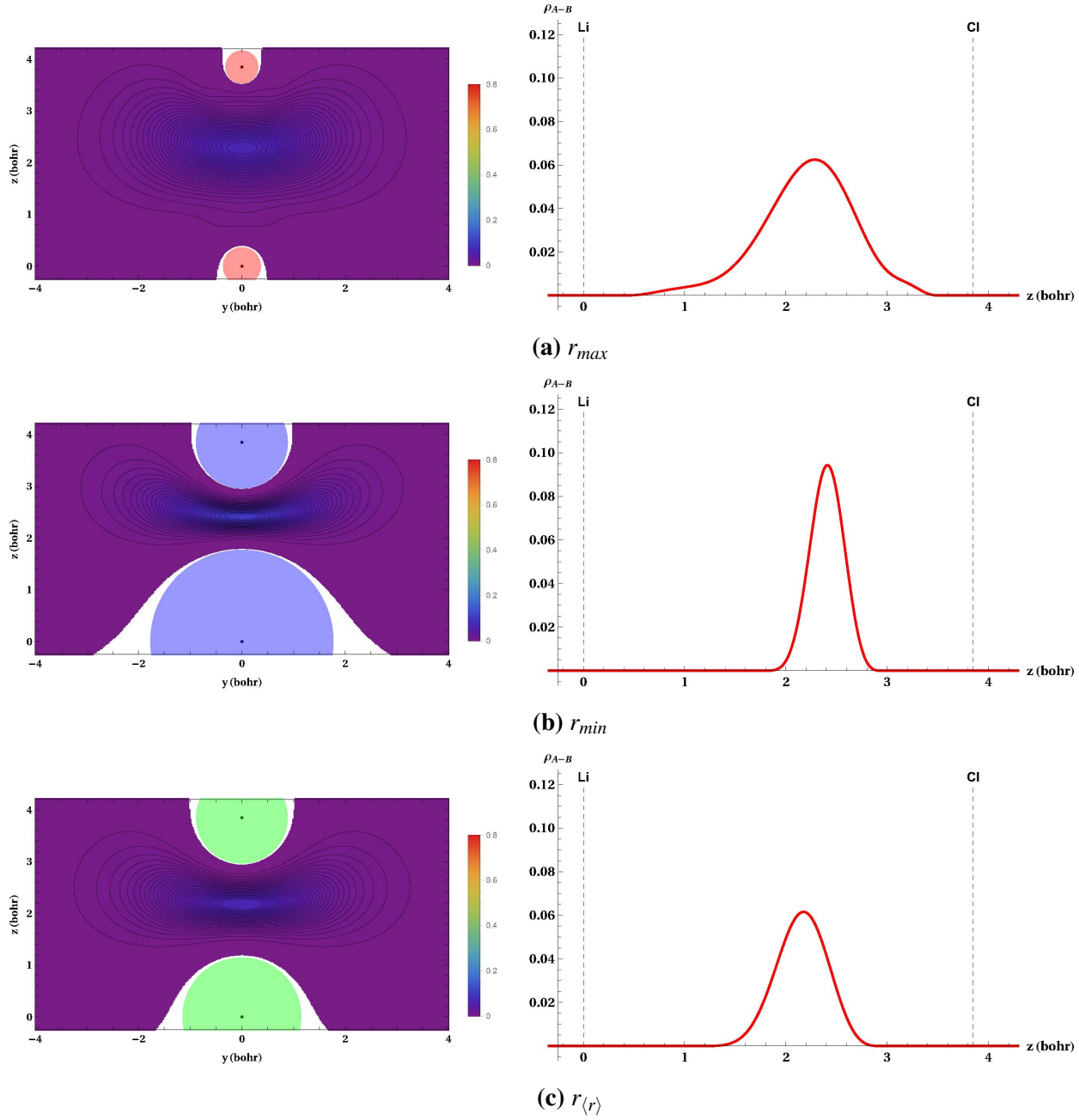
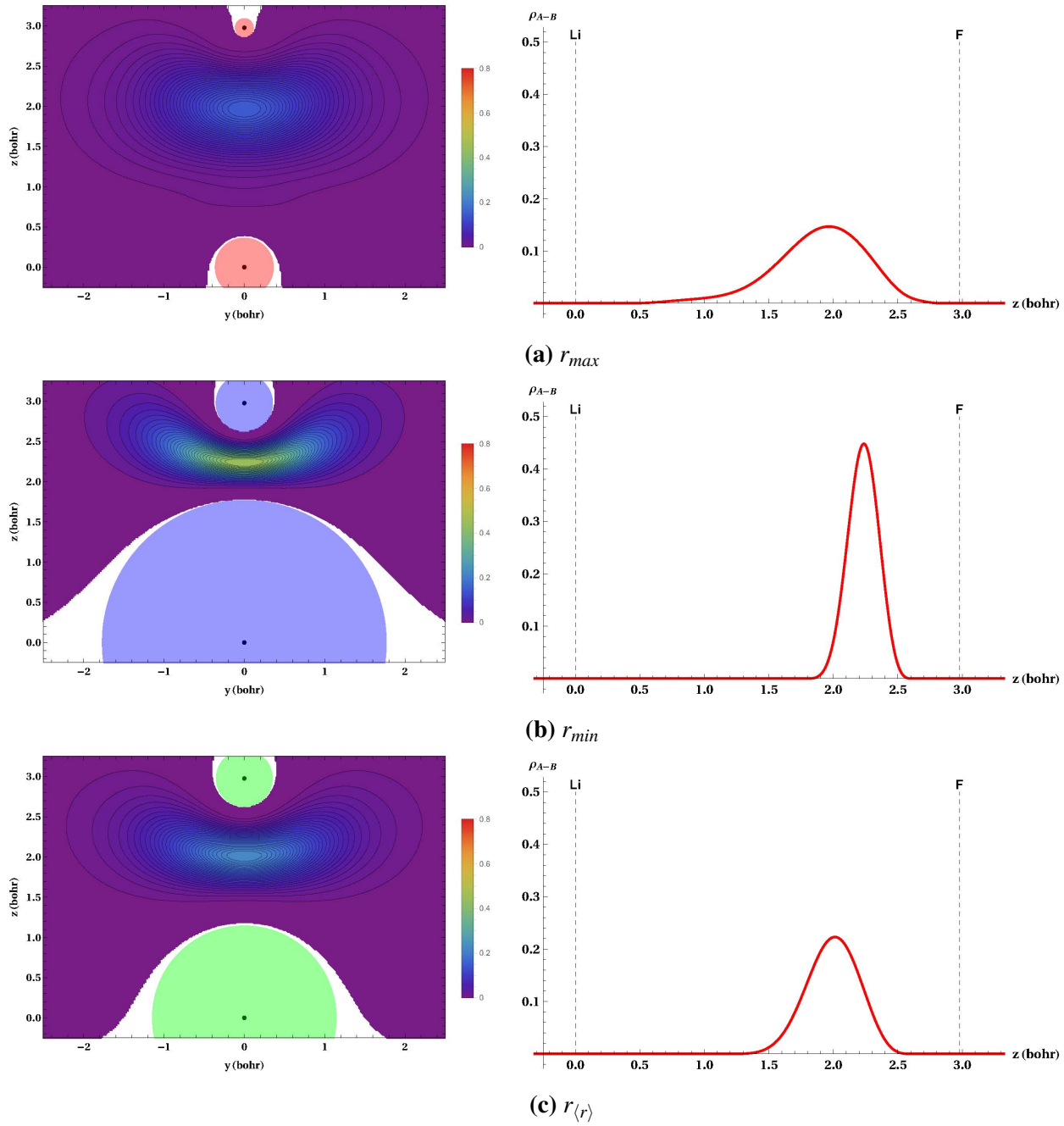


Figure A.9: Bond electron density (BDEN) for LiCl is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r\rangle}$. The atomic coordinates of Li at 0.0 bohr and Cl at 3.8492 bohr along z axis. The red, purple, and green disks are the core regions of r_{max}, r_{min}, and $r_{\langle r\rangle}$, respectively.



(a) r_{max}

(b) r_{min}

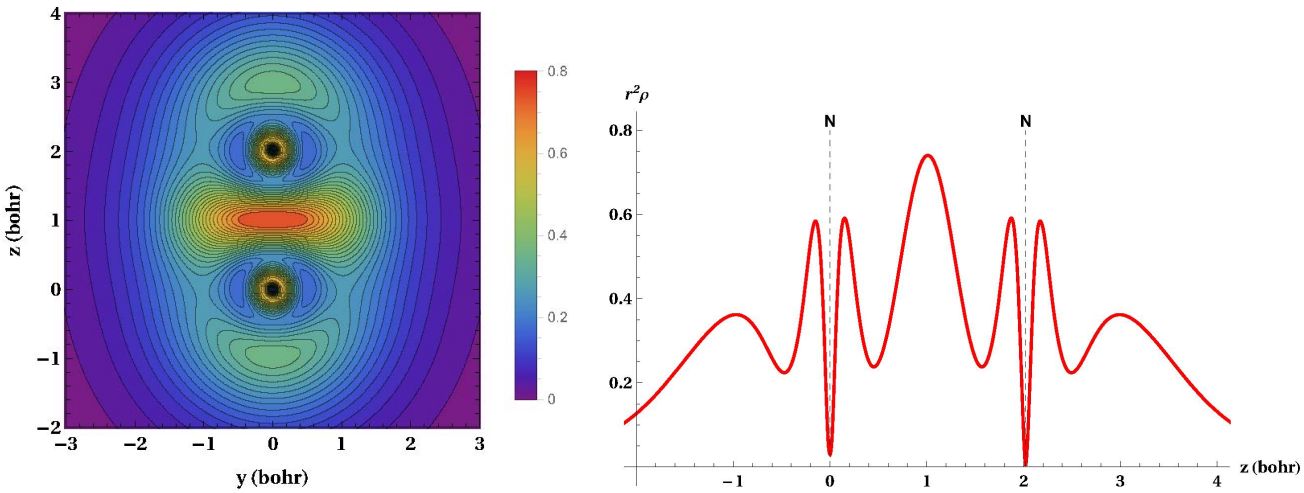
(c) $r_{\langle r \rangle}$

Figure A.10: Bond electron density (BDEN) for LiF is obtained using the Awad weight at: (a) r_{max}, (b) r_{min}, and (c) $r_{\langle r \rangle}$. The atomic coordinates of Li at 0.0 bohr and F at 2.9777 bohr along z axis. The red, purple, and green disks are the core regions of r_{max}, r_{min}, and $r_{\langle r \rangle}$, respectively.

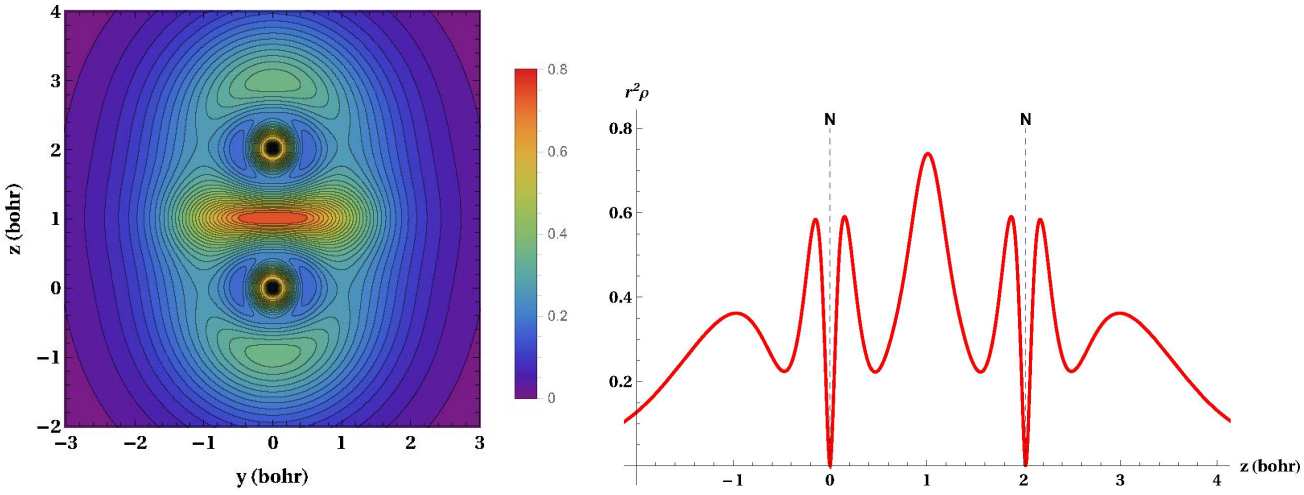
A.3 Comparing the Radial and Bond Electron Densities Between Awad and Becke Weights

In this section, we compare the molecular radial electron density (RDEN) and bond electron density (BDEN) between using Awad and Becke weights.

A.3.1 Comparing the Radial Electron Density Between Awad and Becke Weights

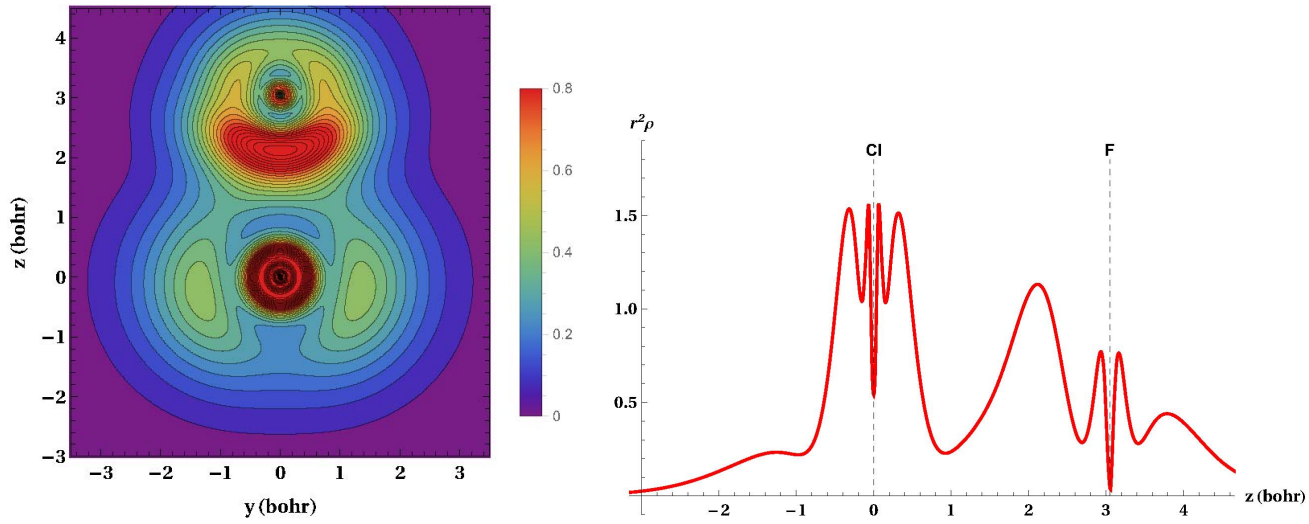


(a) The core size is defined as Becke weight.

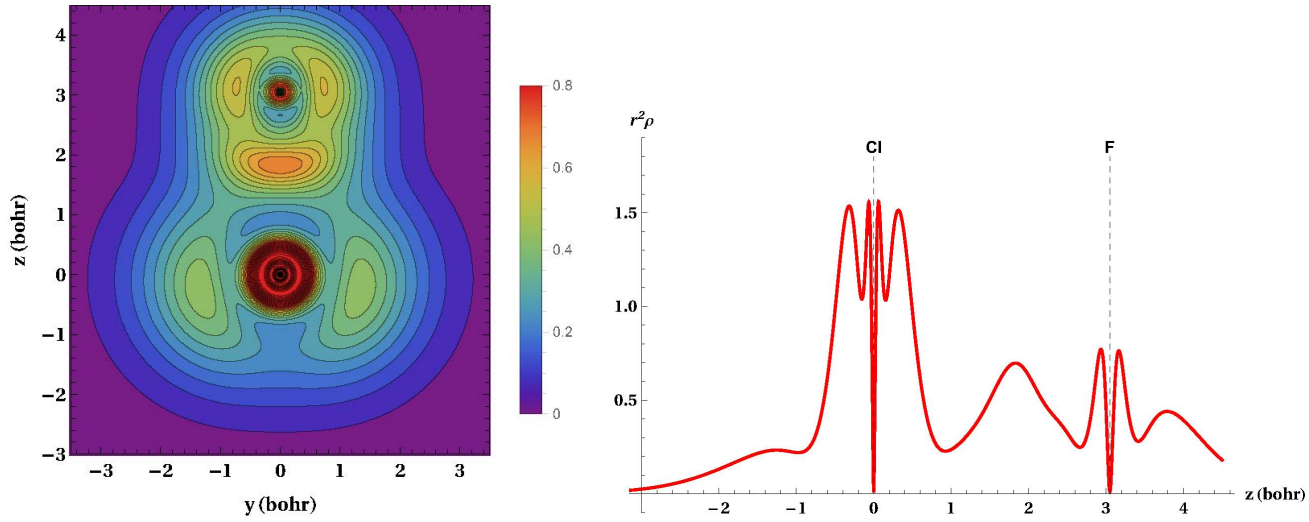


(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.11: Molecular radial electron density (RDEN) for N_2 is obtained using the Becke and Awad weights. The atomic coordinates of the two N atoms are 0.0 and 2.0229 bohr along z axis.

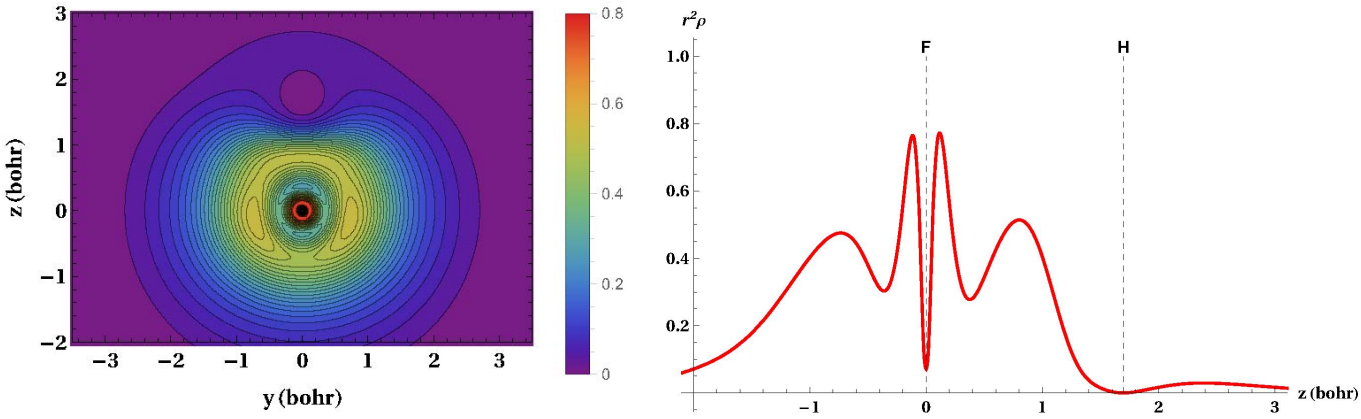


(a) The core size is defined as Becke weight.

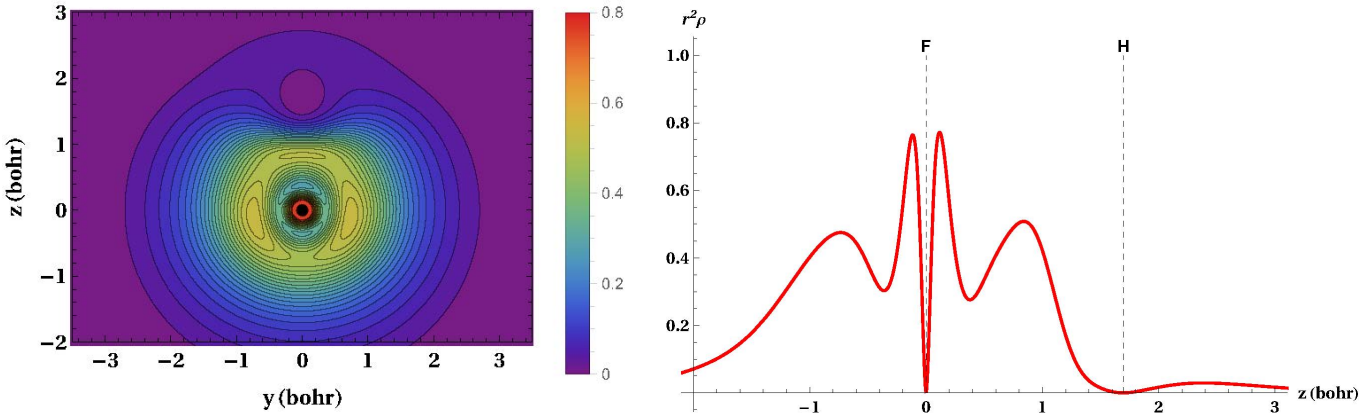


(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.12: Molecular radial electron density (RDEN) for FCl is obtained using the Becke and Awad weights. The atomic coordinates of Cl at 0.0 bohr and F at 3.0504 bohr along z axis.

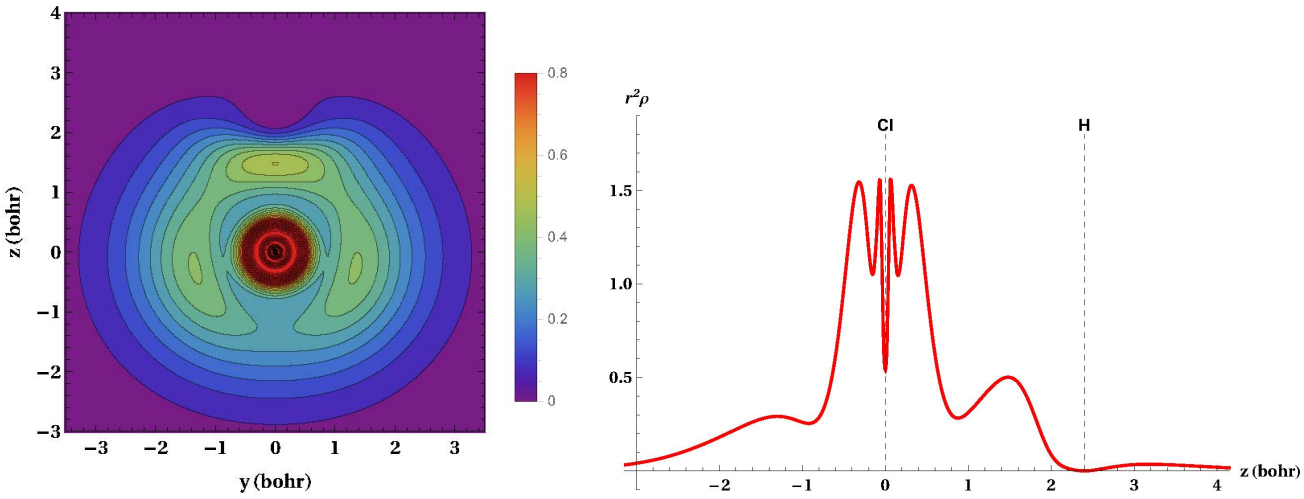


(a) The core size is defined as Becke weight.

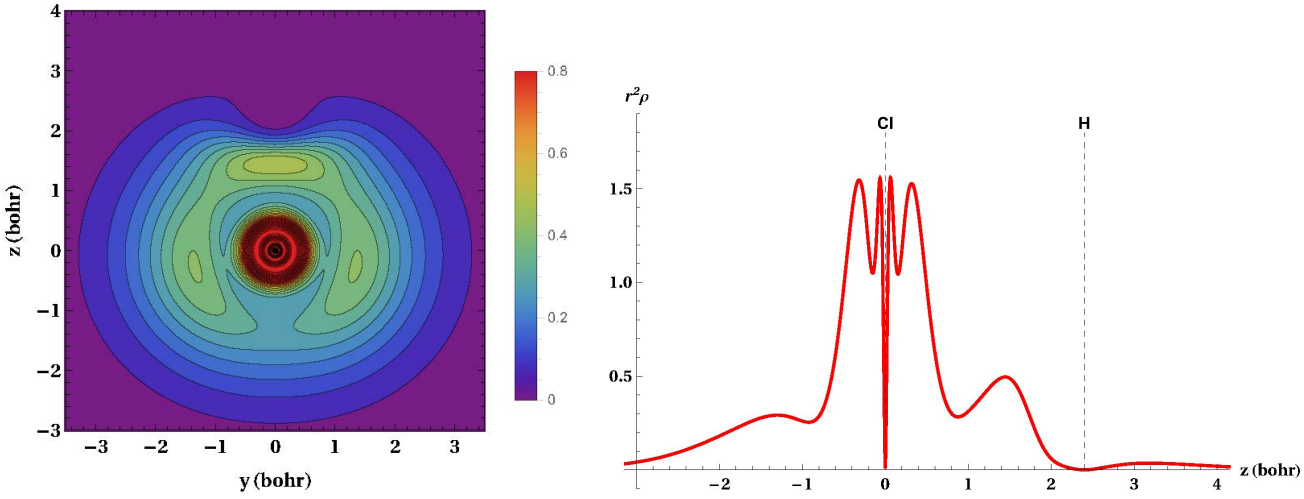


(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.13: Molecular radial electron density (RDEN) for HF is obtained using the Becke and Awad weights. The atomic coordinates of F at 0.0 bohr and H at 1.6960 bohr along z axis.

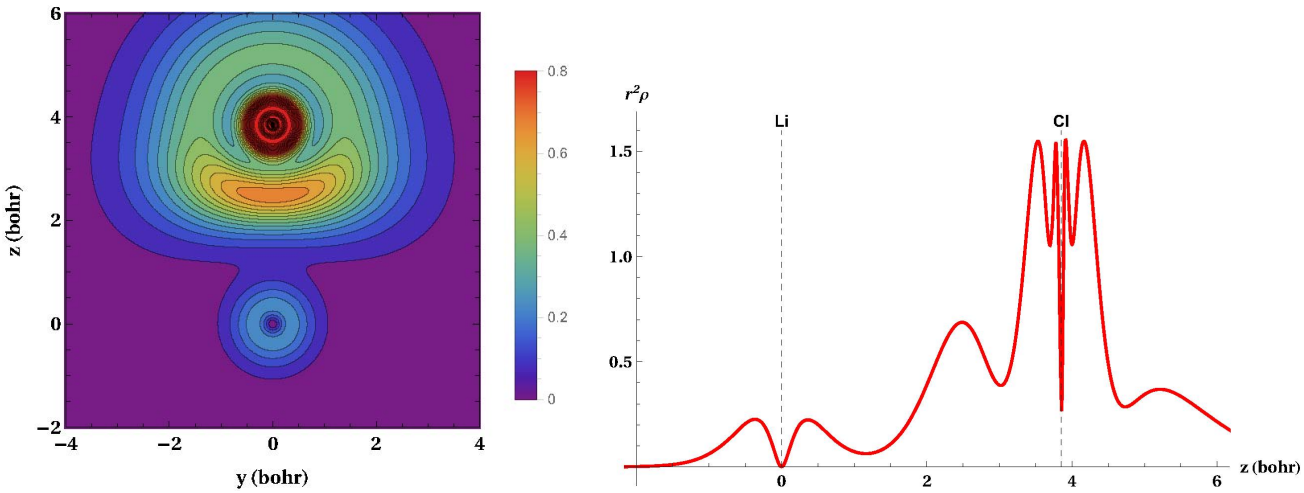


(a) The core size is defined as Becke weight.

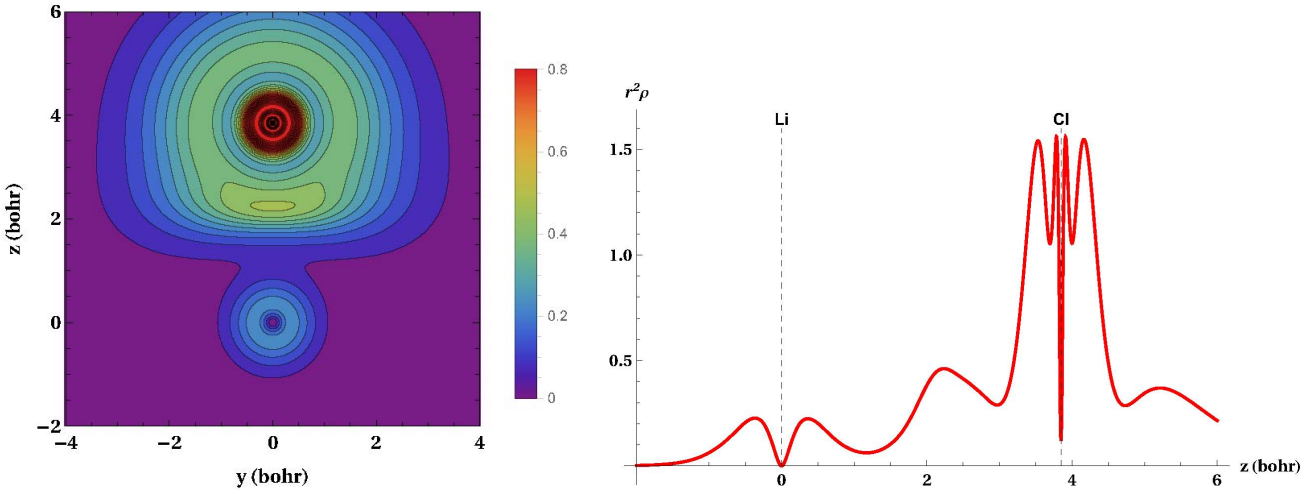


(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.14: Molecular radial electron density (RDEN) for HCl is obtained using the Becke and Awad weights. The atomic coordinates of Cl at 0.0 bohr and H at 2.4000 bohr along z axis.

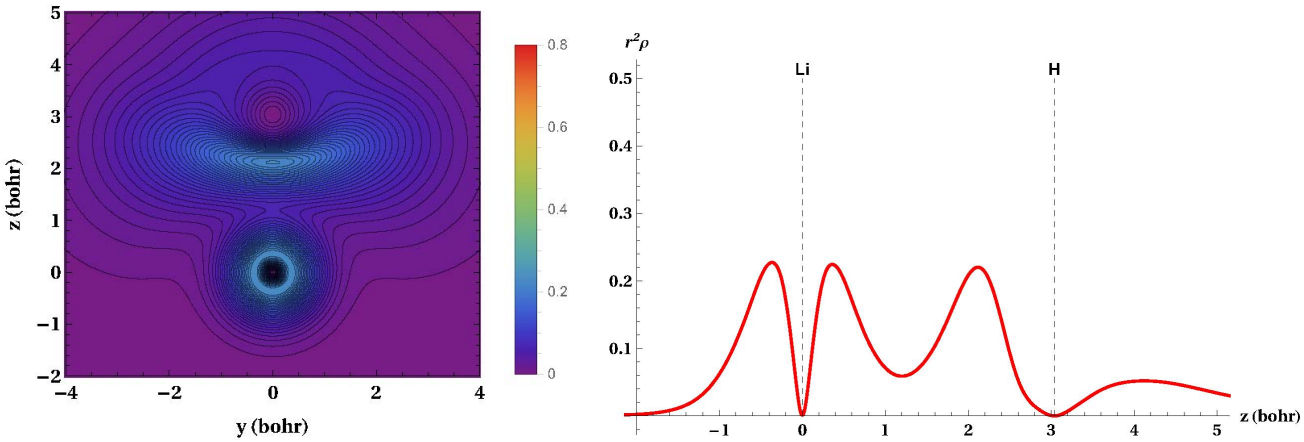


(a) The core size is defined as Becke weight.

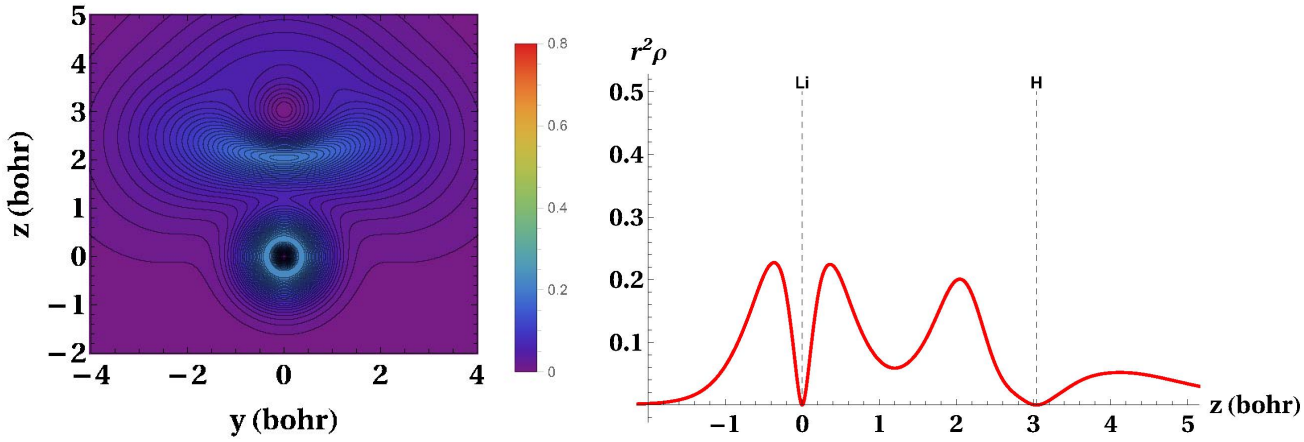


(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.15: Molecular radial electron density (RDEN) for LiCl is obtained using the Becke and Awad weights. The atomic coordinates of Li at 0.0 bohr and Cl at 3.8492 bohr along z axis.

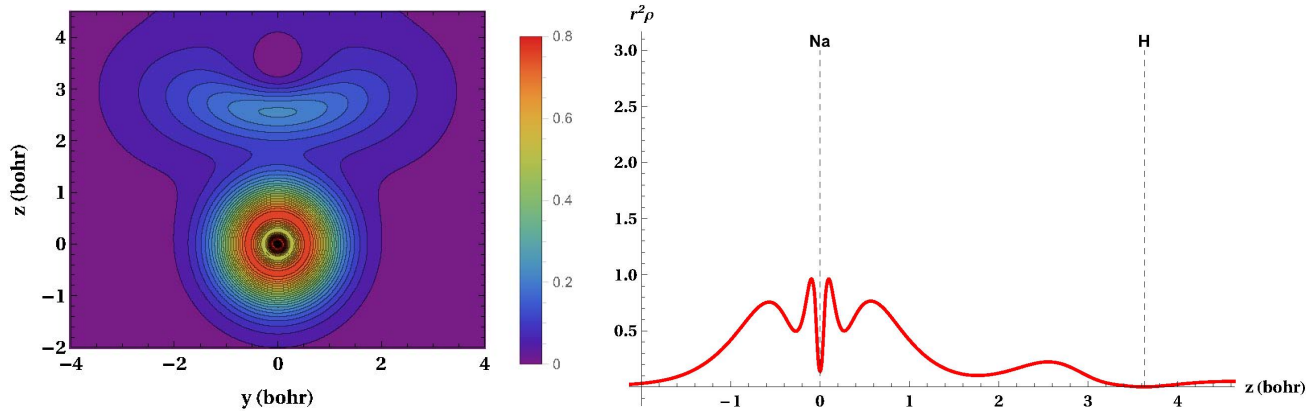


(a) The core size is defined as Becke weight.

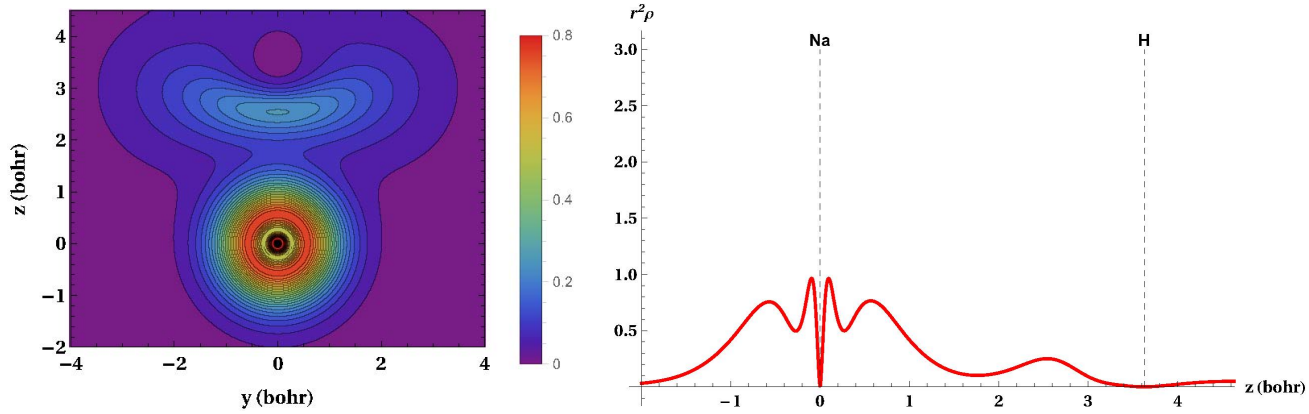


(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.16: Molecular radial electron density (RDEN) for LiH is obtained using the Becke and Awad weights. The atomic coordinates of Li at 0.0 bohr and H at 3.0390 bohr along z axis.



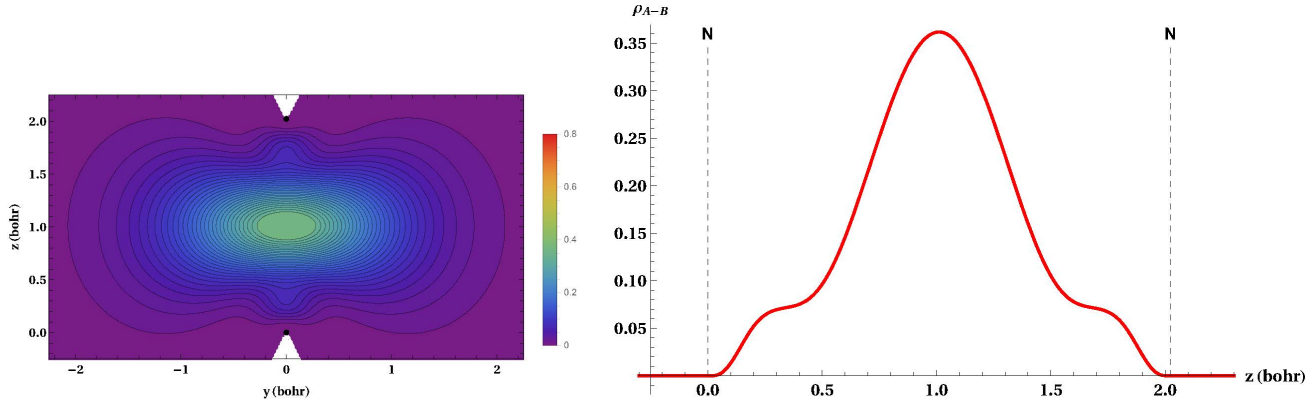
(a) The core size is defined as Becke weight.



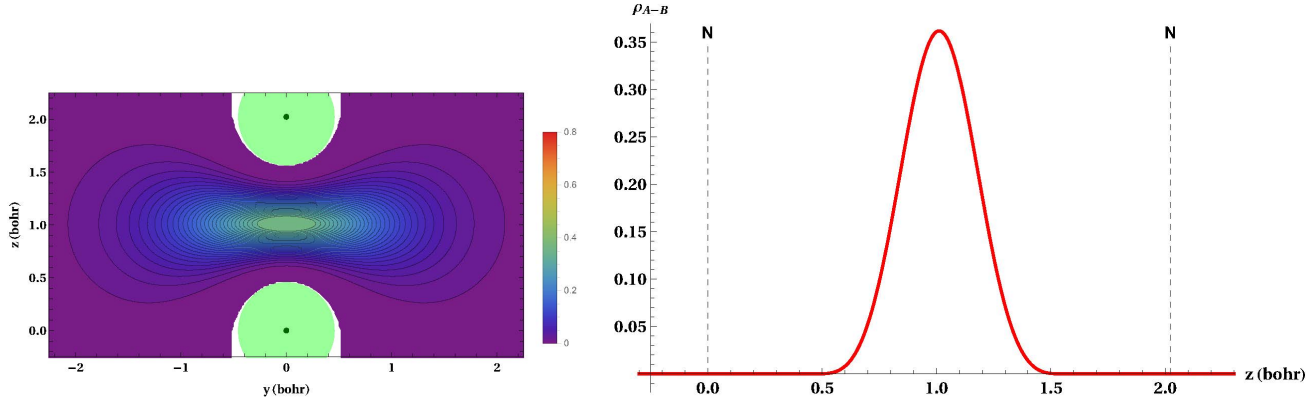
(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.17: Molecular radial electron density (RDEN) for NaH is obtained using the Becke and Awad weights. The atomic coordinates of Na at 0.0 bohr and H at 3.6261 bohr along z axis.

A.3.2 Comparing the Bond Electron Density Between Awad and Becke Weights

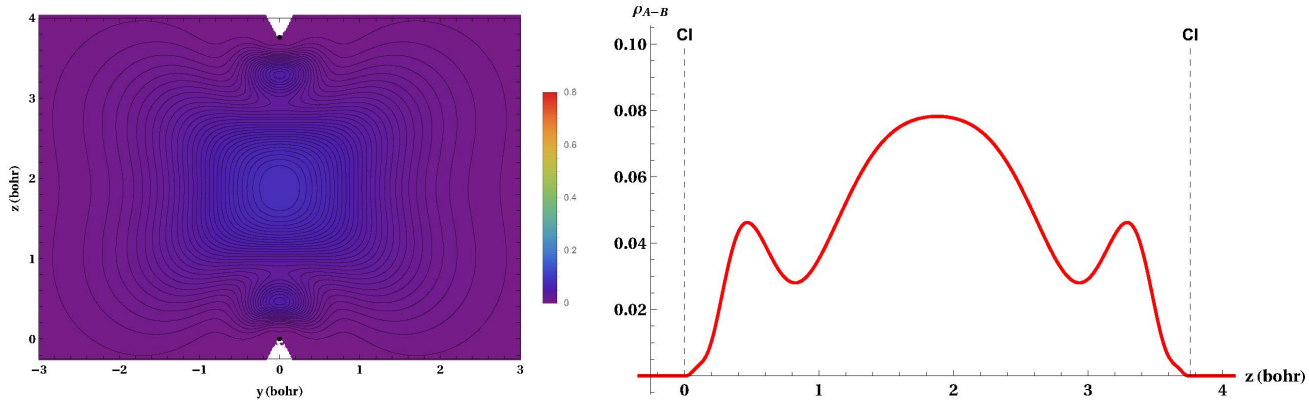


(a) The core size is defined as Becke weight.

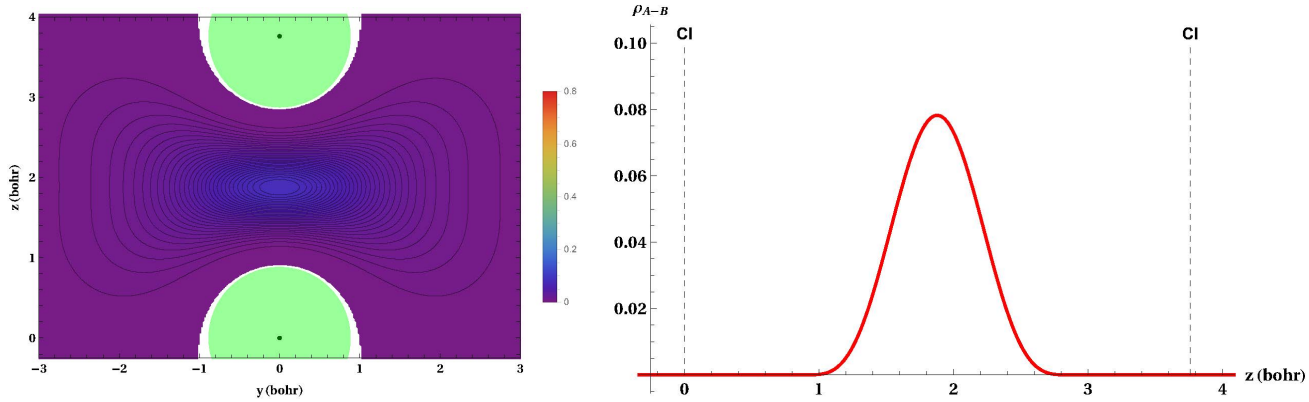


(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.18: Bond electron density (BDEN) for N_2 is obtained using the Becke and Awad weights. The atomic coordinates of the two N atoms are 0.0 and 2.0229 bohr along z axis.

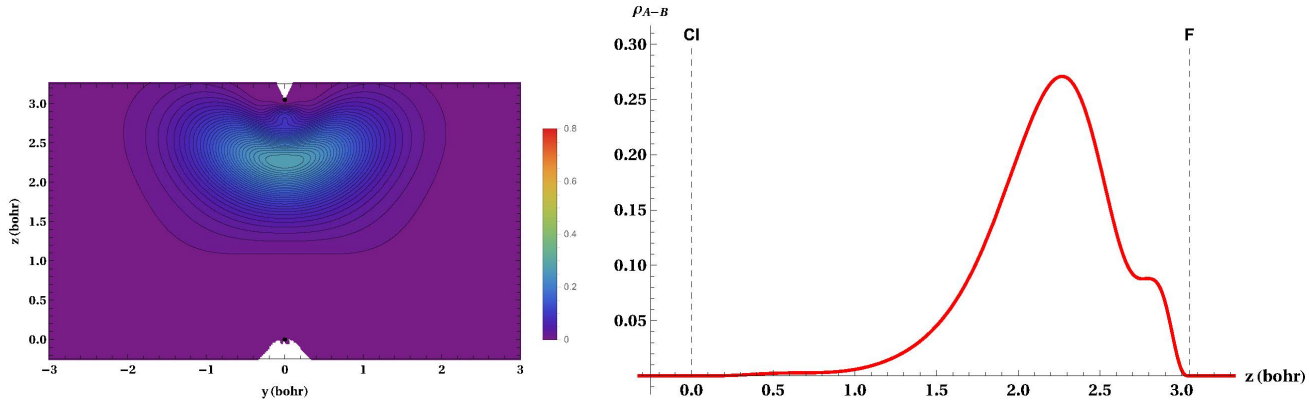


(a) The core size is defined as Becke weight.

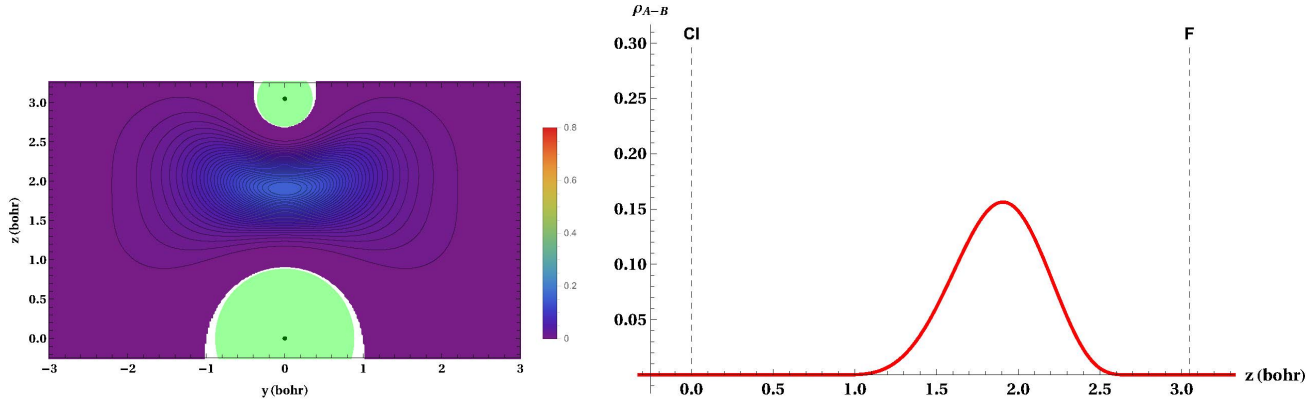


(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.19: Bond electron density (BDEN) for Cl_2 is obtained using the Becke and Awad weights. The atomic coordinates of the two Cl atoms are 0.0 and 3.7600 bohr along z axis.

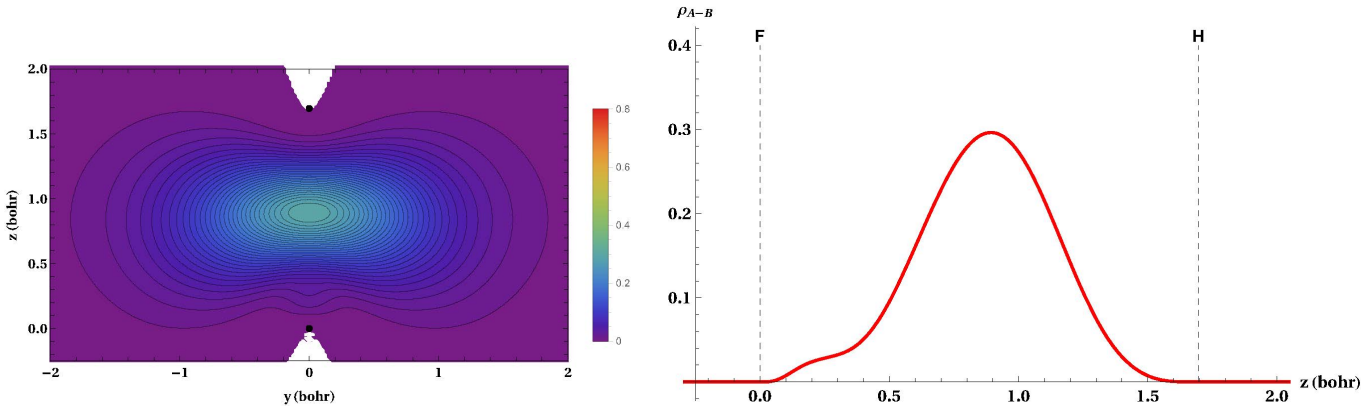


(a) The core size is defined as Becke weight.

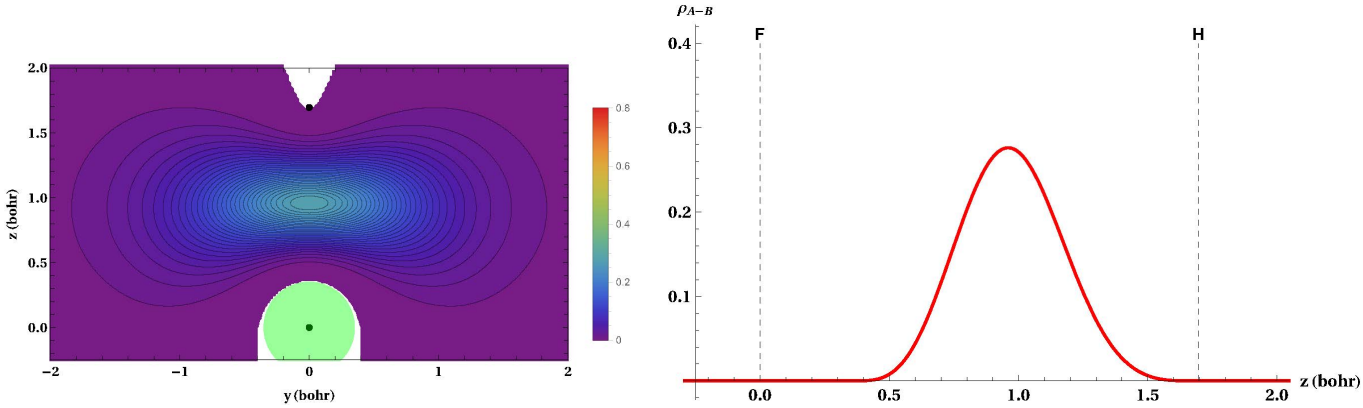


(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.20: Bond electron density (BDEN) for FCl is obtained using the Becke and Awad weights. The atomic coordinates of Cl at 0.0 bohr and F at 3.0504 bohr along z axis.

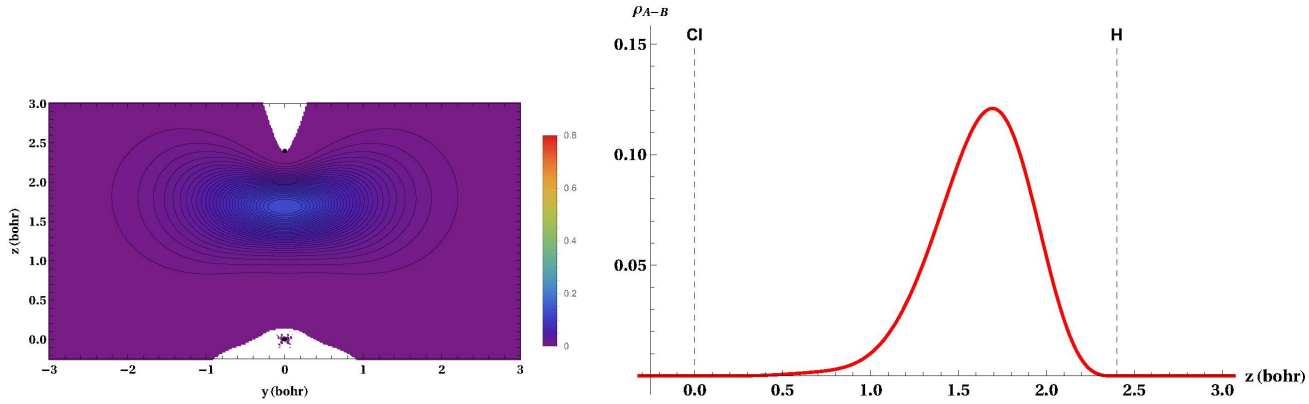


(a) The core size is defined as Becke weight.

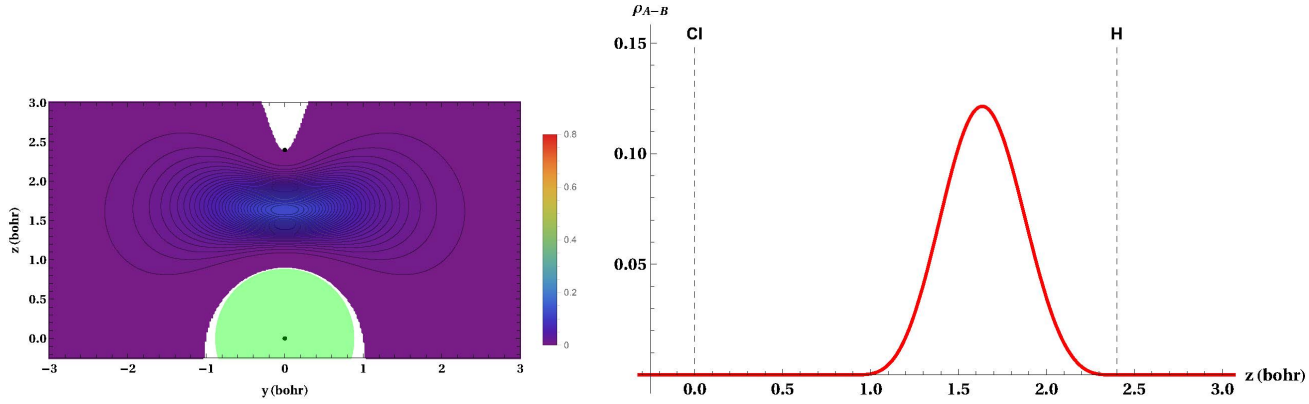


(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.21: Bond electron density (BDEN) for HF is obtained using the Becke and Awad weights. The atomic coordinates of F at 0.0 bohr and H at 1.6960 bohr along z axis.

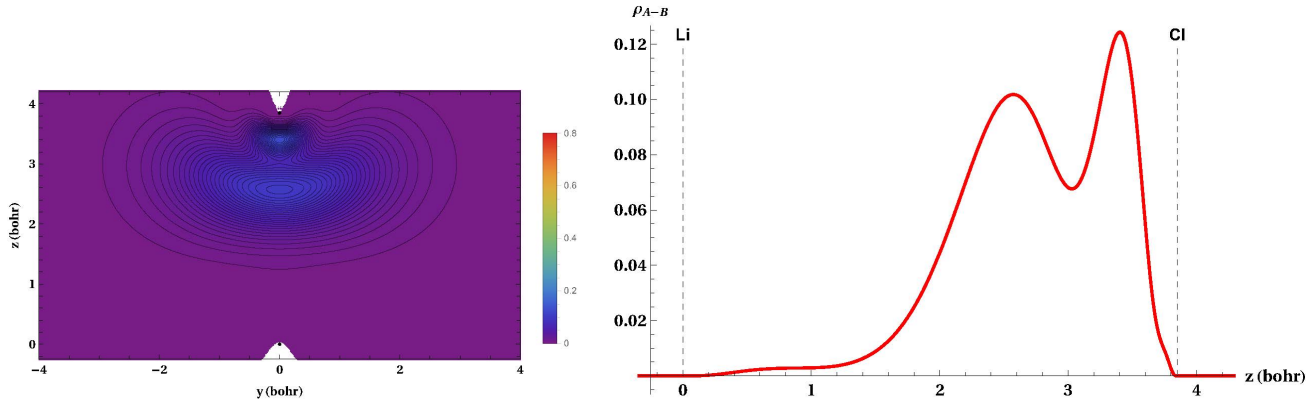


(a) The core size is defined as Becke weight.

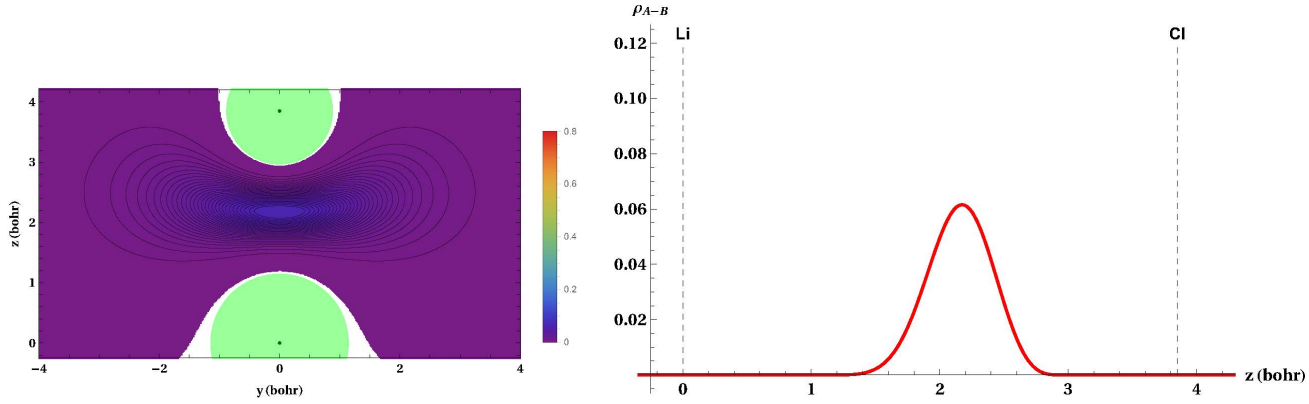


(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.22: Bond electron density (BDEN) for HCl is obtained using the Becke and Awad weights. The atomic coordinates of Cl at 0.0 bohr and H at 2.4000 bohr along z axis.

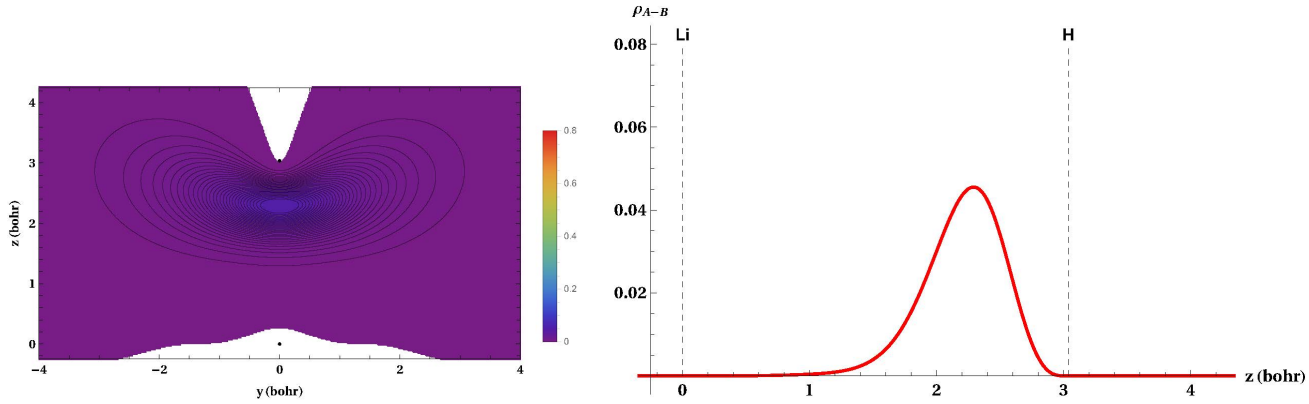


(a) The core size is defined as Becke weight.

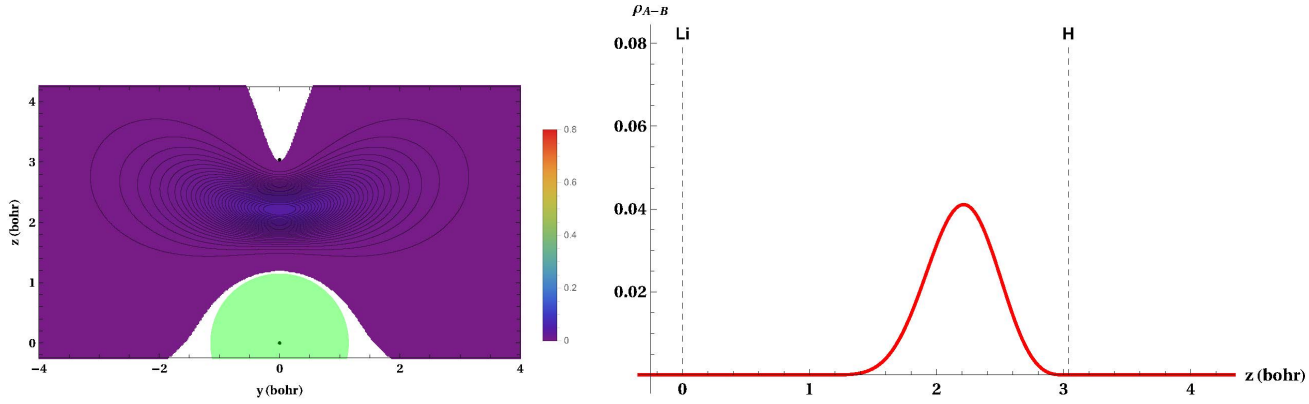


(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.23: Bond electron density (BDEN) for LiCl is obtained using the Becke and Awad weights. The atomic coordinates of Li at 0.0 bohr and Cl at 3.8492 bohr along z axis.

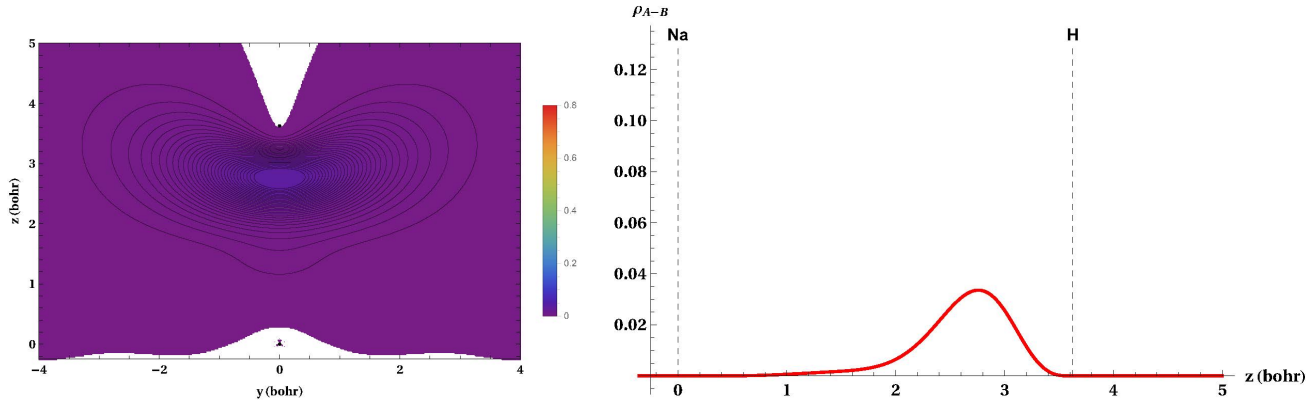


(a) The core size is defined as Becke weight.

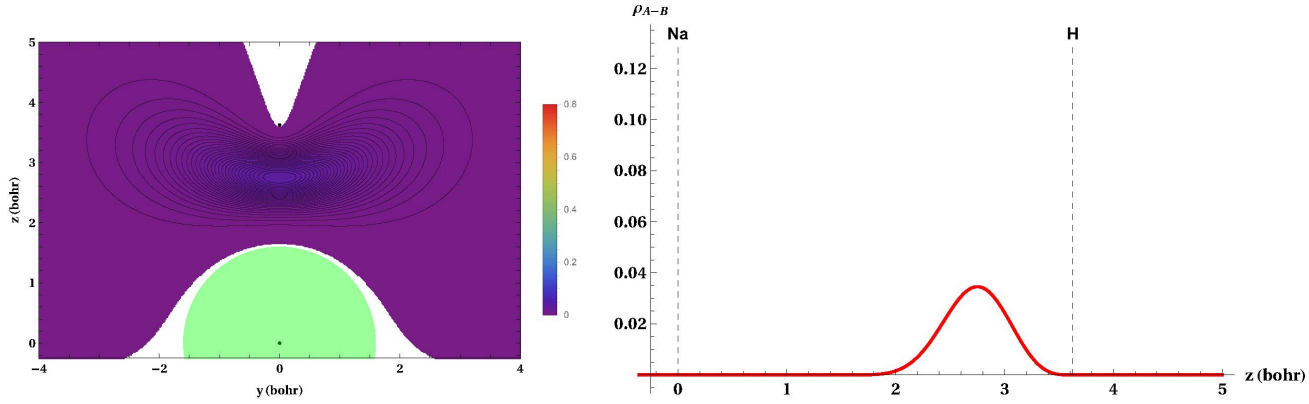


(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.24: Bond electron density (BDEN) for LiH is obtained using the Becke and Awad weights. The atomic coordinates of Li at 0.0 bohr and H at 3.0390 bohr along z axis.



(a) The core size is defined as Becke weight.



(b) Awad weight at $r_{\langle r \rangle}$.

Figure A.25: Bond electron density (BDEN) for NaH is obtained using the Becke and Awad weights. The atomic coordinates of Na at 0.0 bohr and H at 3.6261 bohr along z axis.

Appendix B

AIMD Code

B.1 Input and Output Examples

In this appendix you could find some example of input and output files that has been used during the research.

B.1.1 Displaying the Fragments of the Target Molecule

Here is example of input file to show the cartesian coordinate of the fragments for Glysine molecule,

```
MOLECULE
 UNITS=ANGSTROM !(default)
 CHARGE=0
 MULTIPLICITY=1
FREEZ
N
C    1 B1
C    2 B2 1 A2
O    3 B3 2 A3 1 D3
H    2 B4 1 A4 3 D4
H    2 B5 1 A5 3 D5
H    1 B6 2 A6 3 D6
H    1 B7 2 A7 3 D7
O    3 B8 2 A8 1 D8
H    9 B9 3 A9 2 D9

END
```

```
DEFINE
B1 = 1.44084286              B2 = 1.51781924              B3 = 1.18115749
B4 = 1.08653973              B5 = 1.09386650              B6 = 1.00062635
B7 = 0.99972788              B8 = 1.33306652              B9 = 0.94790958
A2 =  110.31733              A3 =  123.50442              A4 =  109.68864
A5 =  114.69657              A6 =  110.26867              A7 =  111.39727
A8 =  115.54740              A9 =  112.36145              D3 =  -21.55248
D4 =  120.68224              D5 = -119.29373              D6 =   38.90011
D7 =  159.88520              D8 =  161.62252              D9 =   -5.50811
END
end !molecule

BASIS name=6-31G(d) end

AIMDFT
 Level=2
end !AIMDFT

output object=AIMDFT:FRAGMENT%DISPLAY end
stop
```

Here is the output,

```
Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Free format Z-Matrix for: C2H5NO2, (C1)
N
C         N         B1
C         C         B2   N         A2
O         C         B3   C         A3   N         D3
H         C         B4   N         A4   C         D4
H         C         B5   N         A5   C         D5
H         N         B6   C         A6   C         D6
H         N         B7   C         A7   C         D7
O         C         B8   C         A8   N         D8
H         O         B9   C         A9   C         D9

VARIABLES:
B1 = 1.44084286              B2 = 1.51781924              B3 = 1.18115749
B4 = 1.08653973              B5 = 1.09386650              B6 = 1.00062635
B7 = 0.99972788              B8 = 1.33306652              B9 = 0.94790958
A2 =  110.31733              A3 =  123.50442              A4 =  109.68864
A5 =  114.69657              A6 =  110.26867              A7 =  111.39727
A8 =  115.54740              A9 =  112.36145              D3 =  -21.55248
D4 =  120.68224              D5 = -119.29373              D6 =   38.90011
D7 =  159.88520              D8 =  161.62252              D9 =   -5.50811

Z MATRIX FOR: C2H5NO2, (C1)
-----------------------------------------------------------------------------------------------

   I    AN    Z1      BL                  Z2     ALPHA                Z3      BETA
              Z4
-----------------------------------------------------------------------------------------------

   1    7
   2    6     1    1.440843    (  1)
   3    6     2    1.517819    (  2)        1    110.3173    ( 10)
   4    8     3    1.181157    (  3)        2    123.5044    ( 11)        1    -21.5525    (
    18)        0
```

```
   5    1    2   1.086540   (  4)       1   109.6886   ( 12)       3   120.6822   (
    19)       0
   6    1    2   1.093866   (  5)       1   114.6966   ( 13)       3  -119.2937   (
    20)       0
   7    1    1   1.000626   (  6)       2   110.2687   ( 14)       3    38.9001   (
    21)       0
   8    1    1   0.999728   (  7)       2   111.3973   ( 15)       3   159.8852   (
    22)       0
   9    8    3   1.333067   (  8)       2   115.5474   ( 16)       1   161.6225   (
    23)       0
  10    1    9   0.947910   (  9)       3   112.3615   ( 17)       2    -5.5081   (
    24)       0
-----------------------------------------------------------------------------------------------


Cartesian coordinates for: C2H5NO2, (C1)

-----------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                       X
             Y             Z
-----------------------------------------------------------------------------------------------

   1 N          7   0.00000000     0.00000000     0.00000000               0.00000000
       0.00000000     0.00000000
   2 C          6   0.00000000     0.00000000     1.44084286               0.00000000
       0.00000000     2.72279820
   3 C          6   1.42338653     0.00000000     1.96785902               2.68981052
       0.00000000     3.71871433
   4 O          8   2.35288801    -0.36180632     1.33520098               4.44631363
      -0.68371481     2.52316398
   5 H          1  -0.52202178     0.87980599     1.80690743              -0.98647812
       1.66259225     3.41455992
   6 H          1  -0.48626030    -0.86672789     1.89787416              -0.91889872
      -1.63787822     3.58646212
   7 H          1   0.73050930    -0.58944902    -0.34663973               1.38046242
      -1.11389713    -0.65505411
   8 H          1  -0.87404490    -0.32011104    -0.36473314              -1.65170536
      -0.60492216    -0.68924570
   9 O          8   1.56620070     0.37919230     3.23785258               2.95969016
       0.71656954     6.11865417
  10 H          1   0.75176248     0.68026803     3.61808458               1.42062510
       1.28552018     6.83718846
-----------------------------------------------------------------------------------------------

Nuclear repulsion energy:     181.510743561


Distance Matrix for: C2H5NO2, (C1)
             1           2           3           4           5           6
     7           8           9          10
   1      0.000000    1.440843    2.428682    2.729423    2.076410    2.142333
    1.000626    0.999728    3.616692    3.757452
   2      1.440843    0.000000    1.517819    2.382886    1.086540    1.093866
    2.018957    2.031386    2.413714    2.401727
   3      2.428682    1.517819    0.000000    1.181157    2.141163    2.098301
    2.486852    3.289627    1.333067    1.907115
   4      2.729423    2.382886    1.181157    0.000000    3.166893    2.938079
    2.347876    3.647549    2.188159    2.976757
   5      2.076410    1.086540    2.141163    3.166893    0.000000    1.749267
    2.892285    2.505942    2.580483    2.223219
   6      2.142333    1.093866    2.098301    2.938079    1.749267    0.000000
```

```
    2.568123     2.359779     2.749628     2.623932
   7       1.000626     2.018957     2.486852     2.347876     2.892285     2.568123
    0.000000     1.627103     3.805947     4.163132
   8       0.999728     2.031386     3.289627     3.647549     2.505942     2.359779
    1.627103     0.000000     4.407091     4.416655
   9       3.616692     2.413714     1.333067     2.188159     2.580483     2.749628
    3.805947     4.407091     0.000000     0.947910
  10       3.757452     2.401727     1.907115     2.976757     2.223219     2.623932
    4.163132     4.416655     0.947910     0.000000

Charge=    0, Number of electrons=   40

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       85


      ___  _____ ___  _________ ______  _____
     / _ \|_   _||  \/  ||  _  \|  ___||_   _|
    / /_\ \ | |  | .  . || | | || |_     | |
    |  _  | | |  | |\/| || | | ||  _|    | |
    | | | |_| |_ | |  | || |/ / | |      | |
    \_| |_/\___/ \_|  |_/|___/  \_|      \_/

The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

    4124847 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:     113538 IJKJ:     113807 IJJL:     114903 IIKK:       3799
 IJJJ:       1462 IIIL:       1470 IIII:         30 IJKL:    3775838
Number of integrals in INCORE buffers:
IIKK:       2085 IJJL:      64223 IJKJ:      63543
IIKL:      64237 IJKL:    1535810

     937565 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      43244 IJKJ:      44015 IJJL:      43286 IIKK:       2740
 IJJJ:       1152 IIIL:       1158 IIII:         55 IJKL:     801915
Number of integrals in INCORE buffers:
IIKK:       3570 IJJL:      88212 IJKJ:      87691
IIKL:      88349 IJKL:     271361
Number of buffers:
IJJL:          0 IJKJ:          0
IIKL:          0 IJKL:          1
 TOTAL OF    5062412 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF               Nuclear Repulsion Energy is    181.510743561 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE   ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:    1       -463.303052879      -281.792309318
 SCF_CYCLE:    2       -464.156082549      -282.645338988   2.78967E-02
 SCF_CYCLE:    3       -464.252967556      -282.742223996   1.11306E-02
 SCF_CYCLE:    4       -464.282711412      -282.771967851   8.78492E-03
 SCF_CYCLE:    5       -464.301085806      -282.790342245   6.67293E-03
 SCF_CYCLE:    6       -464.310646467      -282.799902906   5.26956E-03
 SCF_CYCLE:    7       -464.325561119      -282.814817558                  4-POINT
 SCF_CYCLE:    8       -464.326970023      -282.816226462   2.39444E-03
 SCF_CYCLE:    9       -464.326980754      -282.816237193   1.12426E-04
 SCF_CYCLE:   10       -464.326984887      -282.816241326   7.27041E-05
```

```
 SCF_CYCLE:  11       -464.326986541      -282.816242980   4.60760E-05
 SCF_CYCLE:  12       -464.326987992      -282.816244431                    4-POINT
 SCF_CYCLE:  13       -464.326987698      -282.816244138   1.87641E-05
At termination total energy is      -282.816244  Hartrees


***********************************
    Current cartesian coordinates
***********************************

Fragment#  1, Atoms# 10
!Atom           X            Y            Z      Ctype    Ftype    Level   Sorted
N          0.00000000   0.00000000   0.00000000              9        0       T
C          0.00000000   0.00000000   1.44084286             29        1       T
H          0.73050930  -0.58944902  -0.34663973              5        1       F
H         -0.87404490  -0.32011104  -0.36473314              5        1       F
C          1.42338653   0.00000000   1.96785902             10        2       T
H         -0.52202178   0.87980599   1.80690743              4        2       F
H         -0.48626030  -0.86672789   1.89787416              4        2       F
H          1.40752011   0.00000000   3.05140244   -CH3       4        3       F
H          1.93964311   0.88478395   1.61422314   -CH3       4        3       F
H          1.93966847  -0.88478388   1.61415464   -CH3       4        3       F

Fragment#  2, Atoms# 10
!Atom           X            Y            Z      Ctype    Ftype    Level   Sorted
C          0.00000000   0.00000000   1.44084286             29        0       T
N          0.00000000   0.00000000   0.00000000              9        1       T
C          1.42338653   0.00000000   1.96785902             34        1       T
H         -0.52202178   0.87980599   1.80690743              4        1       F
H         -0.48626030  -0.86672789   1.89787416              4        1       F
O          1.56620070   0.37919230   3.23785258              7        2       T
O          2.35288801  -0.36180632   1.33520098              4        2       T
H          0.73050930  -0.58944902  -0.34663973              5        2       F
H         -0.87404490  -0.32011104  -0.36473314              5        2       F
H          2.53932814   0.31941883   3.40629958   -OH        6        3       T

Fragment#  3, Atoms# 10
!Atom           X            Y            Z      Ctype    Ftype    Level   Sorted
C          1.42338653   0.00000000   1.96785902             34        0       T
O          1.56620070   0.37919230   3.23785258              7        1       T
O          2.35288801  -0.36180632   1.33520098              4        1       T
C          0.00000000   0.00000000   1.44084286             29        1       T
N          0.00000000   0.00000000   0.00000000              9        2       T
H          0.75176248   0.68026803   3.61808458              6        2       T
H         -0.52202178   0.87980599   1.80690743              4        2       F
H         -0.48626030  -0.86672789   1.89787416              4        2       F
H         -0.95777357   0.00000000  -0.29610147   -NH2       5        3       F
H          0.40141812  -0.86960710  -0.29610147   -NH2       5        3       F

Fragment#  4, Atoms#  8
!Atom           X            Y            Z      Ctype    Ftype    Level   Sorted
O          2.35288801  -0.36180632   1.33520098              4        0       T
C          1.42338653   0.00000000   1.96785902             34        1       T
O          1.56620070   0.37919230   3.23785258              7        2       T
C          0.00000000   0.00000000   1.44084286             10        2       T
H          0.65316574   0.62862176   3.52608235   -OH        6        3       T
H         -0.66879662   0.37530236   2.20646504   -CH3       4        3       F
H         -0.28673497  -1.01059766   1.17457750   -CH3       4        3       F
H         -0.06099919   0.63521777   0.56490118   -CH3       4        3       F

Fragment#  5, Atoms# 10
!Atom           X            Y            Z      Ctype    Ftype    Level   Sorted
H         -0.52202178   0.87980599   1.80690743              4        0       T
C          0.00000000   0.00000000   1.44084286             29        1       T
```

```
N          0.00000000   0.00000000   0.00000000              9       2       T
C          1.42338653   0.00000000   1.96785902             10       2       T
H         -0.48626030  -0.86672789   1.89787416              4       2       T
H          0.48872922  -0.82369531  -0.29610147   -NH2       5       3       F
H         -0.95270540  -0.09851616  -0.29610147   -NH2       5       3       F
H          1.94391057  -0.87751344   1.60269742   -CH3       4       3       F
H          1.93528625   0.89191159   1.62599031   -CH3       4       3       F
H          1.40752416  -0.01421698   3.05139149   -CH3       4       3       F

Fragment#  6, Atoms# 10
!Atom           X           Y           Z      Ctype    Ftype    Level   Sorted
H         -0.48626030  -0.86672789   1.89787416              4       0       T
C          0.00000000   0.00000000   1.44084286             29       1       T
N          0.00000000   0.00000000   0.00000000              9       2       T
C          1.42338653   0.00000000   1.96785902             10       2       T
H         -0.52202178   0.87980599   1.80690743              4       2       T
H          0.46862617   0.83529619  -0.29610147   -NH2       5       3       F
H         -0.95481287   0.07540156  -0.29610147   -NH2       5       3       F
H          1.96357598   0.84116568   1.54958427   -CH3       4       3       F
H          1.91452219  -0.92278526   1.68207074   -CH3       4       3       F
H          1.40861876   0.08144592   3.04843517   -CH3       4       3       F

Fragment#  7, Atoms#  7
!Atom           X           Y           Z      Ctype    Ftype    Level   Sorted
H          0.73050930  -0.58944902  -0.34663973              5       0       T
N          0.00000000   0.00000000   0.00000000              9       1       T
C          0.00000000   0.00000000   1.44084286             14       2       T
H         -0.87404490  -0.32011104  -0.36473314              5       2       T
H         -0.79508214   0.64155293   1.80219079   -CH3       4       3       F
H         -0.15801825  -1.00939572   1.80219079   -CH3       4       3       F
H          0.95326447   0.36771031   1.80219079   -CH3       4       3       F

Fragment#  8, Atoms#  7
!Atom           X           Y           Z      Ctype    Ftype    Level   Sorted
H         -0.87404490  -0.32011104  -0.36473314              5       0       T
N          0.00000000   0.00000000   0.00000000              9       1       T
C          0.00000000   0.00000000   1.44084286             14       2       T
H          0.73050930  -0.58944902  -0.34663973              5       2       T
H          0.95932437   0.35134388   1.80219079   -CH3       4       3       F
H         -0.17544770  -1.00651259   1.80219079   -CH3       4       3       F
H         -0.78407467   0.65509612   1.80219079   -CH3       4       3       F

Fragment#  9, Atoms#  8
!Atom           X           Y           Z      Ctype    Ftype    Level   Sorted
O          1.56620070   0.37919230   3.23785258              7       0       T
C          1.42338653   0.00000000   1.96785902             34       1       T
H          0.75176248   0.68026803   3.61808458              6       1       T
O          2.35288801  -0.36180632   1.33520098              4       2       T
C          0.00000000   0.00000000   1.44084286             10       2       T
H         -0.00222494  -0.32209824   0.40616133   -CH3       4       3       F
H         -0.60406692  -0.67858912   2.03164044   -CH3       4       3       F
H         -0.41037651   1.00075379   1.50851356   -CH3       4       3       F

Fragment# 10, Atoms#  6
!Atom           X           Y           Z      Ctype    Ftype    Level   Sorted
H          0.75176248   0.68026803   3.61808458              6       0       T
O          1.56620070   0.37919230   3.23785258              7       1       T
C          1.42338653   0.00000000   1.96785902             18       2       T
H          2.37886126  -0.33411421   1.58087862   -CH3       4       3       F
H          0.70717887  -0.81134180   1.91135348   -CH3       4       3       F
H          1.06777838   0.83714620   1.37860065   -CH3       4       3       F

***********************************
    Database cartesian coordinates
```

```
************************************

Fragment#  1, Atoms# 10
!Atom            X            Y            Z     Ctype    Ftype    Level   Sorted
N          0.00000000   -0.00000000    0.00000000              9        0       T
C         -0.00000000   -0.00000000    1.44084286             29        1       T
H          0.00000000    0.93866607   -0.34663973              5        1       F
H          0.79799306   -0.47920001   -0.36473314              5        1       F
C         -0.89383628    1.10773910    1.96785902             10        2       T
H         -0.35689075   -0.95874622    1.80690743              4        2       F
H          0.97987823    0.16584623    1.89787416              4        2       F
H         -0.88387273    1.09539118    3.05140244   -CH3       4        3       F
H         -1.90660309    0.95389867    1.61422314   -CH3       4        3       F
H         -0.52946711    2.06514427    1.61415464   -CH3       4        3       F

Fragment#  2, Atoms# 10
!Atom            X            Y            Z     Ctype    Ftype    Level   Sorted
C         -0.00000000    0.00000000    0.00000000             29        0       T
N          0.00000000   -0.00000000    1.44084286              9        1       T
C          0.00000000    1.42338653   -0.52701616             34        1       T
H          0.87980599   -0.52202178   -0.36606457              4        1       F
H         -0.86672789   -0.48626030   -0.45703130              4        1       F
O          0.37919230    1.56620070   -1.79700972              7        2       T
O         -0.36180632    2.35288801    0.10564188              4        2       T
H         -0.58944902    0.73050930    1.78748259              5        2       F
H         -0.32011104   -0.87404490    1.80557600              5        2       F
H          0.31941883    2.53932814   -1.96545672   -OH        6        3       T

Fragment#  3, Atoms# 10
!Atom            X            Y            Z     Ctype    Ftype    Level   Sorted
C         -0.00000000   -0.00000000    0.00000000             34        0       T
O         -0.00000000    0.00000000    1.33306652              7        1       T
O          0.00000000    1.01381600   -0.60606116              4        1       T
C          0.07368323   -1.36743652   -0.65457115             29        1       T
N         -0.35716835   -1.28888329   -2.02724171              9        2       T
H         -0.03707255   -0.87584463    1.69369704              6        2       T
H         -0.55932392   -2.06380818   -0.11149008              4        2       F
H          1.10434945   -1.70738451   -0.51779972              4        2       F
H         -0.29008929   -2.21219942   -2.41194164   -NH2       5        3       F
H          0.30676556   -0.71652891   -2.51368937   -NH2       5        3       F

Fragment#  4, Atoms#  8
!Atom            X            Y            Z     Ctype    Ftype    Level   Sorted
O         -0.00000000    0.00000000   -0.00000000              4        0       T
C         -0.00000000   -0.00000000    1.18115749             34        1       T
O         -0.00000000    1.14420319    1.86516436              7        2       T
C          0.07368323   -1.26347662    2.01899570             10        2       T
H          0.00000000    0.86533235    2.81445637   -OH        6        3       T
H          0.05839270   -1.00128368    3.07034682   -CH3       4        3       F
H          0.99092661   -1.79429459    1.79245948   -CH3       4        3       F
H         -0.77563740   -1.89735718    1.79239856   -CH3       4        3       F

Fragment#  5, Atoms# 10
!Atom            X            Y            Z     Ctype    Ftype    Level   Sorted
H         -0.00000000    0.00000000   -0.00000000              4        0       T
C         -0.00000000   -0.00000000    1.08653973             29        1       T
N          0.00000000    1.35660739    1.57197206              9        2       T
C          1.22412734   -0.74090902    1.59284135             10        2       T
H         -0.86045971   -0.59784588    1.40075830              4        2       T
H          0.00000000    1.31271584    2.57351079   -NH2       5        3       F
H         -0.86960710    1.77063919    1.29378103   -NH2       5        3       F
H          1.22400910   -0.74083745    2.67650093   -CH3       4        3       F
H          2.11948787   -0.24860401    1.23174979   -CH3       4        3       F
H          1.20323095   -1.76248761    1.23168092   -CH3       4        3       F
```

```
Fragment#  6, Atoms# 10
!Atom           X           Y           Z     Ctype    Ftype    Level   Sorted
H          -0.00000000   0.00000000   0.00000000              4       0       T
C           0.00000000   0.00000000   1.09386650             29       1       T
N           0.00000000   1.30905356   1.69586892              9       2       T
C           1.24136788  -0.76979504   1.50641545             10       2       T
H          -0.88574475  -0.54645193   1.40598042              4       2       T
H           0.00000000   1.17790163   2.68975295   -NH2       5       3       F
H          -0.86960710   1.74578916   1.45488219   -NH2       5       3       F
H           1.30090798  -0.80671695   2.58780800   -CH3       4       3       F
H           2.12120534  -0.27430176   1.11297643   -CH3       4       3       F
H           1.18863813  -1.77819294   1.11290613   -CH3       4       3       F

Fragment#  7, Atoms#  7
!Atom           X           Y           Z     Ctype    Ftype    Level   Sorted
H           0.00000000   0.00000000   0.00000000              5       0       T
N          -0.00000000   0.00000000   1.00062635              9       1       T
C           0.00000000   1.35162371   1.49976710             14       2       T
H           0.79799306  -0.50815411   1.32380172              5       2       T
H           0.00000000   1.33667752   2.58332361   -CH3       4       3       F
H           0.88478395   1.86757977   1.14569291   -CH3       4       3       F
H          -0.88478388   1.86760507   1.14562438   -CH3       4       3       F

Fragment#  8, Atoms#  7
!Atom           X           Y           Z     Ctype    Ftype    Level   Sorted
H           0.00000000  -0.00000000  -0.00000000              5       0       T
N          -0.00000000  -0.00000000   0.99972788              9       1       T
C           0.00000000   1.34153017   1.52539407             14       2       T
H           0.80471964  -0.49904836   1.32319369              5       2       T
H           0.00000000   1.30524460   2.60844599   -CH3       4       3       F
H           0.88478395   1.86436016   1.18155110   -CH3       4       3       F
H          -0.88478388   1.86438681   1.18148309   -CH3       4       3       F

Fragment#  9, Atoms#  8
!Atom           X           Y           Z     Ctype    Ftype    Level   Sorted
O          -0.00000000  -0.00000000   0.00000000              7       0       T
C          -0.00000000  -0.00000000   1.33306652             34       1       T
H          -0.00000000   0.87662888  -0.36063052              6       1       T
O           0.04287417  -1.01290903   1.93912768              4       2       T
C          -0.13144606   1.36309713   1.98763767             10       2       T
H          -0.12044877   1.24905513   3.06522367   -CH3       4       3       F
H          -1.06458625   1.82199002   1.68253358   -CH3       4       3       F
H           0.69680779   1.99187626   1.68246767   -CH3       4       3       F

Fragment# 10, Atoms#  6
!Atom           X           Y           Z     Ctype    Ftype    Level   Sorted
H           0.00000000  -0.00000000  -0.00000000              6       0       T
O           0.00000000  -0.00000000   0.94790958              7       1       T
C           0.00000000   1.23282287   1.45507238             18       2       T
H          -0.00000000   1.17831758   2.53736036   -CH3       4       3       F
H           0.88478395   1.76136479   1.12007593   -CH3       4       3       F
H          -0.88478388   1.76139258   1.12000837   -CH3       4       3       F

***********************************
     Molecule atoms symbols
***********************************
    1   7(6(6,1,1)1()1())
    2   6(7(1,1)6(8,8)1()1())
    3   6(8(1)8()6(7,1,1))
    4   8(6(8,6))
    5   1(6(7,6,1))
    6   1(6(7,6,1))
    7   1(7(6,1))
```

```
    8   1(7(6,1))
    9   8(6(8,6)1())
   10   1(8(6))

PROGRAM> end of inputs


Program terminated normally


Job: C2H5NO2_C1_RHF_631Gd ended on :24-Aug-18 at 12:50:13
User: ibrahim
Cpu     time:  00h00m04s46c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m05s00c
```

B.1.2 Saving the Molecular Properties in the Database

Here is example of input file to save the molecular properties of Glysine into the database,

```
MOLECULE
 UNITS=ANGSTROM !(default)
 CHARGE=0
 MULTIPLICITY=1
FREEZ
N
C   1 B1
C   2 B2 1 A2
O   3 B3 2 A3 1 D3
H   2 B4 1 A4 3 D4
H   2 B5 1 A5 3 D5
H   1 B6 2 A6 3 D6
H   1 B7 2 A7 3 D7
O   3 B8 2 A8 1 D8
H   9 B9 3 A9 2 D9

END
DEFINE
B1 = 1.44084286              B2 = 1.51781924              B3 = 1.18115749
B4 = 1.08653973              B5 = 1.09386650              B6 = 1.00062635
B7 = 0.99972788              B8 = 1.33306652              B9 = 0.94790958
A2 =  110.31733              A3 =  123.50442              A4 =  109.68864
A5 =  114.69657              A6 =  110.26867              A7 =  111.39727
A8 =  115.54740              A9 =  112.36145              D3 =  -21.55248
D4 =  120.68224              D5 = -119.29373              D6 =   38.90011
D7 =  159.88520              D8 =  161.62252              D9 =   -5.50811
END
end !molecule

BASIS name=6-31G(d) end

SET RUN NAME = "RUN_Gly_" end end
!RUN NAME = "RUN_Gly_" end

PARtitioning
Scheme=BECKE  ! other examples: ABSw, IAWAD, Fermi, iAwad (default is Becke)
end

AIMDFT
 Level=2
```

```
! OPT  ! Fragments optimization before save in the database (default is NOOPT)
end !AIMDFT

output object=AIMDFT:FRAGMENT%BUILD end
stop
```

Here is the output,

```
Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Free format Z-Matrix for: C2H5NO2, (C1)
N
C         N         B1
C         C         B2   N         A2
O         C         B3   C         A3   N          D3
H         C         B4   N         A4   C          D4
H         C         B5   N         A5   C          D5
H         N         B6   C         A6   C          D6
H         N         B7   C         A7   C          D7
O         C         B8   C         A8   N          D8
H         O         B9   C         A9   C          D9

VARIABLES:
B1 = 1.44084286               B2 = 1.51781924               B3 = 1.18115749
B4 = 1.08653973               B5 = 1.09386650               B6 = 1.00062635
B7 = 0.99972788               B8 = 1.33306652               B9 = 0.94790958
A2 =  110.31733               A3 =  123.50442               A4 =  109.68864
A5 =  114.69657               A6 =  110.26867               A7 =  111.39727
A8 =  115.54740               A9 =  112.36145               D3 =  -21.55248
D4 =  120.68224               D5 = -119.29373               D6 =   38.90011
D7 =  159.88520               D8 =  161.62252               D9 =   -5.50811

Z MATRIX FOR: C2H5NO2, (C1)
-----------------------------------------------------------------------------------------------

   I    AN    Z1      BL                  Z2     ALPHA                 Z3      BETA
              Z4
-----------------------------------------------------------------------------------------------

   1     7
   2     6     1    1.440843    (  1)
   3     6     2    1.517819    (  2)        1    110.3173    ( 10)
   4     8     3    1.181157    (  3)        2    123.5044    ( 11)        1    -21.5525    (
    18)         0
   5     1     2    1.086540    (  4)        1    109.6886    ( 12)        3    120.6822    (
    19)         0
   6     1     2    1.093866    (  5)        1    114.6966    ( 13)        3   -119.2937    (
    20)         0
   7     1     1    1.000626    (  6)        2    110.2687    ( 14)        3     38.9001    (
    21)         0
   8     1     1    0.999728    (  7)        2    111.3973    ( 15)        3    159.8852    (
    22)         0
   9     8     3    1.333067    (  8)        2    115.5474    ( 16)        1    161.6225    (
    23)         0
  10     1     9    0.947910    (  9)        3    112.3615    ( 17)        2     -5.5081    (
    24)         0
-----------------------------------------------------------------------------------------------
```

```
Cartesian coordinates for: C2H5NO2, (C1)

-------------------------------------------------------------------------------------------------------------

                              COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                           X
              Y              Z
-------------------------------------------------------------------------------------------------------------

   1 N          7   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   2 C          6   0.00000000     0.00000000     1.44084286               0.00000000
        0.00000000     2.72279820
   3 C          6   1.42338653     0.00000000     1.96785902               2.68981052
        0.00000000     3.71871433
   4 O          8   2.35288801    -0.36180632     1.33520098               4.44631363
       -0.68371481     2.52316398
   5 H          1  -0.52202178     0.87980599     1.80690743              -0.98647812
        1.66259225     3.41455992
   6 H          1  -0.48626030    -0.86672789     1.89787416              -0.91889872
       -1.63787822     3.58646212
   7 H          1   0.73050930    -0.58944902    -0.34663973               1.38046242
       -1.11389713    -0.65505411
   8 H          1  -0.87404490    -0.32011104    -0.36473314              -1.65170536
       -0.60492216    -0.68924570
   9 O          8   1.56620070     0.37919230     3.23785258               2.95969016
        0.71656954     6.11865417
  10 H          1   0.75176248     0.68026803     3.61808458               1.42062510
        1.28552018     6.83718846
-------------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:     181.510743561


Distance Matrix for: C2H5NO2, (C1)
             1           2           3           4           5           6
     7           8           9          10
   1      0.000000    1.440843    2.428682    2.729423    2.076410    2.142333
    1.000626    0.999728    3.616692    3.757452
   2      1.440843    0.000000    1.517819    2.382886    1.086540    1.093866
    2.018957    2.031386    2.413714    2.401727
   3      2.428682    1.517819    0.000000    1.181157    2.141163    2.098301
    2.486852    3.289627    1.333067    1.907115
   4      2.729423    2.382886    1.181157    0.000000    3.166893    2.938079
    2.347876    3.647549    2.188159    2.976757
   5      2.076410    1.086540    2.141163    3.166893    0.000000    1.749267
    2.892285    2.505942    2.580483    2.223219
   6      2.142333    1.093866    2.098301    2.938079    1.749267    0.000000
    2.568123    2.359779    2.749628    2.623932
   7      1.000626    2.018957    2.486852    2.347876    2.892285    2.568123
    0.000000    1.627103    3.805947    4.163132
   8      0.999728    2.031386    3.289627    3.647549    2.505942    2.359779
    1.627103    0.000000    4.407091    4.416655
   9      3.616692    2.413714    1.333067    2.188159    2.580483    2.749628
    3.805947    4.407091    0.000000    0.947910
  10      3.757452    2.401727    1.907115    2.976757    2.223219    2.623932
    4.163132    4.416655    0.947910    0.000000

Charge=    0, Number of electrons=   40

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       85
```

```
The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

    4124847 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:     113538 IJKJ:     113807 IJJL:     114903 IIKK:       3799
 IJJJ:       1462 IIIL:       1470 IIII:         30 IJKL:    3775838
Number of integrals in INCORE buffers:
IIKK:       2085 IJJL:      64223 IJKJ:      63543
IIKL:      64237 IJKL:    1535810

     937565 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      43244 IJKJ:      44015 IJJL:      43286 IIKK:       2740
 IJJJ:       1152 IIIL:       1158 IIII:         55 IJKL:     801915
Number of integrals in INCORE buffers:
IIKK:       3570 IJJL:      88212 IJKJ:      87691
IIKL:      88349 IJKL:     271361
Number of buffers:
IJJL:          0 IJKJ:          0
IIKL:          0 IJKL:          1
 TOTAL OF    5062412 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF               Nuclear Repulsion Energy is    181.510743561 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE   ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:   1       -463.303052879      -281.792309318
 SCF_CYCLE:   2       -464.156082549      -282.645338988   2.78967E-02
 SCF_CYCLE:   3       -464.252967556      -282.742223996   1.11306E-02
 SCF_CYCLE:   4       -464.282711412      -282.771967851   8.78492E-03
 SCF_CYCLE:   5       -464.301085806      -282.790342245   6.67293E-03
 SCF_CYCLE:   6       -464.310646467      -282.799902906   5.26956E-03
 SCF_CYCLE:   7       -464.325561119      -282.814817558                  4-POINT
 SCF_CYCLE:   8       -464.326970023      -282.816226462   2.39444E-03
 SCF_CYCLE:   9       -464.326980754      -282.816237193   1.12426E-04
 SCF_CYCLE:  10       -464.326984887      -282.816241326   7.27041E-05
 SCF_CYCLE:  11       -464.326986541      -282.816242980   4.60760E-05
 SCF_CYCLE:  12       -464.326987992      -282.816244431                  4-POINT
 SCF_CYCLE:  13       -464.326987698      -282.816244138   1.87641E-05
At termination total energy is      -282.816244  Hartrees


Number of indexs available in the database    =   0
Number of indexs NOT available in the database =  8
MaxIndex  =  114
**********************************
 The unique Symbols in the molecule
**********************************
SYM#    1 is 7(6(6,1,1)1()1())
SYM#    2 is 6(7(1,1)6(8,8)1()1())
SYM#    3 is 6(8(1)8()6(7,1,1))
SYM#    4 is 8(6(8,6))
SYM#    5 is 1(6(7,6,1))
SYM#    6 is 1(7(6,1))
SYM#    7 is 8(6(8,6)1())
SYM#    8 is 1(8(6))
AIMDFT-AIMDFT-AIMDFT-AIMDFT-AIMDFT-AIMDFT-AIMDFT-AIMDFT
```

```
*** RUNing the inputfile :: INPUT_0115.dat


****************************************
 Working on symbol 7(6(6,1,1)1()1())
 IndexNUM          115
****************************************


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: 7(6(6,1,1)1()1())

-------------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                        X
             Y              Z
-------------------------------------------------------------------------------------------------------

   1 N          7   0.00000000     -0.00000000     0.00000000               0.00000000
       -0.00000000     0.00000000
   2 C          6  -0.00000000     -0.00000000     1.44084286              -0.00000000
       -0.00000000     2.72279820
   3 H          1   0.00000000      0.93866607    -0.34663973               0.00000000
        1.77382166    -0.65505411
   4 H          1   0.79799306     -0.47920001    -0.36473314               1.50798822
       -0.90555672    -0.68924570
   5 C          6  -0.89383628      1.10773910     1.96785902              -1.68910566
        2.09332336     3.71871433
   6 H          1  -0.35689075     -0.95874622     1.80690743              -0.67442573
       -1.81176764     3.41455992
   7 H          1   0.97987823      0.16584623     1.89787416               1.85170135
        0.31340393     3.58646212
   8 H          1  -0.88387273      1.09539118     3.05140244              -1.67027728
        2.06998919     5.76631450
   9 H          1  -1.90660309      0.95389867     1.61422314              -3.60295741
        1.80260710     3.05043943
  10 H          1  -0.52946711      2.06514427     1.61415464              -1.00054777
        3.90255681     3.05030997
-------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:      83.927218609


Distance Matrix for: 7(6(6,1,1)1()1())
              1           2           3           4           5           6
     7           8           9          10
   1      0.000000    1.440843    1.000626    0.999728    2.428682    2.076410
    2.142333    3.360382    2.674093    2.674070
   2      1.440843    0.000000    2.018957    2.031386    1.517819    1.086540
    1.093867    2.138928    2.138953    2.138970
   3      1.000626    2.018957    0.000000    1.627103    2.486852    2.892285
    2.568123    3.514610    2.735023    2.322499
   4      0.999728    2.031386    1.627103    0.000000    3.289627    2.505942
    2.359779    4.120436    3.644843    3.485948
   5      2.428682    1.517819    2.486852    3.289627    0.000000    2.141163
    2.098301    1.083660    1.083707    1.083742
```

```
   6      2.076410    1.086540    2.892285    2.505942    2.141163    0.000000
    1.749267    2.458853    2.469199    3.034938
   7      2.142333    1.093867    2.568123    2.359779    2.098301    1.749267
    0.000000    2.380808    3.005538    2.442530
   8      3.360382    2.138928    3.514610    4.120436    1.083660    2.458853
    2.380808    0.000000    1.769599    1.769663
   9      2.674093    2.138953    2.735023    3.644843    1.083707    2.469199
    3.005538    1.769599    0.000000    1.769568
  10      2.674070    2.138970    2.322499    3.485948    1.083742    3.034938
    2.442530    1.769663    1.769568    0.000000

Charge=    0, Number of electrons=   26

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       59

 Partitioning scheme set to: BECKE
Free format Z-Matrix for: 7(6(6,1,1)1()1())
N1
C2        N1        C2N1
H3        N1        H3N1              C2        H3N1C2
H4        N1        H4N1              C2        H4N1C2            H3        H4N1C2H3
C5        C2        C5C2              N1        C5C2N1            H3        C5C2N1H3
H6        C2        H6C2              N1        H6C2N1            H3        H6C2N1H3
H7        C2        H7C2              N1        H7C2N1            H3        H7C2N1H3
H8        C5        H8C5              C2        H8C5C2            N1        H8C5C2N1
H9        C5        H9C5              C2        H9C5C2            N1        H9C5C2N1
H10       C5        H10C5             C2        H10C5C2           N1        H10C5C2N1

VARIABLES:
C2N1       = 1.44084286        H3N1       = 1.00062635        H4N1       = 0.99972788
C5C2       = 1.51781924        H6C2       = 1.08653973        H7C2       = 1.09386650
H8C5       = 1.08365959        H9C5       = 1.08370744        H10C5      = 1.08374183
H3N1C2     =  110.26867        H4N1C2     =  111.39727        C5C2N1     =  110.31733
H6C2N1     =  109.68864        H7C2N1     =  114.69657        H8C5C2     =  109.47840
H9C5C2     =  109.47751        H10C5C2    =  109.47686        H4N1C2H3   =  120.98509
C5C2N1H3   =   38.90011        H6C2N1H3   =  159.58235        H7C2N1H3   =  -80.39362
H8C5C2N1   =  180.00000        H9C5C2N1   =   59.99718        H10C5C2N1  =  -59.99363

Cartesian coordinates for: 7(6(6,1,1)1()1())

--------------------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                        X
              Y              Z
--------------------------------------------------------------------------------------------------------------

   1 N1         7   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   2 C2         6   0.00000000     0.00000000     1.44084286               0.00000000
        0.00000000     2.72279820
   3 H3         1   0.93866607     0.00000000    -0.34663973               1.77382166
        0.00000000    -0.65505411
   4 H4         1  -0.47920001    -0.79799306    -0.36473314              -0.90555672
       -1.50798822    -0.68924570
   5 C5         6   1.10773910     0.89383628     1.96785902               2.09332336
        1.68910566     3.71871433
   6 H6         1  -0.95874622     0.35689075     1.80690743              -1.81176764
        0.67442573     3.41455992
   7 H7         1   0.16584623    -0.97987823     1.89787416               0.31340393
       -1.85170135     3.58646212
```

```
   8 H8          1   1.09539118     0.88387273     3.05140244               2.06998919
        1.67027728     5.76631450
   9 H9          1   0.95389867     1.90660309     1.61422314               1.80260710
        3.60295741     3.05043943
  10 H10         1   2.06514427     0.52946711     1.61415464               3.90255681
        1.00054777     3.05030997
-----------------------------------------------------------------------------------------------

Nuclear repulsion energy:      83.927218609

The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

    1059473 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       36457 IJKJ:       37167 IJJL:       37781 IIKK:        1748
 IJJJ:         618 IIIL:         618 IIII:          18 IJKL:      945066
Number of integrals in INCORE buffers:
IIKK:         891 IJJL:       19339 IJKJ:       19030
IIKL:       18869 IJKL:      330019

     337774 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       19034 IJKJ:       19531 IJJL:       19062 IIKK:        1589
 IJJJ:         666 IIIL:         669 IIII:          41 IJKL:      277182
Number of integrals in INCORE buffers:
IIKK:        1711 IJJL:       29218 IJKJ:       29080
IIKL:       28741 IJKL:      427733
 TOTAL OF    1397247 TWO-ELECTRON INTEGRALS CALCULATED (RAW)



CLOSED SHELL SCF               Nuclear Repulsion Energy is     83.927218609 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE    ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:    1       -217.903421853      -133.976203244
 SCF_CYCLE:    2       -218.150803679      -134.223585070   1.47540E-02
 SCF_CYCLE:    3       -218.169358850      -134.242140241   4.55808E-03
 SCF_CYCLE:    4       -218.171840447      -134.244621838   2.08155E-03
 SCF_CYCLE:    5       -218.172279553      -134.245060944   9.12817E-04
 SCF_CYCLE:    6       -218.172372771      -134.245154162   4.87492E-04
 SCF_CYCLE:    7       -218.172394640      -134.245176031   2.28184E-04
 SCF_CYCLE:    8       -218.172400105      -134.245181496   1.25070E-04
 SCF_CYCLE:    9       -218.172404659      -134.245186051                     4-POINT
 SCF_CYCLE:   10       -218.172402051      -134.245183442   9.80013E-05
At termination total energy is      -134.245183  Hartrees

Energy components:
Kinetic =                   134.181219713
Potential =                -481.008943387
Kinetic + Potential =      -346.827723674
Coulomb repulsion =         148.186452299
Exchange =                  -19.531130678
Coulomb+Exchange =          128.655321622
Nuclear =                    83.927218609
Total electronic =         -218.172402052
Total energy =             -134.245183443

Virial =                     2.000476697
```

```
Atom            J             K              Vee            JHF            KHF
    VeeHF
       1      38.966053     -1.235830     37.730223     44.377644     -6.647421
    37.730223
Atom           Coulomb
       1      46.801541
J_total       46.801541

Atom        Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1     17.933375     18.175964     17.900265     54.009604

Ttotal:      17.933375     18.175964     17.900265     54.009604

Atom           Vne
       1    -169.306443
Vne_total:         -169.306443


****************************************
 The fragment with symbol 7(6(6,1,1)1()1())
 has been added to the database with index #          115
****************************************

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    7.1649528615
Pure Exchange, K ( 2K_ab)         =   -1.2358296867
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -6.6474207082
Kinetic energy Numerical, T      =   54.0096039530
Potential Energy Analytical, Vne = -167.9539733452
Potential Energy Numerical, Vne  = -169.3064434906
Coulomb Energy Anal/Num, Vee     =   46.8015413684
Pure Coulomb, J ( 4J_ab+ Jaa)     =   38.9660529960
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =   44.3776440175
Jaa = Kaa                        =    5.4115910215
Coulomb Numerically Over A       =   27.8333661426


PROGRAM> end of inputs


Program terminated normally


Job: RUN_FRAG_0115 ended on :24-Aug-18 at 15:37:05
User: ibrahim
Cpu     time:  00h00m12s91c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m13s00c



*** RUNing the inputfile :: INPUT_0116.dat


****************************************
 Working on symbol 6(7(1,1)6(8,8)1()1())
 IndexNUM          116
****************************************


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
```

```
Point group: C1
Cartesian coordinates for: 6(7(1,1)6(8,8)1()1())

--------------------------------------------------------------------------------------------------------

                              COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                          X
             Y              Z
--------------------------------------------------------------------------------------------------------

   1 C          6  -0.00000000     0.00000000     0.00000000              -0.00000000
        0.00000000     0.00000000
   2 N          7   0.00000000    -0.00000000     1.44084286               0.00000000
       -0.00000000     2.72279820
   3 C          6   0.00000000     1.42338653    -0.52701616               0.00000000
        2.68981052    -0.99591613
   4 H          1   0.87980599    -0.52202178    -0.36606457               1.66259225
       -0.98647812    -0.69176172
   5 H          1  -0.86672789    -0.48626030    -0.45703130              -1.63787822
       -0.91889872    -0.86366392
   6 O          8   0.37919230     1.56620070    -1.79700972               0.71656954
        2.95969016    -3.39585598
   7 O          8  -0.36180632     2.35288801     0.10564188              -0.68371481
        4.44631363     0.19963421
   8 H          1  -0.58944902     0.73050930     1.78748259              -1.11389713
        1.38046242     3.37785231
   9 H          1  -0.32011104    -0.87404490     1.80557600              -0.60492216
       -1.65170536     3.41204390
  10 H          1   0.31941883     2.53932814    -1.96545672               0.60361407
        4.79863437    -3.71417465
--------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:     181.273968331


Distance Matrix for: 6(7(1,1)6(8,8)1()1())
             1           2           3           4           5           6
     7           8           9          10
   1      0.000000    1.440843    1.517819    1.086540    1.093866    2.413714
    2.382886    2.018957    2.031386    3.226955
   2      1.440843    0.000000    2.428682    2.076410    2.142333    3.616692
    2.729423    1.000626    0.999728    4.260645
   3      1.517819    2.428682    0.000000    2.141163    2.098301    1.333067
    1.181157    2.486852    3.289627    1.848368
   4      1.086540    2.076410    2.141163    0.000000    1.749267    2.580483
    3.166893    2.892285    2.505942    3.499136
   5      1.093866    2.142333    2.098301    1.749267    0.000000    2.749628
    2.938079    2.568123    2.359779    3.582803
   6      2.413714    3.616692    1.333067    2.580483    2.749628    0.000000
    2.188159    3.805947    4.407091    0.989406
   7      2.382886    2.729423    1.181157    3.166893    2.938079    2.188159
    0.000000    2.347876    3.647549    2.188213
   8      2.018957    1.000626    2.486852    2.892285    2.568123    3.805947
    2.347876    0.000000    1.627103    4.264085
   9      2.031386    0.999728    3.289627    2.505942    2.359779    4.407091
    3.647549    1.627103    0.000000    5.126480
  10      3.226955    4.260645    1.848368    3.499136    3.582803    0.989406
    2.188213    4.264085    5.126480    0.000000

Charge=    0, Number of electrons=   40

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
```

```
6-31G(d) Basis Set - Total number of basis functions:       85

 Partitioning scheme set to: BECKE
Free format Z-Matrix for: 6(7(1,1)6(8,8)1()1())
C1
N2       C1       N2C1
C3       C1       C3C1             N2       C3C1N2
H4       C1       H4C1             N2       H4C1N2           C3       H4C1N2C3
H5       C1       H5C1             N2       H5C1N2           C3       H5C1N2C3
O6       C3       O6C3             C1       O6C3C1           N2       O6C3C1N2
O7       C3       O7C3             C1       O7C3C1           N2       O7C3C1N2
H8       N2       H8N2             C1       H8N2C1           C3       H8N2C1C3
H9       N2       H9N2             C1       H9N2C1           C3       H9N2C1C3
H10      O6       H10O6            C3       H10O6C3          C1       H10O6C3C1

VARIABLES:
N2C1      = 1.44084286        C3C1      = 1.51781924        H4C1      = 1.08653973
H5C1      = 1.09386650        O6C3      = 1.33306652        O7C3      = 1.18115749
H8N2      = 1.00062635        H9N2      = 0.99972788        H10O6     = 0.98940602
C3C1N2    =  110.31733        H4C1N2    =  109.68864        H5C1N2    =  114.69657
O6C3C1    =  115.54740        O7C3C1    =  123.50442        H8N2C1    =  110.26867
H9N2C1    =  111.39727        H10O6C3   =  104.50001        H4C1N2C3  =  120.68224
H5C1N2C3  = -119.29373        O6C3C1N2  =  161.62252        O7C3C1N2  =  -21.55248
H8N2C1C3  =   38.90011        H9N2C1C3  =  159.88520        H10O6C3C1 = -180.00000

Cartesian coordinates for: 6(7(1,1)6(8,8)1()1())

--------------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                       X
             Y             Z
--------------------------------------------------------------------------------------------------------

   1 C1         6   0.00000000     0.00000000     0.00000000               0.00000000
       0.00000000     0.00000000
   2 N2         7   0.00000000     0.00000000     1.44084286               0.00000000
       0.00000000     2.72279820
   3 C3         6   1.42338653     0.00000000    -0.52701616               2.68981052
       0.00000000    -0.99591613
   4 H4         1  -0.52202178    -0.87980599    -0.36606457              -0.98647812
      -1.66259225    -0.69176172
   5 H5         1  -0.48626030     0.86672789    -0.45703130              -0.91889872
       1.63787822    -0.86366392
   6 O6         8   1.56620070    -0.37919230    -1.79700972               2.95969016
      -0.71656954    -3.39585598
   7 O7         8   2.35288801     0.36180632     0.10564188               4.44631363
       0.68371481     0.19963421
   8 H8         1   0.73050930     0.58944902     1.78748259               1.38046242
       1.11389713     3.37785231
   9 H9         1  -0.87404490     0.32011104     1.80557600              -1.65170536
       0.60492216     3.41204390
  10 H10        1   2.53932814    -0.31941883    -1.96545672               4.79863437
      -0.60361407    -3.71417465
--------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:     181.273968331

The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16
```

```
    4104066 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:     113140 IJKJ:     113572 IJJL:     114756 IIKK:       3793
 IJJJ:       1462 IIIL:       1470 IIII:         30 IJKL:    3755843
Number of integrals in INCORE buffers:
IIKK:       2085 IJJL:      64260 IJKJ:      63526
IIKL:      64095 IJKL:    1534076

     930964 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      43052 IJKJ:      43862 IJJL:      43150 IIKK:       2734
 IJJJ:       1151 IIIL:       1157 IIII:         55 IJKL:     795803
Number of integrals in INCORE buffers:
IIKK:       3570 IJJL:      88256 IJKJ:      87663
IIKL:      88149 IJKL:     269361
Number of buffers:
IJJL:          0 IJKJ:          0
IIKL:          0 IJKL:          1
 TOTAL OF    5035030 TWO-ELECTRON INTEGRALS CALCULATED (RAW)



CLOSED SHELL SCF               Nuclear Repulsion Energy is    181.273968331 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE    ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:    1       -463.075382663      -281.801414332
 SCF_CYCLE:    2       -463.938846750      -282.664878419   2.63503E-02
 SCF_CYCLE:    3       -464.033491357      -282.759523025   1.04825E-02
 SCF_CYCLE:    4       -464.061188697      -282.787220365   8.09175E-03
 SCF_CYCLE:    5       -464.077105997      -282.803137666   6.06597E-03
 SCF_CYCLE:    6       -464.085279580      -282.811311249   4.81216E-03
 SCF_CYCLE:    7       -464.098916367      -282.824948035                 4-POINT
 SCF_CYCLE:    8       -464.098969593      -282.825001262   2.15105E-03
 SCF_CYCLE:    9       -464.098999780      -282.825031449   1.85118E-04
 SCF_CYCLE:   10       -464.099011460      -282.825043129   1.17512E-04
 SCF_CYCLE:   11       -464.099021079      -282.825052747                 4-POINT
 SCF_CYCLE:   12       -464.099019232      -282.825050900   4.77928E-05
At termination total energy is      -282.825051  Hartrees

Energy components:
Kinetic =                   282.246594234
Potential =                -1028.372855515
Kinetic + Potential =       -746.126261281
Coulomb repulsion =          317.344749503
Exchange =                   -35.317507461
Coulomb+Exchange =           282.027242042
Nuclear =                    181.273968331
Total electronic =          -464.099019239
Total energy =              -282.825050908

Virial =                      2.002049473

Atom           J             K             Vee           JHF           KHF
    VeeHF
       1      43.478567     -0.933213     42.545354     47.819244     -5.273891
    42.545354
Atom           Coulomb
       1      48.232671
J_total       48.232671

Atom         Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1      12.644472     12.655040     12.669234     37.968746
```

```
Ttotal:      12.644472     12.655040     12.669234     37.968746

Atom            Vne
       1    -149.312739
Vne_total:          -149.312739


****************************************
 The fragment with symbol 6(7(1,1)6(8,8)1()1())
 has been added to the database with index #          116
****************************************

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++++
Number of Electrons, N            =    6.1012052215
Pure Exchange, K ( 2K_ab)          =   -0.9332131891
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -5.2738907810
Kinetic energy Numerical, T       =   37.9687460325
Potential Energy Analytical, Vne = -147.1754484683
Potential Energy Numerical, Vne  = -149.3127391196
Coulomb Energy Anal/Num, Vee      =   48.2326712819
Pure Coulomb, J ( 4J_ab+ Jaa)     =   43.4785667800
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =   47.8192443719
Jaa = Kaa                         =    4.3406775919
Coulomb Numerically Over A        =   19.3994839860


PROGRAM> end of inputs


Program terminated normally


Job: RUN_FRAG_0116 ended on :24-Aug-18 at 15:37:44
User: ibrahim
Cpu     time:  00h00m38s54c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m39s00c



*** RUNing the inputfile :: INPUT_0117.dat


****************************************
 Working on symbol 6(8(1)8()6(7,1,1))
 IndexNUM          117
****************************************


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: 6(8(1)8()6(7,1,1))
-------------------------------------------------------------------------------------------------------------

                              COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                        X
             Y             Z
-------------------------------------------------------------------------------------------------------------
```

```
   1 C           6  -0.00000000     -0.00000000      0.00000000               -0.00000000
        -0.00000000      0.00000000
   2 O           8  -0.00000000      0.00000000      1.33306652               -0.00000000
         0.00000000      2.51913045
   3 O           8   0.00000000      1.01381600     -0.60606116                0.00000000
         1.91583445     -1.14528952
   4 C           6   0.07368323     -1.36743652     -0.65457115                0.13924111
        -2.58408033     -1.23696012
   5 N           7  -0.35716835     -1.28888329     -2.02724171               -0.67495031
        -2.43563626     -3.83093134
   6 H           1  -0.03707255     -0.87584463      1.69369704               -0.07005695
        -1.65510637      3.20062332
   7 H           1  -0.55932392     -2.06380818     -0.11149008               -1.05696894
        -3.90003195     -0.21068570
   8 H           1   1.10434945     -1.70738451     -0.51779972                2.08691785
        -3.22648888     -0.97849960
   9 H           1  -0.29008929     -2.21219942     -2.41194164               -0.54818927
        -4.18045073     -4.55790880
  10 H           1   0.30676556     -0.71652891     -2.51368937                0.57970285
        -1.35404331     -4.75018412
-----------------------------------------------------------------------------------------------

Nuclear repulsion energy:     181.310954311


Distance Matrix for: 6(8(1)8()6(7,1,1))
             1           2           3           4           5           6
     7           8           9          10
   1      0.000000    1.333067    1.181157    1.517819    2.428682    1.907115
    2.141163    2.098301    3.285642    2.631759
   2      1.333067    0.000000    2.188159    2.413714    3.616692    0.947910
    2.580483    2.749628    4.359250    3.924927
   3      1.181157    2.188159    0.000000    2.382886    2.729423    2.976757
    3.166893    2.938079    3.708441    2.593693
   4      1.517819    2.413714    2.382886    0.000000    1.440843    2.401727
    1.086540    1.093866    1.983508    1.983514
   5      2.428682    3.616692    2.729423    1.440843    0.000000    3.757452
    2.076410    2.142333    1.002500    1.002511
   6      1.907115    0.947910    2.976757    2.401727    3.757452    0.000000
    2.223219    2.623932    4.325058    4.224418
   7      2.141163    2.580483    3.166893    1.086540    2.076410    2.223219
    0.000000    1.749267    2.320902    2.887184
   8      2.098301    2.749628    2.938079    1.093866    2.142333    2.623932
    1.749267    0.000000    2.405633    2.366751
   9      3.285642    4.359250    3.708441    1.983508    1.002500    4.325058
    2.320902    2.405633    0.000000    1.613573
  10      2.631759    3.924927    2.593693    1.983514    1.002511    4.224418
    2.887184    2.366751    1.613573    0.000000

Charge=    0, Number of electrons=   40

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       85

 Partitioning scheme set to: BECKE
Free format Z-Matrix for: 6(8(1)8()6(7,1,1))
C1
O2       C1       O2C1
O3       C1       O3C1             O2       O3C1O2
C4       C1       C4C1             O2       C4C1O2           O3       C4C1O2O3
N5       C4       N5C4             C1       N5C4C1           O2       N5C4C1O2
H6       O2       H6O2             C1       H6O2C1           O3       H6O2C1O3
H7       C4       H7C4             C1       H7C4C1           O2       H7C4C1O2
```

```
H8        C4        H8C4              C1        H8C4C1            O2        H8C4C1O2
H9        N5        H9N5              C4        H9N5C4            C1        H9N5C4C1
H10       N5        H10N5             C4        H10N5C4           C1        H10N5C4C1

VARIABLES:
O2C1      = 1.33306652        O3C1      = 1.18115749        C4C1      = 1.51781924
N5C4      = 1.44084286        H6O2      = 0.94790958        H7C4      = 1.08653973
H8C4      = 1.09386650        H9N5      = 1.00250002        H10N5     = 1.00251139
O3C1O2    =  120.87106        C4C1O2    =  115.54740        N5C4C1    =  110.31733
H6O2C1    =  112.36145        H7C4C1    =  109.48571        H8C4C1    =  105.77163
H9N5C4    =  107.17931        H10N5C4   =  107.17911        C4C1O2O3  =  176.91565
N5C4C1O2  =  161.62252        H6O2C1O3  =  177.57624        H7C4C1O2  =   40.81887
H8C4C1O2  =  -73.79898        H9N5C4C1  = -180.00000        H10N5C4C1 =   65.22155

Cartesian coordinates for: 6(8(1)8()6(7,1,1))

------------------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                       X
             Y             Z
------------------------------------------------------------------------------------------------------------

   1 C1         6   0.00000000     0.00000000     0.00000000               0.00000000
       0.00000000     0.00000000
   2 O2         8   0.00000000     0.00000000     1.33306652               0.00000000
       0.00000000     2.51913045
   3 O3         8   1.01381600     0.00000000    -0.60606116               1.91583445
       0.00000000    -1.14528952
   4 C4         6  -1.36743652    -0.07368323    -0.65457115              -2.58408033
      -0.13924111    -1.23696012
   5 N5         7  -1.28888329     0.35716835    -2.02724171              -2.43563626
       0.67495031    -3.83093134
   6 H6         1  -0.87584463     0.03707255     1.69369704              -1.65510637
       0.07005695     3.20062332
   7 H7         1  -2.06380818     0.55932392    -0.11149008              -3.90003195
       1.05696894    -0.21068570
   8 H8         1  -1.70738451    -1.10434945    -0.51779972              -3.22648888
      -2.08691785    -0.97849960
   9 H9         1  -2.21219942     0.29008929    -2.41194164              -4.18045073
       0.54818927    -4.55790880
  10 H10        1  -0.71652891    -0.30676556    -2.51368937              -1.35404331
      -0.57970285    -4.75018412
------------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:     181.310954311

The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

    4014630 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:     111868 IJKJ:     112757 IJJL:     113371 IIKK:       3774
 IJJJ:       1448 IIIL:       1461 IIII:         30 IJKL:    3669921
Number of integrals in INCORE buffers:
IIKK:       2085 IJJL:      63881 IJKJ:      63331
IIKL:      63943 IJKL:    1515284

     918968 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      42900 IJKJ:      43766 IJJL:      43075 IIKK:       2721
```

```
 IJJJ:       1151 IIIL:       1156 IIII:         55 IJKL:     784144
Number of integrals in INCORE buffers:
IIKK:       3570 IJJL:      87863 IJKJ:      87429
IIKL:      88004 IJKL:     247089
Number of buffers:
IJJL:          0 IJKJ:          0
IIKL:          0 IJKL:          1
 TOTAL OF    4933598 TWO-ELECTRON INTEGRALS CALCULATED (RAW)



CLOSED SHELL SCF               Nuclear Repulsion Energy is    181.310954311 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE   ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:   1       -463.088972091      -281.778017781
 SCF_CYCLE:   2       -463.951585158      -282.640630848   2.82205E-02
 SCF_CYCLE:   3       -464.049586122      -282.738631811   1.12631E-02
 SCF_CYCLE:   4       -464.079700052      -282.768745742   8.95038E-03
 SCF_CYCLE:   5       -464.098442900      -282.787488590   6.79317E-03
 SCF_CYCLE:   6       -464.108208348      -282.797254037   5.37894E-03
 SCF_CYCLE:   7       -464.123557409      -282.812603099                    4-POINT
 SCF_CYCLE:   8       -464.125156274      -282.814201963   2.45268E-03
 SCF_CYCLE:   9       -464.125167209      -282.814212898   1.14106E-04
 SCF_CYCLE:  10       -464.125171428      -282.814217117   7.40949E-05
 SCF_CYCLE:  11       -464.125173119      -282.814218809   4.69002E-05
 SCF_CYCLE:  12       -464.125174638      -282.814220328                    4-POINT
 SCF_CYCLE:  13       -464.125174309      -282.814219999   1.92028E-05
At termination total energy is      -282.814220  Hartrees

Energy components:
Kinetic =                  282.320136597
Potential =              -1028.401570033
Kinetic + Potential =     -746.081433436
Coulomb repulsion =        317.284416897
Exchange =                 -35.328157773
Coulomb+Exchange =         281.956259124
Nuclear =                  181.310954311
Total electronic =        -464.125174312
Total energy =            -282.814220001

Virial =                    2.001750082

Atom           J             K             Vee           JHF           KHF
    VeeHF
       1      44.469534     -0.959412     43.510122     48.885473     -5.375351
    43.510122
Atom           Coulomb
       1      53.046478
J_total       53.046478

Atom        Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1     12.923600     12.858647     12.914850     38.697097

Ttotal:      12.923600     12.858647     12.914850     38.697097

Atom           Vne
       1    -159.522455
Vne_total:         -159.522455


****************************************
 The fragment with symbol 6(8(1)8()6(7,1,1))
 has been added to the database with index #          117
```

```
****************************************

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    6.1259220110
Pure Exchange, K ( 2K_ab)         =   -0.9594119713
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -5.3753513417
Kinetic energy Numerical, T      =   38.6970966577
Potential Energy Analytical, Vne = -156.5604713149
Potential Energy Numerical, Vne  = -159.5224545210
Coulomb Energy Anal/Num, Vee     =   53.0464780861
Pure Coulomb, J ( 4J_ab+ Jaa)     =   44.4695336088
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =   48.8854729793
Jaa = Kaa                        =    4.4159393704
Coulomb Numerically Over A       =   19.3435272796


PROGRAM> end of inputs


Program terminated normally


Job: RUN_FRAG_0117 ended on :24-Aug-18 at 15:38:24
User: ibrahim
Cpu     time:  00h00m39s51c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m40s00c



*** RUNing the inputfile :: INPUT_0118.dat


****************************************
 Working on symbol 8(6(8,6))
 IndexNUM          118
****************************************


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: 8(6(8,6))

-------------------------------------------------------------------------------------------------------

                             COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                       X
             Y             Z
-------------------------------------------------------------------------------------------------------

   1 O          8  -0.00000000     0.00000000    -0.00000000              -0.00000000
        0.00000000    -0.00000000
   2 C          6  -0.00000000    -0.00000000     1.18115749              -0.00000000
       -0.00000000     2.23206401
   3 O          8  -0.00000000     1.14420319     1.86516436              -0.00000000
        2.16223051     3.52464956
   4 C          6   0.07368323    -1.26347662     2.01899570               0.13924111
       -2.38762460     3.81534865
   5 H          1   0.00000000     0.86533235     2.81445637               0.00000000
        1.63524104     5.31855135
```

```
   6 H           1   0.05839270     -1.00128368      3.07034682               0.11034620
       -1.89215179      5.80211417
   7 H           1   0.99092661     -1.79429459      1.79245948               1.87257978
       -3.39072511      3.38725726
   8 H           1  -0.77563740     -1.89735718      1.79239856              -1.46574214
       -3.58548516      3.38714215
---------------------------------------------------------------------------------------------

Nuclear repulsion energy:     122.224021925


Distance Matrix for: 8(6(8,6))
             1           2           3           4           5           6
     7           8
   1      0.000000    1.181157    2.188159    2.382886    2.944480    3.230017
    2.722928    2.722916
   2      1.181157    0.000000    1.333067    1.517819    1.848368    2.138928
    2.138953    2.138970
   3      2.188159    1.333067    0.000000    2.413714    0.989406    2.461501
    3.101933    3.139745
   4      2.382886    1.517819    2.413714    0.000000    2.273767    1.083660
    1.083707    1.083742
   5      2.944480    1.848368    0.989406    2.273767    0.000000    1.884979
    3.016625    3.046091
   6      3.230017    2.138928    2.461501    1.083660    1.884979    0.000000
    1.769599    1.769663
   7      2.722928    2.138953    3.101933    1.083707    3.016625    1.769599
    0.000000    1.769568
   8      2.722916    2.138970    3.139745    1.083742    3.046091    1.769663
    1.769568    0.000000

Charge=    0, Number of electrons=   32

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       68

 Partitioning scheme set to: BECKE
Free format Z-Matrix for: 8(6(8,6))
O1
C2       O1       C2O1
O3       C2       O3C2             O1       O3C2O1
C4       C2       C4C2             O1       C4C2O1           O3       C4C2O1O3
H5       O3       H5O3             C2       H5O3C2           O1       H5O3C2O1
H6       C4       H6C4             C2       H6C4C2           O1       H6C4C2O1
H7       C4       H7C4             C2       H7C4C2           O1       H7C4C2O1
H8       C4       H8C4             C2       H8C4C2           O1       H8C4C2O1

VARIABLES:
C2O1     = 1.18115749         O3C2     = 1.33306652         C4C2     = 1.51781924
H5O3     = 0.98940602         H6C4     = 1.08365959         H7C4     = 1.08370744
H8C4     = 1.08374183         O3C2O1   =  120.87106         C4C2O1   =  123.50442
H5O3C2   =  104.50001         H6C4C2   =  109.47840         H7C4C2   =  109.47751
H8C4C2   =  109.47686         C4C2O1O3 = -176.66241         H5O3C2O1 = -180.00000
H6C4C2O1 = -180.00000         H7C4C2O1 =   59.99718         H8C4C2O1 =  -59.99363

Cartesian coordinates for: 8(6(8,6))

---------------------------------------------------------------------------------------------

                           COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                       X
             Y             Z
```

```
-----------------------------------------------------------------------------------------------------------

   1 O1         8   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   2 C2         6   0.00000000     0.00000000     1.18115749               0.00000000
        0.00000000     2.23206401
   3 O3         8   1.14420319     0.00000000     1.86516436               2.16223051
        0.00000000     3.52464956
   4 C4         6  -1.26347662    -0.07368323     2.01899570              -2.38762460
       -0.13924111     3.81534865
   5 H5         1   0.86533235     0.00000000     2.81445637               1.63524104
        0.00000000     5.31855135
   6 H6         1  -1.00128368    -0.05839270     3.07034682              -1.89215179
       -0.11034620     5.80211417
   7 H7         1  -1.79429459    -0.99092661     1.79245948              -3.39072511
       -1.87257978     3.38725726
   8 H8         1  -1.89735718     0.77563740     1.79239856              -3.58548516
        1.46574214     3.38714215
-----------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:     122.224021925

The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

    1737317 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       55477 IJKJ:       57072 IJJL:       57795 IIKK:        2543
 IJJJ:         863 IIIL:         872 IIII:          24 IJKL:     1562671
Number of integrals in INCORE buffers:
IIKK:        1332 IJJL:       30556 IJKJ:       30159
IIKL:       29823 IJKL:      578843

     389020 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       21017 IJKJ:       22145 IJJL:       21363 IIKK:        1797
 IJJJ:         688 IIIL:         691 IIII:          44 IJKL:      321275
Number of integrals in INCORE buffers:
IIKK:        2278 IJJL:       42002 IJKJ:       41919
IIKL:       41149 IJKL:      698709
 TOTAL OF    2126337 TWO-ELECTRON INTEGRALS CALCULATED (RAW)



CLOSED SHELL SCF               Nuclear Repulsion Energy is    122.224021925 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE    ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:    1       -349.147748946      -226.923727021
 SCF_CYCLE:    2       -349.864153613      -227.640131688   3.23251E-02
 SCF_CYCLE:    3       -349.947256672      -227.723234748   1.31011E-02
 SCF_CYCLE:    4       -349.974926317      -227.750904392   1.04327E-02
 SCF_CYCLE:    5       -349.992426837      -227.768404912   7.92034E-03
 SCF_CYCLE:    6       -350.001568398      -227.777546473   6.22572E-03
 SCF_CYCLE:    7       -350.018394724      -227.794372799                    4-POINT
 SCF_CYCLE:    8       -350.016674930      -227.792653005   2.80451E-03
 SCF_CYCLE:    9       -350.016704330      -227.792682405   2.31882E-04
 SCF_CYCLE:   10       -350.016716162      -227.792694238   1.55235E-04
 SCF_CYCLE:   11       -350.016725932      -227.792704008                    4-POINT
 SCF_CYCLE:   12       -350.016724621      -227.792702696   6.16034E-05
At termination total energy is      -227.792703  Hartrees
```

```
Energy components:
Kinetic =                    227.331216809
Potential =                 -780.930714443
Kinetic + Potential =       -553.599497634
Coulomb repulsion =          231.888100768
Exchange =                   -28.305327756
Coulomb+Exchange =           203.582773012
Nuclear =                    122.224021925
Total electronic =          -350.016724623
Total energy =              -227.792702698

Virial =                      2.002030015

Atom            J             K             Vee           JHF           KHF
    VeeHF
       1      60.217314     -1.863484     58.353830     66.554675     -8.200845
    58.353830
Atom            Coulomb
       1      63.929851
J_total       63.929851

Atom        Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1     25.117564     24.651336     24.456601     74.225501

Ttotal:      25.117564     24.651336     24.456601     74.225501

Atom           Vne
       1    -231.797194
Vne_total:         -231.797194


****************************************
 The fragment with symbol 8(6(8,6))
 has been added to the database with index #          118
****************************************

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    8.1424852160
Pure Exchange, K ( 2K_ab)         =   -1.8634842591
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -8.2008448054
Kinetic energy Numerical, T      =   74.2255012751
Potential Energy Analytical, Vne = -232.1835859918
Potential Energy Numerical, Vne  = -231.7971941706
Coulomb Energy Anal/Num, Vee     =   63.9298510309
Pure Coulomb, J ( 4J_ab+ Jaa)     =   60.2173142797
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =   66.5546748260
Jaa = Kaa                        =    6.3373605464
Coulomb Numerically Over A       =   37.9275965221


PROGRAM> end of inputs


Program terminated normally


Job: RUN_FRAG_0118 ended on :24-Aug-18 at 15:38:44
User: ibrahim
Cpu     time:  00h00m19s85c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m20s00c
```

```
*** RUNing the inputfile :: INPUT_0119.dat


****************************************
 Working on symbol 1(6(7,6,1))
 IndexNUM          119
****************************************


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: 1(6(7,6,1))

-------------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                       X
             Y             Z
-------------------------------------------------------------------------------------------------------

   1 H          1  -0.00000000     0.00000000    -0.00000000              -0.00000000
        0.00000000    -0.00000000
   2 C          6  -0.00000000    -0.00000000     1.08653973              -0.00000000
       -0.00000000     2.05326237
   3 N          7   0.00000000     1.35660739     1.57197206               0.00000000
        2.56361623     2.97059646
   4 C          6   1.22412734    -0.74090902     1.59284135               2.31326525
       -1.40011502     3.01003370
   5 H          1  -0.86045971    -0.59784588     1.40075830              -1.62603308
       -1.12976490     2.64704936
   6 H          1   0.00000000     1.31271584     2.57351079               0.00000000
        2.48067324     4.86323023
   7 H          1  -0.86960710     1.77063919     1.29378103              -1.64331913
        3.34602290     2.44489164
   8 H          1   1.22400910    -0.74083745     2.67650093               2.31304180
       -1.39997979     5.05785337
   9 H          1   2.11948787    -0.24860401     1.23174979               4.00525131
       -0.46979346     2.32766958
  10 H          1   1.20323095    -1.76248761     1.23168092               2.27377680
       -3.33061865     2.32753944
-------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:      83.935233098


Distance Matrix for: 1(6(7,6,1))
             1           2           3           4           5           6
     7           8           9          10
   1      0.000000    1.086540    2.076410    2.141163    1.749267    2.888976
    2.359078    3.034913    2.463989    2.463973
   2      1.086540    0.000000    1.440843    1.517819    1.093867    1.983508
    1.983514    2.138928    2.138953    2.138970
   3      2.076410    1.440843    0.000000    2.428682    2.142333    1.002500
    1.002511    2.667856    2.680426    3.360404
   4      2.141163    1.517819    2.428682    0.000000    2.098301    2.584100
    3.283449    1.083660    1.083707    1.083742
   5      1.749267    1.093867    2.142333    2.098301    0.000000    2.401247
    2.370917    2.448056    3.005099    2.375668
   6      2.888976    1.983508    1.002500    2.584100    2.401247    0.000000
```

```
    1.613573     2.392882     2.954703     3.564428
   7       2.359078     1.983514     1.002511     3.283449     2.370917     1.613573
    0.000000     3.550022     3.607753     4.096767
   8       3.034913     2.138928     2.667856     1.083660     2.448056     2.392882
    3.550022     0.000000     1.769599     1.769663
   9       2.463989     2.138953     2.680426     1.083707     3.005099     2.954703
    3.607753     1.769599     0.000000     1.769568
  10       2.463973     2.138970     3.360404     1.083742     2.375668     3.564428
    4.096767     1.769663     1.769568     0.000000

Charge=    0, Number of electrons=   26

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       59

 Partitioning scheme set to: BECKE
Free format Z-Matrix for: 1(6(7,6,1))
H1
C2        H1        C2H1
N3        C2        N3C2              H1        N3C2H1
C4        C2        C4C2              H1        C4C2H1            N3        C4C2H1N3
H5        C2        H5C2              H1        H5C2H1            N3        H5C2H1N3
H6        N3        H6N3              C2        H6N3C2            H1        H6N3C2H1
H7        N3        H7N3              C2        H7N3C2            H1        H7N3C2H1
H8        C4        H8C4              C2        H8C4C2            H1        H8C4C2H1
H9        C4        H9C4              C2        H9C4C2            H1        H9C4C2H1
H10       C4        H10C4             C2        H10C4C2           H1        H10C4C2H1

VARIABLES:
C2H1       = 1.08653973         N3C2       = 1.44084286         C4C2       = 1.51781924
H5C2       = 1.09386650         H6N3       = 1.00250002         H7N3       = 1.00251139
H8C4       = 1.08365959         H9C4       = 1.08370744         H10C4      = 1.08374183
N3C2H1     =  109.68864         C4C2H1     =  109.48571         H5C2H1     =  106.69369
H6N3C2     =  107.17931         H7N3C2     =  107.17911         H8C4C2     =  109.47840
H9C4C2     =  109.47751         H10C4C2    =  109.47686         C4C2H1N3   = -121.18463
H5C2H1N3   =  124.79150         H6N3C2H1   = -180.00000         H7N3C2H1   =   65.22155
H8C4C2H1   =  180.00000         H9C4C2H1   =   59.99718         H10C4C2H1  =  -59.99363

Cartesian coordinates for: 1(6(7,6,1))

--------------------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                        X
              Y              Z
--------------------------------------------------------------------------------------------------------------

   1 H1         1   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   2 C2         6   0.00000000     0.00000000     1.08653973               0.00000000
        0.00000000     2.05326237
   3 N3         7   1.35660739     0.00000000     1.57197206               2.56361623
        0.00000000     2.97059646
   4 C4         6  -0.74090902    -1.22412734     1.59284135              -1.40011502
       -2.31326525     3.01003370
   5 H5         1  -0.59784588     0.86045971     1.40075830              -1.12976490
        1.62603308     2.64704936
   6 H6         1   1.31271584     0.00000000     2.57351079               2.48067324
        0.00000000     4.86323023
   7 H7         1   1.77063919     0.86960710     1.29378103               3.34602290
        1.64331913     2.44489164
   8 H8         1  -0.74083745    -1.22400910     2.67650093              -1.39997979
```

```
        -2.31304180     5.05785337
   9 H9         1   -0.24860401    -2.11948787     1.23174979              -0.46979346
        -4.00525131     2.32766958
  10 H10        1   -1.76248761    -1.20323095     1.23168092              -3.33061865
        -2.27377680     2.32753944
--------------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:      83.935233098

The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

    1054921 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       36877 IJKJ:       37743 IJJL:       37996 IIKK:        1743
 IJJJ:         634 IIIL:         637 IIII:          18 IJKL:      939273
Number of integrals in INCORE buffers:
IIKK:         891 IJJL:       19581 IJKJ:       19420
IIKL:       19181 IJKL:      331066

     335016 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       18959 IJKJ:       19633 IJJL:       19108 IIKK:        1589
 IJJJ:         670 IIIL:         674 IIII:          41 IJKL:      274342
Number of integrals in INCORE buffers:
IIKK:        1711 IJJL:       29511 IJKJ:       29541
IIKL:       29052 IJKL:      428347
 TOTAL OF    1389937 TWO-ELECTRON INTEGRALS CALCULATED (RAW)



CLOSED SHELL SCF               Nuclear Repulsion Energy is     83.935233098 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE    ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:    1       -217.901047167       -133.965814069
 SCF_CYCLE:    2       -218.158366446       -134.223133348   1.52834E-02
 SCF_CYCLE:    3       -218.177684138       -134.242451041   4.76679E-03
 SCF_CYCLE:    4       -218.180330431       -134.245097333   2.22284E-03
 SCF_CYCLE:    5       -218.180811678       -134.245578580   9.73653E-04
 SCF_CYCLE:    6       -218.180916474       -134.245683376   5.26826E-04
 SCF_CYCLE:    7       -218.180941559       -134.245708461   2.43426E-04
 SCF_CYCLE:    8       -218.180962327       -134.245729229                 4-POINT
 SCF_CYCLE:    9       -218.180950251       -134.245717154   2.13781E-04
At termination total energy is      -134.245717  Hartrees

Energy components:
Kinetic =                   134.198831358
Potential =                -481.028943959
Kinetic + Potential =      -346.830112601
Coulomb repulsion =         148.179763427
Exchange =                  -19.530601084
Coulomb+Exchange =          128.649162343
Nuclear =                    83.935233098
Total electronic =         -218.180950258
Total energy =             -134.245717160

Virial =                     2.000349376

Atom           J             K             Vee           JHF           KHF
    VeeHF
```

```
       1       4.777116      -0.173768       4.603348       4.935666      -0.332318
    4.603348
Atom            Coulomb
       1        3.666756
J_total         3.666756

Atom         Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1       0.198498      0.199391      0.193105      0.590994

Ttotal:        0.198498      0.199391      0.193105      0.590994

Atom            Vne
       1       -7.870894
Vne_total:            -7.870894


****************************************
 The fragment with symbol 1(6(7,6,1))
 has been added to the database with index #          119
****************************************

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    0.9393195075
Pure Exchange, K ( 2K_ab)         =   -0.1737683920
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -0.3323177771
Kinetic energy Numerical, T      =    0.5909938427
Potential Energy Analytical, Vne =   -8.6399581266
Potential Energy Numerical, Vne  =   -7.8708936682
Coulomb Energy Anal/Num, Vee     =    3.6667562665
Pure Coulomb, J ( 4J_ab+ Jaa)     =    4.7771162122
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =    4.9356655972
Jaa = Kaa                        =    0.1585493850
Coulomb Numerically Over A       =    0.3757362482


PROGRAM> end of inputs


Program terminated normally


Job: RUN_FRAG_0119 ended on :24-Aug-18 at 15:38:55
User: ibrahim
Cpu     time:  00h00m11s63c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m11s00c



*** RUNing the inputfile :: INPUT_0120.dat


****************************************
 Working on symbol 1(7(6,1))
 IndexNUM          120
****************************************


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: 1(7(6,1))
```

```
---------------------------------------------------------------------------------------------------

                             COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                        X
              Y             Z
---------------------------------------------------------------------------------------------------

   1 H          1   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   2 N          7  -0.00000000     0.00000000     1.00062635              -0.00000000
        0.00000000     1.89090962
   3 C          6   0.00000000     1.35162371     1.49976710               0.00000000
        2.55419846     2.83414886
   4 H          1   0.79799306    -0.50815411     1.32380172               1.50798822
       -0.96027202     2.50162252
   5 H          1   0.00000000     1.33667752     2.58332361               0.00000000
        2.52595426     4.88177376
   6 H          1   0.88478395     1.86757977     1.14569291               1.67199922
        3.52921402     2.16504566
   7 H          1  -0.88478388     1.86760507     1.14562438              -1.67199910
        3.52926185     2.16491617
---------------------------------------------------------------------------------------------------

Nuclear repulsion energy:      42.600440492


Distance Matrix for: 1(7(6,1))
             1           2           3           4           5           6
     7
   1      0.000000    1.000626    2.018957    1.627103    2.908654    2.362903
    2.362889
   2      1.000626    0.000000    1.440843    0.999728    2.071627    2.071652
    2.071670
   3      2.018957    1.440843    0.000000    2.031386    1.083660    1.083707
    1.083742
   4      1.627103    0.999728    2.031386    0.000000    2.372044    2.383981
    2.916799
   5      2.908654    2.071627    1.083660    2.372044    0.000000    1.769599
    1.769663
   6      2.362903    2.071652    1.083707    2.383981    1.769599    0.000000
    1.769568
   7      2.362889    2.071670    1.083742    2.916799    1.769663    1.769568
    0.000000

Charge=    0, Number of electrons=   18

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       40

 Partitioning scheme set to: BECKE
Free format Z-Matrix for: 1(7(6,1))
H1
N2       H1       N2H1
C3       N2       C3N2             H1       C3N2H1
H4       N2       H4N2             H1       H4N2H1           C3       H4N2H1C3
H5       C3       H5C3             N2       H5C3N2           H1       H5C3N2H1
H6       C3       H6C3             N2       H6C3N2           H1       H6C3N2H1
H7       C3       H7C3             N2       H7C3N2           H1       H7C3N2H1

VARIABLES:
N2H1     = 1.00062635         C3N2     = 1.44084286         H4N2     = 0.99972788
```

```
H5C3     = 1.08365959         H6C3     = 1.08370744         H7C3     = 1.08374183
C3N2H1   =  110.26867         H4N2H1   =  108.86039         H5C3N2   =  109.47840
H6C3N2   =  109.47751         H7C3N2   =  109.47686         H4N2H1C3 = -122.48858
H5C3N2H1 = -180.00000         H6C3N2H1 =  -59.99718         H7C3N2H1 =   59.99363

Cartesian coordinates for: 1(7(6,1))

--------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                       X
             Y             Z
--------------------------------------------------------------------------------------------------

   1 H1         1   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   2 N2         7   0.00000000     0.00000000     1.00062635               0.00000000
        0.00000000     1.89090962
   3 C3         6   1.35162371     0.00000000     1.49976710               2.55419846
        0.00000000     2.83414886
   4 H4         1  -0.50815411    -0.79799306     1.32380172              -0.96027202
       -1.50798822     2.50162252
   5 H5         1   1.33667752     0.00000000     2.58332361               2.52595426
        0.00000000     4.88177376
   6 H6         1   1.86757977    -0.88478395     1.14569291               3.52921402
       -1.67199922     2.16504566
   7 H7         1   1.86760507     0.88478388     1.14562438               3.52926185
        1.67199910     2.16491617
--------------------------------------------------------------------------------------------------

Nuclear repulsion energy:      42.600440492

The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

     180162 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       8554 IJKJ:       9223 IJJL:       9391 IIKK:        804
 IJJJ:        210 IIIL:        210 IIII:         12 IJKL:     151758
Number of integrals in INCORE buffers:
IIKK:        402 IJJL:       4705 IJKJ:       4616
IIKL:       4293 IJKL:      50857

      67855 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       5131 IJKJ:       5704 IJJL:       5281 IIKK:        748
 IJJJ:        269 IIIL:        270 IIII:         28 IJKL:      50424
Number of integrals in INCORE buffers:
IIKK:        780 IJJL:       7361 IJKJ:       7484
IIKL:       6877 IJKL:      67853
 TOTAL OF     248017 TWO-ELECTRON INTEGRALS CALCULATED (RAW)



CLOSED SHELL SCF               Nuclear Repulsion Energy is     42.600440492 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE   ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:   1       -137.590780390       -94.990339899
 SCF_CYCLE:   2       -137.789680302       -95.189239810   1.90611E-02
```

```
 SCF_CYCLE:   3       -137.805890774       -95.205450282   6.03089E-03
 SCF_CYCLE:   4       -137.808182440       -95.207741948   2.82661E-03
 SCF_CYCLE:   5       -137.808604264       -95.208163772   1.31116E-03
 SCF_CYCLE:   6       -137.808696267       -95.208255775   7.00462E-04
 SCF_CYCLE:   7       -137.808718192       -95.208277700   3.41534E-04
 SCF_CYCLE:   8       -137.808738184       -95.208297692                    4-POINT
 SCF_CYCLE:   9       -137.808725645       -95.208285153   3.23403E-04
At termination total energy is       -95.208285  Hartrees

Energy components:
Kinetic =                    95.147962603
Potential =                 -307.639621452
Kinetic + Potential =       -212.491658849
Coulomb repulsion =           88.293824920
Exchange =                   -13.610891725
Coulomb+Exchange =            74.682933195
Nuclear =                     42.600440492
Total electronic =          -137.808725654
Total energy =               -95.208285162

Virial =                      2.000633987

Atom           J             K             Vee           JHF           KHF
    VeeHF
       1       3.283684      -0.158275      3.125409      3.464068     -0.338659
    3.125409
Atom           Coulomb
       1       3.003021
J_total        3.003021

Atom        Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1      0.222024      0.227975      0.237672      0.687671

Ttotal:       0.222024      0.227975      0.237672      0.687671

Atom           Vne
       1      -6.583313
Vne_total:           -6.583313


****************************************
 The fragment with symbol 1(7(6,1))
 has been added to the database with index #          120
****************************************

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    0.9122004836
Pure Exchange, K ( 2K_ab)         =   -0.1582751924
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -0.3386585185
Kinetic energy Numerical, T      =    0.6876713434
Potential Energy Analytical, Vne =   -7.3027568705
Potential Energy Numerical, Vne  =   -6.5833134021
Coulomb Energy Anal/Num, Vee     =    3.0030205592
Pure Coulomb, J ( 4J_ab+ Jaa)     =    3.2836843595
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =    3.4640676857
Jaa = Kaa                        =    0.1803833262
Coulomb Numerically Over A       =    0.3613871992


PROGRAM> end of inputs


Program terminated normally
```

```
Job: RUN_FRAG_0120 ended on :24-Aug-18 at 15:39:00
User: ibrahim
Cpu     time:  00h00m04s48c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m05s00c



*** RUNing the inputfile :: INPUT_0121.dat


****************************************
 Working on symbol 8(6(8,6)1())
 IndexNUM          121
****************************************


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: 8(6(8,6)1())

-------------------------------------------------------------------------------------------------------

                              COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                         X
             Y             Z
-------------------------------------------------------------------------------------------------------

   1 O          8  -0.00000000    -0.00000000     0.00000000              -0.00000000
       -0.00000000     0.00000000
   2 C          6  -0.00000000    -0.00000000     1.33306652              -0.00000000
       -0.00000000     2.51913045
   3 H          1  -0.00000000     0.87662888    -0.36063052              -0.00000000
        1.65658838    -0.68149287
   4 O          8   0.04287417    -1.01290903     1.93912768               0.08102044
       -1.91412051     3.66441996
   5 C          6  -0.13144606     1.36309713     1.98763767              -0.24839703
        2.57588007     3.75609056
   6 H          1  -0.12044877     1.24905513     3.06522367              -0.22761517
        2.36037195     5.79243283
   7 H          1  -1.06458625     1.82199002     1.68253358              -2.01177631
        3.44306189     3.17952743
   8 H          1   0.69680779     1.99187626     1.68246767               1.31677578
        3.76410033     3.17940288
-------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:     122.214452707


Distance Matrix for: 8(6(8,6)1())
             1           2           3           4           5           6
     7           8
   1      0.000000    1.333067    0.947910    2.188159    2.413714    3.312136
    2.698872    2.698853
   2      1.333067    0.000000    1.907115    1.181157    1.517819    2.138928
    2.138953    2.138970
   3      0.947910    1.907115    0.000000    2.976757    2.401727    3.448142
    2.490295    2.429726
   4      2.188159    1.181157    2.976757    0.000000    2.382886    2.532044
```

```
    3.054335    3.085812
   5      2.413714    1.517819    2.401727    2.382886    0.000000    1.083660
    1.083707    1.083742
   6      3.312136    2.138928    3.448142    2.532044    1.083660    0.000000
    1.769599    1.769663
   7      2.698872    2.138953    2.490295    3.054335    1.083707    1.769599
    0.000000    1.769568
   8      2.698853    2.138970    2.429726    3.085812    1.083742    1.769663
    1.769568    0.000000

Charge=    0, Number of electrons=   32

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       68

 Partitioning scheme set to: BECKE
Free format Z-Matrix for: 8(6(8,6)1())
O1
C2       O1       C2O1
H3       O1       H3O1             C2       H3O1C2
O4       C2       O4C2             O1       O4C2O1           H3       O4C2O1H3
C5       C2       C5C2             O1       C5C2O1           H3       C5C2O1H3
H6       C5       H6C5             C2       H6C5C2           O1       H6C5C2O1
H7       C5       H7C5             C2       H7C5C2           O1       H7C5C2O1
H8       C5       H8C5             C2       H8C5C2           O1       H8C5C2O1

VARIABLES:
C2O1     = 1.33306652         H3O1     = 0.94790958         O4C2     = 1.18115749
C5C2     = 1.51781924         H6C5     = 1.08365959         H7C5     = 1.08370744
H8C5     = 1.08374183         H3O1C2   =  112.36145         O4C2O1   =  120.87106
C5C2O1   =  115.54740         H6C5C2   =  109.47840         H7C5C2   =  109.47751
H8C5C2   =  109.47686         O4C2O1H3 = -177.57624         C5C2O1H3 =    5.50811
H6C5C2O1 = -180.00000         H7C5C2O1 =   59.99718         H8C5C2O1 =  -59.99363

Cartesian coordinates for: 8(6(8,6)1())

----------------------------------------------------------------------------------------------------

                          COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                       X
             Y             Z
----------------------------------------------------------------------------------------------------

   1 O1         8   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   2 C2         6   0.00000000     0.00000000     1.33306652               0.00000000
        0.00000000     2.51913045
   3 H3         1   0.87662888     0.00000000    -0.36063052               1.65658838
        0.00000000    -0.68149287
   4 O4         8  -1.01290903    -0.04287417     1.93912768              -1.91412051
       -0.08102044     3.66441996
   5 C5         6   1.36309713     0.13144606     1.98763767               2.57588007
        0.24839703     3.75609056
   6 H6         1   1.24905513     0.12044877     3.06522367               2.36037195
        0.22761517     5.79243283
   7 H7         1   1.82199002     1.06458625     1.68253358               3.44306189
        2.01177631     3.17952743
   8 H8         1   1.99187626    -0.69680779     1.68246767               3.76410033
       -1.31677578     3.17940288
----------------------------------------------------------------------------------------------------

Nuclear repulsion energy:     122.214452707
```

```
The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

    1941636 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      62701 IJKJ:      63397 IJJL:      64227 IIKK:       2542
 IJJJ:        988 IIIL:        996 IIII:         24 IJKL:    1746761
Number of integrals in INCORE buffers:
IIKK:       1332 IJJL:      33817 IJKJ:      33328
IIKL:      33529 IJKL:     640583

     420642 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      22933 IJKJ:      23388 IJJL:      22937 IIKK:       1798
 IJJJ:        744 IIIL:        747 IIII:         44 IJKL:     348051
Number of integrals in INCORE buffers:
IIKK:       2278 IJJL:      46089 IJKJ:      45739
IIKL:      45855 IJKL:     769781
 TOTAL OF    2362278 TWO-ELECTRON INTEGRALS CALCULATED (RAW)



CLOSED SHELL SCF               Nuclear Repulsion Energy is    122.214452707 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE   ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:   1       -349.154741428      -226.940288720
 SCF_CYCLE:   2       -349.862343589      -227.647890882   3.25587E-02
 SCF_CYCLE:   3       -349.944259438      -227.729806730   1.31475E-02
 SCF_CYCLE:   4       -349.971468508      -227.757015801   1.06269E-02
 SCF_CYCLE:   5       -349.988791077      -227.774338370   8.06760E-03
 SCF_CYCLE:   6       -349.997923915      -227.783471208   6.36459E-03
 SCF_CYCLE:   7       -350.013104804      -227.798652097                 4-POINT
 SCF_CYCLE:   8       -350.013249551      -227.798796844   2.86969E-03
 SCF_CYCLE:   9       -350.013263275      -227.798810568   1.55019E-04
 SCF_CYCLE:  10       -350.013268589      -227.798815882   1.01277E-04
 SCF_CYCLE:  11       -350.013272533      -227.798819825                 4-POINT
 SCF_CYCLE:  12       -350.013272150      -227.798819443   3.91761E-05
At termination total energy is      -227.798819  Hartrees

Energy components:
Kinetic =                    227.373772738
Potential =                 -780.977542957
Kinetic + Potential =       -553.603770218
Coulomb repulsion =          231.913768217
Exchange =                   -28.323270150
Coulomb+Exchange =           203.590498067
Nuclear =                    122.214452707
Total electronic =          -350.013272152
Total energy =              -227.798819445

Virial =                      2.001869374

Atom           J             K             Vee           JHF           KHF
    VeeHF
       1      54.706818     -2.023621     52.683198     60.872873     -8.189676
    52.683198
Atom           Coulomb
       1      63.516302
J_total       63.516302
```

```
Atom         Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1     24.631878     25.067331     24.421903     74.121111

Ttotal:      24.631878     25.067331     24.421903     74.121111

Atom           Vne
       1    -231.651040
Vne_total:         -231.651040


****************************************
 The fragment with symbol 8(6(8,6)1())
 has been added to the database with index #          121
****************************************

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    8.0097633529
Pure Exchange, K ( 2K_ab)         =   -2.0236205654
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -8.1896757065
Kinetic energy Numerical, T      =   74.1211113231
Potential Energy Analytical, Vne = -231.9860332794
Potential Energy Numerical, Vne  = -231.6510404857
Coulomb Energy Anal/Num, Vee     =   63.5163024543
Pure Coulomb, J ( 4J_ab+ Jaa)     =   54.7068180655
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =   60.8728732066
Jaa = Kaa                        =    6.1660551412
Coulomb Numerically Over A       =   37.6296518663


PROGRAM> end of inputs


Program terminated normally


Job: RUN_FRAG_0121 ended on :24-Aug-18 at 15:39:20
User: ibrahim
Cpu     time:  00h00m19s55c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m20s00c



*** RUNing the inputfile :: INPUT_0122.dat


****************************************
 Working on symbol 1(8(6))
 IndexNUM          122
****************************************


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: 1(8(6))

-----------------------------------------------------------------------------------------------------

                              COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
```

```
   I EL        AN        X              Y              Z                        X
                Y              Z
-------------------------------------------------------------------------------------------------

   1 H          1   0.00000000    -0.00000000    -0.00000000               0.00000000
       -0.00000000    -0.00000000
   2 O          8   0.00000000    -0.00000000     0.94790958               0.00000000
       -0.00000000     1.79128937
   3 C          6   0.00000000     1.23282287     1.45507238               0.00000000
        2.32969742     2.74968810
   4 H          1  -0.00000000     1.17831758     2.53736036              -0.00000000
        2.22669735     4.79491582
   5 H          1   0.88478395     1.76136479     1.12007593               1.67199922
        3.32849682     2.11663659
   6 H          1  -0.88478388     1.76139258     1.12000837              -1.67199910
        3.32854934     2.11650893
-------------------------------------------------------------------------------------------------

Nuclear repulsion energy:      41.946468146


Distance Matrix for: 1(8(6))
              1           2           3           4           5           6
   1      0.000000    0.947910    1.907115    2.797611    2.267117    2.267105
   2      0.947910    0.000000    1.333067    1.978582    1.978608    1.978627
   3      1.907115    1.333067    0.000000    1.083660    1.083707    1.083742
   4      2.797611    1.978582    1.083660    0.000000    1.769599    1.769663
   5      2.267117    1.978608    1.083707    1.769599    0.000000    1.769568
   6      2.267105    1.978627    1.083742    1.769663    1.769568    0.000000

Charge=    0, Number of electrons=   18

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       38

 Partitioning scheme set to: BECKE
Free format Z-Matrix for: 1(8(6))
H1
O2       H1       O2H1
C3       O2       C3O2             H1       C3O2H1
H4       C3       H4C3             O2       H4C3O2           H1       H4C3O2H1
H5       C3       H5C3             O2       H5C3O2           H1       H5C3O2H1
H6       C3       H6C3             O2       H6C3O2           H1       H6C3O2H1

VARIABLES:
O2H1     = 0.94790958         C3O2     = 1.33306652         H4C3     = 1.08365959
H5C3     = 1.08370744         H6C3     = 1.08374183         C3O2H1   =  112.36145
H4C3O2   =  109.47840         H5C3O2   =  109.47751         H6C3O2   =  109.47686
H4C3O2H1 =  180.00000         H5C3O2H1 =  -59.99718         H6C3O2H1 =   59.99363

Cartesian coordinates for: 1(8(6))

-------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                        X
                Y              Z
-------------------------------------------------------------------------------------------------

   1 H1         1   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   2 O2         8   0.00000000     0.00000000     0.94790958               0.00000000
```

```
         0.00000000     1.79128937
   3 C3          6   1.23282287     0.00000000     1.45507238               2.32969742
         0.00000000     2.74968810
   4 H4          1   1.17831758     0.00000000     2.53736036               2.22669735
         0.00000000     4.79491582
   5 H5          1   1.76136479    -0.88478395     1.12007593               3.32849682
        -1.67199922     2.11663659
   6 H6          1   1.76139258     0.88478388     1.12000837               3.32854934
         1.67199910     2.11650893
-------------------------------------------------------------------------------------------------

Nuclear repulsion energy:      41.946468146

The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

     139570 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       7031 IJKJ:       7696 IJJL:       7864 IIKK:        756
 IJJJ:        186 IIIL:        186 IIII:         12 IJKL:     115839
Number of integrals in INCORE buffers:
IIKK:        378 IJJL:       3937 IJKJ:       3848
IIKL:       3525 IJKL:      38807

      47218 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       3789 IJKJ:       4371 IJJL:       3937 IIKK:        645
 IJJJ:        217 IIIL:        217 IIII:         26 IJKL:      34016
Number of integrals in INCORE buffers:
IIKK:        703 IJJL:       5917 IJKJ:       6040
IIKL:       5433 IJKL:      50279
 TOTAL OF     186788 TWO-ELECTRON INTEGRALS CALCULATED (RAW)



CLOSED SHELL SCF               Nuclear Repulsion Energy is     41.946468146 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE   ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:   1       -156.599321648      -114.652853501
 SCF_CYCLE:   2       -156.936416209      -114.989948063   3.11167E-02
 SCF_CYCLE:   3       -156.968403650      -115.021935504   9.05693E-03
 SCF_CYCLE:   4       -156.974430396      -115.027962249   5.35023E-03
 SCF_CYCLE:   5       -156.976003655      -115.029535509   2.57563E-03
 SCF_CYCLE:   6       -156.976488501      -115.030020355   1.63447E-03
 SCF_CYCLE:   7       -156.976798256      -115.030330110                 4-POINT
 SCF_CYCLE:   8       -156.976733196      -115.030265049   6.30655E-04
 SCF_CYCLE:   9       -156.976733226      -115.030265080   1.01698E-05
At termination total energy is      -115.030265  Hartrees

Energy components:
Kinetic =                   115.033978657
Potential =                -355.000327289
Kinetic + Potential =      -239.966348632
Coulomb repulsion =          97.903875440
Exchange =                  -14.914260040
Coulomb+Exchange =           82.989615401
Nuclear =                    41.946468146
Total electronic =         -156.976733232
Total energy =             -115.030265085
```

```
Virial =                      1.999967718

Atom           J             K             Vee           JHF           KHF
    VeeHF
       1       3.189304      -0.148166      3.041138      3.381697     -0.340559
    3.041138
Atom           Coulomb
       1       3.134308
J_total        3.134308

Atom        Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1      0.247152      0.256589      0.288540      0.792281

Ttotal:       0.247152      0.256589      0.288540      0.792281

Atom           Vne
       1      -6.886960
Vne_total:           -6.886960


****************************************
 The fragment with symbol 1(8(6))
 has been added to the database with index #          122
****************************************

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    0.8692169593
Pure Exchange, K ( 2K_ab)         =   -0.1481660000
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -0.3405589381
Kinetic energy Numerical, T      =    0.7922814061
Potential Energy Analytical, Vne =   -7.7815163959
Potential Energy Numerical, Vne  =   -6.8869597878
Coulomb Energy Anal/Num, Vee     =    3.1343081800
Pure Coulomb, J ( 4J_ab+ Jaa)     =    3.1893037531
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =    3.3816966912
Jaa = Kaa                        =    0.1923929381
Coulomb Numerically Over A       =    0.3393770945


PROGRAM> end of inputs


Program terminated normally


Job: RUN_FRAG_0122 ended on :24-Aug-18 at 15:39:23
User: ibrahim
Cpu     time:  00h00m03s92c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m03s00c


PROGRAM> end of inputs


Program terminated normally


Job: RUN_Gly_ ended on :24-Aug-18 at 15:39:24
User: ibrahim
Cpu     time:  00h00m04s72c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h02m36s00c
```

B.1.3 Building the Molecular Properties Using AIMD

Here is example of input file to calculate the molecular properties of Glysine from the database and direct methods,

```
MOLECULE
 UNITS=ANGSTROM !(default)
 CHARGE=0
 MULTIPLICITY=1
FREEZ
N
C    1 B1
C    2 B2 1 A2
O    3 B3 2 A3 1 D3
H    2 B4 1 A4 3 D4
H    2 B5 1 A5 3 D5
H    1 B6 2 A6 3 D6
H    1 B7 2 A7 3 D7
O    3 B8 2 A8 1 D8
H    9 B9 3 A9 2 D9

END
DEFINE
B1 = 1.44084286              B2 = 1.51781924              B3 = 1.18115749
B4 = 1.08653973              B5 = 1.09386650              B6 = 1.00062635
B7 = 0.99972788              B8 = 1.33306652              B9 = 0.94790958
A2 =  110.31733              A3 =  123.50442              A4 =  109.68864
A5 =  114.69657              A6 =  110.26867              A7 =  111.39727
A8 =  115.54740              A9 =  112.36145              D3 =  -21.55248
D4 =  120.68224              D5 = -119.29373              D6 =   38.90011
D7 =  159.88520              D8 =  161.62252              D9 =   -5.50811
END
end !molecule

BASIS name=6-31G(d) end

SET RUN NAME = "RUN_Gly_" end end

AIMDFT
 Level=2
 Direct  ! Building of the Molecule (Direct or Database) (default is Database)
end !AIMDFT

output object=AIMDFT:MOLECULE%BUILD end
stop
```

Here is the results using the database,

```
Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Free format Z-Matrix for: C2H5NO2, (C1)
N
C         N         B1
C         C         B2   N         A2
O         C         B3   C         A3   N         D3
H         C         B4   N         A4   C         D4
```

```
H         C         B5  N         A5  C         D5
H         N         B6  C         A6  C         D6
H         N         B7  C         A7  C         D7
O         C         B8  C         A8  N         D8
H         O         B9  C         A9  C         D9

VARIABLES:
B1 = 1.44084286              B2 = 1.51781924              B3 = 1.18115749
B4 = 1.08653973              B5 = 1.09386650              B6 = 1.00062635
B7 = 0.99972788              B8 = 1.33306652              B9 = 0.94790958
A2 =  110.31733              A3 =  123.50442              A4 =  109.68864
A5 =  114.69657              A6 =  110.26867              A7 =  111.39727
A8 =  115.54740              A9 =  112.36145              D3 =  -21.55248
D4 =  120.68224              D5 = -119.29373              D6 =   38.90011
D7 =  159.88520              D8 =  161.62252              D9 =   -5.50811

Z MATRIX FOR: C2H5NO2, (C1)
--------------------------------------------------------------------------------------------------

   I    AN    Z1      BL                  Z2     ALPHA                 Z3      BETA
              Z4
--------------------------------------------------------------------------------------------------

   1    7
   2    6     1    1.440843    (  1)
   3    6     2    1.517819    (  2)        1    110.3173    ( 10)
   4    8     3    1.181157    (  3)        2    123.5044    ( 11)        1    -21.5525    (
    18)         0
   5    1     2    1.086540    (  4)        1    109.6886    ( 12)        3    120.6822    (
    19)         0
   6    1     2    1.093866    (  5)        1    114.6966    ( 13)        3   -119.2937    (
    20)         0
   7    1     1    1.000626    (  6)        2    110.2687    ( 14)        3     38.9001    (
    21)         0
   8    1     1    0.999728    (  7)        2    111.3973    ( 15)        3    159.8852    (
    22)         0
   9    8     3    1.333067    (  8)        2    115.5474    ( 16)        1    161.6225    (
    23)         0
  10    1     9    0.947910    (  9)        3    112.3615    ( 17)        2     -5.5081    (
    24)         0
--------------------------------------------------------------------------------------------------


Cartesian coordinates for: C2H5NO2, (C1)

--------------------------------------------------------------------------------------------------

                              COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                        X
              Y              Z
--------------------------------------------------------------------------------------------------

   1 N          7    0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   2 C          6    0.00000000     0.00000000     1.44084286               0.00000000
        0.00000000     2.72279820
   3 C          6    1.42338653     0.00000000     1.96785902               2.68981052
        0.00000000     3.71871433
   4 O          8    2.35288801    -0.36180632     1.33520098               4.44631363
       -0.68371481     2.52316398
   5 H          1   -0.52202178     0.87980599     1.80690743              -0.98647812
        1.66259225     3.41455992
   6 H          1   -0.48626030    -0.86672789     1.89787416              -0.91889872
```

```
        -1.63787822     3.58646212
   7 H          1   0.73050930    -0.58944902    -0.34663973               1.38046242
        -1.11389713    -0.65505411
   8 H          1  -0.87404490    -0.32011104    -0.36473314              -1.65170536
        -0.60492216    -0.68924570
   9 O          8   1.56620070     0.37919230     3.23785258               2.95969016
         0.71656954     6.11865417
  10 H          1   0.75176248     0.68026803     3.61808458               1.42062510
         1.28552018     6.83718846
--------------------------------------------------------------------------------------------------

Nuclear repulsion energy:     181.510743561


Distance Matrix for: C2H5NO2, (C1)
             1           2           3           4           5           6
     7           8           9          10
   1      0.000000    1.440843    2.428682    2.729423    2.076410    2.142333
    1.000626    0.999728    3.616692    3.757452
   2      1.440843    0.000000    1.517819    2.382886    1.086540    1.093866
    2.018957    2.031386    2.413714    2.401727
   3      2.428682    1.517819    0.000000    1.181157    2.141163    2.098301
    2.486852    3.289627    1.333067    1.907115
   4      2.729423    2.382886    1.181157    0.000000    3.166893    2.938079
    2.347876    3.647549    2.188159    2.976757
   5      2.076410    1.086540    2.141163    3.166893    0.000000    1.749267
    2.892285    2.505942    2.580483    2.223219
   6      2.142333    1.093866    2.098301    2.938079    1.749267    0.000000
    2.568123    2.359779    2.749628    2.623932
   7      1.000626    2.018957    2.486852    2.347876    2.892285    2.568123
    0.000000    1.627103    3.805947    4.163132
   8      0.999728    2.031386    3.289627    3.647549    2.505942    2.359779
    1.627103    0.000000    4.407091    4.416655
   9      3.616692    2.413714    1.333067    2.188159    2.580483    2.749628
    3.805947    4.407091    0.000000    0.947910
  10      3.757452    2.401727    1.907115    2.976757    2.223219    2.623932
    4.163132    4.416655    0.947910    0.000000

Charge=    0, Number of electrons=   40

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       85

The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

    4124847 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:     113538 IJKJ:     113807 IJJL:     114903 IIKK:       3799
 IJJJ:       1462 IIIL:       1470 IIII:         30 IJKL:    3775838
Number of integrals in INCORE buffers:
IIKK:       2085 IJJL:      64223 IJKJ:      63543
IIKL:      64237 IJKL:    1535810

     937565 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      43244 IJKJ:      44015 IJJL:      43286 IIKK:       2740
 IJJJ:       1152 IIIL:       1158 IIII:         55 IJKL:     801915
Number of integrals in INCORE buffers:
IIKK:       3570 IJJL:      88212 IJKJ:      87691
IIKL:      88349 IJKL:     271361
```

```
Number of buffers:
IJJL:          0 IJKJ:          0
IIKL:          0 IJKL:          1
 TOTAL OF    5062412 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF               Nuclear Repulsion Energy is    181.510743561 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE   ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:   1       -463.303052879      -281.792309318
 SCF_CYCLE:   2       -464.156082549      -282.645338988   2.78967E-02
 SCF_CYCLE:   3       -464.252967556      -282.742223996   1.11306E-02
 SCF_CYCLE:   4       -464.282711412      -282.771967851   8.78492E-03
 SCF_CYCLE:   5       -464.301085806      -282.790342245   6.67293E-03
 SCF_CYCLE:   6       -464.310646467      -282.799902906   5.26956E-03
 SCF_CYCLE:   7       -464.325561119      -282.814817558                 4-POINT
 SCF_CYCLE:   8       -464.326970023      -282.816226462   2.39444E-03
 SCF_CYCLE:   9       -464.326980754      -282.816237193   1.12426E-04
 SCF_CYCLE:  10       -464.326984887      -282.816241326   7.27041E-05
 SCF_CYCLE:  11       -464.326986541      -282.816242980   4.60760E-05
 SCF_CYCLE:  12       -464.326987992      -282.816244431                 4-POINT
 SCF_CYCLE:  13       -464.326987698      -282.816244138   1.87641E-05
At termination total energy is      -282.816244  Hartrees


Number of indexs available in the database    =   8
Number of indexs NOT available in the database =  0
**********************************
 The unique Symbols in the molecule
**********************************
SYM#    1 is 7(6(6,1,1)1()1())
SYM#    2 is 6(7(1,1)6(8,8)1()1())
SYM#    3 is 6(8(1)8()6(7,1,1))
SYM#    4 is 8(6(8,6))
SYM#    5 is 1(6(7,6,1))
SYM#    6 is 1(7(6,1))
SYM#    7 is 8(6(8,6)1())
SYM#    8 is 1(8(6))
**********************************
           CART OF INDEX
**********************************
 index #          115 For Symbol 7(6(6,1,1)1()1())
  1   N       -0.00000000      0.00000000      0.00000000
  2   C       -0.00000000     -0.00000000      1.44084286
  3   H        0.00000000      0.93866607     -0.34663973
  4   H        0.79799306     -0.47920001     -0.36473314
  5   C       -0.89383628      1.10773910      1.96785902
  6   H       -0.35689075     -0.95874622      1.80690743
  7   H        0.97987823      0.16584623      1.89787416
  8   H       -0.88387273      1.09539118      3.05140244
  9   H       -1.90660309      0.95389867      1.61422314
 10   H       -0.52946711      2.06514427      1.61415464
 index #          116 For Symbol 6(7(1,1)6(8,8)1()1())
  1   C       -0.00000000     -0.00000000     -0.00000000
  2   N        0.00000000      0.00000000      1.44084286
  3   C       -0.00000000      1.42338653     -0.52701616
  4   H        0.87980599     -0.52202178     -0.36606457
  5   H       -0.86672789     -0.48626030     -0.45703130
  6   O        0.37919230      1.56620070     -1.79700972
  7   O       -0.36180632      2.35288801      0.10564188
  8   H       -0.58944902      0.73050930      1.78748259
  9   H       -0.32011104     -0.87404490      1.80557600
```

```
 10   H         0.31941883      2.53932814     -1.96545672
 index #          117 For Symbol 6(8(1)8()6(7,1,1))
  1   C         0.00000000     -0.00000000      0.00000000
  2   O        -0.00000000      0.00000000      1.33306652
  3   O         0.00000000      1.01381600     -0.60606116
  4   C         0.07368323     -1.36743652     -0.65457115
  5   N        -0.35716835     -1.28888329     -2.02724171
  6   H        -0.03707255     -0.87584463      1.69369704
  7   H        -0.55932392     -2.06380818     -0.11149008
  8   H         1.10434945     -1.70738451     -0.51779972
  9   H        -0.29008929     -2.21219942     -2.41194164
 10   H         0.30676556     -0.71652891     -2.51368937
 index #          118 For Symbol 8(6(8,6))
  1   O         0.00000000      0.00000000      0.00000000
  2   C         0.00000000     -0.00000000      1.18115749
  3   O         0.00000000      1.14420319      1.86516436
  4   C         0.07368323     -1.26347662      2.01899570
  5   H         0.00000000      0.86533235      2.81445637
  6   H         0.05839270     -1.00128368      3.07034682
  7   H         0.99092661     -1.79429459      1.79245948
  8   H        -0.77563740     -1.89735718      1.79239856
 index #          119 For Symbol 1(6(7,6,1))
  1   H        -0.00000000     -0.00000000     -0.00000000
  2   C         0.00000000      0.00000000      1.08653973
  3   N         0.00000000      1.35660739      1.57197206
  4   C         1.22412734     -0.74090902      1.59284135
  5   H        -0.86045971     -0.59784588      1.40075830
  6   H         0.00000000      1.31271584      2.57351079
  7   H        -0.86960710      1.77063919      1.29378103
  8   H         1.22400910     -0.74083745      2.67650093
  9   H         2.11948787     -0.24860401      1.23174979
 10   H         1.20323095     -1.76248761      1.23168092
 index #          120 For Symbol 1(7(6,1))
  1   H        -0.00000000      0.00000000      0.00000000
  2   N         0.00000000      0.00000000      1.00062635
  3   C        -0.00000000      1.35162371      1.49976710
  4   H         0.79799306     -0.50815411      1.32380172
  5   H        -0.00000000      1.33667752      2.58332361
  6   H         0.88478395      1.86757977      1.14569291
  7   H        -0.88478388      1.86760507      1.14562438
 index #          121 For Symbol 8(6(8,6)1())
  1   O         0.00000000     -0.00000000     -0.00000000
  2   C        -0.00000000     -0.00000000      1.33306652
  3   H         0.00000000      0.87662888     -0.36063052
  4   O         0.04287417     -1.01290903      1.93912768
  5   C        -0.13144606      1.36309713      1.98763767
  6   H        -0.12044877      1.24905513      3.06522367
  7   H        -1.06458625      1.82199002      1.68253358
  8   H         0.69680779      1.99187626      1.68246767
 index #          122 For Symbol 1(8(6))
  1   H         0.00000000     -0.00000000     -0.00000000
  2   O         0.00000000      0.00000000      0.94790958
  3   C         0.00000000      1.23282287      1.45507238
  4   H         0.00000000      1.17831758      2.53736036
  5   H        -0.88478395      1.76136479      1.12007593
  6   H         0.88478388      1.76139258      1.12000837
 *********************************
    NEW FRAG CART (FROM DATABASE)
 *********************************
 Atom#             1
  Atom               X               Y               Z
N             -0.00000000      0.00000000      0.00000000
C             -0.00000000     -0.00000000      1.44084286
H              0.00000000      0.93866607     -0.34663973
```

```
H             0.79799306     -0.47920001     -0.36473314
C            -0.89383628      1.10773910      1.96785902
H            -0.35689075     -0.95874622      1.80690743
H             0.97987823      0.16584623      1.89787416
H            -0.88387273      1.09539118      3.05140244
H            -1.90660309      0.95389867      1.61422314
H            -0.52946711      2.06514427      1.61415464
 Atom#            2
  Atom               X              Y              Z
C            -0.00000000     -0.00000000     -0.00000000
N             0.00000000      0.00000000      1.44084286
C            -0.00000000      1.42338653     -0.52701616
H             0.87980599     -0.52202178     -0.36606457
H            -0.86672789     -0.48626030     -0.45703130
O             0.37919230      1.56620070     -1.79700972
O            -0.36180632      2.35288801      0.10564188
H            -0.58944902      0.73050930      1.78748259
H            -0.32011104     -0.87404490      1.80557600
H             0.31941883      2.53932814     -1.96545672
 Atom#            3
  Atom               X              Y              Z
C             0.00000000     -0.00000000      0.00000000
O            -0.00000000      0.00000000      1.33306652
O             0.00000000      1.01381600     -0.60606116
C             0.07368323     -1.36743652     -0.65457115
N            -0.35716835     -1.28888329     -2.02724171
H            -0.03707255     -0.87584463      1.69369704
H            -0.55932392     -2.06380818     -0.11149008
H             1.10434945     -1.70738451     -0.51779972
H            -0.29008929     -2.21219942     -2.41194164
H             0.30676556     -0.71652891     -2.51368937
 Atom#            4
  Atom               X              Y              Z
O             0.00000000      0.00000000      0.00000000
C             0.00000000     -0.00000000      1.18115749
O             0.00000000      1.14420319      1.86516436
C             0.07368323     -1.26347662      2.01899570
H             0.00000000      0.86533235      2.81445637
H             0.05839270     -1.00128368      3.07034682
H             0.99092661     -1.79429459      1.79245948
H            -0.77563740     -1.89735718      1.79239856
 Atom#            5
  Atom               X              Y              Z
H            -0.00000000     -0.00000000     -0.00000000
C             0.00000000      0.00000000      1.08653973
N             0.00000000      1.35660739      1.57197206
C             1.22412734     -0.74090902      1.59284135
H            -0.86045971     -0.59784588      1.40075830
H             0.00000000      1.31271584      2.57351079
H            -0.86960710      1.77063919      1.29378103
H             1.22400910     -0.74083745      2.67650093
H             2.11948787     -0.24860401      1.23174979
H             1.20323095     -1.76248761      1.23168092
 Atom#            6
  Atom               X              Y              Z
H            -0.00000000     -0.00000000     -0.00000000
C             0.00000000      0.00000000      1.08653973
N             0.00000000      1.35660739      1.57197206
C             1.22412734     -0.74090902      1.59284135
H            -0.86045971     -0.59784588      1.40075830
H             0.00000000      1.31271584      2.57351079
H            -0.86960710      1.77063919      1.29378103
H             1.22400910     -0.74083745      2.67650093
H             2.11948787     -0.24860401      1.23174979
```

```
H              1.20323095      -1.76248761       1.23168092
 Atom#             7
  Atom               X                 Y                 Z
H             -0.00000000       0.00000000       0.00000000
N              0.00000000       0.00000000       1.00062635
C             -0.00000000       1.35162371       1.49976710
H              0.79799306      -0.50815411       1.32380172
H             -0.00000000       1.33667752       2.58332361
H              0.88478395       1.86757977       1.14569291
H             -0.88478388       1.86760507       1.14562438
 Atom#             8
  Atom               X                 Y                 Z
H             -0.00000000       0.00000000       0.00000000
N              0.00000000       0.00000000       1.00062635
C             -0.00000000       1.35162371       1.49976710
H              0.79799306      -0.50815411       1.32380172
H             -0.00000000       1.33667752       2.58332361
H              0.88478395       1.86757977       1.14569291
H             -0.88478388       1.86760507       1.14562438
 Atom#             9
  Atom               X                 Y                 Z
O              0.00000000      -0.00000000      -0.00000000
C             -0.00000000      -0.00000000       1.33306652
H              0.00000000       0.87662888      -0.36063052
O              0.04287417      -1.01290903       1.93912768
C             -0.13144606       1.36309713       1.98763767
H             -0.12044877       1.24905513       3.06522367
H             -1.06458625       1.82199002       1.68253358
H              0.69680779       1.99187626       1.68246767
 Atom#            10
  Atom               X                 Y                 Z
H              0.00000000      -0.00000000      -0.00000000
O              0.00000000       0.00000000       0.94790958
C              0.00000000       1.23282287       1.45507238
H              0.00000000       1.17831758       2.53736036
H             -0.88478395       1.76136479       1.12007593
H              0.88478388       1.76139258       1.12000837
 *********************************************
  Where the fragment should be (database form)
 *********************************************
 Atom#             1
  Atom               X                 Y                 Z
N              0.00000000      -0.00000000       0.00000000
C             -0.00000000      -0.00000000       1.44084286
H              0.00000000       0.93866607      -0.34663973
H              0.79799306      -0.47920001      -0.36473314
C             -0.89383628       1.10773910       1.96785902
H             -0.35689075      -0.95874622       1.80690743
H              0.97987823       0.16584623       1.89787416
H             -0.88387273       1.09539118       3.05140244
H             -1.90660309       0.95389867       1.61422314
H             -0.52946711       2.06514427       1.61415464
 Atom#             2
  Atom               X                 Y                 Z
C             -0.00000000       0.00000000       0.00000000
N              0.00000000      -0.00000000       1.44084286
C              0.00000000       1.42338653      -0.52701616
H              0.87980599      -0.52202178      -0.36606457
H             -0.86672789      -0.48626030      -0.45703130
O              0.37919230       1.56620070      -1.79700972
O             -0.36180632       2.35288801       0.10564188
H             -0.58944902       0.73050930       1.78748259
H             -0.32011104      -0.87404490       1.80557600
H              0.31941883       2.53932814      -1.96545672
```

```
 Atom#             3
  Atom               X                 Y                 Z
C             -0.00000000      -0.00000000       0.00000000
O             -0.00000000       0.00000000       1.33306652
O              0.00000000       1.01381600      -0.60606116
C              0.07368323      -1.36743652      -0.65457115
N             -0.35716835      -1.28888329      -2.02724171
H             -0.03707255      -0.87584463       1.69369704
H             -0.55932392      -2.06380818      -0.11149008
H              1.10434945      -1.70738451      -0.51779972
H             -0.29008929      -2.21219942      -2.41194164
H              0.30676556      -0.71652891      -2.51368937
 Atom#             4
  Atom               X                 Y                 Z
O             -0.00000000       0.00000000      -0.00000000
C             -0.00000000      -0.00000000       1.18115749
O             -0.00000000       1.14420319       1.86516436
C              0.07368323      -1.26347662       2.01899570
H              0.00000000       0.86533235       2.81445637
H              0.05839270      -1.00128368       3.07034682
H              0.99092661      -1.79429459       1.79245948
H             -0.77563740      -1.89735718       1.79239856
 Atom#             5
  Atom               X                 Y                 Z
H             -0.00000000       0.00000000      -0.00000000
C             -0.00000000      -0.00000000       1.08653973
N              0.00000000       1.35660739       1.57197206
C              1.22412734      -0.74090902       1.59284135
H             -0.86045971      -0.59784588       1.40075830
H              0.00000000       1.31271584       2.57351079
H             -0.86960710       1.77063919       1.29378103
H              1.22400910      -0.74083745       2.67650093
H              2.11948787      -0.24860401       1.23174979
H              1.20323095      -1.76248761       1.23168092
 Atom#             6
  Atom               X                 Y                 Z
H             -0.00000000       0.00000000       0.00000000
C              0.00000000       0.00000000       1.09386650
N              0.00000000       1.30905356       1.69586892
C              1.24136788      -0.76979504       1.50641545
H             -0.88574475      -0.54645193       1.40598042
H              0.00000000       1.17790163       2.68975295
H             -0.86960710       1.74578916       1.45488219
H              1.30090798      -0.80671695       2.58780800
H              2.12120534      -0.27430176       1.11297643
H              1.18863813      -1.77819294       1.11290613
 Atom#             7
  Atom               X                 Y                 Z
H              0.00000000       0.00000000       0.00000000
N             -0.00000000       0.00000000       1.00062635
C              0.00000000       1.35162371       1.49976710
H              0.79799306      -0.50815411       1.32380172
H              0.00000000       1.33667752       2.58332361
H              0.88478395       1.86757977       1.14569291
H             -0.88478388       1.86760507       1.14562438
 Atom#             8
  Atom               X                 Y                 Z
H              0.00000000      -0.00000000      -0.00000000
N             -0.00000000      -0.00000000       0.99972788
C              0.00000000       1.34153017       1.52539407
H              0.80471964      -0.49904836       1.32319369
H              0.00000000       1.30524460       2.60844599
H              0.88478395       1.86436016       1.18155110
H             -0.88478388       1.86438681       1.18148309
```

```
 Atom#             9
  Atom                X                Y                Z
O             -0.00000000      -0.00000000       0.00000000
C             -0.00000000      -0.00000000       1.33306652
H             -0.00000000       0.87662888      -0.36063052
O              0.04287417      -1.01290903       1.93912768
C             -0.13144606       1.36309713       1.98763767
H             -0.12044877       1.24905513       3.06522367
H             -1.06458625       1.82199002       1.68253358
H              0.69680779       1.99187626       1.68246767
 Atom#            10
  Atom                X                Y                Z
H              0.00000000      -0.00000000      -0.00000000
O              0.00000000      -0.00000000       0.94790958
C              0.00000000       1.23282287       1.45507238
H             -0.00000000       1.17831758       2.53736036
H              0.88478395       1.76136479       1.12007593
H             -0.88478388       1.76139258       1.12000837
 *******************************************
  The error: (FROM DATABASE)- (Original)
 *******************************************
 Atom#             1
  Atom                X                Y                Z
N             -0.00000000       0.00000000       0.00000000
C              0.00000000      -0.00000000       0.00000000
H              0.00000000       0.00000000       0.00000000
H              0.00000000       0.00000000       0.00000000
C              0.00000000       0.00000000      -0.00000000
H             -0.00000000      -0.00000000      -0.00000000
H              0.00000000      -0.00000000      -0.00000000
H              0.00000000       0.00000000      -0.00000000
H              0.00000000      -0.00000000      -0.00000000
H              0.00000000      -0.00000000      -0.00000000
 Atom#             2
  Atom                X                Y                Z
C             -0.00000000      -0.00000000      -0.00000000
N              0.00000000       0.00000000      -0.00000000
C             -0.00000000       0.00000000       0.00000000
H              0.00000000       0.00000000       0.00000000
H             -0.00000000       0.00000000       0.00000000
O              0.00000000       0.00000000       0.00000000
O             -0.00000000      -0.00000000       0.00000000
H             -0.00000000       0.00000000      -0.00000000
H             -0.00000000       0.00000000      -0.00000000
H              0.00000000       0.00000000       0.00000000
 Atom#             3
  Atom                X                Y                Z
C              0.00000000      -0.00000000       0.00000000
O             -0.00000000       0.00000000       0.00000000
O              0.00000000       0.00000000       0.00000000
C              0.00000000      -0.00000000       0.00000000
N             -0.00000000      -0.00000000       0.00000000
H             -0.00000000      -0.00000000      -0.00000000
H              0.00000000       0.00000000       0.00000000
H              0.00000000       0.00000000       0.00000000
H              0.00000000      -0.00000000       0.00000000
H             -0.00000000      -0.00000000       0.00000000
 Atom#             4
  Atom                X                Y                Z
O              0.00000000       0.00000000       0.00000000
C              0.00000000      -0.00000000       0.00000000
O              0.00000000       0.00000000      -0.00000000
C              0.00000000      -0.00000000      -0.00000000
H             -0.00000000       0.00000000      -0.00000000
```

```
H              0.00000000      -0.00000000      -0.00000000
H              0.00000000       0.00000000      -0.00000000
H              0.00000000      -0.00000000      -0.00000000
 Atom#             5
  Atom               X                Y                Z
H             -0.00000000      -0.00000000      -0.00000000
C              0.00000000       0.00000000       0.00000000
N             -0.00000000       0.00000000      -0.00000000
C              0.00000000       0.00000000      -0.00000000
H             -0.00000000       0.00000000      -0.00000000
H             -0.00000000       0.00000000      -0.00000000
H             -0.00000000      -0.00000000      -0.00000000
H              0.00000000       0.00000000      -0.00000000
H             -0.00000000       0.00000000      -0.00000000
H              0.00000000       0.00000000      -0.00000000
 Atom#             6
  Atom               X                Y                Z
H             -0.00000000      -0.00000000      -0.00000000
C              0.00000000       0.00000000      -0.00732677
N              0.00000000       0.04755382      -0.12389686
C             -0.01724054       0.02888602       0.08642590
H              0.02528504      -0.05139395      -0.00522212
H              0.00000000       0.13481421      -0.11624216
H             -0.00000000       0.02485004      -0.16110116
H             -0.07689888       0.06587950       0.08869293
H             -0.00171747       0.02569774       0.11877335
H              0.01459281       0.01570533       0.11877479
 Atom#             7
  Atom               X                Y                Z
H             -0.00000000       0.00000000       0.00000000
N              0.00000000       0.00000000       0.00000000
C             -0.00000000       0.00000000      -0.00000000
H              0.00000000       0.00000000      -0.00000000
H             -0.00000000       0.00000000      -0.00000000
H             -0.00000000      -0.00000000      -0.00000000
H             -0.00000000      -0.00000000      -0.00000000
 Atom#             8
  Atom               X                Y                Z
H             -0.00000000       0.00000000       0.00000000
N              0.00000000       0.00000000       0.00089847
C             -0.00000000       0.01009354      -0.02562697
H             -0.00672658      -0.00910575       0.00060803
H             -0.00000000       0.03143292      -0.02512238
H             -0.00000000       0.00321961      -0.03585820
H             -0.00000000       0.00321826      -0.03585871
 Atom#             9
  Atom               X                Y                Z
O              0.00000000       0.00000000      -0.00000000
C              0.00000000      -0.00000000      -0.00000000
H              0.00000000       0.00000000       0.00000000
O              0.00000000      -0.00000000      -0.00000000
C              0.00000000       0.00000000      -0.00000000
H              0.00000000       0.00000000      -0.00000000
H             -0.00000000      -0.00000000      -0.00000000
H              0.00000000      -0.00000000      -0.00000000
 Atom#            10
  Atom               X                Y                Z
H             -0.00000000      -0.00000000       0.00000000
O              0.00000000       0.00000000       0.00000000
C              0.00000000       0.00000000      -0.00000000
H              0.00000000       0.00000000      -0.00000000
H             -1.76956789      -0.00000000      -0.00000000
H              1.76956776      -0.00000000      -0.00000000
 *********************************
```

```
  CART WANTED (CART OF MOLECULE)
 *********************************
 Atom#            1
  Atom               X               Y               Z
N              0.00000000      0.00000000      0.00000000
C              0.00000000      0.00000000      1.44084286
H              0.73050930     -0.58944902     -0.34663973
H             -0.87404490     -0.32011104     -0.36473314
C              1.42338653      0.00000000      1.96785902
H             -0.52202178      0.87980599      1.80690743
H             -0.48626030     -0.86672789      1.89787416
H              1.40752011      0.00000000      3.05140244
H              1.93964311      0.88478395      1.61422314
H              1.93966847     -0.88478388      1.61415464
 Atom#            2
  Atom               X               Y               Z
C              0.00000000      0.00000000      1.44084286
N              0.00000000      0.00000000      0.00000000
C              1.42338653      0.00000000      1.96785902
H             -0.52202178      0.87980599      1.80690743
H             -0.48626030     -0.86672789      1.89787416
O              1.56620070      0.37919230      3.23785258
O              2.35288801     -0.36180632      1.33520098
H              0.73050930     -0.58944902     -0.34663973
H             -0.87404490     -0.32011104     -0.36473314
H              2.53932814      0.31941883      3.40629958
 Atom#            3
  Atom               X               Y               Z
C              1.42338653      0.00000000      1.96785902
O              1.56620070      0.37919230      3.23785258
O              2.35288801     -0.36180632      1.33520098
C              0.00000000      0.00000000      1.44084286
N              0.00000000      0.00000000      0.00000000
H              0.75176248      0.68026803      3.61808458
H             -0.52202178      0.87980599      1.80690743
H             -0.48626030     -0.86672789      1.89787416
H             -0.95777357      0.00000000     -0.29610147
H              0.40141812     -0.86960710     -0.29610147
 Atom#            4
  Atom               X               Y               Z
O              2.35288801     -0.36180632      1.33520098
C              1.42338653      0.00000000      1.96785902
O              1.56620070      0.37919230      3.23785258
C              0.00000000      0.00000000      1.44084286
H              0.65316574      0.62862176      3.52608235
H             -0.66879662      0.37530236      2.20646504
H             -0.28673497     -1.01059766      1.17457750
H             -0.06099919      0.63521777      0.56490118
 Atom#            5
  Atom               X               Y               Z
H             -0.52202178      0.87980599      1.80690743
C              0.00000000      0.00000000      1.44084286
N              0.00000000      0.00000000      0.00000000
C              1.42338653      0.00000000      1.96785902
H             -0.48626030     -0.86672789      1.89787416
H              0.48872922     -0.82369531     -0.29610147
H             -0.95270540     -0.09851616     -0.29610147
H              1.94391057     -0.87751344      1.60269742
H              1.93528625      0.89191159      1.62599031
H              1.40752416     -0.01421698      3.05139149
 Atom#            6
  Atom               X               Y               Z
H             -0.48626030     -0.86672789      1.89787416
C              0.00000000      0.00000000      1.44084286
```

```
N              0.00000000      0.00000000      0.00000000
C              1.42338653      0.00000000      1.96785902
H             -0.52202178      0.87980599      1.80690743
H              0.46862617      0.83529619     -0.29610147
H             -0.95481287      0.07540156     -0.29610147
H              1.96357598      0.84116568      1.54958427
H              1.91452219     -0.92278526      1.68207074
H              1.40861876      0.08144592      3.04843517
 Atom#             7
  Atom               X               Y               Z
H              0.73050930     -0.58944902     -0.34663973
N              0.00000000      0.00000000      0.00000000
C              0.00000000      0.00000000      1.44084286
H             -0.87404490     -0.32011104     -0.36473314
H             -0.79508214      0.64155293      1.80219079
H             -0.15801825     -1.00939572      1.80219079
H              0.95326447      0.36771031      1.80219079
 Atom#             8
  Atom               X               Y               Z
H             -0.87404490     -0.32011104     -0.36473314
N              0.00000000      0.00000000      0.00000000
C              0.00000000      0.00000000      1.44084286
H              0.73050930     -0.58944902     -0.34663973
H              0.95932437      0.35134388      1.80219079
H             -0.17544770     -1.00651259      1.80219079
H             -0.78407467      0.65509612      1.80219079
 Atom#             9
  Atom               X               Y               Z
O              1.56620070      0.37919230      3.23785258
C              1.42338653      0.00000000      1.96785902
H              0.75176248      0.68026803      3.61808458
O              2.35288801     -0.36180632      1.33520098
C              0.00000000      0.00000000      1.44084286
H             -0.00222494     -0.32209824      0.40616133
H             -0.60406692     -0.67858912      2.03164044
H             -0.41037651      1.00075379      1.50851356
 Atom#            10
  Atom               X               Y               Z
H              0.75176248      0.68026803      3.61808458
O              1.56620070      0.37919230      3.23785258
C              1.42338653      0.00000000      1.96785902
H              2.37886126     -0.33411421      1.58087862
H              0.70717887     -0.81134180      1.91135348
H              1.06777838      0.83714620      1.37860065
 ***********************************
  The DatabaseInform Matrix for each atom
 ***********************************
 Atom#             1
              1           2           3
   1     -0.627965    0.778242    0.000000
   2     -0.778242   -0.627965   -0.000000
   3     -0.000000   -0.000000    1.000000

 Atom#             2
              1           2           3
   1      0.000000    1.000000    0.000000
   2      1.000000   -0.000000    0.000000
   3      0.000000    0.000000   -1.000000

 Atom#             3
              1           2           3
   1     -0.162483    0.980878    0.107132
   2     -0.940310   -0.186830    0.284451
   3      0.299027   -0.054519    0.952686
```

```
Atom#             4
              1           2           3
   1      -0.162483    0.595250   -0.786941
   2      -0.940310    0.148287    0.306315
   3       0.299027    0.789739    0.535625

Atom#             5
              1           2           3
   1       0.860010   -0.171916    0.480444
   2       0.510276    0.289745   -0.809732
   3      -0.000000   -0.941537   -0.336909

Atom#             6
              1           2           3
   1       0.866107   -0.199226    0.458441
   2      -0.499835   -0.336286    0.798171
   3      -0.004849   -0.920446   -0.390840

Atom#             7
              1           2           3
   1      -0.627965    0.269601   -0.730052
   2      -0.778242   -0.217541    0.589080
   3       0.000000    0.938079    0.346423

Atom#             8
              1           2           3
   1       0.343739   -0.339163    0.875678
   2      -0.939065   -0.123900    0.320632
   3       0.000250    0.932533    0.361086

Atom#             9
              1           2           3
   1       0.203818   -0.973129   -0.107132
   2       0.931567    0.226429   -0.284451
   3      -0.301066    0.041824   -0.952686

Atom#            10
              1           2           3
   1      -0.203818   -0.469301    0.859194
   2      -0.931567   -0.176917   -0.317621
   3       0.301066   -0.865134   -0.401127

*********************************
        CART AfteR ROT
*********************************
Atom#             1
  Atom                X               Y               Z
N             -0.00000000      0.00000000      0.00000000
C             -0.00000000     -0.00000000      1.44084286
H              0.73050930     -0.58944902     -0.34663973
H             -0.87404490     -0.32011104     -0.36473314
C              1.42338653     -0.00000000      1.96785902
H             -0.52202178      0.87980599      1.80690743
H             -0.48626030     -0.86672789      1.89787416
H              1.40752011     -0.00000000      3.05140244
H              1.93964311      0.88478395      1.61422314
H              1.93966847     -0.88478388      1.61415464
 Atom#             2
  Atom                X               Y               Z
C             -0.00000000     -0.00000000      1.44084286
N              0.00000000      0.00000000     -0.00000000
C              1.42338653     -0.00000000      1.96785902
H             -0.52202178      0.87980599      1.80690743
```

```
H             -0.48626030      -0.86672789       1.89787416
O              1.56620070       0.37919230       3.23785258
O              2.35288801      -0.36180632       1.33520098
H              0.73050930      -0.58944902      -0.34663973
H             -0.87404490      -0.32011104      -0.36473314
H              2.53932814       0.31941883       3.40629958
 Atom#             3
  Atom               X                Y                Z
C              1.42338653      -0.00000000       1.96785902
O              1.56620070       0.37919230       3.23785258
O              2.35288801      -0.36180632       1.33520098
C             -0.00000000       0.00000000       1.44084286
N             -0.00000000       0.00000000      -0.00000000
H              0.75176248       0.68026803       3.61808458
H             -0.52202178       0.87980599       1.80690743
H             -0.48626030      -0.86672789       1.89787416
H             -0.95777357       0.00000000      -0.29610147
H              0.40141812      -0.86960710      -0.29610147
 Atom#             4
  Atom               X                Y                Z
O              2.35288801      -0.36180632       1.33520098
C              1.42338653       0.00000000       1.96785902
O              1.56620070       0.37919230       3.23785258
C              0.00000000       0.00000000       1.44084286
H              0.65316574       0.62862176       3.52608235
H             -0.66879662       0.37530236       2.20646504
H             -0.28673497      -1.01059766       1.17457750
H             -0.06099919       0.63521777       0.56490118
 Atom#             5
  Atom               X                Y                Z
H             -0.52202178       0.87980599       1.80690743
C             -0.00000000       0.00000000       1.44084286
N             -0.00000000       0.00000000      -0.00000000
C              1.42338653       0.00000000       1.96785902
H             -0.48626030      -0.86672789       1.89787416
H              0.48872922      -0.82369531      -0.29610147
H             -0.95270540      -0.09851616      -0.29610147
H              1.94391057      -0.87751344       1.60269742
H              1.93528625       0.89191159       1.62599031
H              1.40752416      -0.01421698       3.05139149
 Atom#             6
  Atom               X                Y                Z
H             -0.48707570      -0.86598074       1.89431168
C              0.01103837       0.00126345       1.46964890
N             -0.03669094      -0.06748657       0.03123884
C              1.45098101       0.04267375       1.94779700
H             -0.47105478       0.88319877       1.90129715
H              0.43119953       0.74667241      -0.31980238
H             -0.99988359       0.00589677      -0.23691020
H              1.94765803       0.90765413       1.52419460
H              1.96283998      -0.85862697       1.63144402
H              1.47083672       0.10839327       3.02936209
 Atom#             7
  Atom               X                Y                Z
H              0.73050930      -0.58944902      -0.34663973
N              0.00000000       0.00000000       0.00000000
C              0.00000000       0.00000000       1.44084286
H             -0.87404490      -0.32011104      -0.36473314
H             -0.79508214       0.64155293       1.80219079
H             -0.15801825      -1.00939572       1.80219079
H              0.95326447       0.36771031       1.80219079
 Atom#             8
  Atom               X                Y                Z
H             -0.87407016      -0.32011678      -0.36417988
```

```
N             0.00215639      0.00071617     -0.00286791
C            -0.01917809     -0.00670999      1.43779785
H             0.73180217     -0.58207056     -0.35984482
H             0.93473778      0.34256487      1.81511691
H            -0.20009206     -1.01503466      1.79131350
H            -0.80842936      0.64668007      1.79087037
 Atom#            9
  Atom               X               Y               Z
O             1.56620070      0.37919230      3.23785258
C             1.42338653      0.00000000      1.96785902
H             0.75176248      0.68026803      3.61808458
O             2.35288801     -0.36180632      1.33520098
C            -0.00000000      0.00000000      1.44084286
H            -0.00222494     -0.32209824      0.40616133
H            -0.60406692     -0.67858912      2.03164044
H            -0.41037651      1.00075379      1.50851356
 Atom#           10
  Atom               X               Y               Z
H             0.75176248      0.68026803      3.61808458
O             1.56620070      0.37919230      3.23785258
C             1.42338653     -0.00000000      1.96785902
H             2.37886126     -0.33411421      1.58087862
H             1.06784948      0.83712972      1.37859757
H             0.70710779     -0.81132520      1.91135652
 *********************************
  CART Diff error
 *********************************
 Atom#            1
  Atom               X               Y               Z
N            -0.00000000      0.00000000      0.00000000
C            -0.00000000     -0.00000000      0.00000000
H             0.00000000     -0.00000000      0.00000000
H            -0.00000000     -0.00000000      0.00000000
C             0.00000000     -0.00000000     -0.00000000
H             0.00000000      0.00000000     -0.00000000
H            -0.00000000     -0.00000000     -0.00000000
H             0.00000000     -0.00000000     -0.00000000
H            -0.00000000      0.00000000     -0.00000000
H             0.00000000     -0.00000000     -0.00000000
 Atom#            2
  Atom               X               Y               Z
C            -0.00000000     -0.00000000     -0.00000000
N             0.00000000      0.00000000     -0.00000000
C            -0.00000000     -0.00000000     -0.00000000
H             0.00000000      0.00000000     -0.00000000
H            -0.00000000     -0.00000000     -0.00000000
O            -0.00000000      0.00000000     -0.00000000
O            -0.00000000     -0.00000000     -0.00000000
H             0.00000000      0.00000000      0.00000000
H             0.00000000     -0.00000000     -0.00000000
H            -0.00000000      0.00000000     -0.00000000
 Atom#            3
  Atom               X               Y               Z
C            -0.00000000     -0.00000000     -0.00000000
O            -0.00000000     -0.00000000     -0.00000000
O            -0.00000000      0.00000000     -0.00000000
C            -0.00000000      0.00000000     -0.00000000
N            -0.00000000      0.00000000     -0.00000000
H            -0.00000000     -0.00000000     -0.00000000
H            -0.00000000      0.00000000     -0.00000000
H            -0.00000000      0.00000000     -0.00000000
H            -0.00000000      0.00000000     -0.00000000
H            -0.00000000      0.00000000     -0.00000000
 Atom#            4
```

```
  Atom                X              Y              Z
O              0.00000000     0.00000000     0.00000000
C              0.00000000     0.00000000     0.00000000
O              0.00000000     0.00000000     0.00000000
C              0.00000000     0.00000000     0.00000000
H              0.00000000     0.00000000     0.00000000
H              0.00000000     0.00000000     0.00000000
H              0.00000000    -0.00000000     0.00000000
H              0.00000000    -0.00000000    -0.00000000
 Atom#             5
  Atom                X              Y              Z
H             -0.00000000     0.00000000    -0.00000000
C             -0.00000000     0.00000000    -0.00000000
N             -0.00000000     0.00000000    -0.00000000
C             -0.00000000     0.00000000    -0.00000000
H             -0.00000000     0.00000000    -0.00000000
H             -0.00000000     0.00000000    -0.00000000
H             -0.00000000     0.00000000    -0.00000000
H             -0.00000000     0.00000000    -0.00000000
H             -0.00000000     0.00000000    -0.00000000
H             -0.00000000     0.00000000    -0.00000000
 Atom#             6
  Atom                X              Y              Z
H             -0.00081540     0.00074714    -0.00356248
C              0.01103837     0.00126345     0.02880604
N             -0.03669094    -0.06748657     0.03123884
C              0.02759448     0.04267375    -0.02006202
H              0.05096700     0.00339277     0.09438972
H             -0.03742664    -0.08862377    -0.02370091
H             -0.04507072    -0.06950479     0.05919127
H             -0.01591795     0.06648845    -0.02538967
H              0.04831779     0.06415829    -0.05062672
H              0.06221796     0.02694736    -0.01907308
 Atom#             7
  Atom                X              Y              Z
H             -0.00000000     0.00000000    -0.00000000
N              0.00000000     0.00000000     0.00000000
C              0.00000000     0.00000000    -0.00000000
H              0.00000000     0.00000000     0.00000000
H              0.00000000     0.00000000     0.00000000
H              0.00000000     0.00000000    -0.00000000
H              0.00000000    -0.00000000    -0.00000000
 Atom#             8
  Atom                X              Y              Z
H             -0.00002526    -0.00000573     0.00055326
N              0.00215639     0.00071617    -0.00286791
C             -0.01917809    -0.00670999    -0.00304501
H              0.00129286     0.00737846    -0.01320508
H             -0.02458659    -0.00877901     0.01292612
H             -0.02464436    -0.00852207    -0.01087729
H             -0.02435469    -0.00841606    -0.01132042
 Atom#             9
  Atom                X              Y              Z
O             -0.00000000     0.00000000     0.00000000
C             -0.00000000     0.00000000     0.00000000
H             -0.00000000     0.00000000     0.00000000
O             -0.00000000     0.00000000     0.00000000
C             -0.00000000     0.00000000     0.00000000
H             -0.00000000     0.00000000     0.00000000
H             -0.00000000     0.00000000     0.00000000
H             -0.00000000     0.00000000     0.00000000
 Atom#            10
  Atom                X              Y              Z
H              0.00000000    -0.00000000    -0.00000000
```

```
O              0.00000000      -0.00000000      -0.00000000
C              0.00000000      -0.00000000      -0.00000000
H              0.00000000      -0.00000000      -0.00000000
H              0.36067062       1.64847152      -0.53275591
H             -0.36067059      -1.64847141       0.53275587
 *********************************
        CART after TTCORR
 *********************************
 Atom#             1
  Atom               X               Y               Z
N              0.00000000       0.00000000       0.00000000
C             -0.00000000      -0.00000000       1.44084286
H              0.73050930      -0.58944902      -0.34663973
H             -0.87404490      -0.32011104      -0.36473314
C              1.42338653      -0.00000000       1.96785902
H             -0.52202178       0.87980599       1.80690743
H             -0.48626030      -0.86672789       1.89787416
H              1.40752011      -0.00000000       3.05140244
H              1.93964311       0.88478395       1.61422314
H              1.93966847      -0.88478388       1.61415464
 Atom#             2
  Atom               X               Y               Z
C              0.00000000       0.00000000       1.44084286
N              0.00000000       0.00000000      -0.00000000
C              1.42338653      -0.00000000       1.96785902
H             -0.52202178       0.87980599       1.80690743
H             -0.48626030      -0.86672789       1.89787416
O              1.56620070       0.37919230       3.23785258
O              2.35288801      -0.36180632       1.33520098
H              0.73050930      -0.58944902      -0.34663973
H             -0.87404490      -0.32011104      -0.36473314
H              2.53932814       0.31941883       3.40629958
 Atom#             3
  Atom               X               Y               Z
C              1.42338653       0.00000000       1.96785902
O              1.56620070       0.37919230       3.23785258
O              2.35288801      -0.36180632       1.33520098
C              0.00000000       0.00000000       1.44084286
N              0.00000000       0.00000000       0.00000000
H              0.75176248       0.68026803       3.61808458
H             -0.52202178       0.87980599       1.80690743
H             -0.48626030      -0.86672789       1.89787416
H             -0.95777357       0.00000000      -0.29610147
H              0.40141812      -0.86960710      -0.29610147
 Atom#             4
  Atom               X               Y               Z
O              2.35288801      -0.36180632       1.33520098
C              1.42338653      -0.00000000       1.96785902
O              1.56620070       0.37919230       3.23785258
C              0.00000000      -0.00000000       1.44084286
H              0.65316574       0.62862176       3.52608235
H             -0.66879662       0.37530236       2.20646504
H             -0.28673497      -1.01059766       1.17457750
H             -0.06099919       0.63521777       0.56490118
 Atom#             5
  Atom               X               Y               Z
H             -0.52202178       0.87980599       1.80690743
C              0.00000000      -0.00000000       1.44084286
N             -0.00000000       0.00000000      -0.00000000
C              1.42338653       0.00000000       1.96785902
H             -0.48626030      -0.86672789       1.89787416
H              0.48872922      -0.82369531      -0.29610147
H             -0.95270540      -0.09851616      -0.29610147
H              1.94391057      -0.87751344       1.60269742
```

```
H             1.93528625      0.89191159      1.62599031
H             1.40752416     -0.01421698      3.05139149
 Atom#            6
  Atom               X               Y               Z
H            -0.48626030     -0.86672789      1.89787416
C             0.01185377      0.00051630      1.47321138
N            -0.03587554     -0.06823372      0.03480132
C             1.45179641      0.04192661      1.95135948
H            -0.47023938      0.88245162      1.90485963
H             0.43201493      0.74592527     -0.31623990
H            -0.99906819      0.00514962     -0.23334772
H             1.94847343      0.90690698      1.52775708
H             1.96365538     -0.85937412      1.63500650
H             1.47165212      0.10764613      3.03292457
 Atom#            7
  Atom               X               Y               Z
H             0.73050930     -0.58944902     -0.34663973
N             0.00000000      0.00000000      0.00000000
C             0.00000000     -0.00000000      1.44084286
H            -0.87404490     -0.32011104     -0.36473314
H            -0.79508214      0.64155293      1.80219079
H            -0.15801825     -1.00939572      1.80219079
H             0.95326447      0.36771031      1.80219079
 Atom#            8
  Atom               X               Y               Z
H            -0.87404490     -0.32011104     -0.36473314
N             0.00218165      0.00072190     -0.00342117
C            -0.01915283     -0.00670426      1.43724459
H             0.73182743     -0.58206482     -0.36039808
H             0.93476304      0.34257060      1.81456365
H            -0.20006680     -1.01502893      1.79076024
H            -0.80840410      0.64668580      1.79031711
 Atom#            9
  Atom               X               Y               Z
O             1.56620070      0.37919230      3.23785258
C             1.42338653      0.00000000      1.96785902
H             0.75176248      0.68026803      3.61808458
O             2.35288801     -0.36180632      1.33520098
C            -0.00000000      0.00000000      1.44084286
H            -0.00222494     -0.32209824      0.40616133
H            -0.60406692     -0.67858912      2.03164044
H            -0.41037651      1.00075379      1.50851356
 Atom#           10
  Atom               X               Y               Z
H             0.75176248      0.68026803      3.61808458
O             1.56620070      0.37919230      3.23785258
C             1.42338653      0.00000000      1.96785902
H             2.37886126     -0.33411421      1.58087862
H             1.06784948      0.83712972      1.37859757
H             0.70710779     -0.81132520      1.91135652
 *********************************
  CART Diff error after TTCORR
 *********************************
 Atom#            1
  Atom               X               Y               Z
N             0.00000000      0.00000000      0.00000000
C            -0.00000000     -0.00000000      0.00000000
H             0.00000000     -0.00000000      0.00000000
H            -0.00000000     -0.00000000      0.00000000
C             0.00000000     -0.00000000     -0.00000000
H             0.00000000      0.00000000     -0.00000000
H            -0.00000000     -0.00000000     -0.00000000
H             0.00000000     -0.00000000     -0.00000000
H            -0.00000000      0.00000000     -0.00000000
```

```
H              0.00000000      -0.00000000      -0.00000000
 Atom#             2
  Atom               X                Y                Z
C              0.00000000       0.00000000       0.00000000
N              0.00000000       0.00000000      -0.00000000
C              0.00000000      -0.00000000      -0.00000000
H              0.00000000       0.00000000      -0.00000000
H              0.00000000      -0.00000000      -0.00000000
O             -0.00000000       0.00000000       0.00000000
O             -0.00000000      -0.00000000      -0.00000000
H              0.00000000       0.00000000       0.00000000
H              0.00000000      -0.00000000      -0.00000000
H             -0.00000000       0.00000000       0.00000000
 Atom#             3
  Atom               X                Y                Z
C              0.00000000       0.00000000       0.00000000
O             -0.00000000      -0.00000000       0.00000000
O              0.00000000       0.00000000       0.00000000
C              0.00000000       0.00000000       0.00000000
N              0.00000000       0.00000000       0.00000000
H             -0.00000000      -0.00000000       0.00000000
H              0.00000000       0.00000000       0.00000000
H              0.00000000       0.00000000      -0.00000000
H              0.00000000       0.00000000      -0.00000000
H              0.00000000       0.00000000      -0.00000000
 Atom#             4
  Atom               X                Y                Z
O              0.00000000       0.00000000       0.00000000
C             -0.00000000      -0.00000000      -0.00000000
O              0.00000000       0.00000000      -0.00000000
C              0.00000000      -0.00000000      -0.00000000
H              0.00000000      -0.00000000      -0.00000000
H             -0.00000000      -0.00000000      -0.00000000
H              0.00000000      -0.00000000       0.00000000
H              0.00000000      -0.00000000      -0.00000000
 Atom#             5
  Atom               X                Y                Z
H              0.00000000       0.00000000       0.00000000
C              0.00000000      -0.00000000       0.00000000
N             -0.00000000       0.00000000      -0.00000000
C              0.00000000       0.00000000      -0.00000000
H             -0.00000000       0.00000000      -0.00000000
H              0.00000000       0.00000000      -0.00000000
H             -0.00000000       0.00000000       0.00000000
H              0.00000000       0.00000000       0.00000000
H             -0.00000000       0.00000000      -0.00000000
H             -0.00000000       0.00000000       0.00000000
 Atom#             6
  Atom               X                Y                Z
H              0.00000000       0.00000000       0.00000000
C              0.01185377       0.00051630       0.03236852
N             -0.03587554      -0.06823372       0.03480132
C              0.02840988       0.04192661      -0.01649953
H              0.05178240       0.00264563       0.09795221
H             -0.03661124      -0.08937092      -0.02013843
H             -0.04425532      -0.07025193       0.06275375
H             -0.01510255       0.06574131      -0.02182719
H              0.04913319       0.06341114      -0.04706424
H              0.06303336       0.02620021      -0.01551060
 Atom#             7
  Atom               X                Y                Z
H              0.00000000       0.00000000       0.00000000
N              0.00000000       0.00000000       0.00000000
C              0.00000000      -0.00000000       0.00000000
```

```
H              0.00000000      -0.00000000       0.00000000
H              0.00000000      -0.00000000       0.00000000
H              0.00000000      -0.00000000      -0.00000000
H              0.00000000      -0.00000000      -0.00000000
 Atom#             8
  Atom               X                Y                Z
H              0.00000000       0.00000000       0.00000000
N              0.00218165       0.00072190      -0.00342117
C             -0.01915283      -0.00670426      -0.00359827
H              0.00131812       0.00738419      -0.01375834
H             -0.02456133      -0.00877328       0.01237286
H             -0.02461910      -0.00851634      -0.01143055
H             -0.02432943      -0.00841033      -0.01187368
 Atom#             9
  Atom               X                Y                Z
O              0.00000000       0.00000000       0.00000000
C              0.00000000       0.00000000      -0.00000000
H             -0.00000000      -0.00000000      -0.00000000
O              0.00000000       0.00000000       0.00000000
C             -0.00000000       0.00000000       0.00000000
H              0.00000000       0.00000000       0.00000000
H              0.00000000       0.00000000       0.00000000
H              0.00000000       0.00000000       0.00000000
 Atom#            10
  Atom               X                Y                Z
H              0.00000000       0.00000000       0.00000000
O             -0.00000000      -0.00000000      -0.00000000
C             -0.00000000       0.00000000      -0.00000000
H             -0.00000000       0.00000000      -0.00000000
H              0.36067062       1.64847152      -0.53275591
H             -0.36067059      -1.64847141       0.53275587




+-------------------------------------------------------------------------------------------------------

|    Calculated molecule properties using the stored fragments properties
                                                |
|    (already calculated, using fragment weigths and geometry)
                                                |
+-----+--------------+--------------+--------------+--------------+--------------+--------------+---

|Atom#|   Electrons  | Potential Vne| Vee Ana/Num  | HF Coulomb J |HF Exchange K |
    Coulomb J    |  Exchange K  |  Kinetic T   |
+-----+--------------+--------------+--------------+--------------+--------------+--------------+---

|    1|    7.16495286| -167.95397335|   46.80154137|   44.37764402|   -6.64742071|
    38.96605300|   -1.23582969|   54.00960395|
|    2|    6.10120522| -147.17544847|   48.23267128|   47.81924437|   -5.27389078|
    43.47856678|   -0.93321319|   37.96874603|
|    3|    6.12592201| -156.56047131|   53.04647809|   48.88547298|   -5.37535134|
    44.46953361|   -0.95941197|   38.69709666|
|    4|    8.14248522| -232.18358599|   63.92985103|   66.55467483|   -8.20084481|
    60.21731428|   -1.86348426|   74.22550128|
|    5|    0.93931951|   -8.63995813|    3.66675627|    4.93566560|   -0.33231778|
     4.77711621|   -0.17376839|    0.59099384|
|    6|    0.93931951|   -8.63995813|    3.66675627|    4.93566560|   -0.33231778|
     4.77711621|   -0.17376839|    0.59099384|
|    7|    0.91220048|   -7.30275687|    3.00302056|    3.46406769|   -0.33865852|
     3.28368436|   -0.15827519|    0.68767134|
|    8|    0.91220048|   -7.30275687|    3.00302056|    3.46406769|   -0.33865852|
     3.28368436|   -0.15827519|    0.68767134|
```

```
|    9|    8.00976335| -231.98603328|   63.51630245|   60.87287321|   -8.18967571|
    54.70681807|   -2.02362057|   74.12111132|
|   10|    0.86921696|   -7.78151640|    3.13430818|    3.38169669|   -0.34055894|
     3.18930375|   -0.14816600|    0.79228141|
+-----+--------------+--------------+--------------+--------------+--------------+---

|Sum= |   40.11658560| -975.52645879|  292.00070605|  288.69107266|  -35.36969487|
    261.14919063|   -7.82781284|  282.37167102|
+-----+--------------+--------------+--------------+--------------+--------------+---


+-----------------------------------------------------+
| Calculated molecule properties using the stored     |
| electron density in each radial grid point          |
| NOTE: Current partitioning weight                   |
+-----+----------------+----------------+----------------+
|Atom#|   Electrons    |    Vne         | Vee            |
+-----+----------------+----------------+----------------+
|    1|      7.15774153|   -184.05976512|     55.02897510|
|    2|      6.10204839|   -147.49545687|     48.48806546|
|    3|      6.12571670|   -156.63425079|     53.17898690|
|    4|      8.12706805|   -244.31814008|     70.04259434|
|    5|      0.93062947|    -11.19956328|      4.80219154|
|    6|      0.93645619|    -11.16397690|      4.81690395|
|    7|      0.88913919|    -11.38660258|      4.72708503|
|    8|      0.91007803|    -10.30065099|      4.33723234|
|    9|      8.00787444|   -241.06096620|     68.05675198|
|   10|      0.85461607|    -11.49855370|      4.66003608|
+-----+----------------+----------------+----------------+
|Sum= |     40.04136806|  -1029.11792652|    318.13882271|
+-----+----------------+----------------+----------------+

     Nuclear repulsion (Vnn)   =    181.51074356
     ThE Total Energy          =   -282.46638410
     ThE Virial                =      2.00033542


+-----------------------------------------------------+
| Calculated molecule properties using the stored     |
| electron density in each radial grid point          |
| NOTE: Database partitioning weight                  |
+-----+----------------+----------------+----------------+
|Atom#|   Electrons    |    Vne         | Vee            |
+-----+----------------+----------------+----------------+
|    1|      7.16495286|   -184.20943042|     55.14736505|
|    2|      6.10120522|   -147.62537265|     48.54978306|
|    3|      6.12592201|   -156.77003585|     53.24855352|
|    4|      8.14248522|   -244.48499319|     70.21177286|
|    5|      0.93931951|    -11.21837242|      4.85386431|
|    6|      0.93931951|    -11.18001904|      4.83565431|
|    7|      0.91220048|    -11.41270210|      4.86389075|
|    8|      0.91220048|    -10.31592222|      4.35590087|
|    9|      8.00976335|   -241.20509460|     68.13978495|
|   10|      0.86921696|    -11.51829265|      4.74746891|
+-----+----------------+----------------+----------------+
|Sum= |     40.11658560|  -1029.94023513|    318.95403857|
+-----+----------------+----------------+----------------+

     Nuclear repulsion (Vnn)   =    181.51074356
     ThE Total Energy          =   -282.47347685
     ThE Virial                =      2.00036054


 Atom#      AIM_FD        AIM_CW            HF    |Error FD|    |Error CW|    |%Error
```

```
    FD|   |%Error CW|
   1    7.16495286    7.15774153    7.14764497    0.01730789    0.01009656
    0.24214814    0.14125709
   2    6.10120522    6.10204839    6.11995184   -0.01874662   -0.01790345
   -0.30631971   -0.29254238
   3    6.12592201    6.12571670    6.12367153    0.00225048    0.00204517
    0.03675049    0.03339780
   4    8.14248522    8.12706805    8.13922047    0.00326475   -0.01215242
    0.04011128   -0.14930694
   5    0.93931951    0.93062947    0.92410217    0.01521734    0.00652730
    1.64671636    0.70634003
   6    0.93931951    0.93645619    0.92516674    0.01415277    0.01128945
    1.52975359    1.22026147
   7    0.91220048    0.88913919    0.87210384    0.04009664    0.01703535
    4.59769143    1.95336267
   8    0.91220048    0.91007803    0.90122937    0.01097112    0.00884867
    1.21735013    0.98184418
   9    8.00976335    8.00787444    8.00382871    0.00593465    0.00404574
    0.07414761    0.05054752
  10    0.86921696    0.85461607    0.84267594    0.02654102    0.01194012
    3.14961138    1.41692949
Sum=   40.11658560   40.04136806   39.99959557    0.11699003    0.04177249
    0.29247804    0.10443228

PROGRAM> end of inputs


Program terminated normally


Job: RUN_Gly_ ended on :24-Aug-18 at 16:01:05
User: ibrahim
Cpu     time:  00h01m10s23c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h01m11s00c
```

Here is the results using the direct method,

```
Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Free format Z-Matrix for: C2H5NO2, (C1)
N
C         N         B1
C         C         B2  N         A2
O         C         B3  C         A3  N         D3
H         C         B4  N         A4  C         D4
H         C         B5  N         A5  C         D5
H         N         B6  C         A6  C         D6
H         N         B7  C         A7  C         D7
O         C         B8  C         A8  N         D8
H         O         B9  C         A9  C         D9

VARIABLES:
B1 = 1.44084286              B2 = 1.51781924              B3 = 1.18115749
B4 = 1.08653973              B5 = 1.09386650              B6 = 1.00062635
B7 = 0.99972788              B8 = 1.33306652              B9 = 0.94790958
A2 =  110.31733              A3 =  123.50442              A4 =  109.68864
A5 =  114.69657              A6 =  110.26867              A7 =  111.39727
A8 =  115.54740              A9 =  112.36145              D3 =  -21.55248
D4 =  120.68224              D5 = -119.29373              D6 =   38.90011
```

```
D7 =  159.88520               D8 =  161.62252               D9 =   -5.50811

Z MATRIX FOR: C2H5NO2, (C1)
-------------------------------------------------------------------------------------------------

   I    AN    Z1      BL                  Z2     ALPHA                Z3      BETA
              Z4
-------------------------------------------------------------------------------------------------

   1    7
   2    6     1    1.440843    (  1)
   3    6     2    1.517819    (  2)        1    110.3173    ( 10)
   4    8     3    1.181157    (  3)        2    123.5044    ( 11)        1    -21.5525    (
    18)        0
   5    1     2    1.086540    (  4)        1    109.6886    ( 12)        3    120.6822    (
    19)        0
   6    1     2    1.093866    (  5)        1    114.6966    ( 13)        3   -119.2937    (
    20)        0
   7    1     1    1.000626    (  6)        2    110.2687    ( 14)        3     38.9001    (
    21)        0
   8    1     1    0.999728    (  7)        2    111.3973    ( 15)        3    159.8852    (
    22)        0
   9    8     3    1.333067    (  8)        2    115.5474    ( 16)        1    161.6225    (
    23)        0
  10    1     9    0.947910    (  9)        3    112.3615    ( 17)        2     -5.5081    (
    24)        0
-------------------------------------------------------------------------------------------------


Cartesian coordinates for: C2H5NO2, (C1)

-------------------------------------------------------------------------------------------------------

                              COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                        X
              Y             Z
-------------------------------------------------------------------------------------------------------

   1 N          7    0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   2 C          6    0.00000000     0.00000000     1.44084286               0.00000000
        0.00000000     2.72279820
   3 C          6    1.42338653     0.00000000     1.96785902               2.68981052
        0.00000000     3.71871433
   4 O          8    2.35288801    -0.36180632     1.33520098               4.44631363
       -0.68371481     2.52316398
   5 H          1   -0.52202178     0.87980599     1.80690743              -0.98647812
        1.66259225     3.41455992
   6 H          1   -0.48626030    -0.86672789     1.89787416              -0.91889872
       -1.63787822     3.58646212
   7 H          1    0.73050930    -0.58944902    -0.34663973               1.38046242
       -1.11389713    -0.65505411
   8 H          1   -0.87404490    -0.32011104    -0.36473314              -1.65170536
       -0.60492216    -0.68924570
   9 O          8    1.56620070     0.37919230     3.23785258               2.95969016
        0.71656954     6.11865417
  10 H          1    0.75176248     0.68026803     3.61808458               1.42062510
        1.28552018     6.83718846
-------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:     181.510743561
```

```
Distance Matrix for: C2H5NO2, (C1)
             1           2           3           4           5           6
     7           8           9          10
   1      0.000000    1.440843    2.428682    2.729423    2.076410    2.142333
    1.000626    0.999728    3.616692    3.757452
   2      1.440843    0.000000    1.517819    2.382886    1.086540    1.093866
    2.018957    2.031386    2.413714    2.401727
   3      2.428682    1.517819    0.000000    1.181157    2.141163    2.098301
    2.486852    3.289627    1.333067    1.907115
   4      2.729423    2.382886    1.181157    0.000000    3.166893    2.938079
    2.347876    3.647549    2.188159    2.976757
   5      2.076410    1.086540    2.141163    3.166893    0.000000    1.749267
    2.892285    2.505942    2.580483    2.223219
   6      2.142333    1.093866    2.098301    2.938079    1.749267    0.000000
    2.568123    2.359779    2.749628    2.623932
   7      1.000626    2.018957    2.486852    2.347876    2.892285    2.568123
    0.000000    1.627103    3.805947    4.163132
   8      0.999728    2.031386    3.289627    3.647549    2.505942    2.359779
    1.627103    0.000000    4.407091    4.416655
   9      3.616692    2.413714    1.333067    2.188159    2.580483    2.749628
    3.805947    4.407091    0.000000    0.947910
  10      3.757452    2.401727    1.907115    2.976757    2.223219    2.623932
    4.163132    4.416655    0.947910    0.000000

Charge=    0, Number of electrons=   40

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       85

The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

    4124847 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:     113538 IJKJ:     113807 IJJL:     114903 IIKK:       3799
 IJJJ:       1462 IIIL:       1470 IIII:         30 IJKL:    3775838
Number of integrals in INCORE buffers:
IIKK:       2085 IJJL:      64223 IJKJ:      63543
IIKL:      64237 IJKL:    1535810

     937565 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      43244 IJKJ:      44015 IJJL:      43286 IIKK:       2740
 IJJJ:       1152 IIIL:       1158 IIII:         55 IJKL:     801915
Number of integrals in INCORE buffers:
IIKK:       3570 IJJL:      88212 IJKJ:      87691
IIKL:      88349 IJKL:     271361
Number of buffers:
IJJL:          0 IJKJ:          0
IIKL:          0 IJKL:          1
 TOTAL OF    5062412 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF               Nuclear Repulsion Energy is    181.510743561 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE   ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:   1       -463.303052879      -281.792309318
 SCF_CYCLE:   2       -464.156082549      -282.645338988   2.78967E-02
 SCF_CYCLE:   3       -464.252967556      -282.742223996   1.11306E-02
```

```
 SCF_CYCLE:    4       -464.282711412      -282.771967851   8.78492E-03
 SCF_CYCLE:    5       -464.301085806      -282.790342245   6.67293E-03
 SCF_CYCLE:    6       -464.310646467      -282.799902906   5.26956E-03
 SCF_CYCLE:    7       -464.325561119      -282.814817558                    4-POINT
 SCF_CYCLE:    8       -464.326970023      -282.816226462   2.39444E-03
 SCF_CYCLE:    9       -464.326980754      -282.816237193   1.12426E-04
 SCF_CYCLE:   10       -464.326984887      -282.816241326   7.27041E-05
 SCF_CYCLE:   11       -464.326986541      -282.816242980   4.60760E-05
 SCF_CYCLE:   12       -464.326987992      -282.816244431                    4-POINT
 SCF_CYCLE:   13       -464.326987698      -282.816244138   1.87641E-05
At termination total energy is      -282.816244  Hartrees



*** RUNing the inputfile :: INPUT_ALL_0001.dat


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: TEST

-------------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                         X
             Y             Z
-------------------------------------------------------------------------------------------------------

   1 N          7   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   2 C          6   0.00000000     0.00000000     1.44084286               0.00000000
        0.00000000     2.72279820
   3 H          1   0.73050930    -0.58944902    -0.34663973               1.38046242
       -1.11389713    -0.65505411
   4 H          1  -0.87404490    -0.32011104    -0.36473314              -1.65170536
       -0.60492216    -0.68924570
   5 C          6   1.42338653     0.00000000     1.96785902               2.68981052
        0.00000000     3.71871433
   6 H          1  -0.52202178     0.87980599     1.80690743              -0.98647812
        1.66259225     3.41455992
   7 H          1  -0.48626030    -0.86672789     1.89787416              -0.91889872
       -1.63787822     3.58646212
   8 H          1   1.40752011     0.00000000     3.05140244               2.65982733
        0.00000000     5.76631450
   9 H          1   1.93964311     0.88478395     1.61422314               3.66539399
        1.67199922     3.05043943
  10 H          1   1.93966847    -0.88478388     1.61415464               3.66544192
       -1.67199910     3.05030997
-------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:      83.927218609


Distance Matrix for: TEST
             1           2           3           4           5           6
     7           8           9          10
   1      0.000000    1.440843    1.000626    0.999728    2.428682    2.076410
    2.142333    3.360382    2.674093    2.674070
   2      1.440843    0.000000    2.018957    2.031386    1.517819    1.086540
```

```
    1.093866    2.138928    2.138953    2.138970
   3       1.000626    2.018957    0.000000    1.627103    2.486852    2.892285
    2.568123    3.514610    2.735023    2.322499
   4       0.999728    2.031386    1.627103    0.000000    3.289627    2.505942
    2.359779    4.120436    3.644843    3.485948
   5       2.428682    1.517819    2.486852    3.289627    0.000000    2.141163
    2.098301    1.083660    1.083707    1.083742
   6       2.076410    1.086540    2.892285    2.505942    2.141163    0.000000
    1.749267    2.458853    2.469199    3.034938
   7       2.142333    1.093866    2.568123    2.359779    2.098301    1.749267
    0.000000    2.380808    3.005538    2.442530
   8       3.360382    2.138928    3.514610    4.120436    1.083660    2.458853
    2.380808    0.000000    1.769599    1.769663
   9       2.674093    2.138953    2.735023    3.644843    1.083707    2.469199
    3.005538    1.769599    0.000000    1.769568
  10       2.674070    2.138970    2.322499    3.485948    1.083742    3.034938
    2.442530    1.769663    1.769568    0.000000

Charge=    0, Number of electrons=   26

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       59

 Partitioning scheme set to: BECKE
The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

     864066 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       29276 IJKJ:       30830 IJJL:       31306 IIKK:        1744
 IJJJ:         491 IIIL:         491 IIII:          18 IJKL:      769910
Number of integrals in INCORE buffers:
IIKK:         891 IJJL:       16092 IJKJ:       15832
IIKL:       15164 IJKL:      270736

     306612 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       17091 IJKJ:       18220 IJJL:       17413 IIKK:        1588
 IJJJ:         606 IIIL:         609 IIII:          41 IJKL:      251044
Number of integrals in INCORE buffers:
IIKK:        1711 IJJL:       25122 IJKJ:       25236
IIKL:       24040 IJKL:      359668
 TOTAL OF    1170678 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF               Nuclear Repulsion Energy is     83.927218609 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE   ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:    1       -217.903421538      -133.976202929
 SCF_CYCLE:    2       -218.150803414      -134.223584806   1.47304E-02
 SCF_CYCLE:    3       -218.169358592      -134.242139983   4.55696E-03
 SCF_CYCLE:    4       -218.171840190      -134.244621581   2.08161E-03
 SCF_CYCLE:    5       -218.172279297      -134.245060688   9.12731E-04
 SCF_CYCLE:    6       -218.172372515      -134.245153906   4.87522E-04
 SCF_CYCLE:    7       -218.172394384      -134.245175775   2.28158E-04
 SCF_CYCLE:    8       -218.172399849      -134.245181240   1.25079E-04
 SCF_CYCLE:    9       -218.172404403      -134.245185794                    4-POINT
 SCF_CYCLE:   10       -218.172401795      -134.245183186   9.79975E-05
```

```
At termination total energy is      -134.245183  Hartrees


Atom            J             K             Vee           JHF           KHF
    VeeHF
       1      38.966420     -1.235821     37.730599     44.378032     -6.647433
    37.730599

Atom           Coulomb
       1      46.801953
J_total       46.801953

Atom        Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1     17.838931     18.270367     17.899836     54.009134

Ttotal:      17.838931     18.270367     17.899836     54.009134

Atom             Vne
       1    -169.307253
Vne_total:         -169.307253

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    7.1650510475
Pure Exchange, K ( 2K_ab)         =   -1.2358207734
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -6.6474328040
Kinetic energy Numerical, T      =   54.0091341428
Potential Energy Analytical, Vne = -167.9539736333
Potential Energy Numerical, Vne  = -169.3072525485
Coulomb Energy Anal/Num, Vee     =   46.8019525071
Pure Coulomb, J ( 4J_ab+ Jaa)     =   38.9664198835
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =   44.3780319141
Jaa = Kaa                        =    5.4116120306
Coulomb Numerically Over A       =   27.8336653055


PROGRAM> end of inputs


Program terminated normally


Job: C2H7N_C1_RHF_631Gd ended on :24-Aug-18 at 19:43:35
User: ibrahim
Cpu     time:  00h00m13s00c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m13s00c



*** RUNing the inputfile :: INPUT_ALL_0002.dat


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: TEST

-------------------------------------------------------------------------------------------------------

                              COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                        X
```

```
              Y              Z
--------------------------------------------------------------------------------------------------------

   1 C          6   0.00000000     0.00000000     1.44084286               0.00000000
        0.00000000     2.72279820
   2 N          7   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   3 C          6   1.42338653     0.00000000     1.96785902               2.68981052
        0.00000000     3.71871433
   4 H          1  -0.52202178     0.87980599     1.80690743              -0.98647812
        1.66259225     3.41455992
   5 H          1  -0.48626030    -0.86672789     1.89787416              -0.91889872
       -1.63787822     3.58646212
   6 O          8   1.56620070     0.37919230     3.23785258               2.95969016
        0.71656954     6.11865417
   7 O          8   2.35288801    -0.36180632     1.33520098               4.44631363
       -0.68371481     2.52316398
   8 H          1   0.73050930    -0.58944902    -0.34663973               1.38046242
       -1.11389713    -0.65505411
   9 H          1  -0.87404490    -0.32011104    -0.36473314              -1.65170536
       -0.60492216    -0.68924570
  10 H          1   2.53932814     0.31941883     3.40629958               4.79863437
        0.60361407     6.43697285
--------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:     181.273968331


Distance Matrix for: TEST
              1           2           3           4           5           6
     7           8           9          10
   1      0.000000    1.440843    1.517819    1.086540    1.093866    2.413714
    2.382886    2.018957    2.031386    3.226955
   2      1.440843    0.000000    2.428682    2.076410    2.142333    3.616692
    2.729423    1.000626    0.999728    4.260645
   3      1.517819    2.428682    0.000000    2.141163    2.098301    1.333067
    1.181157    2.486852    3.289627    1.848368
   4      1.086540    2.076410    2.141163    0.000000    1.749267    2.580483
    3.166893    2.892285    2.505942    3.499136
   5      1.093866    2.142333    2.098301    1.749267    0.000000    2.749628
    2.938079    2.568123    2.359779    3.582803
   6      2.413714    3.616692    1.333067    2.580483    2.749628    0.000000
    2.188159    3.805947    4.407091    0.989406
   7      2.382886    2.729423    1.181157    3.166893    2.938079    2.188159
    0.000000    2.347876    3.647549    2.188213
   8      2.018957    1.000626    2.486852    2.892285    2.568123    3.805947
    2.347876    0.000000    1.627103    4.264085
   9      2.031386    0.999728    3.289627    2.505942    2.359779    4.407091
    3.647549    1.627103    0.000000    5.126480
  10      3.226955    4.260645    1.848368    3.499136    3.582803    0.989406
    2.188213    4.264085    5.126480    0.000000

Charge=    0, Number of electrons=   40

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       85

 Partitioning scheme set to: BECKE
The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16
```

```
    4104066 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:     113140 IJKJ:     113572 IJJL:     114756 IIKK:       3793
 IJJJ:       1462 IIIL:       1470 IIII:         30 IJKL:    3755843
Number of integrals in INCORE buffers:
IIKK:       2085 IJJL:      64260 IJKJ:      63526
IIKL:      64095 IJKL:    1534076

     930964 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      43052 IJKJ:      43862 IJJL:      43150 IIKK:       2734
 IJJJ:       1151 IIIL:       1157 IIII:         55 IJKL:     795803
Number of integrals in INCORE buffers:
IIKK:       3570 IJJL:      88256 IJKJ:      87663
IIKL:      88149 IJKL:     269361
Number of buffers:
IJJL:          0 IJKJ:          0
IIKL:          0 IJKL:          1
 TOTAL OF    5035030 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF               Nuclear Repulsion Energy is    181.273968331 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE   ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:   1       -463.075382663      -281.801414332
 SCF_CYCLE:   2       -463.938846750      -282.664878419   2.63503E-02
 SCF_CYCLE:   3       -464.033491357      -282.759523025   1.04825E-02
 SCF_CYCLE:   4       -464.061188697      -282.787220365   8.09175E-03
 SCF_CYCLE:   5       -464.077105997      -282.803137666   6.06597E-03
 SCF_CYCLE:   6       -464.085279580      -282.811311249   4.81216E-03
 SCF_CYCLE:   7       -464.098916367      -282.824948035                 4-POINT
 SCF_CYCLE:   8       -464.098969593      -282.825001262   2.15105E-03
 SCF_CYCLE:   9       -464.098999780      -282.825031449   1.85118E-04
 SCF_CYCLE:  10       -464.099011460      -282.825043129   1.17512E-04
 SCF_CYCLE:  11       -464.099021079      -282.825052747                 4-POINT
 SCF_CYCLE:  12       -464.099019232      -282.825050900   4.77928E-05
At termination total energy is      -282.825051  Hartrees


Atom           J             K             Vee           JHF           KHF
    VeeHF
       1      43.478567     -0.933213     42.545354     47.819244     -5.273891
    42.545354

Atom           Coulomb
       1      48.232671
J_total       48.232671

Atom        Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1     12.644472     12.655040     12.669348     37.968860

Ttotal:      12.644472     12.655040     12.669348     37.968860

Atom           Vne
       1    -149.312739
Vne_total:         -149.312739

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    6.1012052215
Pure Exchange, K ( 2K_ab)         =   -0.9332131891
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -5.2738907810
```

```
Kinetic energy Numerical, T      =   37.9688599979
Potential Energy Analytical, Vne = -147.1754484683
Potential Energy Numerical, Vne  = -149.3127391196
Coulomb Energy Anal/Num, Vee     =   48.2326712819
Pure Coulomb, J ( 4J_ab+ Jaa)    =   43.4785667800
HF Coulomb, JHF ( 4J_ab+ 2Jaa)   =   47.8192443719
Jaa = Kaa                        =    4.3406775919
Coulomb Numerically Over A       =   19.3994839860


PROGRAM> end of inputs


Program terminated normally


Job: C2H5NO2_C1_RHF_631Gd ended on :24-Aug-18 at 19:44:14
User: ibrahim
Cpu     time:  00h00m38s67c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m39s00c



*** RUNing the inputfile :: INPUT_ALL_0003.dat


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: TEST

-------------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                        X
              Y              Z
-------------------------------------------------------------------------------------------------------

   1 C          6   1.42338653     0.00000000     1.96785902               2.68981052
        0.00000000     3.71871433
   2 O          8   1.56620070     0.37919230     3.23785258               2.95969016
        0.71656954     6.11865417
   3 O          8   2.35288801    -0.36180632     1.33520098               4.44631363
       -0.68371481     2.52316398
   4 C          6   0.00000000     0.00000000     1.44084286               0.00000000
        0.00000000     2.72279820
   5 N          7   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   6 H          1   0.75176248     0.68026803     3.61808458               1.42062510
        1.28552018     6.83718846
   7 H          1  -0.52202178     0.87980599     1.80690743              -0.98647812
        1.66259225     3.41455992
   8 H          1  -0.48626030    -0.86672789     1.89787416              -0.91889872
       -1.63787822     3.58646212
   9 H          1  -0.95777357     0.00000000    -0.29610147              -1.80992961
        0.00000000    -0.55955064
  10 H          1   0.40141812    -0.86960710    -0.29610147               0.75857025
       -1.64331913    -0.55955064
-------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:     181.310954311
```

```
Distance Matrix for: TEST
             1           2           3           4           5           6
     7           8           9          10
   1      0.000000    1.333067    1.181157    1.517819    2.428682    1.907115
    2.141163    2.098301    3.285642    2.631759
   2      1.333067    0.000000    2.188159    2.413714    3.616692    0.947910
    2.580483    2.749628    4.359250    3.924927
   3      1.181157    2.188159    0.000000    2.382886    2.729423    2.976757
    3.166893    2.938079    3.708441    2.593693
   4      1.517819    2.413714    2.382886    0.000000    1.440843    2.401727
    1.086540    1.093866    1.983508    1.983514
   5      2.428682    3.616692    2.729423    1.440843    0.000000    3.757452
    2.076410    2.142333    1.002500    1.002511
   6      1.907115    0.947910    2.976757    2.401727    3.757452    0.000000
    2.223219    2.623932    4.325058    4.224418
   7      2.141163    2.580483    3.166893    1.086540    2.076410    2.223219
    0.000000    1.749267    2.320902    2.887184
   8      2.098301    2.749628    2.938079    1.093866    2.142333    2.623932
    1.749267    0.000000    2.405633    2.366751
   9      3.285642    4.359250    3.708441    1.983508    1.002500    4.325058
    2.320902    2.405633    0.000000    1.613573
  10      2.631759    3.924927    2.593693    1.983514    1.002511    4.224418
    2.887184    2.366751    1.613573    0.000000

Charge=    0, Number of electrons=   40

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       85

 Partitioning scheme set to: BECKE
The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

    4086004 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:     112311 IJKJ:     113197 IJJL:     113940 IIKK:       3794
 IJJJ:       1449 IIIL:       1456 IIII:         30 IJKL:    3739827
Number of integrals in INCORE buffers:
IIKK:       2085 IJJL:      63789 IJKJ:      63182
IIKL:      63550 IJKL:    1523366

     926721 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      42686 IJKJ:      43768 IJJL:      42806 IIKK:       2735
 IJJJ:       1138 IIIL:       1146 IIII:         55 IJKL:     792387
Number of integrals in INCORE buffers:
IIKK:       3570 IJJL:      87544 IJKJ:      87219
IIKL:      87364 IJKL:     255867
Number of buffers:
IJJL:          0 IJKJ:          0
IIKL:          0 IJKL:          1
 TOTAL OF    5012725 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF               Nuclear Repulsion Energy is    181.310954311 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE   ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
```

```
 SCF_CYCLE:   1       -463.088972146       -281.778017835
 SCF_CYCLE:   2       -463.951585272       -282.640630962   2.82187E-02
 SCF_CYCLE:   3       -464.049586229       -282.738631919   1.12648E-02
 SCF_CYCLE:   4       -464.079700155       -282.768745845   8.95101E-03
 SCF_CYCLE:   5       -464.098443005       -282.787488695   6.79396E-03
 SCF_CYCLE:   6       -464.108208454       -282.797254144   5.37936E-03
 SCF_CYCLE:   7       -464.123558036       -282.812603726                    4-POINT
 SCF_CYCLE:   8       -464.125156392       -282.814202081   2.45294E-03
 SCF_CYCLE:   9       -464.125167327       -282.814213017   1.14132E-04
 SCF_CYCLE:  10       -464.125171546       -282.814217235   7.41020E-05
 SCF_CYCLE:  11       -464.125173237       -282.814218926   4.69041E-05
 SCF_CYCLE:  12       -464.125174755       -282.814220445                    4-POINT
 SCF_CYCLE:  13       -464.125174427       -282.814220116   1.92040E-05
At termination total energy is      -282.814220  Hartrees


Atom            J             K             Vee           JHF           KHF
    VeeHF
       1      44.470645     -0.959420     43.511224     48.886603     -5.375379
    43.511224

Atom           Coulomb
       1      53.047099
J_total       53.047099

Atom         Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1      12.930254     12.860021     12.902546     38.692820

Ttotal:       12.930254     12.860021     12.902546     38.692820

Atom             Vne
       1    -159.523693
Vne_total:         -159.523693

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    6.1261059386
Pure Exchange, K ( 2K_ab)         =   -0.9594200871
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -5.3753785852
Kinetic energy Numerical, T      =   38.6928204030
Potential Energy Analytical, Vne = -156.5604706098
Potential Energy Numerical, Vne  = -159.5236933970
Coulomb Energy Anal/Num, Vee     =   53.0470991061
Pure Coulomb, J ( 4J_ab+ Jaa)     =   44.4706445853
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =   48.8866030835
Jaa = Kaa                        =    4.4159584981
Coulomb Numerically Over A       =   19.3437436031


PROGRAM> end of inputs


Program terminated normally


Job: C2H5NO2_C1_RHF_631Gd ended on :24-Aug-18 at 19:44:54
User: ibrahim
Cpu     time:  00h00m39s66c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m40s00c



*** RUNing the inputfile :: INPUT_ALL_0004.dat
```

```
Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: TEST

----------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                        X
              Y              Z
----------------------------------------------------------------------------------------------------

   1 O          8   2.35288801    -0.36180632     1.33520098               4.44631363
       -0.68371481     2.52316398
   2 C          6   1.42338653     0.00000000     1.96785902               2.68981052
        0.00000000     3.71871433
   3 O          8   1.56620070     0.37919230     3.23785258               2.95969016
        0.71656954     6.11865417
   4 C          6   0.00000000     0.00000000     1.44084286               0.00000000
        0.00000000     2.72279820
   5 H          1   0.65316574     0.62862176     3.52608235               1.23430428
        1.18792287     6.66332945
   6 H          1  -0.66879662     0.37530236     2.20646504              -1.26384235
        0.70921862     4.16961433
   7 H          1  -0.28673497    -1.01059766     1.17457750              -0.54185053
       -1.90975266     2.21962964
   8 H          1  -0.06099919     0.63521777     0.56490118              -0.11527175
        1.20038753     1.06750844
----------------------------------------------------------------------------------------------------

Nuclear repulsion energy:     122.224021925


Distance Matrix for: TEST
              1           2           3           4           5           6
     7           8
   1      0.000000    1.181157    2.188159    2.382886    2.944480    3.230017
    2.722928    2.722916
   2      1.181157    0.000000    1.333067    1.517819    1.848368    2.138928
    2.138953    2.138970
   3      2.188159    1.333067    0.000000    2.413714    0.989406    2.461501
    3.101933    3.139745
   4      2.382886    1.517819    2.413714    0.000000    2.273767    1.083660
    1.083707    1.083742
   5      2.944480    1.848368    0.989406    2.273767    0.000000    1.884979
    3.016625    3.046091
   6      3.230017    2.138928    2.461501    1.083660    1.884979    0.000000
    1.769599    1.769663
   7      2.722928    2.138953    3.101933    1.083707    3.016625    1.769599
    0.000000    1.769568
   8      2.722916    2.138970    3.139745    1.083742    3.046091    1.769663
    1.769568    0.000000

Charge=    0, Number of electrons=   32

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       68

 Partitioning scheme set to: BECKE
```

```
The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

    2017609 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       65274 IJKJ:       65618 IJJL:       66535 IIKK:        2552
 IJJJ:        1037 IIIL:        1043 IIII:          24 IJKL:     1815526
Number of integrals in INCORE buffers:
IIKK:        1332 IJJL:       34689 IJKJ:       34229
IIKL:       34433 IJKL:      653524

     432582 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       23628 IJKJ:       23882 IJJL:       23653 IIKK:        1807
 IJJJ:         776 IIIL:         779 IIII:          44 IJKL:      358013
Number of integrals in INCORE buffers:
IIKK:        2278 IJJL:       47276 IJKJ:       46846
IIKL:       47049 IJKL:      784876
 TOTAL OF    2450191 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF               Nuclear Repulsion Energy is    122.224021925 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE    ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:    1       -349.147748716       -226.923726791
 SCF_CYCLE:    2       -349.864153562       -227.640131637   3.23116E-02
 SCF_CYCLE:    3       -349.947256630       -227.723234705   1.31150E-02
 SCF_CYCLE:    4       -349.974926272       -227.750904347   1.04462E-02
 SCF_CYCLE:    5       -349.992426777       -227.768404852   7.93076E-03
 SCF_CYCLE:    6       -350.001568329       -227.777546404   6.23392E-03
 SCF_CYCLE:    7       -350.018394666       -227.794372741                 4-POINT
 SCF_CYCLE:    8       -350.016674835       -227.792652910   2.80788E-03
 SCF_CYCLE:    9       -350.016704229       -227.792682304   2.31884E-04
 SCF_CYCLE:   10       -350.016716060       -227.792694135   1.55219E-04
 SCF_CYCLE:   11       -350.016725831       -227.792703906                 4-POINT
 SCF_CYCLE:   12       -350.016724517       -227.792702592   6.16014E-05
At termination total energy is      -227.792703  Hartrees


Atom            J             K             Vee           JHF           KHF
    VeeHF
       1      60.217312     -1.863492     58.353820     66.554663     -8.200843
    58.353820

Atom           Coulomb
       1      63.929855
J_total       63.929855

Atom        Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1     24.705415     24.642234     24.878383     74.226033

Ttotal:      24.705415     24.642234     24.878383     74.226033

Atom            Vne
       1    -231.797194
Vne_total:         -231.797194

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++++
```

```
Number of Electrons, N           =    8.1424657625
Pure Exchange, K ( 2K_ab)         =   -1.8634917350
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -8.2008426955
Kinetic energy Numerical, T      =   74.2260329430
Potential Energy Analytical, Vne = -232.1835909075
Potential Energy Numerical, Vne  = -231.7971943095
Coulomb Energy Anal/Num, Vee     =   63.9298549686
Pure Coulomb, J ( 4J_ab+ Jaa)     =   60.2173116265
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =   66.5546625870
Jaa = Kaa                        =    6.3373509605
Coulomb Numerically Over A       =   37.9275985828


PROGRAM> end of inputs


Program terminated normally


Job: C2H4O2_C1_RHF_631Gd ended on :24-Aug-18 at 19:45:13
User: ibrahim
Cpu     time:  00h00m19s69c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m19s00c



*** RUNing the inputfile :: INPUT_ALL_0005.dat


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: TEST

-------------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                        X
              Y             Z
-------------------------------------------------------------------------------------------------------

   1 H          1  -0.52202178     0.87980599     1.80690743              -0.98647812
        1.66259225     3.41455992
   2 C          6   0.00000000     0.00000000     1.44084286               0.00000000
        0.00000000     2.72279820
   3 N          7   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   4 C          6   1.42338653     0.00000000     1.96785902               2.68981052
        0.00000000     3.71871433
   5 H          1  -0.48626030    -0.86672789     1.89787416              -0.91889872
       -1.63787822     3.58646212
   6 H          1   0.48872922    -0.82369531    -0.29610147               0.92356430
       -1.55655844    -0.55955064
   7 H          1  -0.95270540    -0.09851616    -0.29610147              -1.80035215
       -0.18616854    -0.55955064
   8 H          1   1.94391057    -0.87751344     1.60269742               3.67345832
       -1.65825995     3.02865896
   9 H          1   1.93528625     0.89191159     1.62599031               3.65716073
        1.68546850     3.07267615
  10 H          1   1.40752416    -0.01421698     3.05139149               2.65983499
       -0.02686619     5.76629379
```

```
--------------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:      83.935233098


Distance Matrix for: TEST
             1           2           3           4           5           6
     7           8           9          10
   1      0.000000    1.086540    2.076410    2.141163    1.749267    2.888976
    2.359078    3.034913    2.463989    2.463973
   2      1.086540    0.000000    1.440843    1.517819    1.093866    1.983508
    1.983514    2.138928    2.138953    2.138970
   3      2.076410    1.440843    0.000000    2.428682    2.142333    1.002500
    1.002511    2.667856    2.680426    3.360404
   4      2.141163    1.517819    2.428682    0.000000    2.098301    2.584100
    3.283449    1.083660    1.083707    1.083742
   5      1.749267    1.093866    2.142333    2.098301    0.000000    2.401247
    2.370917    2.448056    3.005099    2.375668
   6      2.888976    1.983508    1.002500    2.584100    2.401247    0.000000
    1.613573    2.392882    2.954703    3.564428
   7      2.359078    1.983514    1.002511    3.283449    2.370917    1.613573
    0.000000    3.550022    3.607753    4.096767
   8      3.034913    2.138928    2.667856    1.083660    2.448056    2.392882
    3.550022    0.000000    1.769599    1.769663
   9      2.463989    2.138953    2.680426    1.083707    3.005099    2.954703
    3.607753    1.769599    0.000000    1.769568
  10      2.463973    2.138970    3.360404    1.083742    2.375668    3.564428
    4.096767    1.769663    1.769568    0.000000

Charge=    0, Number of electrons=   26

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       59

 Partitioning scheme set to: BECKE
The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

     887199 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      30123 IJKJ:      31373 IJJL:      31967 IIKK:       1743
 IJJJ:        501 IIIL:        501 IIII:         18 IJKL:     790973
Number of integrals in INCORE buffers:
IIKK:        891 IJJL:      16480 IJKJ:      16136
IIKL:      15693 IJKL:     280648

     312893 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      17502 IJKJ:      18404 IJJL:      17773 IIKK:       1588
 IJJJ:        616 IIIL:        619 IIII:         41 IJKL:     256350
Number of integrals in INCORE buffers:
IIKK:       1711 IJJL:      25722 IJKJ:      25632
IIKL:      24823 IJKL:     372071
 TOTAL OF    1200092 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF               Nuclear Repulsion Energy is     83.935233098 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06
```

```
          CYCLE    ELECTRONIC ENERGY       TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:    1       -217.901046920       -133.965813822
 SCF_CYCLE:    2       -218.158366283       -134.223133185   1.52595E-02
 SCF_CYCLE:    3       -218.177683980       -134.242450883   4.76579E-03
 SCF_CYCLE:    4       -218.180330274       -134.245097176   2.22264E-03
 SCF_CYCLE:    5       -218.180811521       -134.245578424   9.73581E-04
 SCF_CYCLE:    6       -218.180916317       -134.245683220   5.26773E-04
 SCF_CYCLE:    7       -218.180941403       -134.245708305   2.43399E-04
 SCF_CYCLE:    8       -218.180962171       -134.245729073                      4-POINT
 SCF_CYCLE:    9       -218.180950095       -134.245716997   2.13774E-04
At termination total energy is      -134.245717  Hartrees


Atom            J             K             Vee           JHF           KHF
    VeeHF
       1       4.777099     -0.173767      4.603332      4.935648     -0.332316
    4.603332

Atom            Coulomb
       1       3.666742
J_total         3.666742

Atom         Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1       0.199270      0.193756      0.197964      0.590990

Ttotal:        0.199270      0.193756      0.197964      0.590990

Atom            Vne
       1      -7.870855
Vne_total:           -7.870855

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    0.9393179250
Pure Exchange, K ( 2K_ab)         =   -0.1737670983
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -0.3323162223
Kinetic energy Numerical, T      =    0.5909901007
Potential Energy Analytical, Vne =   -8.6399581474
Potential Energy Numerical, Vne  =   -7.8708548747
Coulomb Energy Anal/Num, Vee     =    3.6667417499
Pure Coulomb, J ( 4J_ab+ Jaa)     =    4.7770989645
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =    4.9356480886
Jaa = Kaa                        =    0.1585491240
Coulomb Numerically Over A       =    0.3757287077


PROGRAM> end of inputs


Program terminated normally


Job: C2H7N_C1_RHF_631Gd ended on :24-Aug-18 at 19:45:25
User: ibrahim
Cpu     time:  00h00m11s63c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m12s00c



*** RUNing the inputfile :: INPUT_ALL_0006.dat


Welcome to MUNgauss - July 9, 2018 Version 7
```

```
 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: TEST

-----------------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                         X
             Y             Z
-----------------------------------------------------------------------------------------------------------

   1 H          1  -0.48626030    -0.86672789     1.89787416              -0.91889872
       -1.63787822     3.58646212
   2 C          6   0.00000000     0.00000000     1.44084286               0.00000000
        0.00000000     2.72279820
   3 N          7   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   4 C          6   1.42338653     0.00000000     1.96785902               2.68981052
        0.00000000     3.71871433
   5 H          1  -0.52202178     0.87980599     1.80690743              -0.98647812
        1.66259225     3.41455992
   6 H          1   0.46862617     0.83529619    -0.29610147               0.88557506
        1.57848092    -0.55955064
   7 H          1  -0.95481287     0.07540156    -0.29610147              -1.80433469
        0.14248828    -0.55955064
   8 H          1   1.96357598     0.84116568     1.54958427               3.71062056
        1.58957264     2.92828967
   9 H          1   1.91452219    -0.92278526     1.68207074               3.61792234
       -1.74381129     3.17865279
  10 H          1   1.40861876     0.08144592     3.04843517               2.66190347
        0.15391046     5.76070717
-----------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:      83.933362249


Distance Matrix for: TEST
             1           2           3           4           5           6
     7           8           9          10
   1      0.000000    1.093866    2.142333    2.098301    1.749267    2.936362
    2.433244    3.006643    2.411114    2.411097
   2      1.093866    0.000000    1.440843    1.517819    1.086540    1.983508
    1.983514    2.138928    2.138953    2.138970
   3      2.142333    1.440843    0.000000    2.428682    2.076410    1.002500
    1.002511    2.639015    2.710404    3.359136
   4      2.098301    1.517819    2.428682    0.000000    2.141163    2.595150
    3.284362    1.083660    1.083707    1.083742
   5      1.749267    1.086540    2.076410    2.141163    0.000000    2.325083
    2.292819    2.499181    3.033425    2.430256
   6      2.936362    1.983508    1.002500    2.595150    2.325083    0.000000
    1.613573    2.375177    3.015731    3.554969
   7      2.433244    1.983514    1.002511    3.284362    2.292819    1.613573
    0.000000    3.536940    3.625276    4.095335
   8      3.006643    2.138928    2.639015    1.083660    2.499181    2.375177
    3.536940    0.000000    1.769599    1.769663
   9      2.411114    2.138953    2.710404    1.083707    3.033425    3.015731
    3.625276    1.769599    0.000000    1.769568
  10      2.411097    2.138970    3.359136    1.083742    2.430256    3.554969
    4.095335    1.769663    1.769568    0.000000

Charge=    0, Number of electrons=   26
```

```
The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       59

 Partitioning scheme set to: BECKE
The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

     889780 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      30174 IJKJ:      31418 IJJL:      32001 IIKK:       1740
 IJJJ:        502 IIIL:        501 IIII:         18 IJKL:     793426
Number of integrals in INCORE buffers:
IIKK:        891 IJJL:      16491 IJKJ:      16154
IIKL:      15699 IJKL:     281062

     313488 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      17511 IJKJ:      18396 IJJL:      17787 IIKK:       1588
 IJJJ:        617 IIIL:        619 IIII:         41 IJKL:     256929
Number of integrals in INCORE buffers:
IIKK:       1711 IJJL:      25737 IJKJ:      25653
IIKL:      24829 IJKL:     372612
 TOTAL OF    1203268 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF               Nuclear Repulsion Energy is     83.933362249 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE    ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:   1       -217.899573327      -133.966211078
 SCF_CYCLE:   2       -218.155122907      -134.221760658   1.52152E-02
 SCF_CYCLE:   3       -218.174192099      -134.240829850   4.70431E-03
 SCF_CYCLE:   4       -218.176781933      -134.243419684   2.19982E-03
 SCF_CYCLE:   5       -218.177250021      -134.243887771   9.52724E-04
 SCF_CYCLE:   6       -218.177351734      -134.243989485   5.20729E-04
 SCF_CYCLE:   7       -218.177376093      -134.244013844   2.39070E-04
 SCF_CYCLE:   8       -218.177396346      -134.244034097                      4-POINT
 SCF_CYCLE:   9       -218.177384544      -134.244022295   2.12360E-04
At termination total energy is      -134.244022  Hartrees


Atom            J             K             Vee           JHF           KHF
    VeeHF
       1       4.731195     -0.172328      4.558866      4.888402     -0.329536
    4.558866

Atom           Coulomb
       1       3.635814
J_total        3.635814

Atom        Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1      0.199600      0.192762      0.192561      0.584923

Ttotal:       0.199600      0.192762      0.192561      0.584923

Atom            Vne
       1      -7.805313
Vne_total:           -7.805313
```

```
  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    0.9362212077
Pure Exchange, K ( 2K_ab)         =   -0.1723284183
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -0.3295360224
Kinetic energy Numerical, T      =    0.5849226569
Potential Energy Analytical, Vne =   -8.5931541860
Potential Energy Numerical, Vne  =   -7.8053134927
Coulomb Energy Anal/Num, Vee     =    3.6358137319
Pure Coulomb, J ( 4J_ab+ Jaa)     =    4.7311946586
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =    4.8884022626
Jaa = Kaa                        =    0.1572076040
Coulomb Numerically Over A       =    0.3719479049


PROGRAM> end of inputs


Program terminated normally


Job: C2H7N_C1_RHF_631Gd ended on :24-Aug-18 at 19:45:37
User: ibrahim
Cpu     time:  00h00m11s69c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m12s00c



*** RUNing the inputfile :: INPUT_ALL_0007.dat


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: TEST

-----------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                       X
             Y             Z
-----------------------------------------------------------------------------------------------------

   1 H          1   0.73050930    -0.58944902    -0.34663973               1.38046242
       -1.11389713    -0.65505411
   2 N          7   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
   3 C          6   0.00000000     0.00000000     1.44084286               0.00000000
        0.00000000     2.72279820
   4 H          1  -0.87404490    -0.32011104    -0.36473314              -1.65170536
       -0.60492216    -0.68924570
   5 H          1  -0.79508214     0.64155293     1.80219079              -1.50248738
        1.21235925     3.40564677
   6 H          1  -0.15801825    -1.00939572     1.80219079              -0.29861119
       -1.90748133     3.40564677
   7 H          1   0.95326447     0.36771031     1.80219079               1.80140864
        0.69487173     3.40564677
-----------------------------------------------------------------------------------------------------

Nuclear repulsion energy:      42.600440492
```

```
Distance Matrix for: TEST
              1           2           3           4           5           6
     7
   1      0.000000    1.000626    2.018957    1.627103    2.908654    2.362903
    2.362889
   2      1.000626    0.000000    1.440843    0.999728    2.071627    2.071652
    2.071670
   3      2.018957    1.440843    0.000000    2.031386    1.083660    1.083707
    1.083742
   4      1.627103    0.999728    2.031386    0.000000    2.372044    2.383981
    2.916799
   5      2.908654    2.071627    1.083660    2.372044    0.000000    1.769599
    1.769663
   6      2.362903    2.071652    1.083707    2.383981    1.769599    0.000000
    1.769568
   7      2.362889    2.071670    1.083742    2.916799    1.769663    1.769568
    0.000000

Charge=    0, Number of electrons=   18

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       40

 Partitioning scheme set to: BECKE
The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

     183482 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       8340 IJKJ:       8637 IJJL:       9078 IIKK:        802
 IJJJ:        194 IIIL:        194 IIII:         12 IJKL:     156225
Number of integrals in INCORE buffers:
IIKK:        402 IJJL:       4546 IJKJ:       4323
IIKL:       4193 IJKL:      52435

      71542 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       5336 IJKJ:       5623 IJJL:       5351 IIKK:        748
 IJJJ:        269 IIIL:        270 IIII:         28 IJKL:      53917
Number of integrals in INCORE buffers:
IIKK:        780 IJJL:       7236 IJKJ:       7151
IIKL:       6883 IJKL:      70608
 TOTAL OF     255024 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF               Nuclear Repulsion Energy is     42.600440492 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE    ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:    1       -137.590780320       -94.990339828
 SCF_CYCLE:    2       -137.789680229       -95.189239737   1.90274E-02
 SCF_CYCLE:    3       -137.805890701       -95.205450209   6.03033E-03
 SCF_CYCLE:    4       -137.808182367       -95.207741875   2.82643E-03
 SCF_CYCLE:    5       -137.808604191       -95.208163699   1.31109E-03
 SCF_CYCLE:    6       -137.808696194       -95.208255702   7.00445E-04
 SCF_CYCLE:    7       -137.808718120       -95.208277628   3.41503E-04
 SCF_CYCLE:    8       -137.808738113       -95.208297621                 4-POINT
 SCF_CYCLE:    9       -137.808725573       -95.208285081   3.23420E-04
```

```
At termination total energy is       -95.208285  Hartrees


Atom            J             K             Vee           JHF           KHF
    VeeHF
       1       3.283657      -0.158273      3.125384      3.464038      -0.338654
    3.125384

Atom            Coulomb
       1       3.002990
J_total        3.002990

Atom        Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1      0.227447      0.233828      0.226529      0.687805

Ttotal:       0.227447      0.233828      0.226529      0.687805

Atom            Vne
       1      -6.583226
Vne_total:           -6.583226

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    0.9122002778
Pure Exchange, K ( 2K_ab)         =   -0.1582727854
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -0.3386542079
Kinetic energy Numerical, T      =    0.6878047756
Potential Energy Analytical, Vne =   -7.3027568430
Potential Energy Numerical, Vne  =   -6.5832255905
Coulomb Energy Anal/Num, Vee     =    3.0029900015
Pure Coulomb, J ( 4J_ab+ Jaa)     =    3.2836565308
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =    3.4640379533
Jaa = Kaa                        =    0.1803814224
Coulomb Numerically Over A       =    0.3613782943


PROGRAM> end of inputs


Program terminated normally


Job: CH5N_C1_RHF_631Gd ended on :24-Aug-18 at 19:45:41
User: ibrahim
Cpu     time:  00h00m04s67c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m04s00c



*** RUNing the inputfile :: INPUT_ALL_0008.dat


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: TEST

-------------------------------------------------------------------------------------------------------

                              COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                        X
```

```
              Y              Z
--------------------------------------------------------------------------------------------------------------

   1 H          1   -0.87404490     -0.32011104     -0.36473314               -1.65170536
       -0.60492216     -0.68924570
   2 N          7    0.00000000      0.00000000      0.00000000                0.00000000
        0.00000000      0.00000000
   3 C          6    0.00000000      0.00000000      1.44084286                0.00000000
        0.00000000      2.72279820
   4 H          1    0.73050930     -0.58944902     -0.34663973                1.38046242
       -1.11389713     -0.65505411
   5 H          1    0.95932437      0.35134388      1.80219079                1.81286020
        0.66394366      3.40564677
   6 H          1   -0.17544770     -1.00651259      1.80219079               -0.33154808
       -1.90203300      3.40564677
   7 H          1   -0.78407467      0.65509612      1.80219079               -1.48168628
        1.23795217      3.40564677
--------------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:      42.600440385


Distance Matrix for: TEST
              1           2           3           4           5           6
     7
   1      0.000000    0.999728    2.031386    1.627103    2.916788    2.377971
    2.377958
   2      0.999728    0.000000    1.440843    1.000626    2.071627    2.071652
    2.071670
   3      2.031386    1.440843    0.000000    2.018957    1.083660    1.083707
    1.083742
   4      1.627103    1.000626    2.018957    0.000000    2.356888    2.369002
    2.908665
   5      2.916788    2.071627    1.083660    2.356888    0.000000    1.769599
    1.769663
   6      2.377971    2.071652    1.083707    2.369002    1.769599    0.000000
    1.769568
   7      2.377958    2.071670    1.083742    2.908665    1.769663    1.769568
    0.000000

Charge=    0, Number of electrons=   18

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       40

 Partitioning scheme set to: BECKE
The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

     183462 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       8340 IJKJ:       8637 IJJL:       9078 IIKK:        802
 IJJJ:        194 IIIL:        194 IIII:         12 IJKL:     156205
Number of integrals in INCORE buffers:
IIKK:        402 IJJL:       4546 IJKJ:       4323
IIKL:       4193 IJKL:      52437

      71530 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       5335 IJKJ:       5624 IJJL:       5351 IIKK:        748
```

```
 IJJJ:         269 IIIL:         270 IIII:          28 IJKL:       53905
Number of integrals in INCORE buffers:
IIKK:         780 IJJL:        7236 IJKJ:        7151
IIKL:        6883 IJKL:       70610
 TOTAL OF     254992 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF               Nuclear Repulsion Energy is     42.600440385 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE   ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:   1       -137.590779928       -94.990339544
 SCF_CYCLE:   2       -137.789680198       -95.189239813   1.90274E-02
 SCF_CYCLE:   3       -137.805890762       -95.205450378   6.03040E-03
 SCF_CYCLE:   4       -137.808182454       -95.207742069   2.82647E-03
 SCF_CYCLE:   5       -137.808604285       -95.208163901   1.31112E-03
 SCF_CYCLE:   6       -137.808696290       -95.208255906   7.00458E-04
 SCF_CYCLE:   7       -137.808718216       -95.208277832   3.41509E-04
 SCF_CYCLE:   8       -137.808738210       -95.208297826                    4-POINT
 SCF_CYCLE:   9       -137.808725670       -95.208285285   3.23428E-04
At termination total energy is       -95.208285  Hartrees


Atom            J             K             Vee           JHF           KHF
    VeeHF
       1       3.268000     -0.158437      3.109563      3.448276     -0.338712
    3.109563

Atom            Coulomb
       1       2.998325
J_total        2.998325

Atom        Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1      0.230469      0.231221      0.226525      0.688215

Ttotal:       0.230469      0.231221      0.226525      0.688215

Atom            Vne
       1      -6.574218
Vne_total:           -6.574218

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    0.9120293225
Pure Exchange, K ( 2K_ab)         =   -0.1584367635
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -0.3387121932
Kinetic energy Numerical, T      =    0.6882153433
Potential Energy Analytical, Vne =   -7.2924262401
Potential Energy Numerical, Vne  =   -6.5742177172
Coulomb Energy Anal/Num, Vee     =    2.9983253516
Pure Coulomb, J ( 4J_ab+ Jaa)     =    3.2680001524
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =    3.4482755821
Jaa = Kaa                        =    0.1802754296
Coulomb Numerically Over A       =    0.3612338864


PROGRAM> end of inputs


Program terminated normally


Job: CH5N_C1_RHF_631Gd ended on :24-Aug-18 at 19:45:46
```

```
User: ibrahim
Cpu     time:  00h00m04s64c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m05s00c



*** RUNing the inputfile :: INPUT_ALL_0009.dat


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: TEST

-----------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                        X
             Y             Z
-----------------------------------------------------------------------------------------------------

   1 O          8   1.56620070     0.37919230     3.23785258               2.95969016
         0.71656954     6.11865417
   2 C          6   1.42338653     0.00000000     1.96785902               2.68981052
         0.00000000     3.71871433
   3 H          1   0.75176248     0.68026803     3.61808458               1.42062510
         1.28552018     6.83718846
   4 O          8   2.35288801    -0.36180632     1.33520098               4.44631363
        -0.68371481     2.52316398
   5 C          6   0.00000000     0.00000000     1.44084286               0.00000000
         0.00000000     2.72279820
   6 H          1  -0.00222494    -0.32209824     0.40616133              -0.00420453
        -0.60867742     0.76753362
   7 H          1  -0.60406692    -0.67858912     2.03164044              -1.14152095
        -1.28234749     3.83924375
   8 H          1  -0.41037651     1.00075379     1.50851356              -0.77549915
         1.89115044     2.85067729
-----------------------------------------------------------------------------------------------------

Nuclear repulsion energy:     122.214452707


Distance Matrix for: TEST
             1           2           3           4           5           6
     7           8
   1      0.000000    1.333067    0.947910    2.188159    2.413714    3.312136
    2.698872    2.698853
   2      1.333067    0.000000    1.907115    1.181157    1.517819    2.138928
    2.138953    2.138970
   3      0.947910    1.907115    0.000000    2.976757    2.401727    3.448142
    2.490295    2.429726
   4      2.188159    1.181157    2.976757    0.000000    2.382886    2.532044
    3.054335    3.085812
   5      2.413714    1.517819    2.401727    2.382886    0.000000    1.083660
    1.083707    1.083742
   6      3.312136    2.138928    3.448142    2.532044    1.083660    0.000000
    1.769599    1.769663
   7      2.698872    2.138953    2.490295    3.054335    1.083707    1.769599
    0.000000    1.769568
   8      2.698853    2.138970    2.429726    3.085812    1.083742    1.769663
    1.769568    0.000000
```

```
Charge=    0, Number of electrons=   32

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       68

 Partitioning scheme set to: BECKE
The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

    2016067 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      65204 IJKJ:      65543 IJJL:      66540 IIKK:       2555
 IJJJ:       1037 IIIL:       1043 IIII:         24 IJKL:    1814121
Number of integrals in INCORE buffers:
IIKK:       1332 IJJL:      34694 IJKJ:      34199
IIKL:      34436 IJKL:     653603

     432221 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:      23612 IJKJ:      23852 IJJL:      23629 IIKK:       1807
 IJJJ:        773 IIIL:        778 IIII:         44 IJKL:     357726
Number of integrals in INCORE buffers:
IIKK:       2278 IJJL:      47269 IJKJ:      46807
IIKL:      47049 IJKL:     784963
 TOTAL OF    2448288 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF               Nuclear Repulsion Energy is    122.214452707 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE   ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:   1       -349.154741317      -226.940288610
 SCF_CYCLE:   2       -349.862343544      -227.647890837   3.25505E-02
 SCF_CYCLE:   3       -349.944259385      -227.729806678   1.31496E-02
 SCF_CYCLE:   4       -349.971468451      -227.757015743   1.06281E-02
 SCF_CYCLE:   5       -349.988791020      -227.774338313   8.06894E-03
 SCF_CYCLE:   6       -349.997923860      -227.783471152   6.36537E-03
 SCF_CYCLE:   7       -350.013106898      -227.798654191                 4-POINT
 SCF_CYCLE:   8       -350.013249493      -227.798796786   2.87013E-03
 SCF_CYCLE:   9       -350.013263223      -227.798810515   1.55083E-04
 SCF_CYCLE:  10       -350.013268538      -227.798815831   1.01305E-04
 SCF_CYCLE:  11       -350.013272483      -227.798819775                 4-POINT
 SCF_CYCLE:  12       -350.013272100      -227.798819393   3.91899E-05
At termination total energy is      -227.798819  Hartrees


Atom           J             K             Vee           JHF           KHF
    VeeHF
       1      54.706447     -2.023613     52.682834     60.872489     -8.189655
    52.682834

Atom           Coulomb
       1      63.515892
J_total       63.515892

Atom        Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1     24.705261     24.959704     24.455530     74.120495
```

```
Ttotal:      24.705261     24.959704     24.455530     74.120495

Atom           Vne
       1    -231.650205
Vne_total:          -231.650205

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    8.0096836596
Pure Exchange, K ( 2K_ab)         =   -2.0236129019
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -8.1896546098
Kinetic energy Numerical, T      =   74.1204946192
Potential Energy Analytical, Vne = -231.9860318506
Potential Energy Numerical, Vne  = -231.6502053280
Coulomb Energy Anal/Num, Vee     =   63.5158922613
Pure Coulomb, J ( 4J_ab+ Jaa)     =   54.7064470025
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =   60.8724887104
Jaa = Kaa                        =    6.1660417079
Coulomb Numerically Over A       =   37.6294775256


PROGRAM> end of inputs


Program terminated normally


Job: C2H4O2_C1_RHF_631Gd ended on :24-Aug-18 at 19:46:05
User: ibrahim
Cpu     time:  00h00m19s37c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m19s00c



*** RUNing the inputfile :: INPUT_ALL_0010.dat


Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is an asymmetric top.
Point group: C1
Cartesian coordinates for: TEST

-------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                        X
              Y              Z
-------------------------------------------------------------------------------------------------

   1 H          1   0.75176248     0.68026803     3.61808458               1.42062510
        1.28552018     6.83718846
   2 O          8   1.56620070     0.37919230     3.23785258               2.95969016
        0.71656954     6.11865417
   3 C          6   1.42338653     0.00000000     1.96785902               2.68981052
        0.00000000     3.71871433
   4 H          1   2.37886126    -0.33411421     1.58087862               4.49539595
       -0.63138430     2.98742741
   5 H          1   0.70717887    -0.81134180     1.91135348               1.33637428
       -1.53321369     3.61193435
   6 H          1   1.06777838     0.83714620     1.37860065               2.01780856
        1.58197694     2.60517747
```

```
-----------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:      41.946468146


Distance Matrix for: TEST
              1           2           3           4           5           6
   1      0.000000    0.947910    1.907115    2.797611    2.267117    2.267105
   2      0.947910    0.000000    1.333067    1.978582    1.978608    1.978627
   3      1.907115    1.333067    0.000000    1.083660    1.083707    1.083742
   4      2.797611    1.978582    1.083660    0.000000    1.769599    1.769663
   5      2.267117    1.978608    1.083707    1.769599    0.000000    1.769568
   6      2.267105    1.978627    1.083742    1.769663    1.769568    0.000000

Charge=    0, Number of electrons=   18

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-31G(d) Basis Set - Total number of basis functions:       38

 Partitioning scheme set to: BECKE
The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-31G(d)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

     202149 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       10353 IJKJ:       10370 IJJL:       10576 IIKK:         756
 IJJJ:         264 IIIL:         264 IIII:          12 IJKL:      169554
Number of integrals in INCORE buffers:
IIKK:         378 IJJL:        5294 IJKJ:        5186
IIKL:        5186 IJKL:       56669

      59744 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:        4891 IJKJ:        4939 IJJL:        4865 IIKK:         645
 IJJJ:         265 IIIL:         265 IIII:          26 IJKL:       43848
Number of integrals in INCORE buffers:
IIKK:         703 IJJL:        7740 IJKJ:        7662
IIKL:        7644 IJKL:       71391
 TOTAL OF     261893 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF               Nuclear Repulsion Energy is     41.946468146 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE    ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
 SCF_CYCLE:    1       -156.599321711       -114.652853564
 SCF_CYCLE:    2       -156.936416265       -114.989948119   3.10786E-02
 SCF_CYCLE:    3       -156.968403705       -115.021935559   9.05486E-03
 SCF_CYCLE:    4       -156.974430451       -115.027962305   5.34933E-03
 SCF_CYCLE:    5       -156.976003710       -115.029535564   2.57505E-03
 SCF_CYCLE:    6       -156.976488557       -115.030020410   1.63418E-03
 SCF_CYCLE:    7       -156.976798318       -115.030330172                 4-POINT
 SCF_CYCLE:    8       -156.976733251       -115.030265104   6.30541E-04
 SCF_CYCLE:    9       -156.976733281       -115.030265135   1.01697E-05
At termination total energy is      -115.030265  Hartrees


Atom           J             K             Vee           JHF           KHF
    VeeHF
```

```
       1       3.189251      -0.148162       3.041089       3.381640      -0.340551
    3.041089

Atom            Coulomb
       1        3.134249
J_total         3.134249

Atom         Kinetic(x)    Kinetic(y)    Kinetic(z)       Total
       1       0.275231      0.259927      0.257344      0.792503

Ttotal:        0.275231      0.259927      0.257344      0.792503

Atom            Vne
       1       -6.886800
Vne_total:            -6.886800

  Atomic properties for atom #    1
  ++++++++++++++++++++++++++++++++++++++
Number of Electrons, N           =    0.8692118318
Pure Exchange, K ( 2K_ab)         =   -0.1481619415
HF Exchange, KHF ( 2K_ab+ Kaa)    =   -0.3405505985
Kinetic energy Numerical, T      =    0.7925026501
Potential Energy Analytical, Vne =   -7.7815164006
Potential Energy Numerical, Vne  =   -6.8868003991
Coulomb Energy Anal/Num, Vee     =    3.1342492136
Pure Coulomb, J ( 4J_ab+ Jaa)     =    3.1892511907
HF Coulomb, JHF ( 4J_ab+ 2Jaa)    =    3.3816398477
Jaa = Kaa                        =    0.1923886570
Coulomb Numerically Over A       =    0.3393663631


PROGRAM> end of inputs


Program terminated normally


Job: CH4O_C1_RHF_631Gd ended on :24-Aug-18 at 19:46:09
User: ibrahim
Cpu     time:  00h00m03s97c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m04s00c
```

```
+-------------------------------------------------------------------------------------------------

|    Calculated molecule properties using the stored fragments properties
                                                |
|    (already calculated, using fragment weigths and geometry)
                                                |
+-----+-------------+-------------+-------------+-------------+-------------+-------------+---

|Atom#|   Electrons  | Potential Vne| Vee Ana/Num  | HF Coulomb J |HF Exchange K |
    Coulomb J   |  Exchange K  |  Kinetic T   |
+-----+-------------+-------------+-------------+-------------+-------------+-------------+---

|    1|    7.16505105| -167.95397363|   46.80195251|   44.37803191|   -6.64743280|
    38.96641988|   -1.23582077|   54.00913414|
|    2|    6.10120522| -147.17544847|   48.23267128|   47.81924437|   -5.27389078|
    43.47856678|   -0.93321319|   37.96886000|
|    3|    6.12610594| -156.56047061|   53.04709911|   48.88660308|   -5.37537859|
```

```
    44.47064459|    -0.95942009|    38.69282040|
|     4|     8.14246576| -232.18359091|    63.92985497|    66.55466259|    -8.20084270|
    60.21731163|    -1.86349173|    74.22603294|
|     5|     0.93931793|    -8.63995815|     3.66674175|     4.93564809|    -0.33231622|
     4.77709896|    -0.17376710|     0.59099010|
|     6|     0.93622121|    -8.59315419|     3.63581373|     4.88840226|    -0.32953602|
     4.73119466|    -0.17232842|     0.58492266|
|     7|     0.91220028|    -7.30275684|     3.00299000|     3.46403795|    -0.33865421|
     3.28365653|    -0.15827279|     0.68780478|
|     8|     0.91202932|    -7.29242624|     2.99832535|     3.44827558|    -0.33871219|
     3.26800015|    -0.15843676|     0.68821534|
|     9|     8.00968366| -231.98603185|    63.51589226|    60.87248871|    -8.18965461|
    54.70644700|    -2.02361290|    74.12049462|
|    10|     0.86921183|    -7.78151640|     3.13424921|     3.38163985|    -0.34055060|
     3.18925119|    -0.14816194|     0.79250265|
+-----+--------------+--------------+--------------+--------------+--------------+--------------+---

|Sum= |    40.11349219| -975.46932729|   291.96559017|   288.62903440|   -35.36696872|
    261.08859137|    -7.82652569|   282.36177763|
+-----+--------------+--------------+--------------+--------------+--------------+--------------+---


+-----------------------------------------------------------+
| Calculated molecule properties using the stored           |
| electron density in each radial grid point                |
| NOTE: Current partitioning weight                         |
+-----+----------------+----------------+----------------+
|Atom#|   Electrons    |     Vne        |  Vee           |
+-----+----------------+----------------+----------------+
|     1|       7.15777802|    -184.04853311|      55.02393274|
|     2|       6.10204839|    -147.47178143|      48.47777008|
|     3|       6.12586589|    -156.61882673|      53.16678291|
|     4|       8.12711617|    -244.30502587|      70.03631834|
|     5|       0.93063294|     -11.19681140|       4.80074561|
|     6|       0.92887027|     -11.15165216|       4.76876720|
|     7|       0.88913766|     -11.38502637|       4.72612464|
|     8|       0.90912020|     -10.29924806|       4.33377802|
|     9|       8.00779253|    -241.04401767|      68.04177982|
|    10|       0.85461039|     -11.49622610|       4.65863240|
+-----+----------------+----------------+----------------+
|Sum= |      40.03297246|   -1029.01714890|     318.03463175|
+-----+----------------+----------------+----------------+

     Nuclear repulsion (Vnn)   =     181.51074356
     ThE Total Energy          =    -282.47696467
     ThE Virial                =       2.00040794


+-----------------------------------------------------------+
| Calculated molecule properties using the stored           |
| electron density in each radial grid point                |
| NOTE: Database partitioning weight                        |
+-----+----------------+----------------+----------------+
|Atom#|   Electrons    |     Vne        |  Vee           |
+-----+----------------+----------------+----------------+
|     1|       7.16505105|    -184.20665368|      55.14699849|
|     2|       6.10120522|    -147.61809518|      48.54746317|
|     3|       6.12610594|    -156.76366675|      53.24110912|
|     4|       8.14246576|    -244.47968969|      70.20929978|
|     5|       0.93931793|     -11.21750473|       4.85323402|
|     6|       0.93622121|     -11.17199079|       4.81545229|
|     7|       0.91220028|     -11.41231399|       4.86341981|
|     8|       0.91202932|     -10.31552881|       4.35566492|
|     9|       8.00968366|    -241.19764270|      68.12952172|
```

```
|   10|       0.86921183|     -11.51729460|       4.74661561|
+-----+-----------------+-----------------+-----------------+
|Sum= |      40.11349219|   -1029.90038093|     318.90877893|
+-----+-----------------+-----------------+-----------------+

     Nuclear repulsion (Vnn)   =     181.51074356
     ThE Total Energy          =    -282.48604952
     ThE Virial                =       2.00044012


 Atom#       AIM_FD         AIM_CW             HF     |Error FD|     |Error CW|    |%Error
    FD|    |%Error CW|
   1     7.16505105     7.15777802     7.14766342     0.01738763     0.01011460
    0.24326312     0.14150921
   2     6.10120522     6.10204839     6.11995184    -0.01874662    -0.01790345
   -0.30631971    -0.29254238
   3     6.12610594     6.12586589     6.12381199     0.00229395     0.00205390
    0.03745947     0.03353960
   4     8.14246576     8.12711617     8.13943840     0.00302737    -0.01232223
    0.03719379    -0.15138918
   5     0.93931793     0.93063294     0.92410932     0.01520860     0.00652362
    1.64575819     0.70593633
   6     0.93622121     0.92887027     0.92516871     0.01105249     0.00370156
    1.19464621     0.40009548
   7     0.91220028     0.88913766     0.87210041     0.04009987     0.01703725
    4.59807925     1.95358780
   8     0.91202932     0.90912020     0.90122908     0.01080024     0.00789112
    1.19839033     0.87559554
   9     8.00968366     8.00779253     8.00374504     0.00593862     0.00404749
    0.07419805     0.05056994
  10     0.86921183     0.85461039     0.84266991     0.02654192     0.01194048
    3.14974090     1.41698156
Sum=    40.11349219    40.03297246    39.99988812     0.11360407     0.03308434
    0.28401098     0.08271108

PROGRAM> end of inputs


Program terminated normally


Job: RUN_Gly_ ended on :24-Aug-18 at 19:47:15
User: ibrahim
Cpu     time:  00h01m09s88c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h03m58s00c
```

B.1.4 Plotting

Here is an example of input file for plotting some molecular and atomic properties along the CO bond,

```
MOLECULE Multiplicity =  1 Charge =  0
Title = "CO"
Z-Matrix
C
O        C        BL
END ! Z-Matrix
```

```
DEFINE ! 6-311++G(d,p)
BL       = 1.10444701
END !DEFINE
END ! MOLECULE

BASIS name=6-311++G(d,p) end

PARtitioning
Scheme=IAWAD  ! other examples: ABSw, IAWAD, Fermi, Becke (default is Becke)
COre=Average  ! the core (Average, MAX1, MAX2, MIN) (default is Average)
! STeepness=0.5 ! For IAwad weight (default is 0.5)
! KValue=15.0   ! For Femi weight (default is 7.0)
end

GRID
  MESH
    ORigin= ( 0.0  0.0 -3.5 )  ! orign x,y,z
    MESH = ( 0.02 0.02 0.02 )  ! step size
    NX=0 NY=0 NZ=400           ! number of steps
  end ! mesh
end ! grid

PLOT ATOM=false Molecule=true end

AIMDFT ROT=( 2 1 ) end  ! Rotate atoms: 2 to (0,0,0), 1 to +Z, None to YZ plane

output object=AIMDFT:MOLECULE%ROTATE end     !  Rotate

!output object=GRID:WEIGHTS%MESH end          !  plot the weight
!output object=GRID:EXCHANGE%MESH end        !  plot the exchange
!output object=GRID:KINETIC%MESH end         !  plot the kinetic
output object=GRID:RADIAL_DENSITY%MESH end  !  plot the RDEN
!output object=GRID:COULOMB%MESH end         !  plot J
!output object=GRID:VNE%MESH end             !  plot Vne
stop
```

Here is the output file,

```
Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
 Molecule is a symmetric top.
Point group: Cinf_v
Free format Z-Matrix for: CO
C
O        C        BL

VARIABLES:
BL = 1.10444701

Z MATRIX FOR: CO
-----------------------------------------------------------------------------------------------

   I    AN    Z1      BL                  Z2     ALPHA                Z3      BETA
              Z4
-----------------------------------------------------------------------------------------------

   1    6
   2    8     1    1.104447    (  1)
-----------------------------------------------------------------------------------------------
```

```
Cartesian coordinates for: CO

-----------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X              Y              Z                        X
              Y              Z
-----------------------------------------------------------------------------------------------------

   1 C          6   0.00000000     0.00000000     0.00000000               0.00000000
         0.00000000     0.00000000
   2 O          8   0.00000000     0.00000000     1.10444701               0.00000000
         0.00000000     2.08710222
-----------------------------------------------------------------------------------------------------

Nuclear repulsion energy:      22.998394420

Charge=    0, Number of electrons=   14

The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-311++G(d,p) Basis Set - Total number of basis functions:       46

 Partitioning scheme set to: IAWAD
 Last atomic core set to: r_Average
Total number of grid points:      401

RotUpdate is TRUE
The Cartesian coordinates were updated
The new coordinates are: (Angstrom unit)
Atom             X             Y             Z
C       0.00000000    0.00000000    1.10444701
O       0.00000000    0.00000000    0.00000000
 ---------------------------------------------
The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-311++G(d,p)
All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

     104707 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       4482 IJKJ:       5770 IJJL:       6500 IIKK:        948
 IJJJ:        118 IIIL:        118 IIII:         12 IJKL:      86759
Number of integrals in INCORE buffers:
IIKK:        474 IJJL:       3252 IJKJ:       2885
IIKL:       2243 IJKL:      28925

      49463 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (COMBINATIONS)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:       3651 IJKJ:       5050 IJJL:       3459 IIKK:       1117
 IJJJ:        169 IIIL:        169 IIII:         34 IJKL:      35814
Number of integrals in INCORE buffers:
IIKK:       1035 IJJL:       4984 IJKJ:       5413
IIKL:       4071 IJKL:      40877
 TOTAL OF     154170 TWO-ELECTRON INTEGRALS CALCULATED (COMBINATIONS)



CLOSED SHELL SCF              Nuclear Repulsion Energy is     22.998394420 Hartrees
Convergence on Density Matrix Required to Exit is  5.0000E-06

          CYCLE   ELECTRONIC ENERGY      TOTAL ENERGY    CONVERGENCE   EXTRAPOLATION
```

```
 SCF_CYCLE:    1       -135.138449669       -112.140055248
 SCF_CYCLE:    2       -135.652326582       -112.653932162   2.93131E-02
 SCF_CYCLE:    3       -135.686265729       -112.687871309   1.62184E-02
 SCF_CYCLE:    4       -135.711088854       -112.712694434   1.17709E-02
 SCF_CYCLE:    5       -135.723210145       -112.724815724   1.11787E-02
 SCF_CYCLE:    6       -135.754336026       -112.755941606                 4-POINT
 SCF_CYCLE:    7       -135.769718423       -112.771324003   5.07036E-03
 SCF_CYCLE:    8       -135.769733450       -112.771339030   2.97065E-04
 SCF_CYCLE:    9       -135.769736659       -112.771342239                 4-POINT
 SCF_CYCLE:   10       -135.769736274       -112.771341854   6.45217E-05
 SCF_CYCLE:   11       -135.769741458       -112.771347038   1.96897E-04
 SCF_CYCLE:   12       -135.769745261       -112.771350841   1.71384E-04
 SCF_CYCLE:   13       -135.769752379       -112.771357959                 4-POINT
 SCF_CYCLE:   14       -135.769756054       -112.771361634   7.83639E-05
At termination total energy is      -112.771362  Hartrees


PROGRAM> end of inputs


Program terminated normally


Job: CO_Cinf_v_RHF_6311ppGdp ended on :29-Aug-18 at 13:44:11
User: ibrahim
Cpu     time:  00h00m00s10c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m00s00c
```

Here is an example of plot file,

```
Results calculated at: RHF/6-311++G(d,p)
Molecular Radial Density
X and Y fixed to:     0.000000    0.000000
Z, Molecular Radial Density
   -3.500000    0.005525
   -3.480000    0.005790
   -3.460000    0.006066
   -3.440000    0.006354
   -3.420000    0.006653
   -3.400000    0.006964
   -3.380000    0.007288
   -3.360000    0.007624
   -3.340000    0.007973
   -3.320000    0.008335
   -3.300000    0.008711
```

B.1.5 Expectation Value of Last Core Shell

Here is example of input file to calculate the expectation value $\langle r \rangle$ of Cl atom,

```
MOLECULE Multiplicity =  2 Charge =  0
Title = "Cl"
Z-Matrix
Cl
END ! Z-Matrix
END ! MOLECULE
BASIS name=6-311++G(d,p) end
ROHF Ncore=6 Nopen=3 ITEration=1000 run end
```

```
NUmercial
property=Radial
RAdial GRID=SG1 end
end
output object=QM:DENSITY%NUMERICAL end
stop
```

Here is the output file,

```
Welcome to MUNgauss - July 9, 2018 Version 7

 N_molecules:  1
Point group: Kh
Cartesian coordinates for: Cl

-------------------------------------------------------------------------------------------------------------

                            COORDINATES IN ANGSTROMS
     COORDINATES IN BOHR
   I EL        AN        X             Y             Z                       X
             Y             Z
-------------------------------------------------------------------------------------------------------------

   1 Cl        17   0.00000000     0.00000000     0.00000000               0.00000000
        0.00000000     0.00000000
-------------------------------------------------------------------------------------------------------------

Nuclear repulsion energy:       0.000000000

Charge=    0, Number of electrons=   17

NOpen=    1  NCore=    8
NOTE: Assuming no degeneracy amongst open-shell orbitals
The basis set has now been re-ordered FDPS
The basis set has now been re-ordered FDPS
6-311++G(d,p) Basis Set - Total number of basis functions:       31

         MENU_SCF> Number of closed shells set to          6
         MENU_SCF> Number of open shells set to          3
         MENU_SCF> Maximum SCF iterations set to       1000
The basis set has now been re-ordered FDPS
Projecting extended Huckel matrix (STO-3G)  to 6-311++G(d,p)

RHF OPEN SHELL SPECIFICATION
GROUP     : FFFFFFFFAVVVVVVVVVVVVVVVVVVVVV
ALPHA OCC : 11111111100000000000000000000000
BETA OCC  : 11111111000000000000000000000000

ORBITAL GROUP:          F       A       V
NUMBER OF MO:           8       1      22

OLEVEL    DEGENERACY    DOUBLE OCC    SINGLE ALPHA    SINGLE BETA    CONFIGURATIONS
      SAMPLE CONFIGURATION
0  1           3             2              1               0               3
      AB  AB  A
+
      --  --  --

COULOMB COUPLING COEFFICIENTS   A(I,J)
  MO          OCCA      OCCB         CLOSED   OPEN 1   OPEN 2   OPEN 3
CLOSED       1.0000    1.0000         2.0000   1.6667   1.6667   1.6667
OPEN 2       1.0000    0.6667         1.6667   1.3333   1.3333   1.3333
```

```
OPEN 3        1.0000    0.6667         1.6667   1.3333   1.3333   1.3333
OPEN 4        1.0000    0.6667         1.6667   1.3333   1.3333   1.3333

EXCHANGE COUPLING COEFFICIENTS   B(I,J)
  MO           OCCA      OCCB          CLOSED   OPEN 1   OPEN 2   OPEN 3
CLOSED        1.0000    1.0000         1.0000   0.8333   0.8333   0.8333
OPEN 2        1.0000    0.6667         0.8333   0.6667   0.6667   0.6667
OPEN 3        1.0000    0.6667         0.8333   0.6667   0.6667   0.6667
OPEN 4        1.0000    0.6667         0.8333   0.6667   0.6667   0.6667


All integrals will be kept INCORE
NOTE: INTEGRALS .LE.  1.00E-07 (I2EACC) WERE NOT KEPT
Exponent cutoff used:  2.00E+01 PQCUT2 cutoff used:  1.00E-16

       9888 TWO-ELECTRON INTEGRALS CALCULATED IN IDFCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:         296 IJKJ:         836 IJJL:        1052 IIKK:         327
 IJJJ:          24 IIIL:          24 IIII:           6 IJKL:        7323
Number of integrals in INCORE buffers:
IIKK:         165 IJJL:         526 IJKJ:         418
IIKL:         148 IJKL:        2453

       8887 TWO-ELECTRON INTEGRALS CALCULATED IN ISPCLC (RAW)
 TOTAL NUMBER OF EACH OF THE 8 TYPES OF INTEGRALS SAVED:
 IIKL:        1147 IJKJ:        1276 IJJL:         610 IIKK:         597
 IJJJ:          66 IIIL:          66 IIII:          25 IJKL:        5100
Number of integrals in INCORE buffers:
IIKK:         465 IJJL:         831 IJKJ:        1056
IIKL:         723 IJKL:        4153
 TOTAL OF      18775 TWO-ELECTRON INTEGRALS CALCULATED (RAW)


RHF OPEN SHELL SCF
CONVERGENCE ON DENSITY MATRIX REQUIRED TO EXIT IS  5.0000D-06,  INITIAL SCALE FACTOR
    (SCALST) IS 0.60000
SCALE FACTOR CUT RATIO (FACTDN) IS 0.66667,  SCALE FACTOR INCREASE RATIO (FACTUP) IS
    1.20000
SCALE FACTOR LOWER LIMIT (SLIMDN) IS 0.00050,  SCALE FACTOR UPPER LIMIT (SLIMUP) IS
    0.80000
SWITCH TO INCREASE MODE AT D CONVERGENCE  2.0000D-03 (SCRIT, RHF=INCR MODE ONLY).
Nuclear repulsion energy is       0.000000000 Hartrees
CYCLE                  ENERGY                    SCF CONVERGENCE    EXTRAPOLATION
    SCALE     RESET
 SCF          ELECTRONIC           TOTAL         COEFF      DENSITY
   0     -458.772202561     -458.772202561
   1     -458.891001035     -458.891001035              4.9546D-02
    6.0000D-01    YES
   2     -459.451471361     -459.451471361   1.6572D-02 2.9781D-02
    6.0000D-01    YES
   3     -459.224788562     -459.224788562   1.5073D-02 2.5790D-02
    6.0000D-01    YES
   4     -459.289075435     -459.289075435   2.0801D-02 3.0468D-02
    6.0000D-01    YES
   5     -459.021625012     -459.021625012   3.4345D-02 4.9075D-02
    6.0000D-01    YES
   6     -458.280660837     -458.280660837   6.3981D-02 7.9415D-02
    6.0000D-01    YES
   7     -453.860960916     -453.860960916   1.3775D-01 1.7536D-01
    6.0000D-01    YES
   8     -455.707007894     -455.707007894   1.6493D-01 2.9103D-01
    1.1852D-01    YES
   9     -455.612617494     -455.612617494   1.5923D-01 2.6469D-01
    6.0000D-01    YES
```

```
10      -447.162448243      -447.162448243   3.6744D-02 2.9407D-01
  4.0000D-01    YES
11      -449.900249936      -449.900249936   1.4882D-02 3.1609D-01
  2.6667D-01    YES
12      -457.075125871      -457.075125871   6.0375D-02 3.4565D-01
  1.1852D-01    YES
13      -455.905703310      -455.905703310   8.0477D-03 3.4317D-01
  7.9012D-02    YES
14      -453.294097820      -453.294097820   1.2178D-02 3.0694D-01
  1.7778D-01    YES
15      -454.843032412      -454.843032412   3.4627D-03 3.7677D-01
  1.0405D-02    YES
16      -456.592350352      -456.592350352   2.4616D-03 3.2049D-01
  5.2675D-02    YES
17      -455.560469276      -455.560469276   2.7767D-03 3.5625D-01
  1.7778D-01    YES
18      -448.898879015      -448.898879015   8.7994D-03 9.6854D-01
  6.0000D-01    YES
19      -457.700926667      -457.700926667   3.2563D-03 1.0743D+00
  1.7778D-01    YES
20      -447.435208609      -447.435208609   8.6254D-03 1.0155D+00
  6.0000D-01    YES
21      -456.234213849      -456.234213849   2.1204D-03 1.0590D+00
  1.7778D-01    YES
22      -457.100355907      -457.100355907   1.8084D-01 3.3080D-01
  1.1852D-01    YES
23      -455.679752922      -455.679752922   1.2540D-01 3.4269D-01
  3.5117D-02    YES
24      -460.800574864      -460.800574864   9.5108D-03 3.5824D-01
  6.0000D-01    YES
25      -459.008498553      -459.008498553   9.1972D-02 2.4129D-01
  6.0000D-01    YES
26      -457.744912026      -457.744912026   1.4814D-02 3.3384D-01
  6.0000D-01    YES
27      -457.579926153      -457.579926153   9.9838D-02 4.4324D-01
  6.0000D-01    YES
28      -458.917448500      -458.917448500   1.0648D-02 4.6793D-01
  2.6667D-01    YES
29      -458.226714489      -458.226714489   7.8724D-02 5.2723D-01
  3.0829D-03    YES
30      -460.260373064      -460.260373064   1.4062D-01 3.9332D-01
  6.0000D-01    YES
31      -457.625850891      -457.625850891   1.2905D-01 2.5640D-01
  1.1852D-01    YES
32      -458.832022002      -458.832022002   1.3738D-01 2.5314D-01
  6.0000D-01    YES
33      -454.939760389      -454.939760389   8.3623D-04 3.7504D-01
  5.2675D-02    YES
34      -456.739664891      -456.739664891   7.7015D-04 3.8178D-01
  5.2675D-02    YES
35      -456.945230394      -456.945230394   6.8413D-04 2.0000D-01
  5.2675D-02    YES
36      -457.195639916      -457.195639916   7.3119D-04 1.6140D-01
  7.9012D-02    YES
37      -454.197688783      -454.197688783   2.7958D-03 3.7077D-01
  4.0000D-01    YES
38      -458.316579026      -458.316579026   8.0862D-02 3.5366D-01
  2.0553D-03    YES
39      -457.542854523      -457.542854523   9.0986D-03 1.2178D-01
  4.0000D-01    YES
40      -455.875478405      -455.875478405   1.0238D-01 8.6068D-02
  6.0000D-01    YES
41      -457.655165507      -457.655165507   1.0550D-03 2.0983D-01
  7.9012D-02    YES
```

```
42      -458.042087729      -458.042087729   1.4133D-03 1.7545D-01
  1.1852D-01    YES
43      -457.940168781      -457.940168781   1.1831D-03 1.8627D-01
  1.1852D-01    YES
44      -456.682022967      -456.682022967   1.3207D-01 2.0372D-01
  2.6667D-01    YES
45      -458.263330179      -458.263330179   4.1016D-04 2.0688D-01
  5.2675D-02    YES
46      -447.322880391      -447.322880391   3.0212D-03 4.1350D-01
  4.0000D-01    YES
47      -457.726647483      -457.726647483   8.3823D-02 3.8888D-01
  3.5117D-02    YES
48      -456.971708875      -456.971708875   7.1904D-02 1.3138D-01
  6.0000D-01    YES
49      -454.245572658      -454.245572658   2.9059D-03 3.1669D-01
  1.7778D-01    YES
50      -444.366751154      -444.366751154   3.0660D-03 3.9233D-01
  2.6667D-01    YES
51      -453.384895324      -453.384895324   1.5424D-03 3.6964D-01
  1.7778D-01    YES
52      -454.391573147      -454.391573147   1.1290D-03 2.5721D-01
  1.7778D-01    YES
53      -457.626406762      -457.626406762   3.7611D-04 2.0999D-01
  5.2675D-02    YES
54      -447.357529946      -447.357529946   1.0202D-03 2.8888D-01
  2.6667D-01    YES
55      -458.294503303      -458.294503303   4.5901D-02 3.0412D-01
  4.6244D-03    YES
56      -454.954413052      -454.954413052   8.8269D-02 1.6685D-01
  6.0000D-01    YES
57      -456.079224307      -456.079224307   1.1615D-01 2.0551D-01
  7.9012D-02    YES
58      -456.478964339      -456.478964339   1.0304D-01 1.6610D-01
  7.9012D-02    YES
59      -452.428015695      -452.428015695   1.4993D-03 2.3197D-01
  1.7778D-01    YES
60      -437.866322770      -437.866322770   1.8896D-03 4.7869D-01
  2.6667D-01    YES
61      -449.135513951      -449.135513951   1.2285D-03 4.6380D-01
  1.7778D-01    YES
62      -448.138431820      -448.138431820   9.6326D-04 2.0784D-01
  1.7778D-01    YES
63      -456.893586077      -456.893586077   9.0528D-02 1.9815D-01
  7.9012D-02    YES
64      -456.903750176      -456.903750176   1.2922D-01 2.5547D-01
  7.9012D-02    YES
65      -451.915269653      -451.915269653   2.7655D-03 2.9528D-01
  2.6667D-01    YES
66      -455.327229310      -455.327229310   5.8918D-04 3.2280D-01
  1.5607D-02    YES
67      -436.887744424      -436.887744424   4.2152D-03 4.4820D-01
  6.0000D-01    YES
68      -447.067971151      -447.067971151   2.6786D-03 4.6887D-01
  4.0000D-01    YES
69      -441.190981179      -441.190981179   9.0666D-03 4.3297D-01
  6.0000D-01    YES
70      -452.406584642      -452.406584642   1.5222D-02 4.2577D-01
  1.7778D-01    YES
71      -448.804980190      -448.804980190   2.2324D-02 4.6997D-01
  2.6667D-01    YES
72      -431.798180374      -431.798180374   7.6692D-02 6.0945D-01
  6.0000D-01    YES
73      -455.101912192      -455.101912192   2.1836D-02 6.1540D-01
  2.3411D-02    YES
```

```
 74      -453.899689856      -453.899689856   1.7685D-02 2.3582D-01
   5.2675D-02    YES
 75      -448.312914967      -448.312914967   1.4525D-02 3.6670D-01
   1.1852D-01    YES
 76      -455.236117297      -455.236117297   1.1749D-01 2.7521D-01
   7.9012D-02    YES
 77      -449.169209766      -449.169209766   5.0789D-03 3.3640D-01
   1.1852D-01    YES
 78      -448.387891593      -448.387891593   5.9289D-03 5.5145D-01
   1.1852D-01    YES
 79      -451.560113800      -451.560113800   3.5098D-03 4.7033D-01
   7.9012D-02    YES
 80      -448.738088207      -448.738088207   3.2885D-02 6.0441D-01
   2.6667D-01    YES
 81      -453.181303442      -453.181303442   8.6422D-03 5.4121D-01
   1.1852D-01    YES
 82      -456.112617133      -456.112617133   1.5943D-03 2.3072D-01
   5.2675D-02    YES
 83      -455.638201172      -455.638201172   2.1297D-03 3.4554D-01
   5.2675D-02    YES
 84      -450.339918600      -450.339918600   7.7392D-03 6.6935D-01
   1.7778D-01    YES
 85      -456.923733136      -456.923733136   1.0000D-01 6.9107D-01
   4.6244D-03    YES
 86      -456.694962016      -456.694962016   2.1827D-02 3.0850D-01
   6.0000D-01    YES
 87      -454.729010130      -454.729010130   2.0747D-02 3.5780D-01
   6.0000D-01    YES
 88      -453.438188031      -453.438188031   1.0312D-01 3.2874D-01
   6.0000D-01    YES
 89      -456.011614567      -456.011614567   9.9600D-02 3.9415D-01
   5.2675D-02    YES
 90      -449.448233250      -449.448233250   3.1992D-02 7.9750D-01
   6.0000D-01    YES
 91      -455.344774594      -455.344774594   1.1749D-01 7.2010D-01
   4.0000D-01    YES
 92      -454.762043639      -454.762043639   9.6388D-02 2.2241D-01
   1.3702D-03    YES
 93      -431.712627116      -431.712627116   1.5712D-01 2.1443D+00
   6.0000D-01    YES
 94      -455.513311722      -455.513311722   6.1955D-02 2.1559D+00
   2.3411D-02    YES
 95      -455.750871685      -455.750871685   2.0241D-02 1.7350D-01
   3.5117D-02    YES
 96      -449.427045980      -449.427045980   7.8276D-02 1.0687D+00
   7.9012D-02    YES
 97      -457.517141473      -457.517141473   2.1674D-02 9.5998D-01
   5.2675D-02    YES
 98      -453.569703593      -453.569703593   2.3082D-02 4.7470D-01
   1.1852D-01    YES
 99      -453.664651395      -453.664651395   2.4803D-02 7.9378D-01
   1.7778D-01    YES
100      -456.526812179      -456.526812179   4.2875D-03 9.2407D-01
   3.5117D-02    YES
101      -438.884921053      -438.884921053   0.0000D+00 1.7528D+00  NO MATCH
   4.0000D-01    YES
102      -453.656537816      -453.656537816   4.1753D-01 1.8939D+00
   3.5117D-02    YES
103      -456.547793192      -456.547793192   8.6809D-03 7.0902D-01
   1.1852D-01    YES
104      -458.113195660      -458.113195660   2.0644D-03 2.6528D-01
   3.5117D-02    YES
105      -456.035628537      -456.035628537   2.9774D-03 4.5746D-01
   5.2675D-02    YES
```

```
106      -457.390485014      -457.390485014   3.1543D-03 5.9934D-01
   1.5607D-02    YES
107      -450.494088975      -450.494088975   8.3199D-03 6.9104D-01
   1.7778D-01    YES
108      -439.483169905      -439.483169905   2.7253D-02 9.0450D-01
   6.0000D-01    YES
109      -447.631577692      -447.631577692   2.0383D-02 9.1803D-01
   2.6667D-01    YES
110      -456.988950620      -456.988950620   9.4336D-02 1.2380D+00
   5.2675D-02    YES
111      -455.970837211      -455.970837211   3.4148D-02 5.9403D-01
   6.0000D-01    YES
112      -455.177960301      -455.177960301   2.5303D-02 1.8506D-01
   7.9012D-02    YES
113      -455.391677589      -455.391677589   3.1686D-03 4.8206D-01
   7.9012D-02    YES
114      -456.128288715      -456.128288715   1.8729D-02 8.0846D-01
   6.0000D-01    YES
115      -456.028505180      -456.028505180   1.4637D-02 6.9377D-01
   4.0000D-01    YES
116      -455.560838470      -455.560838470   5.6778D-02 5.0960D-01
   6.0000D-01    YES
117      -441.202514437      -441.202514437   2.8229D-01 8.0787D-01
   6.0000D-01    YES
118      -452.011202344      -452.011202344   4.3893D-01 6.3418D-01
   7.9012D-02    YES
119      -446.844288444      -446.844288444   3.2707D-02 7.1145D-01
   2.6667D-01    YES
120      -456.193020147      -456.193020147   2.9450D-01 6.7176D-01
   1.1852D-01    YES
121      -454.982137538      -454.982137538   1.2830D-01 5.8592D-01
   6.9366D-03    YES
122      -456.060060596      -456.060060596   1.5909D-01 4.8046D-01
   1.5607D-02    YES
123      -455.803818422      -455.803818422   9.4987D-02 2.1697D-01
   4.6244D-03    YES
124      -458.582057119      -458.582057119   1.1612D-01 2.5448D-01
   1.1852D-01    YES
125      -457.615073226      -457.615073226   1.0170D-01 2.1476D-01
   4.0000D-01    YES
126      -456.237165736      -456.237165736   1.9626D-01 5.0461D-01
   6.0000D-01    YES
127      -454.733095648      -454.733095648   3.7172D-02 5.7665D-01
   1.1852D-01    YES
128      -454.393732753      -454.393732753   1.0321D-02 4.8821D-01
   1.1852D-01    YES
129      -456.334497002      -456.334497002   2.6595D-01 4.6979D-01
   1.7778D-01    YES
130      -452.753890116      -452.753890116   1.5173D-02 3.7527D-01
   6.0000D-01    YES
131      -458.545643216      -458.545643216   1.5767D-01 4.4122D-01
   3.5117D-02    YES
132      -453.847443961      -453.847443961   1.2508D-02 3.5267D-01
   6.0000D-01    YES
133      -455.738598519      -455.738598519   2.0058D-03 3.8019D-01
   2.3411D-02    YES
134      -455.505378108      -455.505378108   4.4980D-03 4.0308D-01
   1.7778D-01    YES
135      -455.586577343      -455.586577343   8.6627D-03 2.7458D-01
   2.6667D-01    YES
136      -455.853594719      -455.853594719   9.2160D-03 2.4010D-01
   4.0000D-01    YES
137      -453.940010542      -453.940010542   7.9523D-03 2.0088D-01
   1.1852D-01    YES
```

```
138      -457.687669885      -457.687669885   1.0274D-01 2.1307D-01
   7.9012D-02    YES
139      -457.118423226      -457.118423226   1.0074D-02 1.5201D-01
   6.0000D-01    YES
140      -457.172213836      -457.172213836   8.1226D-02 1.3011D-01
   6.0000D-01    YES
141      -455.583567341      -455.583567341   4.8841D-02 3.5927D-01
   4.0000D-01    YES
142      -455.975833331      -455.975833331   1.7003D-01 4.8339D-01
   4.0000D-01    YES
143      -453.121149627      -453.121149627   6.8089D-02 8.5461D-01
   1.7778D-01    YES
144      -453.372902920      -453.372902920   1.9321D-02 8.2024D-01
   4.0000D-01    YES
145      -454.561286980      -454.561286980   1.7298D-01 5.2852D-01
   1.1852D-01    YES
146      -453.794757371      -453.794757371   1.1233D-02 4.5954D-01
   2.6667D-01    YES
147      -447.353572473      -447.353572473   7.1233D-03 1.5410D+00
   4.0000D-01    YES
148      -446.308673831      -446.308673831   1.2180D-02 1.2186D+00
   6.0000D-01    YES
149      -451.900338927      -451.900338927   2.1935D-01 5.8358D-01
   1.7778D-01    YES
150      -450.901291059      -450.901291059   1.2171D-03 1.0038D+00
   1.1852D-01    YES
151      -457.905806337      -457.905806337   6.4717D-04 8.2476D-01
   5.2675D-02    YES
152      -442.083114884      -442.083114884   3.5373D-03 1.0845D+00
   4.0000D-01    YES
153      -458.048735208      -458.048735208   2.7017D-01 1.1782D+00
   2.3411D-02    YES
154      -457.391264532      -457.391264532   1.0364D-01 3.4627D-01
   5.2675D-02    YES
155      -457.460444985      -457.460444985   9.6426D-02 2.2765D-01
   7.9012D-02    YES
156      -456.761152731      -456.761152731   3.5376D-03 1.7529D-01
   4.0000D-01    YES
157      -457.841611865      -457.841611865   5.4545D-03 1.4129D-01
   6.0000D-01    YES
158      -457.113956243      -457.113956243   5.4431D-03 2.5876D-01
   6.0000D-01    YES
159      -456.071374353      -456.071374353   4.5220D-03 3.0893D-01
   2.6667D-01    YES
160      -457.456643367      -457.456643367   4.7129D-03 3.3357D-01
   1.7778D-01    YES
161      -456.724853201      -456.724853201   6.7561D-03 4.0987D-01
   6.0000D-01    YES
162      -457.639588798      -457.639588798   1.0201D-03 2.4081D-01
   3.5117D-02    YES
163      -457.467337082      -457.467337082   8.9818D-03 2.2318D-01
   6.0000D-01    YES
164      -458.527509174      -458.527509174   6.0769D-03 2.0607D-01
   2.6667D-01    YES
165      -457.902628945      -457.902628945   4.8515D-03 3.0217D-01
   6.0000D-01    YES
166      -455.887828830      -455.887828830   7.3446D-02 2.9623D-01
   6.0000D-01    YES
167      -460.265262456      -460.265262456   8.4497D-02 2.3123D-01
   2.6667D-01    YES
168      -459.663599144      -459.663599144   5.4585D-02 1.0157D-01
   6.0000D-01    YES
169      -458.883295440      -458.883295440   7.5442D-02 1.1264D-01
   6.0000D-01    YES
```

```
170      -459.756666018      -459.756666018   1.3061D-01 1.6793D-01
   6.0000D-01    YES
171      -449.887182141      -449.887182141   2.8125D-01 5.7231D-01
   6.0000D-01    YES
172      -453.862703638      -453.862703638   1.7487D-01 6.1297D-01
   4.0000D-01    YES
173      -452.822856134      -452.822856134   1.0897D-01 5.6887D-01
   2.6667D-01    YES
174      -454.410414965      -454.410414965   3.8329D-02 5.8423D-01
   4.0000D-01    YES
175      -452.076775963      -452.076775963   2.4586D-01 6.0878D-01
   2.6667D-01    YES
176      -455.014357087      -455.014357087   3.0426D-03 4.0386D-01
   1.7778D-01    YES
177      -438.157465778      -438.157465778   4.2669D-03 7.0351D-01
   4.0000D-01    YES
178      -454.577415384      -454.577415384   3.1389D-01 1.0121D+00
   1.1852D-01    YES
179      -459.996999966      -459.996999966   2.4606D-01 1.0423D+00
   2.3411D-02    YES
180      -455.103725753      -455.103725753   4.4992D-03 2.2080D-01
   4.0000D-01    YES
181      -454.576065383      -454.576065383   9.6354D-03 2.6841D-01
   6.0000D-01    YES
182      -455.016049434      -455.016049434   1.1865D-01 3.5135D-01
   6.0000D-01    YES
183      -458.189079984      -458.189079984   1.2007D-01 3.2480D-01
   2.6667D-01    YES
184      -450.508559328      -450.508559328   4.3101D-02 7.2652D-01
   6.0000D-01    YES
185      -450.180101788      -450.180101788   4.1646D-02 7.2691D-01
   6.0000D-01    YES
186      -450.062407129      -450.062407129   1.4628D-01 5.5950D-01
   6.0000D-01    YES
187      -444.609479598      -444.609479598   4.6065D-02 7.4726D-01
   6.0000D-01    YES
188      -454.672721924      -454.672721924   4.4608D-02 4.0973D-01
   4.0000D-01    YES
189      -447.087937427      -447.087937427   5.6410D-02 3.4835D-01
   6.0000D-01    YES
190      -451.483647306      -451.483647306   3.4277D-03 6.6457D-01
   6.0000D-01    YES
191      -445.479046315      -445.479046315   3.6888D-02 7.1563D-01
   6.0000D-01    YES
192      -453.415938716      -453.415938716   6.3666D-02 5.2770D-01
   6.0000D-01    YES
193      -445.774095890      -445.774095890   5.0452D-02 5.0065D-01
   6.0000D-01    YES
194      -453.427510716      -453.427510716   3.9955D-02 7.6799D-01
   6.0000D-01    YES
195      -446.406717481      -446.406717481   4.5605D-02 7.4540D-01
   6.0000D-01    YES
196      -451.941588175      -451.941588175   6.0722D-02 4.1957D-01
   6.0000D-01    YES
197      -447.220859642      -447.220859642   9.9924D-03 3.5630D-01
   6.0000D-01    YES
198      -451.749806560      -451.749806560   1.9007D-01 3.3944D-01
   6.0000D-01    YES
199      -450.394486712      -450.394486712   1.0113D-01 3.8526D-01
   2.6667D-01    YES
200      -451.641583153      -451.641583153   6.7024D-02 3.4714D-01
   4.0000D-01    YES
201      -457.009545797      -457.009545797   1.4699D-01 3.7683D-01
   1.1852D-01    YES
```

```
202      -457.383351980      -457.383351980   9.7237D-02 1.4901D-01
   6.0000D-01    YES
203      -458.224109275      -458.224109275   1.1273D-01 1.5200D-01
   6.0000D-01    YES
204      -456.174700871      -456.174700871   3.5312D-02 2.0588D-01
   2.6667D-01    YES
205      -455.699523581      -455.699523581   2.9008D-02 2.0776D-01
   4.0000D-01    YES
206      -456.965784185      -456.965784185   1.0039D-01 1.5532D-01
   2.6667D-01    YES
207      -458.604008375      -458.604008375   9.4526D-02 1.1695D-01
   6.0000D-01    YES
208      -458.340131743      -458.340131743   9.6752D-02 1.1716D-01
   6.0000D-01    YES
209      -458.249458253      -458.249458253   1.0680D-01 1.2698D-01
   6.0000D-01    YES
210      -458.689944251      -458.689944251   1.2053D-01 1.4777D-01
   6.0000D-01    YES
211      -458.037168822      -458.037168822   1.3802D-01 1.6271D-01
   6.0000D-01    YES
212      -458.720931977      -458.720931977   1.6471D-01 2.1054D-01  3-POINT
   6.0000D-01    YES
213      -458.680166912      -458.680166912              8.6318D-03
   6.0000D-01    YES
214      -458.674371048      -458.674371048   1.0934D-02 1.5283D-02
   6.0000D-01    YES
215      -458.823355541      -458.823355541   1.3365D-02 1.7375D-02
   6.0000D-01    YES
216      -458.942775578      -458.942775578   2.0542D-02 2.7008D-02
   6.0000D-01    YES
217      -459.146685169      -459.146685169   2.4869D-02 3.1265D-02
   6.0000D-01    YES
218      -459.303688800      -459.303688800   3.2133D-02 3.9672D-02
   6.0000D-01    YES
219      -459.431924182      -459.431924182   3.7558D-02 4.5656D-02
   6.0000D-01    YES
220      -459.419594052      -459.419594052   4.2024D-02 4.8926D-02
   6.0000D-01    YES
221      -459.496578143      -459.496578143   4.6352D-02 5.4352D-02
   6.0000D-01    YES
222      -459.359028266      -459.359028266   4.7314D-02 5.3017D-02
   6.0000D-01    YES
223      -459.524106000      -459.524106000   4.8962D-02 5.5752D-02
   6.0000D-01    YES
224      -459.307006991      -459.307006991   4.6665D-02 5.0995D-02
   6.0000D-01    YES
225      -459.563164716      -459.563164716   4.5481D-02 5.0688D-02
   6.0000D-01    YES
226      -459.297365026      -459.297365026   4.0815D-02 4.3842D-02  4-POINT
   6.0000D-01    YES
227      -459.706147025      -459.706147025              6.9408D-03
   6.0000D-01    YES
228      -459.391281466      -459.391281466   4.7995D-03 5.2560D-03
   6.0000D-01    YES
229      -459.531637871      -459.531637871   3.8063D-03 4.0181D-03
   6.0000D-01    YES
230      -459.418970990      -459.418970990   3.1986D-03 3.4004D-03
   6.0000D-01    YES
231      -459.501625567      -459.501625567   2.6673D-03 2.8064D-03
   6.0000D-01    YES
232      -459.436218733      -459.436218733   2.2513D-03 2.3777D-03  4-POINT
   6.0000D-01    YES
233      -459.466309910      -459.466309910              6.9392D-04
   6.0000D-01    YES
```

```
 234      -459.470459626      -459.470459626   4.3247D-04 4.4145D-04
    6.0000D-01    YES
 235      -459.471109554      -459.471109554   3.9884D-04 4.0840D-04
    6.0000D-01    YES
 236      -459.467996095      -459.467996095   2.7762D-04 2.8240D-04
    6.0000D-01    YES
 237      -459.472715890      -459.472715890   2.6793D-04 2.7520D-04
    6.0000D-01    YES
 238      -459.467373199      -459.467373199   1.9970D-04 2.0415D-04
    6.0000D-01    YES
 239      -459.473045374      -459.473045374   1.9565D-04 2.0199D-04  4-POINT
    6.0000D-01    YES
 240      -459.468461663      -459.468461663              5.8131D-05
    6.0000D-01    YES
 241      -459.471708491      -459.471708491   4.3680D-05 4.5798D-05
    6.0000D-01    YES
 242      -459.469106097      -459.469106097   3.4395D-05 3.5837D-05
    6.0000D-01    YES
 243      -459.471207868      -459.471207868   3.0939D-05 3.2744D-05
    6.0000D-01    YES
 244      -459.469460388      -459.469460388   2.4190D-05 2.5321D-05
    6.0000D-01    YES
 245      -459.470929708      -459.470929708   2.4075D-05 2.5708D-05
    6.0000D-01    YES
 246      -459.469671268      -459.469671268   1.8459D-05 1.9412D-05
    6.0000D-01    YES
 247      -459.470758951      -459.470758951   1.9605D-05 2.0853D-05
    6.0000D-01    YES
 248      -459.469806459      -459.469806459   1.4660D-05 1.5463D-05  4-POINT
    6.0000D-01    YES
 249      -459.470162194      -459.470162194              6.2938D-06
    6.0000D-01    YES
 250      -459.470301566      -459.470301566   3.8669D-06 4.0741D-06
    6.0000D-01    YES
 251      -459.470211928      -459.470211928   3.4584D-06 3.6241D-06
    6.0000D-01    YES
 252      -459.470268047      -459.470268047   2.2615D-06 2.3675D-06
    6.0000D-01    YES
 253      -459.470246996      -459.470246996              0.0000D+00
    6.0000D-01    YES
At termination total energy is     -459.470247  Hartrees
 Orbital convergence is 4.61331D-06
Energy components:
Kinetic =                   459.478881804
Potential =                -1094.308816416
Kinetic + Potential =       -634.829934612
Coulomb repulsion =          202.868115385
Exchange =                   -27.311163122
Coulomb+Exchange =           175.556952264
Nuclear =                      0.000000000
Total electronic =          -459.272982348
Total energy =              -459.272982348

Virial =                      1.999551885

Dipole moment (Debye):
**********************
       X: -3.727308E-05     Y: -8.172788E-05     Z: -2.095933E-04     TOTAL: 0.000228

Radial grid used for numerical integration: SG1
Weights used for numerical integration:     BECKE
The Bragg-Slater radius of H has been set to 0.35
************************************************
Radial by atom and MO:  <r> (bohr)
```

```
-------------------------------------------------------------------------------------------

   Atom,MO:       1         2         3         4         5         6         7
     1          0.18261   0.88353   0.81147   0.81147   0.81147   3.11692   3.07513
                   8         9
                3.07513   3.07515
Total <r>:     15.84289
-------------------------------------------------------------------------------------------

Radial by atom and MO:  <r^2> (bohr^2)
-------------------------------------------------------------------------------------------

   Atom,MO:       1         2         3         4         5         6         7
     1          0.02240   0.46262   0.40870   0.40870   0.40870   5.65437   6.80701
                   8         9
                6.80703   6.80708
Total <r^2>:   27.78660
-------------------------------------------------------------------------------------------

Total number of grid points:     3760


PROGRAM> end of inputs


Program terminated normally


Job: Cl_Kh_ROHF_6311ppGdp ended on :24-Aug-18 at 13:05:06
User: ibrahim
Cpu     time:  00h00m01s00c on ibrahim-Lenovo-IdeaPad-P500
Elapsed time:  00h00m01s00c
```

B.2 Code Source

Here you can find the source code of AIMD,

```
      SUBROUTINE GET_AIMDFT_object (class, Objname, Modality)
!*****************************************************************************
!     Date last modified: July 22, 2015                                      *
!     Author: Ibrahim Awad I. Awad                                           *
!     Description: Objects belonging to the class AIMDFT                     *
!*****************************************************************************
! MODULEs:
      USE program_files
      USE program_objects

      implicit none
!
! Input scalar:
      character*(*) :: class
      character*(*) :: Objname
      character*(*) :: Modality
!
! Local scalars:
      integer :: Object_number
!
! Begin:
```

```
call PRG_manager ('enter', 'GET_AIMDFT_object', 'UTILITY')

call GET_object_number (OBJ_AIMDFT,     &
                        NAIMDFTobjects, &
                        FirstAIMDFT,    &
                        Objname,        &
                        Modality,       &
                        Object_number)

if(OBJ_AIMDFT(Object_number)%current)then
  call PRG_manager ('exit', 'GET_AIMDFT_object', 'UTILITY')
  return
end if

OBJ_AIMDFT(Object_number)%exist=.true.
OBJ_AIMDFT(Object_number)%Current=.true.
Object(ObjNum)%exist=.true.
Object(ObjNum)%Current=.true.

select_Object: select case (Objname)
case ('FRAGCART') ! >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
  select case (Modality)
  case('DBASE')
   call Add_To_DB
  case('GRIDS')
   call Store_result_direct
  case default
     write(UNIout,*)'No such object "',Objname(1:len_trim(Objname)), &
                 '" for Modality "',Modality(1:len_trim(Modality)),'"'
 stop'No such object'
end select

case ('MOLECULE') ! >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
  select case (Modality)
  case('BUILD')
   call CalcMoleProperties
  case('ROTATE')
   call get_databaseform
  case default
     write(UNIout,*)'No such object "',Objname(1:len_trim(Objname)), &
                 '" for Modality "',Modality(1:len_trim(Modality)),'"'
 stop'No such object'
end select

case ('FRAGMENT') ! >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
  select case (Modality)
  case('DISPLAY')
   call FragmentsDisplay
  case('BUILD')
   call Build_FragFiles
  case default
     write(UNIout,*)'No such object "',Objname(1:len_trim(Objname)), &
                 '" for Modality "',Modality(1:len_trim(Modality)),'"'
 stop'No such object'
end select

case default
  write(UNIout,*)'No such object "',Objname(1:len_trim(Objname)), &
                  '" for Modality "',Modality(1:len_trim(Modality)),'"'
  stop'No such object'

end select select_Object

call PRG_manager ('exit', 'GET_AIMDFT_object', 'UTILITY')
```

```
      return
      end SUBROUTINE GET_AIMDFT_object

      SUBROUTINE BLD_AIMDFT_objects
!*****************************************************************************
!       Date last modified July 22, 2015                                     *
!       Author: Ibrahim Awad                                                 *
!       Description:                                                         *
!*****************************************************************************
! MODULEs:
      USE program_objects

      implicit none

! Local scalar:
      integer :: Iobject
!
! Begin:
      OBJ_AIMDFT(1:Max_objects)%modality = 'other'
      OBJ_AIMDFT(1:Max_objects)%class = 'AIMDFT'
      OBJ_AIMDFT(1:Max_objects)%depend = .true.
      NAIMDFTobjects = 0
!
! Class of (AIMDFT)
      NAIMDFTobjects = NAIMDFTobjects + 1
      OBJ_AIMDFT(NAIMDFTobjects)%name = 'FRAGCART'
      OBJ_AIMDFT(NAIMDFTobjects)%modality = 'DBASE'
      OBJ_AIMDFT(NAIMDFTobjects)%routine = 'Add_To_DB'

      NAIMDFTobjects = NAIMDFTobjects + 1
      OBJ_AIMDFT(NAIMDFTobjects)%name = 'FRAGCART'
      OBJ_AIMDFT(NAIMDFTobjects)%modality = 'GRIDS'
      OBJ_AIMDFT(NAIMDFTobjects)%routine = 'Store_result_direct'

      NAIMDFTobjects = NAIMDFTobjects + 1
      OBJ_AIMDFT(NAIMDFTobjects)%name = 'MOLECULE'
      OBJ_AIMDFT(NAIMDFTobjects)%modality = 'BUILD'
      OBJ_AIMDFT(NAIMDFTobjects)%routine = 'CalcMoleProperties'

      NAIMDFTobjects = NAIMDFTobjects + 1
      OBJ_AIMDFT(NAIMDFTobjects)%name = 'MOLECULE'
      OBJ_AIMDFT(NAIMDFTobjects)%modality = 'ROTATE'
      OBJ_AIMDFT(NAIMDFTobjects)%routine = 'get_databaseform'

      NAIMDFTobjects = NAIMDFTobjects + 1
      OBJ_AIMDFT(NAIMDFTobjects)%name = 'FRAGMENT'
      OBJ_AIMDFT(NAIMDFTobjects)%modality = 'DISPLAY'
      OBJ_AIMDFT(NAIMDFTobjects)%routine = 'FragmentsDisplay'

      NAIMDFTobjects = NAIMDFTobjects + 1
      OBJ_AIMDFT(NAIMDFTobjects)%name = 'FRAGMENT'
      OBJ_AIMDFT(NAIMDFTobjects)%modality = 'BUILD'
      OBJ_AIMDFT(NAIMDFTobjects)%routine = 'Build_FragFiles'

!     NAIMDFTobjects = NAIMDFTobjects + 1
!     OBJ_AIMDFT(NAIMDFTobjects)%name = 'FRAGCART'
!     OBJ_AIMDFT(NAIMDFTobjects)%modality = 'MESH'
!     OBJ_AIMDFT(NAIMDFTobjects)%routine = 'DENFRAG'

! Dummy Class
      NAIMDFTobjects = NAIMDFTobjects + 1
      OBJ_AIMDFT(NAIMDFTobjects)%name = '?'
      OBJ_AIMDFT(NAIMDFTobjects)%modality = '?'
      OBJ_AIMDFT(NAIMDFTobjects)%routine = '?'
```

```
!
      do Iobject=1,NAIMDFTobjects
        OBJ_AIMDFT(Iobject)%exist=.false.
        OBJ_AIMDFT(Iobject)%Current=.false.
      end do

      return
      end SUBROUTINE BLD_AIMDFT_objects
```

```
      SUBROUTINE MENU_AIMDFT
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
! MODULEs:
      USE program_parser
      USE program_constants
      USE MENU_gets
      USE AIMDFT_type

      implicit none
!
! Local scalars:
      integer :: length
      logical done
      ChangeTerminalAtoms=.false.
!
! Begin:
      call PRG_manager ('enter', 'MENU_AIMDFT', 'UTILITY')
! Defaults:
      done=.false.
! Menu:
      do while (.not.done)
      call new_token (' AIMDFT:')
!
      if(token(1:4).eq.'HELP')then
         write(UNIout,'(a)') &
      ' Command AIMDFT:', &
      '   Purpose: Request AIMDFT codes', &
      '   Syntax :', &
      '   Level_Number = <integer>, Override the default', &
      '   TErminal = <integer>, Override the default', &
      '   end'
!
! ROTlist
      else if(token(1:3).eq.'ROT')then
        call GET_value(ROTlist, NROTlist)

! Level_Number
      else if(token(1:1).EQ.'L')then
          call GET_value (Level_Number)

! Terminal Atom
      else if(token(1:1).EQ.'T')then
         call GET_value (TerminalAtom)

! INDexNUM (needed to RUN the input files for the fragments)
      else if(token(1:1).EQ.'I')then
         call GET_value (IndexNumber)

!DIrectMethod
      else if(token(1:1).EQ.'D')then
         DirectMethod=.true.
```

```
!Use the Becke weight from the Database
!      else if(token(1:5).EQ.'NODBW')then
!         DatabaseBWeight=.False.

! CHangeTerm
      else if(token(1:2).EQ.'CH')then
         ChangeTerminalAtoms=.true.

! OPT
      else if(token(1:1).EQ.'O')then
         OPT_status="OPT"

      else
         call MENU_end (done)
      end if
!
      end do !(.not.done)

!
! end of routine MENU_AIMDFT
      call PRG_manager ('exit', 'MENU_AIMDFT', 'UTILITY')
      return
      end
```

```
      MODULE AIMDFT_type
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
!

      USE program_constants

      implicit none

!     Parameters for fragmant atom
      integer               :: Level_Number
      integer, parameter    :: SymMax = 64
      integer               :: TerminalAtom              ! Add H as termenal
      logical               :: ChangeTerminalAtoms
      logical               :: DirectMethod
      logical               :: DatabaseBWeight
      integer               :: IndexNumber
      integer               :: MaxGridPnt
      logical               :: RotUpdate

      integer, parameter :: MAX_ROTlist=4 ! Maximum number of ROT in ROTlist
      integer :: NROTlist
      integer, dimension(:) :: ROTlist(Max_ROTlist)

      ! the output unit (Angstrom)
      double precision, parameter :: CartPrnFactor = Bohr_to_Angstrom

      ! Database types
      character (128)  :: DirLoc             ! Database Location
      character (128)  :: DBFileExt          ! Database file type (Default txt)
      character (128)  :: CsvSep             !
      character (128)  :: IndexFileName      !
      character (128)  :: CartFileName
      character (128)  :: PropFileName

      character (128)  :: homedir
```

```
    character (len=:), allocatable :: IndexFilePath
    character (len=:), allocatable :: CartFilePath
    character (len=:), allocatable :: PropFilePath
    character (len=:), allocatable :: GridPath

    character (128)  :: fmt_index
    character (128)  :: fmt_cart
    character (128)  :: fmt_prop
    character (128)  :: fmt_index_title
    character (128)  :: fmt_cart_title
    character (128)  :: fmt_prop_title

    character(len=8)   :: OPT_status

    character(len=8)   :: fmt_xyz
    character(len=8)   :: fmt_w
    character(len=8)   :: fmt_GridPts
    character(len=8)   :: fmt_rho
    character(len=32)  :: Pfmt_rho
    character(len=32)  :: Pfmt_rho_w
    character(len=256) :: Pfmt_rho_w_v
    character(len=64)  :: Pfmt_rho_dd

    type :: CartesianCoordinate
        double precision  :: X
        double precision  :: Y
        double precision  :: Z
    end type CartesianCoordinate

    type :: GridsDataBase
        double precision  :: X
        double precision  :: Y
        double precision  :: Z
        double precision  :: W        ! Total Angular Weight
        double precision  :: BW       ! Becke Weight
        double precision  :: D        ! Electron Density
        double precision  :: Q        ! W*BW*D
        double precision  :: VpotA    ! I1E_V12dr2 (analytical term)
        double precision  :: VpotN    ! I1E_V12dr2 (numerical term)
    end type GridsDataBase

    type  ::  FragAtomInfo
        integer             :: Atomic_Number
        integer             :: MUNIdx  ! the index of atom in MUNgauss
        double precision    :: X
        double precision    :: Y
        double precision    :: Z
        character(len=8)    :: ELEMENT   ! Atom Symbol (e.g H for Hydrogen)
        integer             :: type      ! Atom type
        integer             :: number    ! Total number of a given atom type
        integer             :: factor    ! Valence factor
        character(len=16)   :: Ctype     ! The atom type as a character string
        integer             :: level = 0
        double precision    :: bond_order
        logical             :: sorted = .false. ! Have UNiqE Type
    end type FragAtomInfo

    type :: FragsInfo
        ! Number of fragment atoms
        integer                                     :: NAtoms
        ! Number of neighbour atoms
        integer                                     :: NumNeighbourAtoms
        ! (atoms number : level)
        character (SymMax) , dimension(10000,10) :: symbol
```

```
        ! Rotation matrix
        double precision, dimension (3,3)           :: RMat
        ! Translation matrix
        double precision, dimension (3)             :: TMat
        ! connectivity matrix
        integer, dimension(100,100)                 :: Connect
    end type FragsInfo

    ! Two dimension array (integer Values)
    type :: TwoDiArrInt
        integer, dimension(:,:), allocatable    :: ijarray
    end type TwoDiArrInt

    type :: TypeFragMat
        double precision, dimension (3,3)    :: RMat
        double precision, dimension (3)      :: TMat
        double precision, dimension (3)      :: TMatCorr
    end type TypeFragMat

    type :: IndexFile
        integer                    :: IndexNum          ! column # 1
        character(len=SymMax) :: Symbol           ! column # 2
        integer                    :: NAtoms            ! column # 3
        integer                    :: GridsNum          ! column # 4 Grids number
        character (len=32)         :: GridType          ! column # 5 Grid type
        character (len=32)         :: GridTypeExtra     ! column # 6
        character (len=32)         :: Method            ! column # 7 RHF, ....
        character (len=32)         :: BasisSet          ! column # 8 Basis set
        character (len=16)         :: DEN_Partitioning ! column # 9 Becke, ...
        character (len=16)         :: CoreSize          ! column # 10 Average,...
        character (len=8)          :: OPT               ! column # 11 NO_OPT, OPT
        logical                    :: Availability      ! Availability
    end type IndexFile

    type :: PropFile
        integer                    :: IndexNum          ! column # 1
        double precision           :: ElectonNum        ! column # 2
        double precision           :: K                 ! column # 3
        double precision           :: KHF               ! column # 4
        double precision           :: T                 ! column # 5
        double precision           :: Vne_A             ! column # 6
        double precision           :: Vne_N             ! column # 7
        double precision           :: Vee_AN            ! column # 8
        double precision           :: J                 ! column # 9
        double precision           :: JHF               ! column # 10
        double precision           :: Jaa               ! column # 11
        double precision           :: VeeA              ! column # 12
        logical                    :: Availability      ! Availability
    end type PropFile

    end MODULE AIMDFT_type
```

```
    SUBROUTINE SET_defaults_AIMDFT
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
!
    USE AIMDFT_type

    implicit none

!     Parameters for fragmant atom
```

```
      Level_Number    = 2
      TerminalAtom    = 1       ! Add H as termenal
      IndexNumber     = 0
      MaxGridPnt  = 150000

      DirectMethod    = .False.  ! Read from the database.
      DatabaseBWeight = .True.   ! Use the Bweight from the database
                                 ! (Do not generate new weights)
      OPT_status      = "NO_OPT"
      RotUpdate       = .True. ! update the new coordiants rotation

!     DataBase Parameters
      CALL get_environment_variable("HOME",homedir)
      DirLoc                     = trim(homedir)//"/MUNgauss/AIMDFT/DATABASE/"

      DBFileExt      = "txt"
      CsvSep                     = ","
      IndexFileName              = "Tables_Index"
      CartFileName               = "Carts_Index"
      PropFileName         = "Properties_Index"

!     Database file locations
      IndexFilePath   = trim(DirLoc)//trim(IndexFileName)//"."//trim(DBFileExt)
      CartFilePath    = trim(DirLoc)//trim(CartFileName)//"."//trim(DBFileExt)
      PropFilePath    = trim(DirLoc)//trim(PropFileName)//"."//trim(DBFileExt)
      GridPath = trim(DirLoc)

!     Output format
      fmt_GridPts='f26.16'
      fmt_rho='f26.16'
      fmt_xyz='f24.16'
      fmt_w='E24.16'

      Pfmt_rho='(3'//trim(fmt_GridPts)//','//trim(fmt_rho)//')'
      Pfmt_rho_w='(3'//trim(fmt_w)//','//trim(fmt_w)//','//trim(fmt_w)//')'
      Pfmt_rho_w_v='(3'//trim(fmt_w)//',4'//trim(fmt_w)//','//trim(fmt_xyz)//')'
      Pfmt_rho_dd='(10'//trim(fmt_w)//')'

      write(fmt_index,'(A8,I0,A22)')"(I5,2X,A",SymMax,",I2,I8,A8,A15,A8,4A14)"
      fmt_index_title='(A5,A16,A30,A7,A7,A26,A12,3A12)'

      fmt_cart='(I5,I3,3X,A3,3'//fmt_xyz//')'
      fmt_cart_title='(A5,A3,3X,A3,A22,A26,A28)'

      fmt_prop='(I5,11'//fmt_xyz//')'
      fmt_prop_title='(A5,11A24)'

      end SUBROUTINE SET_defaults_AIMDFT
```

```
      MODULE GetDataBaseInfo
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
!
      USE mod_rotation
      USE AIMDFT_Files
      USE program_constants

      implicit none

CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
      SUBROUTINE GetMoleculeAtomsIndex(MolIdxInfo,&
                                       SearchSym,&
                                       SearchSymNum,&
                                       IndexFoundNum)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description: Return the required information from index file           *
!                  for the given unique symbols, where diffrent              *
!                  conditions were applied.                                  *
!*****************************************************************************
!
      USE module_grid_points
      USE NI_defaults
      USE type_Weights
      USE QM_defaults


      implicit none

      integer, intent(out) :: IndexFoundNum
      integer :: iSymbol
      integer :: File_Unit
      integer :: Reason
      logical :: Lerror
      character (SymMax):: CurrentSearchSymbol
      character (20) :: Radnote
      type(IndexFile) :: CurrentIndexInfo

      integer, intent(in) :: SearchSymNum
      character (SymMax), dimension (:), allocatable, intent(in) ::  SearchSym
      type (IndexFile), dimension(:), allocatable, intent(out) :: MolIdxInfo

      allocate (MolIdxInfo(Natoms))

      call GET_object ('GRID', 'RADIAL', RADIAL_grid)
      ! check the database and return index
      IndexFoundNum=0 ! count number of unique symbols in the database for mol.
      MolIdxInfo%Availability=.false.
      do iSymbol=1, SearchSymNum
         call GET_unit (IndexFilePath, File_Unit, Lerror)
         open(UNIT=File_Unit,file=IndexFilePath, status='old',form='formatted')
         CurrentSearchSymbol=SearchSym(iSymbol)
         read(File_Unit,*)
         do ! read the index file
            read(File_Unit,fmt_index,IOSTAT=Reason)                        &
                                       CurrentIndexInfo%IndexNum,          &
                                       CurrentIndexInfo%Symbol,            &
                                       CurrentIndexInfo%NAtoms,            &
                                       CurrentIndexInfo%GridsNum,          &
                                       CurrentIndexInfo%GridType,          &
                                       CurrentIndexInfo%GridTypeExtra,     &
                                       CurrentIndexInfo%Method,            &
                                       CurrentIndexInfo%BasisSet,          &
                                       CurrentIndexInfo%DEN_Partitioning,&
                                       CurrentIndexInfo%CoreSize,          &
                                       CurrentIndexInfo%OPT
            if (Reason.lt.0) exit
            if (trim(ADJUSTL(CurrentIndexInfo%Symbol)).ne.&
            trim(ADJUSTL(CurrentSearchSymbol))) cycle  ! first condition
            if (trim(ADJUSTL(CurrentIndexInfo%GridType)).ne.&
            trim(ADJUSTL(RADIAL_grid))) cycle  ! second condition
            if (trim(RADIAL_grid).eq.'GILL') then
               Radnote=""
```

```
               write(Radnote,'(I2,A1,I2,A1,I2,A1,I3,A1)') NRPoints_Gill, &
                                                   & '(',NApoints_Gill(1),',',&
                                                         NApoints_Gill(2),',',&
                                                         NApoints_Gill(3),')'
               ! second condition for GILL
               if (trim(ADJUSTL(Radnote)).ne.&
               trim(ADJUSTL(CurrentIndexInfo%GridTypeExtra))) cycle
            end if
            if (trim(ADJUSTL(CurrentIndexInfo%BasisSet)).ne.&
            trim(ADJUSTL(Basis_set_name))) cycle  ! third condition
            if (trim(ADJUSTL(CurrentIndexInfo%DEN_Partitioning)).ne.&
            trim(ADJUSTL(DEN_Partitioning))) cycle  ! forth condition
            if (trim(ADJUSTL(CurrentIndexInfo%OPT)).ne.&
            trim(ADJUSTL(OPT_status))) cycle  ! fifth condition

            IndexFoundNum=IndexFoundNum+1
            MolIdxInfo(iSymbol) = CurrentIndexInfo
            MolIdxInfo(iSymbol)%Availability = .true.
         end do ! end of read index file
         close(UNIT=File_Unit)
      end do ! end loop of condition

      write (*,'(/A,I0)') "Number of indexs available in the database    =   ",&
                          IndexFoundNum
      write (*,'(A,I0)') "Number of indexs NOT available in the database =  ", &
                          SearchSymNum-IndexFoundNum
      return
      end SUBROUTINE GetMoleculeAtomsIndex
      SUBROUTINE GetMaxIndex (MaxIndex)
!*********************************************************************
!     Date last modified:                                            *
!     Author: Ibrahim Awad                                           *
!     Description: Return the maximum index number from index file   *
!*********************************************************************
!
      implicit none

      type(IndexFile) :: CurrentIndexInfo
      integer :: File_unit
      integer :: Reason
      logical :: Lerror

      integer, intent(out) :: MaxIndex
      CurrentIndexInfo%IndexNum=0
      MaxIndex=0
      call GET_unit (trim(IndexFilePath), File_unit, Lerror)
      open(UNIT=File_unit, file=trim(IndexFilePath), &
           status='old' ,form='formatted')
      read(File_Unit,*)
         Do
            read(File_Unit,fmt_index,IOSTAT=Reason)  &
                                       CurrentIndexInfo%IndexNum,        &
                                       CurrentIndexInfo%Symbol,          &
                                       CurrentIndexInfo%NAtoms,          &
                                       CurrentIndexInfo%GridsNum,        &
                                       CurrentIndexInfo%GridType,        &
                                       CurrentIndexInfo%GridTypeExtra,   &
                                       CurrentIndexInfo%Method,          &
                                       CurrentIndexInfo%BasisSet,        &
                                       CurrentIndexInfo%DEN_Partitioning,&
                                       CurrentIndexInfo%CoreSize,        &
                                       CurrentIndexInfo%OPT

            if(CurrentIndexInfo%IndexNum.gt.MaxIndex) MaxIndex=&
```

```
                                                        CurrentIndexInfo%IndexNum
            if(Reason.lt.0) exit
         end do
      close(unit=File_unit)
      write(*,'(A,I0)')"MaxIndex   =  ", MaxIndex
      return
      end SUBROUTINE GetMaxIndex
      SUBROUTINE GetFragmentDBCart (MolIdxInfo,UniqueSymNum,FragmentCart)
!*********************************************************************
!     Date last modified:                                            *
!     Author: Ibrahim Awad                                           *
!     Description: Return the cartesian coordinates of the fragments   *
!                  from the database cartesian file.                 *
!*********************************************************************
!
      implicit none

      integer, intent(in) :: UniqueSymNum
      type(IndexFile),dimension(:),allocatable, intent(in) :: MolIdxInfo
      type(FragAtomInfo),dimension(:,:),allocatable, intent(out) :: FragmentCart

      integer :: iatom, latom, indexi, newindex
      integer :: File_unit, Reason
      character(len=8) :: Element
      double precision :: XCart, YCart, ZCart
      logical :: Lerror

      allocate (FragmentCart(Natoms,Natoms))

      ! Return the coordinate of fragments in the database for the wanted index
      ! loop over the wanted index
      do indexi=1, UniqueSymNum
        ! check if the symbol is available within the database
        if (.not.MolIdxInfo(indexi)%Availability) cycle
        ! loop over the atoms of the fragment
        do iatom=1 , MolIdxInfo(indexi)%NAtoms
            call GET_unit (CartFilePath, File_unit, Lerror)
            open(UNIT=File_unit,file=CartFilePath, &
                 status='old',form='formatted')
            read(File_unit,*)
            do ! start, reading the file
              read(File_unit,fmt_cart,IOSTAT=Reason) newindex, latom, Element
              read(File_unit,fmt_cart,IOSTAT=Reason) XCart, YCart, ZCart
              if (Reason.lt.0) exit
              if(newindex.eq.MolIdxInfo(indexi)%IndexNum.and.latom.eq.iatom)then
                 FragmentCart(indexi,iatom)%element=Element
                 FragmentCart(indexi,iatom)%x=XCart
                 FragmentCart(indexi,iatom)%y=YCart
                 FragmentCart(indexi,iatom)%z=ZCart
                 exit
              end if
            end do ! reading the file
            close(unit=File_unit)
        end do ! iatom
      end do ! indexi
      end SUBROUTINE GetFragmentDBCart
      SUBROUTINE GetFragmentsProperties (MolIdxInfo,UniqueSymNum,FragProp)
!*********************************************************************
!     Date last modified:                                            *
!     Author: Ibrahim Awad                                           *
!     Description: Return the cartesian coordinates of the fragments   *
!                  from the database cartesian file.                 *
!*********************************************************************
```

```
!

      implicit none

      integer, intent(in) :: UniqueSymNum
      type (IndexFile), dimension(:), allocatable, intent(in) :: MolIdxInfo
      type (PropFile), dimension(:), allocatable, intent(out) :: FragProp

      integer :: indexi, newindex
      integer :: File_unit, Reason
      double precision :: EleNum, EV_Anal, EV_Num, EX, EXJ, ET
      double precision :: EC, Kaa, Jaa, Vee_A, ECK, EC_AnaNum
      logical :: Lerror

      allocate (FragProp(UniqueSymNum))

      ! Return the coordinate of fragments in the database for the wanted index
      ! loop over the wanted index
      FragProp%Availability=.false.
      do indexi=1, UniqueSymNum
         ! check if the symbol is available within the database
         if (.not.MolIdxInfo(indexi)%Availability) cycle
         FragProp(indexi)%Availability=.true.
         ! loop over the atoms of the fragment
         call GET_unit (PropFilePath, File_unit, Lerror)
         open(UNIT=File_unit,file=PropFilePath, status='old',form='formatted')
         read(File_unit,*)
         do ! start, reading the file
             read(File_unit,fmt_prop,IOSTAT=Reason) &
                & newindex,                  &
                & EleNum,            &
                & EX,                        &
                & EXJ,                       &
                & ET,                        &
                & EV_Anal,                   &
                & EV_Num,                    &
                & EC_AnaNum,                 &
                & EC,                        &
                & ECK,                       &
                & Jaa,                       &
                & vee_A


             if (Reason.lt.0) exit
             if (newindex.eq.MolIdxInfo(indexi)%IndexNum) then
                 FragProp(indexi)%IndexNum=newindex
                 FragProp(indexi)%ElectonNum=EleNum
                 FragProp(indexi)%K=EX
                 FragProp(indexi)%KHF=EXJ
                 FragProp(indexi)%T=ET
                 FragProp(indexi)%Vne_A=EV_Anal
                 FragProp(indexi)%Vne_N=EV_Num
                 FragProp(indexi)%Vee_AN=EC_AnaNum
                 FragProp(indexi)%J=EC
                 FragProp(indexi)%JHF=ECK
                 FragProp(indexi)%Jaa=Jaa
                 FragProp(indexi)%VeeA=Vee_A
                 exit
             end if
         end do ! reading the file
         close(unit=File_unit)
      end do ! indexi
      end SUBROUTINE GetFragmentsProperties
      end MODULE GetDataBaseInfo
```

```
      SUBROUTINE Add_To_DB
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
! MODULEs

      USE type_density
      USE QM_defaults
      USE N_integration
      USE NI_defaults
      USE Sorted_AIMDFT
      USE symbol_AIMDFT
      USE type_plotting
      USE GetMolecularProperties

      implicit none

      character (SymMax), dimension (:), allocatable ::  symbols
      type (FragsInfo), dimension(:), allocatable :: FragInfo
      type (FragAtomInfo), dimension(:,:), allocatable :: DatabaseFragAtoms
      type (FragAtomInfo), dimension(:,:), allocatable :: SortedFragAtoms
      integer, dimension(:), allocatable :: atomAdj
      double precision, dimension(:), allocatable :: Vpotl
      double precision, dimension(:), allocatable :: V12dr2
      double precision :: TraceAB
      double precision :: Xpt,Ypt,Zpt

      integer :: Iatom, newindex, NApts_atom_New, Jatom
      integer :: kfound
      integer :: IApoint
      integer :: Znum

      allocate (FragInfo(Natoms))
      allocate (SortedFragAtoms(Natoms,Natoms))
      allocate (DatabaseFragAtoms(Natoms,Natoms))
      allocate (symbols(Natoms))
      allocate (atomAdj(Natoms))

      CALL GET_object ('GRID', 'RADIAL', RADIAL_grid)
      CALL GET_object ('QM', 'ENERGY_COMPONENTS', Wavefunction)
      CALL GetSortCartTerm(NAtoms,SortedFragAtoms,FragInfo)
      CALL GetDBFormFragments(SortedFragAtoms,DatabaseFragAtoms,FragInfo)
      CALL GetFragSymbols(Level_Number,symbols)

      newindex=IndexNumber ! add the new symbols

      Iatom=1 ! Just add the intrest atom within the fragment.
      NAIMprint=1
      AIMprint(1)=1

!      write(*,*)
      CALL GET_object ('QM', 'ENERGY_VEE', 'NUMERICAL')
!      CALL GET_object ('QM', 'ENERGY_EXCHANGE', 'NUMERICAL')

!      CALL GET_object ('QM', 'ENERGY_COULOMB', 'MO')
!      write(*,*)
      CALL GET_object ('QM', 'ENERGY_COULOMB', 'NUMERICAL')
      write(*,*)
      CALL GET_object ('QM', 'ENERGY_KINETIC', 'NUMERICAL')
      write(*,*)
      CALL GET_object ('QM', 'ENERGY_VNE', 'NUMERICAL')
      write(*,*)
```

```
      Znum=CARTESIAN(Iatom)%Atomic_Number
      kfound=GRID_loc(Znum)
      NApts_atom=NApoints_atom(kfound)
      allocate(grid_points(1:NApts_atom))
      ! Save grid points
      do IApoint=1,NApts_atom
         grid_points(IApoint)%X=Egridpts(Iatom,IApoint)%X+CARTESIAN(Iatom)%X
         grid_points(IApoint)%Y=Egridpts(Iatom,IApoint)%Y+CARTESIAN(Iatom)%Y
         grid_points(IApoint)%Z=Egridpts(Iatom,IApoint)%Z+CARTESIAN(Iatom)%Z
         grid_points(IApoint)%w=Egridpts(Iatom,IApoint)%w
      end do

      ! compute the weight for all grid points
      allocate(Bweights(NApts_atom))
      CALL GET_weights (grid_points, NApts_atom, Iatom, Bweights)

      ! compute the charge for all grid points
      allocate (rho_Atom(NApts_atom))
      CALL GET_density (grid_points, NApts_atom, rho_Atom)

      CALL FindAdj(1,Iatom,atomAdj)

      ! compute the V12dr2 for all grid points
      allocate(Vpotl(1:NApts_atom))
      allocate(V12dr2(1:MATlen))
      do IApoint=1,NApts_atom
         Xpt=grid_points(IApoint)%X
         Ypt=grid_points(IApoint)%Y
         Zpt=grid_points(IApoint)%Z
         CALL I1E_V12dr2 (V12dr2, MATlen,Xpt,Ypt,Zpt)
         Vpotl(IApoint)=-TraceAB (PM0, V12dr2, NBasis)
      end do ! IApoint

      CALL Store_grid_points()
      CALL add_to_index_file()
      CALL add_to_cart_file()
      CALL add_to_prop_file(PropFilePath)

      deallocate (rho_Atom)
      deallocate (V12dr2)
      deallocate (Vpotl)
      deallocate (grid_points)
      deallocate (FragInfo)
      deallocate (SortedFragAtoms)
      deallocate (DatabaseFragAtoms)
      deallocate (symbols)

CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

      SUBROUTINE Store_grid_points()
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
! MODULEs

      implicit none

      type (FragAtomInfo), dimension(:), allocatable :: temp1
      type (FragAtomInfo), dimension(:), allocatable :: temp2
      character (len=:), allocatable:: GridFilenames
      logical :: Lerror
```

```
      integer :: File_unit

     ! Save grid points
      allocate (temp1(NApts_atom))
      allocate (temp2(NApts_atom))

      temp1%x=grid_points%x;temp1%y=grid_points%y;temp1%z=grid_points%z

      CALL DoCartRotation(temp1,temp2,NApts_atom,&
                          FragInfo(Iatom)%RMat,FragInfo(Iatom)%TMat)
      CALL INPUT_File_GRID (newindex, GridFilenames) ! just frag atom
      CALL GET_unit (GridFilenames, File_unit, Lerror)

      open(UNIT=File_unit,file=GridFilenames, status='REPLACE',form='formatted')
      write(File_unit,'(A20,7A24)') "X","Y","Z","Rho",&
                                    "BW","AW","Q=w.W.Rho","Vpol"
      NApts_atom_New=0
      do IApoint=1,NApts_atom
!     if(rho_Atom(IApoint)*grid_points(IApoint)%w*Bweights(IApoint).lt.1.0D-09)&
!     cycle
        NApts_atom_New=NApts_atom_New+1
        write(File_unit,Pfmt_rho_w_v) temp2(IApoint)%X,                          &
              &   temp2(IApoint)%Y,                                              &
              &   temp2(IApoint)%Z,                                              &
              &   rho_Atom(IApoint),                                             &
              &   Bweights(IApoint),                                             &
              &   grid_points(IApoint)%w,                                        &
              &   rho_Atom(IApoint)*grid_points(IApoint)%w*Bweights(IApoint),    &
              &   Vpotl(IApoint)
      end do ! Ipoint
      close(UNIT=File_unit)
      deallocate(temp1)
      deallocate(temp2)
      end SUBROUTINE Store_grid_points
      SUBROUTINE add_to_index_file()
!*****e*****************************************************************************
!     Date last modified:                                                          *
!     Author: Ibrahim Awad                                                         *
!     Description:                                                                 *
!*******************************************************************************
! MODULEs

      USE type_Weights

      implicit none

      character (28) :: Radnote, coreSize
      logical :: Lerror
      integer :: File_unit

      Radnote=""
      coreSize=""
      if (trim(RADIAL_grid).eq.'GILL') then
         write(Radnote,'(I2,A1,I2,A1,I2,A1,I3,A1)') NRPoints_Gill,&
         & '(',NApoints_Gill(1),',',NApoints_Gill(2),',',NApoints_Gill(3),')'
      endif
      if (trim(DEN_Partitioning).eq.'IAWAD') then
         write(coreSize,'(A)') Last_Core
      endif

      write(*,'(/A)') "*****************************************"
      write(*,*) "The fragment with symbol ", trim(Symbols(Iatom))
      write(*,*) "has been added to the database with index # ", newindex
      write(*,'(A/)') "*****************************************"
```

```
      CALL GET_unit (trim(IndexFilePath), File_unit, Lerror)
      open(UNIT=File_unit,file=trim(IndexFilePath), &
           status='old',form='formatted',position="append")
      write(File_unit,fmt_index) newindex,  &
                & Symbols(Iatom),           &
                & NAtoms,                   &
                & NApts_atom_New,           &
                & trim(RADIAL_grid),        &
                & trim(Radnote),            &
                & trim(Wavefunction),       &
                & trim(Basis_set_name),     &
                & trim(DEN_Partitioning),   &
                & trim(coreSize),           &
                & trim(OPT_status)
      close(unit=File_unit)

      end SUBROUTINE add_to_index_file
      SUBROUTINE add_to_prop_file(PropFilePath)
!*****e*****************************************************************************
!     Date last modified:                                                          *
!     Author: Ibrahim Awad                                                         *
!     Description:                                                                 *
!*******************************************************************************
! MODULEs

      implicit none

      character (20) :: Radnote
      logical :: Lerror
      integer :: File_unit
      double precision :: EV_Anal, EV_Num, EX, ET, EC, EXJ, ECK, EC_AnaNum
      double precision :: EleNum, vee_A
      character (*) :: PropFilePath


      write(*,'(a,I5)') "  Atomic properties for atom #", Iatom
      write(*,'(a)')    "  ++++++++++++++++++++++++++++++++++++++"
      CALL CalAtomicEle(grid_points, NApts_atom, rho_Atom, Bweights, EleNum) !
      write(*,'(a,F15.10)')"Number of Electrons, N           = ", EleNum

      EX = K_Atomic(Iatom)
      write(*,'(a,F15.10)')"Pure Exchange, K ( 2K_ab)        = ", EX

      EXJ = KHF_Atomic(Iatom)
      write(*,'(a,F15.10)')"HF Exchange, KHF ( 2K_ab+ Kaa)   = ", EXJ

      ET = Atomic_Kinetic(Iatom)
      write(*,'(a,F15.10)')"Kinetic energy Numerical, T      = ", ET

      ! Calculate the atomic potential energy for intrest fragment atom.
      CALL get_Vne_Atom_Analytical(EV_Anal)
      write(*,'(a,F15.10)')"Potential Energy Analytical, Vne = ", EV_Anal

      EV_Num = Atomic_Vne(Iatom)
      write(*,'(a,F15.10)')"Potential Energy Numerical, Vne  = ", EV_Num

      EC_AnaNum = Atomic_Coulomb(Iatom)
      write(*,'(a,F15.10)')"Coulomb Energy Anal/Num, Vee     = ", EC_AnaNum

      EC = J_Atomic(Iatom)
      write(*,'(a,F15.10)')"Pure Coulomb, J ( 4J_ab+ Jaa)    = ", EC

      ECK = JHF_Atomic(Iatom)
      write(*,'(a,F15.10)')"HF Coulomb, JHF ( 4J_ab+ 2Jaa)   = ", ECK
```

```
      write(*,'(a,F15.10)')"Jaa = Kaa                          = ", ECK-EC

      CALL CalcVeeSelfA(grid_points, NApts_atom, rho_Atom, Bweights, vee_A)
      write(*,'(a,F15.10)')"Coulomb Numerically Over A         = ", vee_A
      write(*,*)


      CALL GET_unit (trim(PropFilePath), File_unit, Lerror)
      open(UNIT=File_unit,file=trim(PropFilePath), &
           status='old',form='formatted',position="append")
      write(File_unit,fmt_prop) newindex,   &
                & EleNum,             &
                & EX,                       &
                & EXJ,                      &
                & ET,                       &
                & EV_Anal,                  &
                & EV_Num,                   &
                & EC_AnaNum,                &
                & EC,                       &
                & ECK,                      &
                & ECK-EC,                   &
                & vee_A

      close(unit=File_unit)

      end SUBROUTINE add_to_prop_file
      SUBROUTINE add_to_cart_file()
!*****e*****************************************************************************
!     Date last modified:                                                            *
!     Author: Ibrahim Awad                                                           *
!     Description:                                                                   *
!*****************************************************************************
! MODULEs

      implicit none
      integer :: katom
      integer :: File_unit
      logical :: Lerror

      CALL GET_unit (trim(CartFilePath), File_unit, Lerror)
      open(UNIT=File_unit,file=trim(CartFilePath), &
           status='old',form='formatted',position="append")

      do katom=1,NAtoms
         write(File_unit,fmt_cart) newindex,                                  &
                                 & katom,                                     &
                                 & DatabaseFragAtoms(Iatom,katom)%element,    &
                                 & DatabaseFragAtoms(Iatom,katom)%x,          &
                                 & DatabaseFragAtoms(Iatom,katom)%y,          &
                                 & DatabaseFragAtoms(Iatom,katom)%z
      end do
      close(unit=File_unit)

      end SUBROUTINE add_to_cart_file
      end SUBROUTINE Add_To_DB
```

```
      SUBROUTINE CalcMolePropertiesDB(DatabaseInform, FragProp, FRgrids)
!*****************************************************************************
!     Date last modified:                                                       *
!     Author: Ibrahim Awad                                                      *
!     Description:                                                              *
!*****************************************************************************
! MODULEs:
```

```
USE module_grid_points
USE mod_build
USE GetDataBaseInfo

implicit none

! Grid points by Jatom after RR
type (GridsDataBase), dimension(:,:), allocatable, INTENT(IN) :: FRgrids
type (IndexFile), dimension (:), allocatable, INTENT(IN) :: DatabaseInform
type (PropFile), dimension(:), allocatable, INTENT(IN) :: FragProp
type (type_grid_points), dimension(:,:), allocatable :: grid_point_DB

double precision, dimension(:,:), allocatable :: Bweights
double precision, dimension(:,:), allocatable :: PWeights
double precision, dimension(:,:), allocatable :: Vpot_r2

double precision, dimension(:), allocatable :: EleNum
double precision, dimension(:), allocatable :: pot_energy
double precision, dimension(:), allocatable :: colA_energy
double precision, dimension(:), allocatable :: colN_energy

double precision :: EleNum_tot
double precision :: pot_energy_tot
double precision :: colA_energy_tot
double precision :: colN_energy_tot
double precision :: EleNum_tot_prop
double precision :: pot_energy_A_tot_prop
double precision :: VeeAN_energy_tot_prop
double precision :: HFcoul_energy_tot_prop
double precision :: HFexchange_tot_prop
double precision :: coul_energy_tot_prop
double precision :: exchange_tot_prop
double precision :: kinetic_tot_prop
double precision :: Vnn_tot
double precision :: col_FullNum
double precision :: Virial

integer :: avlN, Jatom, points_num

CHARACTER (64) :: message

allocate (EleNum(Natoms))
allocate (pot_energy(Natoms))
allocate (colA_energy(Natoms))
allocate (colN_energy(Natoms))

call GET_object ('GRID', 'RADIAL', RADIAL_grid)

! compute the weight for all grid points
allocate (grid_point_DB(Natoms,MaxGridPnt))
allocate(PWeights(NAtoms,MaxGridPnt))
allocate(Bweights(NAtoms,MaxGridPnt))
allocate(Vpot_r2(NAtoms,MaxGridPnt))

grid_point_DB(1:Natoms,1:MaxGridPnt)%x=FRgrids(1:Natoms,1:MaxGridPnt)%x
grid_point_DB(1:Natoms,1:MaxGridPnt)%y=FRgrids(1:Natoms,1:MaxGridPnt)%y
grid_point_DB(1:Natoms,1:MaxGridPnt)%z=FRgrids(1:Natoms,1:MaxGridPnt)%z

do Jatom=1, Natoms
   if (.not.DatabaseInform(Jatom)%Availability) cycle
   points_num=DatabaseInform(Jatom)%GridsNum
   CALL GET_weights (grid_point_DB(Jatom,1:points_num), points_num, &
                     Jatom, Bweights(Jatom,1:points_num))
```

```fortran
      end do
      deallocate (grid_point_DB)

      write(*,'(/A)') '                         '
      write(*,'(A)')  '          '
      write(*,'(A)')  '                                                       '

      call calc_atomic_prop()

      message = 'Current partitioning weight'
      PWeights(1:Natoms,1:MaxGridPnt)=Bweights(1:Natoms,1:MaxGridPnt)
      call calc_density_prop(message)
      call prt_tot_energy

      message = 'Database partitioning weight'
      PWeights(1:Natoms,1:MaxGridPnt)=FRgrids(1:Natoms,1:MaxGridPnt)%BW
      call calc_density_prop(message)
      call prt_tot_energy

      call calc_error_density()

      !Print warning message
      avlN=0
      do Jatom=1, NAtoms
      if (Cartesian(JAtom)%Atomic_Number.eq.0) cycle
      if (.not.DatabaseInform(Jatom)%Availability) then
          avlN=avlN+1
          write(*,'(A36,I3,A17)') "WARNING :: the fragment of atom# ", Jatom,&
                                  " is not avaliable"
      end if
      end do
      if (avlN.ne.0) write(*,'(A12,I2,A19)') 'There are ', avlN, &
                                             ' warning messege[s]'
      call Stores_error_files()
      deallocate(EleNum)
CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

      SUBROUTINE prt_tot_energy ()
!***********************************************************************
!     Date last modified:                                              *
!     Author: Ibrahim Awad                                             *
!     Description:   Calculate Number of electrons                     *
!***********************************************************************
! MODULEs:
      implicit none

      call calc_Vnn()
      write(*,'(A32, F16.8)') "ThE Total Energy          =", Vnn_tot + &
                                                 pot_energy_tot + &
                                                colN_energy_tot + &
                                               kinetic_tot_prop + &
                                               HFexchange_tot_prop

      if (kinetic_tot_prop .ne. ZERO) then
         write(*,'(A32, F16.8/)')  "ThE Virial                =", -(Vnn_tot + &
                                                      pot_energy_tot + &
                                                     colN_energy_tot + &
                                                 HFexchange_tot_prop) / &
                                                 kinetic_tot_prop
      end if
```

```
      end SUBROUTINE  prt_tot_energy
      SUBROUTINE calc_density_prop(cw)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:   Calculate Number of electrons                             *
!*****************************************************************************
! MODULEs:
      implicit none

      CHARACTER (64), INTENT(IN) :: cw


      call calc_elec()
      call calc_Vne()
      call calc_Vee_FullNum()
      call calc_Vee_anal()

      write(*,'(/A)')'+-------------------------------------------------------+'
      write(*,'(A)') '| Calculated molecule properties using the stored       |'
      write(*,'(A)') '| electron density in each radial grid point            |'
      write(*,'(A8,A48,A2)') '| NOTE: ',cw,' |'
      write(*,'(A)') '+-----+---------------+---------------+----------------+'
      write(*,'(A)') '|Atom#|   Electrons    |    Vne        | Vee            |'
      write(*,'(A)') '+-----+---------------+---------------+----------------+'

      do Jatom=1, NAtoms
        if (.not.FragProp(Jatom)%Availability) cycle
      write(*,'(A1,I5,3(A1,F16.8),A1)') '|',Jatom, &
                                    &  '|',EleNum(Jatom), &
                                    &  '|',-pot_energy(Jatom), &
                                    &  '|', colN_energy(Jatom), &
                                    &  '|'
      end do
      write(*,'(A)') '+-----+---------------+---------------+----------------+'
      write(*,'(A6,3(A1,F16.8),A1)') '|Sum= ', &
                                     & '|',EleNum_tot ,&
                                     & '|',pot_energy_tot  ,&
                                     & '|',colN_energy_tot       ,'|'
      write(*,'(A)') '+-----+---------------+---------------+----------------+'

      end SUBROUTINE calc_density_prop
      SUBROUTINE calc_elec()
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:   Calculate Number of electrons                             *
!*****************************************************************************
! MODULEs:
      implicit none

      double precision :: point_charge
      integer :: Ipoint, Jatom

      EleNum=0.0d0
      EleNum_tot=0.0d0
      do Jatom=1, NAtoms
        if (.not.DatabaseInform(Jatom)%Availability) cycle
        do Ipoint=1, DatabaseInform(Jatom)%GridsNum
          point_charge  =  FRgrids(Jatom,Ipoint)%D*&
                           FRgrids(Jatom,Ipoint)%W*&
                           PWeights(Jatom,Ipoint)
          EleNum(Jatom) = EleNum(Jatom)+point_charge
        end do
```

```
        EleNum(Jatom) = FourPi*EleNum(Jatom)
        EleNum_tot = EleNum_tot + EleNum(Jatom)
      end do
      return
      end SUBROUTINE calc_elec
      SUBROUTINE calc_Vne()
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:   Calculate V_ne, Potential                               *
!*****************************************************************************
! MODULEs:
      implicit none

      double precision :: point_charge, NWeight, temp
      integer :: Ipoint, Jatom, Iatom, JZ_num

      pot_energy_tot=ZERO
      pot_energy = ZERO
      do Iatom=1, Natoms
      JZ_num = CARTESIAN(Iatom)%Atomic_Number
      if (JZ_num .le. 0) cycle
      do Jatom = 1,Natoms
        if (.not.DatabaseInform(Jatom)%Availability) cycle
        do Ipoint = 1,DatabaseInform(Jatom)%GridsNum
          point_charge  =  FRgrids(Jatom,Ipoint)%D*&
                           FRgrids(Jatom,Ipoint)%W*&
                           PWeights(Jatom,Ipoint)
          temp = JZ_num*point_charge/    &
             dsqrt((FRgrids(Jatom,Ipoint)%x-CARTESIAN(Iatom)%X)**2 + &
                   (FRgrids(Jatom,Ipoint)%y-CARTESIAN(Iatom)%Y)**2 + &
                   (FRgrids(Jatom,Ipoint)%z-CARTESIAN(Iatom)%Z)**2)
          pot_energy(Iatom) = pot_energy(Iatom) + temp
         end do ! Ipoint
      end do ! Jatom
      pot_energy(Iatom)= FourPi*pot_energy(Iatom)
      pot_energy_tot = pot_energy_tot - pot_energy(Iatom)
      end do
      return
      end SUBROUTINE calc_Vne
      SUBROUTINE calc_Vnn()
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:   Calculate V_nn, Potential                               *
!*****************************************************************************
! MODULEs:
      implicit none

      double precision :: temp
      integer :: Jatom, Iatom, JZ_num, IZ_num
      double precision ::  pot_energy_point

      Vnn_tot=ZERO
      do Iatom=1, Natoms
      pot_energy_point = ZERO
      IZ_num = CARTESIAN(Iatom)%Atomic_Number
      if (IZ_num .le. 0) cycle
      do Jatom = Iatom+1, Natoms
        JZ_num = CARTESIAN(Jatom)%Atomic_Number
        if (JZ_num .le. 0) cycle
          temp = JZ_num*IZ_num/    &
             dsqrt((CARTESIAN(Jatom)%x-CARTESIAN(Iatom)%X)**2 + &
                   (CARTESIAN(Jatom)%y-CARTESIAN(Iatom)%Y)**2 + &
```

```
                  (CARTESIAN(Jatom)%z-CARTESIAN(Iatom)%Z)**2)
         pot_energy_point = pot_energy_point + temp
      end do ! Jatom
      Vnn_tot = Vnn_tot + pot_energy_point
      end do
      write(*,'(/A32,F16.8)')   "Nuclear repulsion (Vnn)   =", Vnn_tot
      return
      end SUBROUTINE calc_Vnn
      SUBROUTINE calc_Vee_anal()
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:   Calculate V_ee (half analytical, from the fragments),   *
!                    Coulomb repulsion                                       *
!*****************************************************************************
! MODULEs:
      implicit none

      double precision :: point_charge, NWeight
      integer :: Ipoint, Jatom, JZ_num

      colA_energy = ZERO
      colA_energy_tot=ZERO
      do Jatom=1, Natoms
      JZ_num = CARTESIAN(Jatom)%Atomic_Number
      if (JZ_num .le. 0) cycle
      do Ipoint = 1,DatabaseInform(Jatom)%GridsNum
        if (.not.DatabaseInform(Jatom)%Availability) cycle
          point_charge  =  FRgrids(Jatom,Ipoint)%VpotA*&
                           FRgrids(Jatom,Ipoint)%D*&
                           FRgrids(Jatom,Ipoint)%W*&
                           PWeights(Jatom,Ipoint)/TWO
          colA_energy(Jatom) = colA_energy(Jatom) + point_charge
      end do
      colA_energy(Jatom) = FourPi*colA_energy(Jatom)
      colA_energy_tot = colA_energy_tot + colA_energy(Jatom)
      end do
      return
      end SUBROUTINE calc_Vee_anal
      SUBROUTINE calc_Vee_FullNum()
 !*****************************************************************************
 !    Date last modified:                                                     *
 !    Author: Ibrahim Awad                                                    *
 !    Description:   Calculate V_ee (Full Numerical), Coulomb repulsion       *
 !*****************************************************************************
 ! MODULEs:
      implicit none

      double precision :: point_charge, NWeight
      integer :: Ipoint, Jatom, JZ_num
      double precision :: NWeightI, NWeightJ, temp, Pcharge
      integer :: Jpoint, Iatom

      colN_energy=ZERO
      colN_energy_tot=ZERO
      do Iatom=1, Natoms
         JZ_num = CARTESIAN(Iatom)%Atomic_Number
         if (JZ_num .le. 0) cycle
         do Ipoint = 1,DatabaseInform(Iatom)%GridsNum
            if (.not.DatabaseInform(Iatom)%Availability) cycle
            do Jatom=1, Natoms
               JZ_num = CARTESIAN(Jatom)%Atomic_Number
               if (JZ_num .le. 0) cycle
               do Jpoint = 1,DatabaseInform(Jatom)%GridsNum
```

```
                if (.not.DatabaseInform(Jatom)%Availability) cycle
                if(Ipoint.eq.Jpoint .and. Iatom.eq.Jatom) cycle
                NWeightJ=PWeights(Jatom,Jpoint)
                NWeightI=PWeights(Iatom,Ipoint)
                Pcharge=FRgrids(Iatom,Ipoint)%D*FRgrids(Jatom,Jpoint)%D &
                     & *FRgrids(Iatom,Ipoint)%W*FRgrids(Jatom,Jpoint)%W &
                     & *NWeightJ*NWeightI
                temp = Pcharge/    &
                DSQRT((FRgrids(Jatom,Jpoint)%x-FRgrids(Iatom,Ipoint)%x)**2+ &
                      (FRgrids(Jatom,Jpoint)%y-FRgrids(Iatom,Ipoint)%y)**2+ &
                      (FRgrids(Jatom,Jpoint)%z-FRgrids(Iatom,Ipoint)%z)**2)
                colN_energy(Iatom)=colN_energy(Iatom)+temp
             end do ! Jpoint = Ipoint + 1,DatabaseInform(Jatom)%GridsNum
          end do ! Jatom=1, Natoms
       end do ! Ipoint = 1,DatabaseInform(Iatom)%GridsNum
       colN_energy(Iatom)=0.5d0*FourPi*FourPi*colN_energy(Iatom)
       colN_energy_tot=colN_energy_tot+colN_energy(Iatom)
    end do ! Iatom=1, Natoms
    return
    end SUBROUTINE calc_Vee_FullNum
    SUBROUTINE calc_Vpotr2_pt(Iatom,Ipoint,vpot_pt)
!**********************************************************************
!    Date last modified:                                              *
!    Author: Ibrahim Awad                                             *
!    Description:   Calculate V_ee (Full Numerical), Coulomb repulsion*
!**********************************************************************
! MODULEs:
    implicit none

    integer :: Jatom,Jpoint,JZ_num,Iatom,Ipoint
    double precision :: NWeightJ, PchargeJ, densityXYZ,GET_density_at_xyz
    double precision :: vpot_pt, SQRTr

    vpot_pt=ZERO
    do Jatom=1, Natoms
       JZ_num = CARTESIAN(Jatom)%Atomic_Number
       if (JZ_num .le. 0) cycle
       if (.not.DatabaseInform(Jatom)%Availability) cycle
       do Jpoint = 1,DatabaseInform(Jatom)%GridsNum
          if(Ipoint.eq.Jpoint .and. Iatom.eq.Jatom) cycle
            NWeightJ=PWeights(Jatom,Jpoint)
            PchargeJ=FRgrids(Jatom,Jpoint)%W*&
                     NWeightJ*FRgrids(Jatom,Jpoint)%D  !densityXYZ!
            SQRTr=DSQRT((FRgrids(Jatom,Jpoint)%x-FRgrids(Iatom,Ipoint)%x)**2+&
                (FRgrids(Jatom,Jpoint)%y-FRgrids(Iatom,Ipoint)%y)**2+ &
                (FRgrids(Jatom,Jpoint)%z-FRgrids(Iatom,Ipoint)%z)**2)
            vpot_pt = vpot_pt + PchargeJ/SQRTr
       end do ! Ipoint = 1,DatabaseInform(Iatom)%GridsNum
    end do ! Iatom=1, Natoms
    vpot_pt=vpot_pt*FourPi
    return
    end SUBROUTINE calc_Vpotr2_pt
    SUBROUTINE calc_Vee_FullNum2()
!**********************************************************************
!    Date last modified:                                              *
!    Author: Ibrahim Awad                                             *
!    Description:   Calculate V_ee (Full Numerical), Coulomb repulsion*
!**********************************************************************
! MODULEs:
    implicit none

    double precision :: point_chargeI, NWeight
    integer :: Ipoint, Jatom, JZ_num
    double precision :: NWeightI, vpot_pt, temp
```

```
      double precision :: PchargeI, densityXYZ, GET_density_at_xyz
      integer :: Jpoint, Iatom

      colN_energy=ZERO
      colN_energy_tot=ZERO
      do Iatom=1, Natoms
         JZ_num = CARTESIAN(Iatom)%Atomic_Number
         if (JZ_num .le. 0) cycle
         do Ipoint = 1,DatabaseInform(Iatom)%GridsNum
            if (.not.DatabaseInform(Iatom)%Availability) cycle
            NWeightI=PWeights(Iatom,Ipoint)
            PchargeI=FRgrids(Iatom,Ipoint)%W*NWeightI*FRgrids(Iatom,Ipoint)%D
            call calc_Vpotr2_pt(Iatom,Ipoint,vpot_pt)
            temp = PchargeI*vpot_pt!FRgrids(Iatom,Ipoint)%VpotA!
            colN_energy(Iatom)=colN_energy(Iatom)+temp
         end do ! Ipoint = 1,DatabaseInform(Iatom)%GridsNum
         colN_energy(Iatom)=0.50d0*FourPi*colN_energy(Iatom)
         colN_energy_tot=colN_energy_tot+colN_energy(Iatom)
      end do ! Iatom=1, Natoms
      return
      end SUBROUTINE calc_Vee_FullNum2
      SUBROUTINE calc_atomic_prop()
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:   Calculate Number of electrons                           *
!*****************************************************************************
! MODULEs:
      implicit none

      integer :: Jatom

      EleNum_tot_prop=ZERO
      pot_energy_A_tot_prop=ZERO
!      pot_energy_N_tot_prop=ZERO
      VeeAN_energy_tot_prop=ZERO
      HFexchange_tot_prop=ZERO
      HFcoul_energy_tot_prop=ZERO
      exchange_tot_prop=ZERO
      coul_energy_tot_prop=ZERO
      kinetic_tot_prop=ZERO

      write(*,'(/A)')'+-------------------------------------------------------------&
        &-----------------------------------------------------------------+'
      write(*,'(A)') '|    Calculated molecule properties using the stored frag&
        &ments properties                                                  |'
      write(*,'(A)') '|    (already calculated, using fragment weigths and geom&
        &etry)                                                             |'
      write(*,'(A)') '+-----+-------------+-------------+-------------+-----&
        &--------+-------------+-------------+-------------+-------------+'
      write(*,'(A)') '|Atom#|   Electrons  | Potential Vne| Vee Ana/Num  | HF &
       &Coulomb J |HF Exchange K |  Coulomb J   |  Exchange K  |  Kinetic T   |'
      write(*,'(A)') '+-----+-------------+-------------+-------------+-----&
        &--------+-------------+-------------+-------------+-------------+'
      do Jatom=1, NAtoms
        if (.not.FragProp(Jatom)%Availability) cycle
        write(*,'(A1,I5,8(A1,F14.8),A1)') '|',Jatom, &
                    & '|',FragProp(Jatom)%ElectonNum, &
                    & '|',FragProp(Jatom)%Vne_A, &
         !          & '|',FragProp(Jatom)%Vne_N, & ! No need same as Vne_A
                    & '|',FragProp(Jatom)%Vee_AN, &
                    & '|',FragProp(Jatom)%JHF, &
                    & '|',FragProp(Jatom)%KHF, &
                    & '|',FragProp(Jatom)%J,   &
```

```
                    & '|',FragProp(Jatom)%K,   &
                    & '|',FragProp(Jatom)%T, '|'

       EleNum_tot_prop=EleNum_tot_prop+FragProp(Jatom)%ElectonNum
       pot_energy_A_tot_prop=pot_energy_A_tot_prop+FragProp(Jatom)%Vne_A
!       pot_energy_N_tot_prop = pot_energy_N_tot_prop + FragProp(Jatom)%Vne_N
       VeeAN_energy_tot_prop=VeeAN_energy_tot_prop+FragProp(Jatom)%Vee_AN
       HFcoul_energy_tot_prop=HFcoul_energy_tot_prop + FragProp(Jatom)%JHF
       HFexchange_tot_prop=HFexchange_tot_prop + FragProp(Jatom)%KHF
       coul_energy_tot_prop=coul_energy_tot_prop + FragProp(Jatom)%J
       exchange_tot_prop=exchange_tot_prop + FragProp(Jatom)%K
       kinetic_tot_prop=kinetic_tot_prop + FragProp(Jatom)%T
     end do
     write(*,'(A)') '+-----+--------------+--------------+--------------+-----&
       &---------+--------------+--------------+--------------+--------------+'
     write(*,'(A6,8(A1,F14.8),A1)') '|Sum= ', &
                                    & '|',EleNum_tot_prop ,&
                                    & '|',pot_energy_A_tot_prop  ,&
                           !        No need same as Vne_A
                           !        & '|',pot_energy_N_tot_prop  ,&
                                    & '|',VeeAN_energy_tot_prop ,&
                                    & '|',HFcoul_energy_tot_prop ,&
                                    & '|',HFexchange_tot_prop    ,&
                                    & '|',coul_energy_tot_prop   ,&
                                    & '|',exchange_tot_prop      ,&
                                    & '|',kinetic_tot_prop       ,'|'
     write(*,'(A)') '+-----+--------------+--------------+--------------+-----&
                    &---------+--------------+--------------+--------------+--&
                    &------------+'
     return
     end SUBROUTINE calc_atomic_prop
     SUBROUTINE calc_error_density()
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description: Print the correct and the AIMDFT number of electrons      *
!*****************************************************************************
! MODULEs:
     implicit none

     double precision :: densityXYZ, GET_density_at_xyz
     integer :: Ipoint
     integer :: Jatom
     logical :: Lerror
     character (60) :: PlotFilenames
     double precision :: xpoint, ypoint, zpoint
     double precision :: density_w,ABSerror1, ABSerror2
     double precision :: AtomTotError,TotError1,TotError2
     double precision :: TDensity, TAIMDensity1,TAIMDensity2
     double precision :: AIMDFTsity_w1,AIMDFTsity_w2
     double precision, dimension(:), allocatable ::  AtomDensity
     double precision, dimension(:), allocatable ::  AtomAIMDFTDensity1
     double precision, dimension(:), allocatable ::  AtomAIMDFTDensity2
     double precision, dimension(:), allocatable ::  ErrorDensity_w1
     double precision, dimension(:), allocatable ::  ErrorDensity_w2

     allocate(AtomDensity(NAtoms))
     allocate(AtomAIMDFTDensity1(NAtoms))
     allocate(AtomAIMDFTDensity2(NAtoms))
     allocate(ErrorDensity_w1(NAtoms))
     allocate(ErrorDensity_w2(NAtoms))

     TDensity=ZERO
     TAIMDensity1=ZERO
```

```
      TAIMDensity2=ZERO
      TotError1=ZERO
      TotError2=ZERO
      AtomDensity=ZERO
      AtomAIMDFTDensity1=ZERO
      AtomAIMDFTDensity2=ZERO
      ErrorDensity_w1=ZERO
      ErrorDensity_w2=ZERO
      write(*,'(/A6,A12,6A14)') 'Atom#','AIM_FD','AIM_CW','HF','|Error FD|', &
                                & '|Error CW|','|%Error FD|','|%Error CW|'
      do Jatom=1, NAtoms
         if (.not.DatabaseInform(Jatom)%Availability) cycle
         do Ipoint=1, DatabaseInform(Jatom)%GridsNum

            xpoint=FRgrids(Jatom,Ipoint)%x-CARTESIAN(Jatom)%x
            ypoint=FRgrids(Jatom,Ipoint)%y-CARTESIAN(Jatom)%y
            zpoint=FRgrids(Jatom,Ipoint)%z-CARTESIAN(Jatom)%z

            densityXYZ=GET_density_at_xyz(FRgrids(Jatom,Ipoint)%x,&
                                          FRgrids(Jatom,Ipoint)%y,&
                                          FRgrids(Jatom,Ipoint)%z)
            density_w=densityXYZ*FRgrids(Jatom,Ipoint)%W*BWeights(Jatom,Ipoint)
            AIMDFTsity_w1=FRgrids(Jatom,Ipoint)%D*&
                          FRgrids(Jatom,Ipoint)%W*&
                          FRgrids(Jatom,Ipoint)%BW
            AIMDFTsity_w2=FRgrids(Jatom,Ipoint)%D*&
                          FRgrids(Jatom,Ipoint)%W*&
                          BWeights(Jatom,Ipoint)
            ABSerror1=AIMDFTsity_w1-density_w
            ABSerror2=AIMDFTsity_w2-density_w

            AtomDensity(Jatom)=AtomDensity(Jatom)+density_w
            AtomAIMDFTDensity1(Jatom)=AtomAIMDFTDensity1(Jatom)+AIMDFTsity_w1
            AtomAIMDFTDensity2(Jatom)=AtomAIMDFTDensity2(Jatom)+AIMDFTsity_w2
         end do

         AtomDensity(Jatom)=FourPi*AtomDensity(Jatom)
         AtomAIMDFTDensity1(Jatom)=FourPi*AtomAIMDFTDensity1(Jatom)
         AtomAIMDFTDensity2(Jatom)=FourPi*AtomAIMDFTDensity2(Jatom)

         TDensity=TDensity+AtomDensity(Jatom)
         TAIMDensity1=TAIMDensity1+AtomAIMDFTDensity1(Jatom)
         TAIMDensity2=TAIMDensity2+AtomAIMDFTDensity2(Jatom)

         ErrorDensity_w1(Jatom)=AtomAIMDFTDensity1(Jatom)-AtomDensity(Jatom)
         ErrorDensity_w2(Jatom)=AtomAIMDFTDensity2(Jatom)-AtomDensity(Jatom)

         TotError1=TotError1+ErrorDensity_w1(Jatom)
         TotError2=TotError2+ErrorDensity_w2(Jatom)

         if(AtomDensity(Jatom).ne.ZERO) then
         write(*,'(I4,7F14.8)') Jatom, &
                       & AtomAIMDFTDensity1(Jatom),  &
                       & AtomAIMDFTDensity2(Jatom),  &
                       & AtomDensity(Jatom),         &
                       & ErrorDensity_w1(Jatom),     &
                       & ErrorDensity_w2(Jatom),     &
                       & ErrorDensity_w1(Jatom)/AtomDensity(Jatom)*100.0d0, &
                       & ErrorDensity_w2(Jatom)/AtomDensity(Jatom)*100.0d0
         end if
      end do
      if (TDensity.ne.ZERO) then
         write(*,'(A4,7F14.8)') 'Sum=', TAIMDensity1,TAIMDensity2, &
                                & TDensity,TotError1,TotError2, &
```

```
                                    & TotError1/TDensity*100.0d0, &
                                    & TotError2/TDensity*100.0d0
      end if
      return
      end SUBROUTINE calc_error_density
      SUBROUTINE Stores_error_files()
!*************************************************************************
!     Date last modified:                                                *
!     Author: Ibrahim Awad                                               *
!     Description: ! Just to check the denisty at each radial grid point....   *
!*************************************************************************
! MODULEs:
      implicit none

      double precision :: densityXYZ, GET_density_at_xyz
      integer :: File_unit
      integer :: Ipoint
      integer :: Jatom
      logical :: Lerror
      character (60) :: PlotFilenames
      double precision :: xpoint, ypoint, zpoint
      double precision :: ABSerror1, ABSerror2
      double precision :: density_w, AIMDFTsity_w1,AIMDFTsity_w2

      do Jatom=1, NAtoms
         if (.not.DatabaseInform(Jatom)%Availability) cycle
         !<<<<<<<<<<<IMPORTANT:::::: should be change
         call INPUT_File_FRAG ('GRIDError_','.dat',Jatom, PlotFilenames)
         call GET_unit (PlotFilenames, File_unit, Lerror)
         open(UNIT=File_unit,file=PlotFilenames, &
              status='REPLACE',form='formatted')
         write(File_unit,'(3A12,A18,6A17)') 'X','Y','Z','Bweight','RadWeight', &
                                         & 'AIM_Density_FW','AIM_Density_CW', &
                                         & 'Density_w','Diff1','Diff2'
         do Ipoint=1, DatabaseInform(Jatom)%GridsNum

            xpoint=FRgrids(Jatom,Ipoint)%x-CARTESIAN(Jatom)%x
            ypoint=FRgrids(Jatom,Ipoint)%y-CARTESIAN(Jatom)%y
            zpoint=FRgrids(Jatom,Ipoint)%z-CARTESIAN(Jatom)%z

            densityXYZ=GET_density_at_xyz (FRgrids(Jatom,Ipoint)%x, &
                                           FRgrids(Jatom,Ipoint)%y, &
                                           FRgrids(Jatom,Ipoint)%z)
            density_w=densityXYZ*&
                      FRgrids(Jatom,Ipoint)%W*&
                      BWeights(Jatom,Ipoint)
            AIMDFTsity_w1=FRgrids(Jatom,Ipoint)%D*&
                          FRgrids(Jatom,Ipoint)%W*&
                          FRgrids(Jatom,Ipoint)%BW
            AIMDFTsity_w2=FRgrids(Jatom,Ipoint)%D*&
                          FRgrids(Jatom,Ipoint)%W*&
                          BWeights(Jatom,Ipoint)
            ABSerror1=AIMDFTsity_w1-density_w
            ABSerror2=AIMDFTsity_w2-density_w

            write(File_unit,Pfmt_rho_dd)   xpoint,   &
                                         & ypoint,   &
                                         & zpoint,   &
                                         & FRgrids(Jatom,Ipoint)%BW,   &
                                         & FRgrids(Jatom,Ipoint)%W,   &
                                         & AIMDFTsity_w1,   &
                                         & AIMDFTsity_w2,   &
                                         & density_w   ,    &
                                         & ABSerror1,       &
```

```
                                        & ABSerror2
         end do
         close(UNIT=File_unit)
      end do
      return
      end SUBROUTINE Stores_error_files
      end SUBROUTINE CalcMolePropertiesDB
```

```
      SUBROUTINE Build_Molecule_DB(DatabaseInform,&
                                   FragProp,&
                                   FRgrids,&
                                   CorrFragAtom)
!*****e*****************************************************************************
!     Date last modified:                                                            *
!     Author: Ibrahim Awad                                                           *
!     Description:                                                                   *
!*****************************************************************************
! MODULEs:

      USE symbol_AIMDFT
      USE sorted_AIMDFT
      USE matrix_print

      implicit none

      ! Grid points by Jatom after RR
      type(GridsDataBase),dimension(:,:),allocatable,intent(out)::FRgrids
      type(IndexFile),dimension (:),allocatable,intent(out)::DatabaseInform
      type(PropFile),dimension(:),allocatable,intent(out)::FragProp
      ! Adjusted the rotated database coordinate to fit the current fragements
      ! coordinates and the intreseted atom in the center.
      type(FragAtomInfo),dimension(:,:),allocatable,intent(out)::CorrFragAtom
      type(FragAtomInfo),dimension(:,:),allocatable :: &
       & CurrFragAtoms,     &! current fragment coordinates of the molecule
       & DBCurrFragAtoms,   &! database form of current fragment coordinates
       & DBUniqueFragAtoms, &! The database coordinates for the unique fragments
       & DBFragAtoms,       &! The database coordinates for all atoms fragments
       & RotDBFragAtoms      ! Rotated database coordinate to fit the
                             ! current fragemnts coordinates.
      type(FragAtomInfo),dimension(:),allocatable  :: term1, term2, term3
      type(TypeFragMat),dimension (:),allocatable :: FragAtomsMatrix
      ! Information for current fragments of molecule.
      type(FragsInfo),dimension(:),allocatable :: CurrFragInfo
      ! Contain index file information.
      type(IndexFile),dimension(:),allocatable :: MolIdxInfo
      ! The atomic properties for the fragments from the datbase
      type(PropFile),dimension(:),allocatable :: DBFragProp

      double precision,dimension (:,:),allocatable :: RR
      double precision,dimension (:),allocatable :: TT
      double precision,dimension (:),allocatable :: TTCORR

      character (SymMax),dimension (:),allocatable ::  symbols
      character (SymMax),dimension (:),allocatable ::  UniSym
      integer,dimension (:),allocatable :: UniSymMUNIdx

      integer :: UniSymMum
      integer :: IndexFoundNum
      integer :: ifound
      integer :: indexi
      integer :: Ipoint
      integer :: File_unit
      integer :: Reason
      integer :: Jatom
```

```
      integer :: iatom
      integer :: symI


      character (SymMax):: SymbolINDX
      character (SymMax):: searchsym
      character (25) :: GridType
      character(len=:),allocatable :: GridFilenames
      character (60) :: PlotFilenames
      character (60) :: PlotFilenames2

      logical :: Lerror2
      logical :: Lerror

      double precision,dimension (1:3) :: NewPoint
      double precision,dimension (1:3) :: point


      double precision :: DEN, BeckW, AngW, Qval, Vpot
      character(len=8) :: ele

      if(.not. allocated(FRgrids))  allocate (FRgrids(Natoms,MaxGridPnt))
      if(.not. allocated(DatabaseInform)) allocate (DatabaseInform(Natoms))
      if(.not. allocated(CorrFragAtom)) allocate (CorrFragAtom(Natoms,Natoms))
      if(.not. allocated(FragProp)) allocate (FragProp(Natoms))
      allocate (FragAtomsMatrix(Natoms))
      allocate (CurrFragAtoms(Natoms,Natoms))
      allocate (DBCurrFragAtoms(Natoms,Natoms))
      allocate (DBUniqueFragAtoms(Natoms,Natoms))
      allocate (DBFragAtoms(Natoms,Natoms))
      allocate (RotDBFragAtoms(Natoms,Natoms))
      allocate (CurrFragInfo(Natoms))
      allocate (MolIdxInfo(Natoms))
      allocate (DBFragProp(Natoms))
      allocate (RR(1:3,1:3))
      allocate (TT(1:3))
      allocate (TTCORR(1:3))
      allocate (symbols(Natoms))
      allocate (UniSym(Natoms))
      allocate (UniSymMUNIdx(Natoms))

      ! Partioning the molecule to its fragemnt and complete the valance
      CALL GetSortCartTerm(Level_Number,CurrFragAtoms,CurrFragInfo)
     ! CALL GenerateFragments(Level_Number,CurrFragAtoms,CurrFragInfo)
      ! find the symbols of the fragments
      CALL GetFragSymbols(Level_Number,symbols)
      ! find the unique symbols for the molecule and their number
      CALL GetUniSym (Level_Number, UniSym, UniSymMUNIdx, UniSymMum)
      ! return the MolIdxInfo (all information from the index file)
      ! and the availability, ......
      ! from the database at specific conditions
      CALL GetMoleculeAtomsIndex (MolIdxInfo,UniSym,UniSymMum,IndexFoundNum)
      ! Return the coordinate of fragments from
      ! the database for the wanted index
      CALL GetFragmentDBCart (MolIdxInfo,UniSymMum,DBUniqueFragAtoms)
      ! Return the atomic properties of fragments
      ! from the database for the wanted index
      CALL GetFragmentsProperties (MolIdxInfo,UniSymMum,DBFragProp)
      ! Project the database information to the molecule atoms
      DBFragAtoms%x=0.d0
      DBFragAtoms%y=0.d0
      DBFragAtoms%z=0.d0
      DBFragAtoms%element="NOT"
      DatabaseInform(1:NAtoms)%Availability=.false.
```

```
      FragProp(1:NAtoms)%Availability=.false.
      DatabaseInform(1:Natoms)%NAtoms=CurrFragInfo(1:Natoms)%NAtoms
      do Jatom=1, NAtoms  ! loop over all atoms
         do indexi=1, UniSymMum
            ! loop over the available database
            if (.not.MolIdxInfo(indexi)%Availability) cycle
            if (symbols(Jatom).ne.MolIdxInfo(indexi)%Symbol) cycle

            ! first: project the index information
            DatabaseInform(Jatom)=MolIdxInfo(indexi)
            ! There is available index for the Jatom
            DatabaseInform(Jatom)%Availability=.true.

            ! Second: Project the cartizeian coordinates for the fragmnets
            DBFragAtoms(Jatom,1:MolIdxInfo(indexi)%NAtoms)= &
          & DBUniqueFragAtoms(indexi,1:MolIdxInfo(indexi)%NAtoms)

            ! Third: Project the atomic properties for the fragments
            FragProp(Jatom)=DBFragProp(indexi)
            ! There is available atomic properties for the Jatom
            FragProp(Jatom)%Availability=.true.
         end do
      end do
      DBFragAtoms%type=0
      DBFragAtoms%number=0
      DBFragAtoms%factor=0
      DBFragAtoms%Ctype="NOT"

      ! Calculate RR and TT for each atom
      do JAtom=1, NAtoms
         if (.not.DatabaseInform(Jatom)%Availability) cycle
         allocate (term1(DatabaseInform(Jatom)%NAtoms))
         allocate (term2(DatabaseInform(Jatom)%NAtoms))
         allocate (term3(DatabaseInform(Jatom)%NAtoms))
         ! Needed to be in one dimension to get RR and TT
         term1(1:DatabaseInform(Jatom)%NAtoms)=&
         CurrFragAtoms(Jatom,1:DatabaseInform(Jatom)%NAtoms)
         term2(1:DatabaseInform(Jatom)%NAtoms)=&
         DBFragAtoms(Jatom,1:DatabaseInform(Jatom)%NAtoms)
         call GetCartRotMatrixW(term1,term2,DatabaseInform(Jatom)%NAtoms,RR,TT)
         call DoCartRotation(term2,term3,DatabaseInform(Jatom)%NAtoms,RR,TT)
         FragAtomsMatrix(Jatom)%RMAT=RR
         FragAtomsMatrix(Jatom)%TMAT=TT
         RotDBFragAtoms(Jatom,1:DatabaseInform(Jatom)%NAtoms)=&
                   term3(1:DatabaseInform(Jatom)%NAtoms)
         RotDBFragAtoms(Jatom,1:DatabaseInform(Jatom)%NAtoms)%element=&
                  & DBFragAtoms(Jatom,1:DatabaseInform(Jatom)%NAtoms)%element
         deallocate (term1)
         deallocate (term2)
         deallocate (term3)
      end do

      ! calculate TTCORR (let first atom at 0,0,0 )
      do JAtom=1, NAtoms
         if (.not.DatabaseInform(Jatom)%Availability) cycle
         do ifound=1, DatabaseInform(Jatom)%NAtoms
            FragAtomsMatrix(Jatom)%TMATCORR(1) = -RotDBFragAtoms(Jatom,1)%x+&
                                                  CurrFragAtoms(Jatom,1)%x
            FragAtomsMatrix(Jatom)%TMATCORR(2) = -RotDBFragAtoms(Jatom,1)%y+&
                                                  CurrFragAtoms(Jatom,1)%y
            FragAtomsMatrix(Jatom)%TMATCORR(3) = -RotDBFragAtoms(Jatom,1)%z+&
                                                  CurrFragAtoms(Jatom,1)%z
         end do
      end do
```

```
      ! PUT the Data in An Array
      ! the GriD points
      do Jatom=1, NAtoms
         if (.not.DatabaseInform(Jatom)%Availability) cycle
         call INPUT_File_GRID (DatabaseInform(Jatom)%IndexNum, GridFilenames)
         call GET_unit (GridFilenames, File_unit, Lerror)
         open(UNIT=File_unit,file=GridFilenames, status='OLD',form='formatted')
         read(File_unit,*)
         Ipoint=0
         do
            read(File_unit,Pfmt_rho_w_v,IOSTAT=Reason) point(1),&
                                                       point(2),&
                                                       point(3),&
                                                       DEN,&
                                                       BeckW,&
                                                       AngW,&
                                                       Qval,&
                                                       Vpot
            if (Reason.lt.0) exit
            Ipoint=Ipoint+1
            NewPoint= Matmul(FragAtomsMatrix(Jatom)%RMAT,point) + &
                           & FragAtomsMatrix(Jatom)%TMAT + &
                           & FragAtomsMatrix(Jatom)%TMATCORR
            FRgrids(Jatom,Ipoint)%x=NewPoint(1)
            FRgrids(Jatom,Ipoint)%y=NewPoint(2)
            FRgrids(Jatom,Ipoint)%z=NewPoint(3)
            FRgrids(Jatom,Ipoint)%D=DEN
            FRgrids(Jatom,Ipoint)%BW=BeckW
            FRgrids(Jatom,Ipoint)%W=AngW
            FRgrids(Jatom,Ipoint)%Q=Qval
            FRgrids(Jatom,Ipoint)%VpotA=Vpot
         end do
         close(UNIT=File_unit)
      end do

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  end  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      ! JUST PRINTING (YOU CAN DELETE THE BELOW LINES)
      !
      write(*,'(A)') "***********************************"
      write(*,'(A)') " The unique Symbols in the molecule "
      write(*,'(A)') "***********************************"
      do symI=1, UniSymMum
        write (*,'(A,I4,A,A)') "SYM# ", symI, " is ", trim(UniSym(symI))
      end do
      !!!!!!!!!!!!!!!!!! end OF PRINTING
      ! JUST PRINTING (YOU CAN DELETE THE BELOW LINES)
      !
      write(*,'(A)') "***********************************"
      write(*,'(A)') "           CART OF INDEX           "
      write(*,'(A)') "***********************************"
      do indexi=1, UniSymMum
        if (.not.MolIdxInfo(indexi)%Availability) cycle
        write(*,*) "index # ",MolIdxInfo(indexi)%IndexNum, &
                   "For Symbol ", MolIdxInfo(indexi)%Symbol
        do iatom=1 , MolIdxInfo(indexi)%NAtoms
            write(*,'(I3,3X,A3,3F16.8)') &
            & iatom, &
            & DBUniqueFragAtoms(indexi,iatom)%element, &
            & DBUniqueFragAtoms(indexi,iatom)%x*CartPrnFactor, &
            & DBUniqueFragAtoms(indexi,iatom)%y*CartPrnFactor, &
            & DBUniqueFragAtoms(indexi,iatom)%z*CartPrnFactor
        end do
      end do
```

```
!!!!!!!!!!!!!!!!!! end OF PRINTING
! JUST PRINTING (YOU CAN DELETE THE BELOW LINES)
!
write(*,*) "*********************************"
write(*,*) "   NEW FRAG CART (FROM DATABASE)  "
write(*,*) "*********************************"
do JAtom=1, NAtoms
  write(UNIout,*) "Atom# ", JAtom
  if (.not.DatabaseInform(Jatom)%Availability) then
      write(UNIout,*) "the carts for atom#", JAtom, &
                      "are not available in the database"
      cycle
  end if
  write(UNIout,'(A6,3A16)') "Atom","X","Y","Z"
  do ifound=1, DatabaseInform(Jatom)%NAtoms
      write(UNIout,'(A6,2X,3F16.8)') &
      & DBFragAtoms(Jatom,ifound)%element ,&
      & DBFragAtoms(Jatom,ifound)%x*CartPrnFactor, &
      & DBFragAtoms(Jatom,ifound)%y*CartPrnFactor, &
      & DBFragAtoms(Jatom,ifound)%z*CartPrnFactor
  end do
end do
!!!!!!!!!!!!!!!!!! end OF PRINTING
! JUST PRINTING (YOU CAN DELETE THE BELOW LINES)
!
write(*,*) "*********************************************"
write(*,*) " Where the fragment should be (database form) "
write(*,*) "*********************************************"
call GetDBFormFragments(CurrFragAtoms,DBCurrFragAtoms,CurrFragInfo)
do JAtom=1, NAtoms
  write(UNIout,*) "Atom# ", JAtom
  if (.not.DatabaseInform(Jatom)%Availability) then
      write(UNIout,*) "the carts for atom#", JAtom, &
                      "are not available in the database"
      cycle
  end if
  write(UNIout,'(A6,3A16)') "Atom","X","Y","Z"
  do ifound=1, DatabaseInform(Jatom)%NAtoms
      write(UNIout,'(A6,2X,3F16.8)') &
      & DBCurrFragAtoms(Jatom,ifound)%element, &
      & DBCurrFragAtoms(Jatom,ifound)%x*CartPrnFactor, &
      & DBCurrFragAtoms(Jatom,ifound)%y*CartPrnFactor, &
      & DBCurrFragAtoms(Jatom,ifound)%z*CartPrnFactor
  end do
end do
!!!!!!!!!!!!!!!!!! end OF PRINTING
! JUST PRINTING (YOU CAN DELETE THE BELOW LINES)
!
write(*,*) "*********************************************"
write(*,*) " The error: (FROM DATABASE)- (Original)       "
write(*,*) "*********************************************"
call GetDBFormFragments(CurrFragAtoms,DBCurrFragAtoms,CurrFragInfo)
do JAtom=1, NAtoms
  write(UNIout,*) "Atom# ", JAtom
  if (.not.DatabaseInform(Jatom)%Availability) then
      write(UNIout,*) "the carts for atom#", JAtom,&
                      "are not available in the database"
      cycle
  end if
  write(UNIout,'(A6,3A16)') "Atom","X","Y","Z"
  do ifound=1, DatabaseInform(Jatom)%NAtoms
      write(UNIout,'(A6,2X,3F16.8)') &
      & DBCurrFragAtoms(Jatom,ifound)%element, &
      & (DBFragAtoms(Jatom,ifound)%x-&
```

```
          &  DBCurrFragAtoms(Jatom,ifound)%x)*CartPrnFactor, &
          & (DBFragAtoms(Jatom,ifound)%y-&
          &  DBCurrFragAtoms(Jatom,ifound)%y)*CartPrnFactor, &
          & (DBFragAtoms(Jatom,ifound)%z-&
          &  DBCurrFragAtoms(Jatom,ifound)%z)*CartPrnFactor
    end do
  end do
  !!!!!!!!!!!!!!!!!! end OF PRINTING
  ! JUST PRINTING (YOU CAN DELETE THE BELOW LINES)
  !
  write(*,*) "*********************************"
  write(*,*) " CART WANTED (CART OF MOLECULE)   "
  write(*,*) "*********************************"
  do JAtom=1, NAtoms
    if (.not.DatabaseInform(Jatom)%Availability) cycle
    write(UNIout,*) "Atom# ", JAtom
    write(UNIout,'(A6,3A16)') "Atom","X","Y","Z"
    do ifound=1, DatabaseInform(Jatom)%NAtoms
        write(UNIout,'(A6,2X,3F16.8)') &
        & CurrFragAtoms(Jatom,ifound)%element, &
        & CurrFragAtoms(Jatom,ifound)%x*CartPrnFactor, &
        & CurrFragAtoms(Jatom,ifound)%y*CartPrnFactor, &
        & CurrFragAtoms(Jatom,ifound)%z*CartPrnFactor
    end do
  end do
  !!!!!!!!!!!!!!!!!! end OF PRINTING
  ! JUST PRINTING (YOU CAN DELETE THE BELOW LINES)
  !
  write(*,*) "*********************************"
  write(*,*) " The DatabaseInform Matrix for each atom  "
  write(*,*) "*********************************"
  do JAtom=1, NAtoms
   if (.not.DatabaseInform(Jatom)%Availability) cycle
   write(UNIout,*) "Atom# ", JAtom
   call PRT_matrix(FragAtomsMatrix(Jatom)%RMAT,3,3)
  end do
  !!!!!!!!!!!!!!!!!! end OF PRINTING
  ! JUST PRINTING (YOU CAN DELETE THE BELOW LINES)
  !
  write(*,*) "*********************************"
  write(*,*) "        CART AfteR ROT            "
  write(*,*) "*********************************"
  do JAtom=1, NAtoms
   if (.not.DatabaseInform(Jatom)%Availability) cycle
   write(UNIout,*) "Atom# ", JAtom
   write(UNIout,'(A6,3A16)') "Atom","X","Y","Z"
   do ifound=1, DatabaseInform(Jatom)%NAtoms
       write(UNIout,'(A6,2X,3F16.8)') &
       & RotDBFragAtoms(Jatom,ifound)%element, &
       & RotDBFragAtoms(Jatom,ifound)%x*CartPrnFactor,       &
       & RotDBFragAtoms(Jatom,ifound)%y*CartPrnFactor,       &
       & RotDBFragAtoms(Jatom,ifound)%z*CartPrnFactor
   end do
  end do
  !!!!!!!!!!!!!!!!!! end OF PRINTING
  ! JUST PRINTING (YOU CAN DELETE THE BELOW LINES)
  !
  write(*,*) "*********************************"
  write(*,*) " CART Diff error                  "
  write(*,*) "*********************************"
  do JAtom=1, NAtoms
   if (.not.DatabaseInform(Jatom)%Availability) cycle
   write(UNIout,*) "Atom# ", JAtom
   write(UNIout,'(A6,3A16)') "Atom","X","Y","Z"
```

```
      do ifound=1, DatabaseInform(Jatom)%NAtoms
           write(UNIout,'(A6,2X,3F16.8)') &
           & RotDBFragAtoms(Jatom,ifound)%element, &
           & (RotDBFragAtoms(Jatom,ifound)%x-&
           &  CurrFragAtoms(Jatom,ifound)%x)*CartPrnFactor, &
           & (RotDBFragAtoms(Jatom,ifound)%y-&
           &  CurrFragAtoms(Jatom,ifound)%y)*CartPrnFactor, &
           & (RotDBFragAtoms(Jatom,ifound)%z-&
           &  CurrFragAtoms(Jatom,ifound)%z)*CartPrnFactor
      end do
     end do
     !!!!!!!!!!!!!!!!! end OF PRINTING
     ! JUST PRINTING (YOU CAN DELETE THE BELOW LINES)
     !
     write(*,*) "*********************************"
     write(*,*) "       CART after TTCORR         "
     write(*,*) "*********************************"
     CorrFragAtom=RotDBFragAtoms
     do JAtom=1, NAtoms
      if (.not.DatabaseInform(Jatom)%Availability) cycle
      write(UNIout,*) "Atom# ", JAtom
      write(UNIout,'(A6,3A16)') "Atom","X","Y","Z"

      do ifound=1, DatabaseInform(Jatom)%NAtoms
         CorrFragAtom(Jatom,ifound)%element=&
                                     & RotDBFragAtoms(Jatom,ifound)%element
         CorrFragAtom(Jatom,ifound)%x=RotDBFragAtoms(Jatom,ifound)%x + &
                                     &  FragAtomsMatrix(Jatom)%TMATCORR(1)
         CorrFragAtom(Jatom,ifound)%y=RotDBFragAtoms(Jatom,ifound)%y + &
                                     &  FragAtomsMatrix(Jatom)%TMATCORR(2)
         CorrFragAtom(Jatom,ifound)%z=RotDBFragAtoms(Jatom,ifound)%z + &
                                     &  FragAtomsMatrix(Jatom)%TMATCORR(3)

         write(UNIout,'(A6,2X,3F16.8)') &
                          & CorrFragAtom(Jatom,ifound)%element,         &
                          & CorrFragAtom(Jatom,ifound)%x*CartPrnFactor, &
                          & CorrFragAtom(Jatom,ifound)%y*CartPrnFactor, &
                          & CorrFragAtom(Jatom,ifound)%z*CartPrnFactor
      end do
     end do
     !!!!!!!!!!!!!!!!! end OF PRINTING
     ! JUST PRINTING (YOU CAN DELETE THE BELOW LINES)
     !
     write(*,*) "*********************************"
     write(*,*) " CART Diff error after TTCORR     "
     write(*,*) "*********************************"
     do JAtom=1, NAtoms
      if (.not.DatabaseInform(Jatom)%Availability) cycle
      write(UNIout,*) "Atom# ", JAtom
      write(UNIout,'(A6,3A16)') "Atom","X","Y","Z"
      do ifound=1, DatabaseInform(Jatom)%NAtoms
           write(UNIout,'(A6,2X,3F16.8)') &
           &  RotDBFragAtoms(Jatom,ifound)%element,         &
           & (CorrFragAtom(Jatom,ifound)%x-&
           &  CurrFragAtoms(Jatom,ifound)%x)*CartPrnFactor, &
           & (CorrFragAtom(Jatom,ifound)%y-&
           &  CurrFragAtoms(Jatom,ifound)%y)*CartPrnFactor, &
           & (CorrFragAtom(Jatom,ifound)%z-&
           &  CurrFragAtoms(Jatom,ifound)%z)*CartPrnFactor
      end do
     end do
     !!!!!!!!!!!!!!!!! end OF PRINTING

!    deallocate (FRgrids)
```

```
      deallocate (FragAtomsMatrix)
  !   deallocate (CorrFragAtom)
      deallocate (CurrFragAtoms)
      deallocate (DBCurrFragAtoms)
      deallocate (DBUniqueFragAtoms)
      deallocate (DBFragAtoms)
      deallocate (RotDBFragAtoms)
      deallocate (CurrFragInfo)
      deallocate (MolIdxInfo)
      deallocate (DBFragProp)
      deallocate (RR)
      deallocate (TT)
      deallocate (TTCORR)
      deallocate (symbols)
      deallocate (UniSym)
      deallocate (UniSymMUNIdx)
      return
      end SUBROUTINE Build_Molecule_DB
```

```
      SUBROUTINE Build_Molecule_Direct(DatabaseInform,FragProp,FRgrids)
!*****e*****************************************************************************
!     Date last modified:                                                           *
!     Author: Ibrahim Awad                                                          *
!     Description:                                                                  *
!*****************************************************************************
! MODULEs:

      USE AIMDFT_type
      USE type_molecule

      implicit none

      ! Grid points by Iatom
      type(GridsDataBase), dimension(:,:), allocatable, intent(out) :: FRgrids
      type(IndexFile), dimension (:), allocatable, INTENT(out) :: DatabaseInform
      type(PropFile), dimension(:), allocatable, INTENT(out) :: FragProp

      integer :: File_unit
      logical :: Lerror

      ! < should be change
      CALL GET_unit ('Molecule_Properties.txt', File_unit, Lerror)
      open(UNIT=File_unit,file=trim('Molecule_Properties.txt'), &
           status='REPLACE',form='formatted')
      write(File_unit,fmt_prop_title) &
                                      & "#",            &
                                      & "Electrons",    &
                                      & "K",            &
                                      & "K_HF",         &
                                      & "T",            &
                                      & "Vne_Ana",      &
                                      & "Vne_Num",      &
                                      & "Vee(Ana/Num)", &
                                      & "J",            &
                                      & "J_HF",         &
                                      & "Jaa=Kaa",      &
                                      & "Vee_DFT_AtomA"
      close(unit=File_unit)

      CALL GenFilesDirect()
      CALL GetGridsInform(DatabaseInform,FRgrids)
      CALL GetDirectFragmentsProperties(FragProp)
```

```fortran
CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      SUBROUTINE GenFilesDirect()
!*************************************************************************
!      Date last modified:                                               *
!      Author: Ibrahim Awad                                              *
!      Description:                                                      *
!*************************************************************************
! MODULEs:

      USE module_grid_points
      USE NI_defaults
      USE type_Weights
      USE QM_defaults
      USE Sorted_AIMDFT
      USE symbol_AIMDFT

      implicit none
!
! Local scalars:

      type(FragsInfo), dimension(:), allocatable :: FragInfo
      type(FragAtomInfo), dimension(:,:), allocatable :: DBFragmentAtom
      type(GridsDataBase), dimension(:,:), allocatable :: FRgrids

      integer :: katom
      integer :: fileunit
      integer :: Iatom
      integer :: iSym

      character (60) :: cmdd
      character (60) :: radline
      character (60) :: outfile
      character (60) :: inputfile
      logical :: Lerror

      allocate (FragInfo(Natoms))
      allocate (DBFragmentAtom(Natoms,Natoms))

      CALL GetSortCartTerm(Level_Number,DBFragmentAtom,FragInfo)
      do Iatom=1, NAtoms
         if (CARTESIAN(Iatom)%Atomic_Number.le.0) cycle
         CALL INPUT_File_FRAG ('INPUT_ALL_','.dat',Iatom,inputfile)
         CALL INPUT_File_FRAG ('INPUT_ALL_','.out',Iatom,outfile)
         CALL GET_unit (inputfile, fileunit, Lerror)
         open(UNIT=fileunit, file=inputfile, status='REPLACE')
         write(fileunit,'(A)') &
         "MOLECULE MUltiplicity =  1 CHarge =  0  UNit = Angstrom"
         write(fileunit,'(A)')'TITle = "'//"TEST"//'"'
         write(fileunit,'(A)')'CArtesian'
         do katom=1,FragInfo(Iatom)%NAtoms
             write(fileunit,'(A5,3f26.20)') &
             &  DBFragmentAtom(Iatom,katom)%element, &
             &  DBFragmentAtom(Iatom,katom)%x*Bohr_to_Angstrom, &
             &  DBFragmentAtom(Iatom,katom)%y*Bohr_to_Angstrom, &
             &  DBFragmentAtom(Iatom,katom)%z*Bohr_to_Angstrom
         end do
         write(fileunit,'(A)')'end ! CArtesian'
         write(fileunit,'(A/)')'end ! MOLECULE'
         write(fileunit,'(A12,A14,A4/)')'basis name= ',&
                                     trim(Basis_set_name),' end'
         write(fileunit,'(A/)')'PLOT ATOM=false Molecule=false end'

         write(fileunit,'(A20,I4,A5,I3,A4/)')'AIMDFT IndexNumber=',Iatom,&
                                        ' Level=',Level_Number, 'end'
```

```
         write(fileunit,'(A20,A10,A4/)') 'PARtitioning Scheme=',&
                                         trim(DEN_Partitioning),' end'

         if (trim(RADIAL_grid).eq.'GILL') then
            write(radline,'(A20,I3,A10,I2,I3,I4,A10)') &
                         'RADial Gill RPoints=',&
                         NRPoints_Gill ,&
                         ' APoints=(',&
                         NApoints_Gill(1),&
                         NApoints_Gill(2),&
                         NApoints_Gill(3),' ) end end'
         else
            write(radline,'(A22,A7,A8)') 'NUmercial RADial GRID=',&
                                         trim(RADIAL_grid),' end end'
         endif

         write(fileunit,'(A/)') radline

         write(fileunit,'(A/)')'output object=AIMDFT:FRAGCART%GRIDS end'
         write(fileunit,'(A)')'stop'
         close(UNIT=fileunit)
         write(*,'(///A,A/,A/)') "*** RUNing the inputfile :: ", inputfile
         write(cmdd,'(a7,a20)')"mgauss<",inputfile
         CALL SYSTEM(cmdd)
       !  write(cmdd,'(a3,a20)')"rm ",inputfile
        ! CALL SYSTEM(cmdd)
      end do
      write(*,'(A///)')
      deallocate (FragInfo)
      deallocate (DBFragmentAtom)
      return
      end SUBROUTINE GenFilesDirect
      SUBROUTINE GetGridsInform(DatabaseInform,FRgrids)
!*****e*****************************************************************************
!     Date last modified:                                                           *
!     Author: Ibrahim Awad                                                          *
!     Description:                                                                  *
!*****************************************************************************
! MODULEs:

      USE AIMDFT_Files
      USE type_molecule
      USE AIMDFT_type

      implicit none

      integer :: Ipoint
      integer :: Iatom
      integer :: File_unit
      integer :: Reason

      character (60) :: PlotFilenames
      character (60) :: cmdd
      logical :: Lerror
      double precision :: DEN, BeckW, AngW, Qval, Vpot
      double precision, dimension (1:3) :: point

      type(GridsDataBase), dimension(:,:), allocatable, intent(out) :: FRgrids
      type(IndexFile), dimension (:), allocatable, INTENT(out) :: DatabaseInform

      allocate (FRgrids(Natoms,MaxGridPnt))
      allocate(DatabaseInform(NAtoms))

      DatabaseInform%Availability=.false.
```

```
      do Iatom=1, NAtoms
         if (CARTESIAN(Iatom)%Atomic_Number.le.0) cycle
         call INPUT_File_FRAG ('GRID_DIRECT_','.dat',Iatom, PlotFilenames)
         call GET_unit (PlotFilenames, File_unit, Lerror)
         open(UNIT=File_unit,file=PlotFilenames, status='OLD',form='formatted')
         Ipoint=0
         read(File_unit,*)
         do
            read(File_unit,Pfmt_rho_w_v,IOSTAT=Reason) point(1),&
                 point(2),point(3), DEN, BeckW, AngW, Qval, Vpot
            if (Reason.lt.0) exit
            Ipoint=Ipoint+1
            FRgrids(Iatom,Ipoint)%x=Point(1)
            FRgrids(Iatom,Ipoint)%y=Point(2)
            FRgrids(Iatom,Ipoint)%z=Point(3)
            FRgrids(Iatom,Ipoint)%D=DEN
            FRgrids(Iatom,Ipoint)%BW=BeckW
            FRgrids(Iatom,Ipoint)%W=AngW
            FRgrids(Iatom,Ipoint)%Q=Qval
            FRgrids(Iatom,Ipoint)%VpotA=Vpot
         end do
         close(UNIT=File_unit)
      !   write(cmdd,'(a3,a36)')"rm ",PlotFilenames
      !   CALL SYSTEM(cmdd)

         DatabaseInform(Iatom)%Availability=.true.
         DatabaseInform(Iatom)%IndexNum=Iatom
         DatabaseInform(Iatom)%GridsNum=Ipoint

      end do
      end SUBROUTINE GetGridsInform
      SUBROUTINE GetDirectFragmentsProperties(FragProp)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description: Return the cartesian coordinates of the fragments         *
!                  from the database cartesian file.                         *
!*****************************************************************************
!

      implicit none

      type(PropFile), dimension(:), allocatable, intent(out) :: FragProp

      integer :: newindex
      integer :: File_unit, Reason
      logical :: Lerror
      double precision :: EleNum, EV_Anal, EV_Num
      double precision :: EX, EXJ, ET, EC, Kaa, Jaa, Vee_A, ECK, EC_AnaNum
      character (60) :: cmdd

      allocate (FragProp(NAtoms))

      ! loop over the wanted index
         FragProp%Availability=.false.
         call GET_unit ('Molecule_Properties.txt', File_unit, Lerror)
         open(UNIT=File_unit,file='Molecule_Properties.txt', &
             status='old',form='formatted')
         read(File_unit,*)
         do ! start, reading the file
             read(File_unit,fmt_prop,IOSTAT=Reason) &
                & newindex,                 &
                & EleNum,                   &
```

```
                & EX,                          &
                & EXJ,                         &
                & ET,                          &
                & EV_Anal,                     &
                & EV_Num,                      &
                & EC_AnaNum,                   &
                & EC,                          &
                & ECK,                         &
                & Jaa,                         &
                & vee_A

            if (Reason.lt.0) exit
                FragProp(newindex)%IndexNum=newindex
                FragProp(newindex)%ElectonNum=EleNum
                FragProp(newindex)%K=EX
                FragProp(newindex)%KHF=EXJ
                FragProp(newindex)%T=ET
                FragProp(newindex)%Vne_A=EV_Anal
                FragProp(newindex)%Vne_N=EV_Num
                FragProp(newindex)%Vee_AN=EC_AnaNum
                FragProp(newindex)%J=EC
                FragProp(newindex)%JHF=ECK
                FragProp(newindex)%Jaa=Jaa
                FragProp(newindex)%VeeA=Vee_A
                FragProp(newindex)%Availability=.true.
        end do ! reading the file

        close(unit=File_unit)
        write(cmdd,'(a3,a24)')"rm ","Molecule_Properties.txt"
        CALL SYSTEM(cmdd)

    end SUBROUTINE GetDirectFragmentsProperties
    end SUBROUTINE Build_Molecule_Direct
```

```
    SUBROUTINE Store_result_direct()
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
! MODULEs

    USE type_density
    USE QM_defaults
    USE QM_objects
    USE N_integration
    USE NI_defaults
    USE type_molecule
    USE AIMDFT_type
    USE AIMDFT_files
    USE type_plotting
    USE GetMolecularProperties

    implicit none

    character (60):: PlotFilenames

    double precision, dimension(:), allocatable :: Vpotl
    double precision, dimension(:), allocatable :: V12dr2
    double precision :: TraceAB
    double precision :: Xpt,Ypt,Zpt

    logical :: Lerror
    integer :: IApoint
```

```
      integer :: Znum, Iatom
      integer :: kfound
      integer :: File_unit
      integer ::  newindex

      CALL GET_object ('GRID', 'RADIAL', RADIAL_grid)
!      CALL GET_object ('QM', 'ENERGY_COMPONENTS', Wavefunction)

      newindex=IndexNumber

      Iatom=1 ! Just add the intrest atom within the fragment.
      NAIMprint=1
      AIMprint(1)=1

!      write(*,*)
      CALL GET_object ('QM', 'ENERGY_VEE', 'NUMERICAL')
!      CALL GET_object ('QM', 'ENERGY_EXCHANGE', 'NUMERICAL')

!      CALL GET_object ('QM', 'ENERGY_COULOMB', 'MO')
      write(*,*)
      CALL GET_object ('QM', 'ENERGY_COULOMB', 'NUMERICAL')
      write(*,*)
      CALL GET_object ('QM', 'ENERGY_KINETIC', 'NUMERICAL')
      write(*,*)
      CALL GET_object ('QM', 'ENERGY_VNE', 'NUMERICAL')
      write(*,*)

      Znum=CARTESIAN(Iatom)%Atomic_Number
      kfound=GRID_loc(Znum)
      NApts_atom=NApoints_atom(kfound)

      allocate(grid_points(1:NApts_atom))
     ! Save grid points
      do IApoint=1,NApts_atom
         grid_points(IApoint)%X=Egridpts(Iatom,IApoint)%X+CARTESIAN(Iatom)%X
         grid_points(IApoint)%Y=Egridpts(Iatom,IApoint)%Y+CARTESIAN(Iatom)%Y
         grid_points(IApoint)%Z=Egridpts(Iatom,IApoint)%Z+CARTESIAN(Iatom)%Z
         grid_points(IApoint)%w=Egridpts(Iatom,IApoint)%w
      end do ! IApoint

      ! compute the weight for all grid points

      allocate(Bweights(NApts_atom))
      CALL GET_weights (grid_points, NApts_atom, Iatom, Bweights)

      ! compute the charge for all grid points
      allocate (rho_Atom(NApts_atom))
      CALL GET_density (grid_points, NApts_atom, rho_Atom)

      ! compute the V12dr2 for all grid points!
      ! < IT IS Not nessessary to calculate it.
      allocate(Vpotl(1:NApts_atom))
      allocate(V12dr2(1:MATlen))
      do IApoint=1,NApts_atom
         Xpt=grid_points(IApoint)%X
         Ypt=grid_points(IApoint)%Y
         Zpt=grid_points(IApoint)%Z
         CALL I1E_V12dr2 (V12dr2, MATlen,Xpt,Ypt,Zpt)
         Vpotl(IApoint)=-TraceAB (PM0, V12dr2, NBasis)
      end do ! IApoint

      CALL Store_grid_points_direct
      CALL add_to_prop_file('Molecule_Properties.txt')
```

```
      deallocate (rho_Atom)
      deallocate (V12dr2)
      deallocate (Vpotl)
      deallocate (grid_points)

CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      SUBROUTINE Store_grid_points_direct()

      implicit none

      CALL INPUT_File_FRAG ('GRID_DIRECT_','.dat',IndexNumber, PlotFilenames)
      CALL GET_unit (PlotFilenames, File_unit, Lerror)
      open(UNIT=File_unit,file=PlotFilenames, status='REPLACE',form='formatted')
      write(File_unit,'(A20,7A24)') "X","Y","Z","Rho","BW","AW","Q=w.W.Rho","Vpol"
      do IApoint=1,NApts_atom
!    if (rho_Atom(IApoint)*grid_points(IApoint)%w*Bweights(IApoint).lt.1.0D-15)&
!    cycle
         write(File_unit,Pfmt_rho_w_v) grid_points(IApoint)%X,  &
                                  &   grid_points(IApoint)%Y,  &
                                  &   grid_points(IApoint)%Z,  &
                                  &   rho_Atom(IApoint),       &
                                  &   Bweights(IApoint),       &
                                  &   grid_points(IApoint)%w,  &
                                  &   rho_Atom(IApoint)*&
                                  grid_points(IApoint)%w*Bweights(IApoint),  &
                                  &   Vpotl(IApoint)
      end do ! Ipoint
      close(UNIT=File_unit)
      end SUBROUTINE Store_grid_points_direct
      SUBROUTINE add_to_prop_file(FilePath)
!*****e*****************************************************************************
!     Date last modified:                                                          *
!     Author: Ibrahim Awad                                                         *
!     Description:                                                                 *
!*******************************************************************************
! MODULEs


      implicit none

      character (20) :: Radnote
      logical :: Lerror
      integer :: File_unit
      double precision :: EV_Anal, EV_Num, EX, ET, EC, EXJ, ECK, EC_AnaNum
      double precision :: EleNum, vee_A
      character (*) :: FilePath


      write(*,'(a,I5)') "  Atomic properties for atom #", Iatom
      write(*,'(a)')    "  ++++++++++++++++++++++++++++++++++++++"
      CALL CalAtomicEle(grid_points, NApts_atom, rho_Atom, Bweights, EleNum) !
      write(*,'(a,F15.10)')"Number of Electrons, N           = ", EleNum

      EX = K_Atomic(Iatom)
      write(*,'(a,F15.10)')"Pure Exchange, K ( 2K_ab)        = ", EX

      EXJ = KHF_Atomic(Iatom)
      write(*,'(a,F15.10)')"HF Exchange, KHF ( 2K_ab+ Kaa)   = ", EXJ

      ET = Atomic_Kinetic(Iatom)
      write(*,'(a,F15.10)')"Kinetic energy Numerical, T      = ", ET

      ! Calculate the atomic potential energy for intrest fragment atom.
      CALL get_Vne_Atom_Analytical(EV_Anal)
```

```
      write(*,'(a,F15.10)')"Potential Energy Analytical, Vne = ", EV_Anal

      EV_Num = Atomic_Vne(Iatom)
      write(*,'(a,F15.10)')"Potential Energy Numerical, Vne  = ", EV_Num

      EC_AnaNum = Atomic_Coulomb(Iatom)
      write(*,'(a,F15.10)')"Coulomb Energy Anal/Num, Vee     = ", EC_AnaNum

      EC = J_Atomic(Iatom)
      write(*,'(a,F15.10)')"Pure Coulomb, J ( 4J_ab+ Jaa)    = ", EC

      ECK = JHF_Atomic(Iatom)
      write(*,'(a,F15.10)')"HF Coulomb, JHF ( 4J_ab+ 2Jaa)   = ", ECK

      write(*,'(a,F15.10)')"Jaa = Kaa                        = ", ECK-EC

      CALL CalcVeeSelfA(grid_points, NApts_atom, rho_Atom, Bweights, vee_A)
      write(*,'(a,F15.10)')"Coulomb Numerically Over A       = ", vee_A
      write(*,*)

      CALL GET_unit (trim(FilePath), File_unit, Lerror)
      open(UNIT=File_unit,file=trim(FilePath), &
           status='old',form='formatted',position="append")
      write(File_unit,fmt_prop) newindex,   &
                & EleNum,                   &
                & EX,                       &
                & EXJ,                      &
                & ET,                       &
                & EV_Anal,                  &
                & EV_Num,                   &
                & EC_AnaNum,                &
                & EC,                       &
                & ECK,                      &
                & ECK-EC,                   &
                & vee_A

      close(unit=File_unit)

      end SUBROUTINE add_to_prop_file

      end SUBROUTINE Store_result_direct
```

```
      MODULE symbol_AIMDFT
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:                                                             *
!*****************************************************************************
!
      USE Sorted_AIMDFT

      implicit none

CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

      SUBROUTINE GetUniSym(level, UniSym, UniSymMUNIdx, UniSymMum)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description: Return a unique symbols array and its count                 *
!                  within the symbols list                                     *
!*****************************************************************************
!
      implicit none
```

```
      integer, intent(in) ::  level
      character (SymMax), dimension (:), allocatable, intent(out) :: UniSym
      integer, dimension (:), allocatable, intent(out) :: UniSymMUNIdx
      character (SymMax), dimension (:), allocatable :: symbols

      integer, intent(out) :: UniSymMum
      integer :: Jatom

      allocate (UniSym(NAtoms))
      allocate (symbols(Natoms))
      allocate (UniSymMUNIdx(Natoms))

      CALL GetFragSymbols(level,symbols)

      UniSym = ""
      UniSymMum = 0   ! count the unique symbols within the molecule
      do Jatom=1, NAtoms
         if (Cartesian(Jatom)%Atomic_Number.eq.0) cycle ! exclude dummy atoms
         if (ANY(UniSym.eq.symbols(Jatom))) cycle ! Founded before
         UniSymMum=UniSymMum+1
         UniSym(UniSymMum)=symbols(Jatom)
         UniSymMUNIdx(UniSymMum)=Jatom  ! Just the first match
      end do
      deallocate(symbols)
      return
      end SUBROUTINE GetUniSym
      SUBROUTINE GetFragSymbols(level,symbols)
!*************************************************************************
!     Date last modified:                                                *
!     Author: Ibrahim Awad                                               *
!     Description: Generate unique symbols for the molecule fragements   *
!                  [e.g. 7( 6( 6( 8 8 6) 1( 6) 1( 6)) 1() 1()) ]         *
!*************************************************************************
!

      implicit none

      integer, intent(in) :: level
      character (SymMax), dimension (:), intent(out), allocatable :: symbols

      type (FragAtomInfo), dimension(:,:), allocatable :: FragAtoms
      type (FragsInfo), dimension(:), allocatable :: FragInfo ! (main atom)

      character (SymMax), dimension(:), allocatable :: tempOrdValue
      integer, dimension(10) :: tempOrd
      integer :: NtempOrd, TempCopy, curr,next
      character (SymMax) :: TempCopyChar
      character (10) :: atomicN
      character (len=1), parameter :: SepSymbol=","

      integer :: ilevel
      integer :: ifound
      integer :: Matom
      integer :: Jatom
      logical :: swap

      allocate (symbols(Natoms))
      allocate (FragAtoms(Natoms,Natoms))
      allocate (FragInfo(Natoms))
      allocate (tempOrdValue(Natoms))


      CALL GetSortCart(level,FragAtoms,FragInfo)
```

```
! JUST ::::defined symbol for level 1
do Jatom=1, NAtoms
if (Cartesian(JAtom)%Atomic_Number.eq.0) cycle ! exclude dummay atoms
   do Matom=1, Natoms  ! loop for subatoms
      ! Defind the MainAtom
      write(atomicN,'(I0)') Cartesian(Jatom)%Atomic_Number
      FragInfo(Matom)%symbol(Jatom,1)=trim(atomicN)//"("
      NtempOrd=0
      tempOrd=0
      do ifound=1, FragInfo(Jatom)%NAtoms  ! loop over subatoms
         if (FragAtoms(Jatom,ifound)%level.ne.1) cycle !Just level 1
         ! exclude subatom=Matom
         if (FragAtoms(Jatom,ifound)%MUNIdx.eq.Matom) cycle
         ! exclude Mainatom=Mainatom
         if (FragAtoms(Jatom,ifound)%MUNIdx.eq.Jatom) cycle
         NtempOrd=NtempOrd+1
         tempOrd(NtempOrd)=FragAtoms(Jatom,ifound)%Atomic_Number
      end do
      swap=.true.
      do while(swap)
         swap=.false.
         do curr=1,NtempOrd-1
            next=curr+1
            if (tempOrd(curr).lt.tempOrd(next)) then
               TempCopy=tempOrd(curr)
               tempOrd(curr)=tempOrd(next)
               tempOrd(next)=TempCopy
               swap=.true.
            endif !(valueB.lt.valueA)
         end do !curr
      end do !while(swap)
      do ifound=1, NtempOrd
         if (ifound.ne.NtempOrd) then
         ! the new subatom
         write(atomicN,'(I0,A1)') tempOrd(ifound),Sepsymbol
         else
         write(atomicN,'(I0)') tempOrd(ifound)   ! the new subatom
         end if
         ! add the new subatom
         FragInfo(Matom)%symbol(Jatom,1)=&
         trim(FragInfo(Matom)%symbol(Jatom,1))//trim(atomicN)//""
      end do
      ! just level 1, the neighber
      FragInfo(Matom)%symbol(Jatom,1)=&
      trim(FragInfo(Matom)%symbol(Jatom,1))//")"
      if (Matom.eq.Jatom) symbols(Jatom)=&
      trim(FragInfo(Matom)%symbol(Jatom,1))
   end do  !Matom
end do ! Jatom

! Now from the symbols of the 1st level,
! we will build the symbols for ALL levels
do ilevel=2, level
   do Jatom=1, NAtoms
      do Matom=1, Natoms ! loop over dumatoms
         ! Defind the Mainatom
         write(atomicN,'(I0)') Cartesian(Jatom)%Atomic_Number
         FragInfo(Matom)%symbol(Jatom,ilevel)=trim(atomicN)//"("
         NtempOrd=0
         tempOrd=0
         tempOrdValue=""
         do ifound=1, FragInfo(Jatom)%NAtoms  ! loop over subatoms
           if (FragAtoms(Jatom,ifound)%level.ne.1) cycle !Just level 1
```

```
                  ! exclude subatom=Matom
                  if (FragAtoms(Jatom,ifound)%MUNIdx.eq.Matom) cycle
                  ! exclude MainAtom=Mainatom
                  if (FragAtoms(Jatom,ifound)%MUNIdx.eq.Jatom) cycle
                  NtempOrd=NtempOrd+1
                  tempOrd(NtempOrd)= FragAtoms(Jatom,ifound)%Atomic_Number
                  tempOrdValue(NtempOrd)=&
                  FragInfo(Matom)%symbol(FragAtoms(Jatom,ifound)%MUNIdx,ilevel-1)
               end do
               swap=.true.
               do while(swap)
                  swap=.false.
                  do curr=1,NtempOrd-1
                     next=curr+1
                     if (tempOrd(curr).lt.tempOrd(next)) then
                        TempCopy=tempOrd(curr)
                        TempCopyChar=tempOrdValue(curr)
                        tempOrd(curr)=tempOrd(next)
                        tempOrdValue(curr)=tempOrdValue(next)
                        tempOrd(next)=TempCopy
                        tempOrdValue(next)=trim(TempCopyChar)
                        swap=.true.
                     end if !(valueB.lt.valueA)
                  end do !curr
               end do !while(swap)
               do ifound=1, NtempOrd
                  FragInfo(Matom)%symbol(Jatom,ilevel)=&
                  trim(FragInfo(Matom)%symbol(Jatom,ilevel))// & ! like x=x+??
                  & tempOrdValue(ifound)//"" ! adding the new subatom
               end do
               !just level 1, the neighber
               FragInfo(Matom)%symbol(Jatom,ilevel)=&
               trim(FragInfo(Matom)%symbol(Jatom,ilevel))//")"
               if (Matom.eq.Jatom) symbols(Jatom) =&
                trim(FragInfo(Matom)%symbol(Jatom,ilevel))
            end do ! Matom
         end do ! Jatom
      end do ! level
      deallocate(FragAtoms, FragInfo, tempOrdValue)
      return
      end SUBROUTINE GetFragSymbols
      end MODULE symbol_AIMDFT
```

```
      MODULE AddTerminal
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
!
      USE mod_fragment

      implicit none

CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

      SUBROUTINE AddTerminalAtoms(adlevel,FragAtoms,FragInfo)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
!
      USE mod_math
```

```
implicit none

type (FragAtomInfo),dimension(:,:),intent(inout),allocatable :: FragAtoms
type (FragsInfo),dimension(:),intent(inout),allocatable :: FragInfo
integer, intent(in) :: adlevel

type (FragAtomInfo), dimension(:), allocatable :: trematoms
type (FragAtomInfo), dimension(:), allocatable :: DBterm
type (FragAtomInfo), dimension(:), allocatable :: Newtrematoms
type (FragAtomInfo), dimension(:), allocatable :: NewDBterm

double precision, dimension (:,:), allocatable :: RR,RRI
double precision, dimension (:) , allocatable :: TT
double precision, dimension (1:3) :: NewPoint, point

integer :: loop3, oldloop3, iloop, NewLoop
integer :: Jatom, iMUNatom, ifound, ifound2, ifound3
double precision :: bondtype
logical :: termenalTF

allocate (trematoms(Natoms))
allocate (DBterm(Natoms))
allocate (Newtrematoms(Natoms))
allocate (NewDBterm(Natoms))
allocate (RR(3,3))
allocate (RRI(3,3))
allocate (TT(3))

CALL GET_object ('MOL', 'GRAPH', 'CONNECT')
CALL GET_object ('QM', 'BOND_ORDER', 'MEYER')

do JAtom=1, NAtoms
   ! loop over number of adjacent atoms for Jatom
   do ifound=1, FragInfo(Jatom)%NAtoms
      iMUNatom=FragAtoms(Jatom,ifound)%MUNIdx
      if(ChangeTerminalAtoms) then
         termenalTF=.true.
      else
         ! In the case if the atom is termenal then leave it same
         termenalTF=(FragInfo(iMUNatom)%NumNeighbourAtoms.ne.1)
      end if
      if ((FragAtoms(Jatom,ifound)%level.ne.(adlevel))) cycle
      if (.not.termenalTF) cycle

      ! found one atom (2 >>> to be in z-axes)
      trematoms(2)=FragAtoms(Jatom,ifound)
      do ifound2=1, FragInfo(Jatom)%NAtoms
         ! Look for atom with (level-1) of the intrest atom
         if (FragAtoms(Jatom,ifound2)%level.ne.(adlevel-1)) cycle
         ! find the parent atom
         if (Connect(FragAtoms(Jatom,ifound2)%MUNIdx,&
                        trematoms(2)%MUNIdx).ne.1) cycle
         trematoms(1)=FragAtoms(Jatom,ifound2) ! found the origin atom
      end do
      loop3=2  ! number of termenal atoms
      do ifound3=1, FragInfo(Jatom)%NAtoms
         if (Connect(trematoms(1)%MUNIdx,&
           FragAtoms(Jatom,ifound3)%MUNIdx).ne.1) cycle
         if (FragAtoms(Jatom,ifound3)%MUNIdx.eq.trematoms(2)%MUNIdx) cycle
         loop3=loop3+1
         trematoms(loop3)=FragAtoms(Jatom,ifound3) ! the termenal atoms
      end do
      ! find the bond type of the intrest atom with the origin
```

```
            bondtype= bond_order(trematoms(2)%MUNIdx,trematoms(1)%MUNIdx)
   !         write(*,*) trematoms(2)%MUNIdx,trematoms(1)%MUNIdx,bondtype
            CALL GetDBCart(trematoms,DBterm,loop3,RR,TT)
            CALL INV33(RR,RRI)

            oldloop3=loop3
            if(bondtype.le.0.5)then ! Partial single bond (TS?)
               CALL getterm1 (DBterm,loop3,NewDBterm,NewLoop)
            else if(bondtype.le.1.15)then ! Single bond
               CALL getterm2 (DBterm,loop3,NewDBterm,NewLoop)
            else if(bondtype.le.1.6)then ! Aromatic double bond
               CALL getterm3 (DBterm,loop3,NewDBterm,NewLoop)
            else if(bondtype.le.2.2)then ! Double bond
               CALL getterm4 (DBterm,loop3,NewDBterm,NewLoop)
            else if(bondtype.le.2.7)then ! Aromatic triple bond
               CALL getterm5 (DBterm,loop3,NewDBterm,NewLoop)
            else if(bondtype.le.3.2)then ! Triple bond
               CALL getterm6 (DBterm,loop3,NewDBterm,NewLoop)
            end if

            do iloop=loop3+1, NewLoop
              point=(/NewDBterm(iloop)%x,NewDBterm(iloop)%y,NewDBterm(iloop)%z/)
              NewPoint=Matmul(RRI,(point-TT))
              Newtermatoms(iloop)%x=NewPoint(1)
              Newtermatoms(iloop)%y=NewPoint(2)
              Newtermatoms(iloop)%z=NewPoint(3)
            end do

            FragAtoms(Jatom,ifound)%Atomic_Number=NewDBterm(2)%Atomic_Number
            FragAtoms(Jatom,ifound)%element=NewDBterm(2)%element

            do iloop=loop3+1, NewLoop
               FragInfo(Jatom)%NAtoms=FragInfo(Jatom)%NAtoms+1
               FragAtoms(Jatom,FragInfo(Jatom)%NAtoms)=Newtermatoms(iloop)
               FragAtoms(Jatom,FragInfo(Jatom)%NAtoms)%Atomic_Number=&
                                 NewDBterm(iloop)%Atomic_Number
               FragAtoms(Jatom,FragInfo(Jatom)%NAtoms)%element=&
                                 NewDBterm(iloop)%element
               FragAtoms(Jatom,FragInfo(Jatom)%NAtoms)%type=999
               FragAtoms(Jatom,FragInfo(Jatom)%NAtoms)%ctype=&
                                 NewDBterm(iloop)%ctype
               FragAtoms(Jatom,FragInfo(Jatom)%NAtoms)%factor=-1
               FragAtoms(Jatom,FragInfo(Jatom)%NAtoms)%level=adlevel+1
               FragAtoms(Jatom,FragInfo(Jatom)%NAtoms)%MUNIdx=-1
            end do
         end do ! ifound=1, SATOMS(Jatom)
      end do ! JAtom=1, NAtoms

      CALL CalcFragConnect(FragAtoms,FragInfo)
      CALL UpdateFragTypes(FragAtoms,FragInfo)
      deallocate(trematoms,DBterm,Newtermatoms,NewDBterm,RR,TT,RRI)
      return
      end SUBROUTINE AddTerminalAtoms
      SUBROUTINE getterm1 (DBterm,loop3,NewDBterm,NewLoop)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
!
      implicit none

      type (FragAtomInfo), dimension(:), allocatable, intent(in) :: DBterm
      integer, intent(in) :: loop3  ! +1 for new atoms
```

```
      type (FragAtomInfo), dimension(:), allocatable, intent(out) :: NewDBterm
      integer, intent(out) :: NewLoop  ! +1 for new atoms
      type (FragAtomInfo), dimension(:), allocatable  :: newatoms

      double precision :: CH_bond, Anglea
      integer :: NNewatoms, iloop

      allocate (NewDBterm(Natoms))

      NewDBterm=DBterm
      NNewatoms=0

    !  select case(NewDBterm(2)%Atomic_Number)
    !    case(1)  ! H atom
         !!!!!!!!!!!!!!!!!
    !    case(2)  ! He atom
         !!!!!!!!!!!!!!!!!
    !    case(3)
         !!!!!!!!!!!!!!!!!
    !    case(6)  ! C atom
         !!!!!!!!!!!!!!!!!
    !    case(8)  ! O atom
         !!!!!!!!!!!!!!!!!
    !    CASE DEFAULT
    !      NewDBterm(2)%Atomic_Number=TerminalAtom
    !      NewDBterm(2)%element=element_symbols(TerminalAtom)
    !  end select

      NewLoop=loop3
      do iloop=1, NNewatoms
          NewLoop=NewLoop+1
          NewDBterm(NewLoop)%x=newatoms(iloop)%x+DBterm(2)%x
          NewDBterm(NewLoop)%y=newatoms(iloop)%y+DBterm(2)%y
          NewDBterm(NewLoop)%z=newatoms(iloop)%z+DBterm(2)%z
          NewDBterm(NewLoop)%Atomic_Number=newatoms(iloop)%Atomic_Number
          NewDBterm(NewLoop)%element=newatoms(iloop)%element
          NewDBterm(NewLoop)%ctype=newatoms(iloop)%ctype
      end do

      if (allocated(newatoms)) deallocate(newatoms) ! if claUSE is necessary
      end SUBROUTINE getterm1

      SUBROUTINE getterm2 (DBterm,loop3,NewDBterm,NewLoop)
!*********************************************************************
!     Date last modified:                                            *
!     Author: Ibrahim Awad                                           *
!     Description:                                                   *
!*********************************************************************
!
      implicit none

      type (FragAtomInfo), dimension(:), allocatable, intent(in) :: DBterm
      integer, intent(in) :: loop3  ! +1 for new atoms
      type (FragAtomInfo), dimension(:), allocatable, intent(out) :: NewDBterm
      integer, intent(out) :: NewLoop  ! +1 for new atoms
      type (FragAtomInfo), dimension(:), allocatable  :: newatoms

      double precision :: HO_bond, Anglea, CH_bond, NH_bond
      integer :: NNewatoms, iloop

      allocate (NewDBterm(Natoms))

      NewDBterm=DBterm
      NNewatoms=0
```

```
    select case(NewDBterm(2)%Atomic_Number)
      case(6)  ! C atom
       NNewatoms=3
       allocate (newatoms(NNewatoms))
        ! NEW ATOMS FOR THE TERMENAL
        ! Single bond (-CH3)
        CH_bond=2.04781965 ! Bohr
        !Anglea=31.65 *0.0174533 ! Need Degree to RAD
        !H1
        newatoms(1)%x=0.0d0
        newatoms(1)%y=-1.93061734
        newatoms(1)%z=0.68284856
        newatoms(1)%Atomic_Number=TerminalAtom
        newatoms(1)%element=element_symbols(TerminalAtom)
        newatoms(1)%ctype="-CH3"
        !H2
        newatoms(2)%x=1.67199927
        newatoms(2)%y=0.96543892
        newatoms(2)%z=0.68284856
        newatoms(2)%Atomic_Number=TerminalAtom
        newatoms(2)%element=element_symbols(TerminalAtom)
        newatoms(2)%ctype="-CH3"
        !H3
        newatoms(3)%x=-1.67199914
        newatoms(3)%y= 0.96557695
        newatoms(3)%z= 0.68284856
        newatoms(3)%Atomic_Number=TerminalAtom
        newatoms(3)%element=element_symbols(TerminalAtom)
        newatoms(3)%ctype="-CH3"

      case(7)  ! N atom
       NNewatoms=2
       allocate (newatoms(NNewatoms))
        ! NEW ATOMS FOR THE TERMENAL
        ! Single bond (-NH2)
        NH_bond=1.89445030 ! Bohr
        !Anglea=31.65 *0.0174533 ! Need Degree to RAD
        !H1
        newatoms(1)%x=0.0d0
        newatoms(1)%y=-1.80992957
        newatoms(1)%z=0.55955062
        newatoms(1)%Atomic_Number=TerminalAtom
        newatoms(1)%element=element_symbols(TerminalAtom)
        newatoms(1)%ctype="-NH2"
        !H2
        newatoms(2)%x=-1.64331915
        newatoms(2)%y=0.75857027
        newatoms(2)%z=0.55955062
        newatoms(2)%Atomic_Number=TerminalAtom
        newatoms(2)%element=element_symbols(TerminalAtom)
        newatoms(2)%ctype="-NH2"


      case(8)  ! O atom

        ! NEW ATOMS FOR THE TERMENAL
        ! Hydrogen bond (H-O)
        NNewatoms=1
        allocate (newatoms(NNewatoms))

        HO_bond=1.86970623 ! Bohr
        Anglea=14.5d0 *0.0174533 ! Need Degree to RAD
        !H1 for O atom
```

```
          newatoms(1)%x=0.0d0
          newatoms(1)%y=-HO_bond*Cos(Anglea)
          newatoms(1)%z=HO_bond*Sin(Anglea)
          newatoms(1)%Atomic_Number=TerminalAtom
          newatoms(1)%element=element_symbols(TerminalAtom)
          newatoms(1)%ctype="-OH"
    !     case(8)  ! O atom
          !!!!!!!!!!!!!!!!!
        CASE DEFAULT
          NewDBterm(2)%Atomic_Number=TerminalAtom
          NewDBterm(2)%element=element_symbols(TerminalAtom)
      end select

      NewLoop=loop3
      do iloop=1, NNewatoms
          NewLoop=NewLoop+1
          NewDBterm(NewLoop)%x=newatoms(iloop)%x+DBterm(2)%x
          NewDBterm(NewLoop)%y=newatoms(iloop)%y+DBterm(2)%y
          NewDBterm(NewLoop)%z=newatoms(iloop)%z+DBterm(2)%z
          NewDBterm(NewLoop)%Atomic_Number=newatoms(iloop)%Atomic_Number
          NewDBterm(NewLoop)%element=newatoms(iloop)%element
          NewDBterm(NewLoop)%ctype=newatoms(iloop)%ctype
      end do

      if (allocated(newatoms)) deallocate(newatoms) ! if claUSE is necessary


      end SUBROUTINE getterm2

      SUBROUTINE getterm3 (DBterm,loop3,NewDBterm,NewLoop)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
!
      implicit none

      type (FragAtomInfo), dimension(:), allocatable, intent(in) :: DBterm
      integer, intent(in) :: loop3  ! +1 for new atoms
      type (FragAtomInfo), dimension(:), allocatable, intent(out) :: NewDBterm
      integer, intent(out) :: NewLoop  ! +1 for new atoms
      type (FragAtomInfo), dimension(:), allocatable  :: newatoms

      double precision :: CH_bond, Anglea
      integer :: NNewatoms, iloop

      allocate (NewDBterm(Natoms))

      NewDBterm=DBterm
      NNewatoms=0

    !  select case(NewDBterm(2)%Atomic_Number)
    !    case(1)  ! H atom
         !!!!!!!!!!!!!!!!!
    !    case(2)  ! He atom
         !!!!!!!!!!!!!!!!!
    !    case(3)
         !!!!!!!!!!!!!!!!!
    !    case(6)  ! C atom
         !!!!!!!!!!!!!!!!!
    !    case(8)  ! O atom
         !!!!!!!!!!!!!!!!!
    !    CASE DEFAULT
```

```
    !      NewDBterm(2)%Atomic_Number=TerminalAtom
    !      NewDBterm(2)%element=element_symbols(TerminalAtom)
    !   end select

      NewLoop=loop3
      do iloop=1, NNewatoms
          NewLoop=NewLoop+1
          NewDBterm(NewLoop)%x=newatoms(iloop)%x+DBterm(2)%x
          NewDBterm(NewLoop)%y=newatoms(iloop)%y+DBterm(2)%y
          NewDBterm(NewLoop)%z=newatoms(iloop)%z+DBterm(2)%z
          NewDBterm(NewLoop)%Atomic_Number=newatoms(iloop)%Atomic_Number
          NewDBterm(NewLoop)%element=newatoms(iloop)%element
          NewDBterm(NewLoop)%ctype=newatoms(iloop)%ctype
      end do

      if (allocated(newatoms)) deallocate(newatoms) ! if claUSE is necessary
      end SUBROUTINE getterm3
      SUBROUTINE getterm4 (DBterm,loop3,NewDBterm,NewLoop)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
!
      implicit none

      type (FragAtomInfo), dimension(:), allocatable, intent(in) :: DBterm
      integer, intent(in) :: loop3  ! +1 for new atoms
      type (FragAtomInfo), dimension(:), allocatable, intent(out) :: NewDBterm
      integer, intent(out) :: NewLoop  ! +1 for new atoms
      type (FragAtomInfo), dimension(:), allocatable  :: newatoms

      double precision :: CH_bond, Anglea
      integer :: NNewatoms, iloop

      allocate (NewDBterm(Natoms))

      NewDBterm=DBterm
      NNewatoms=0

      select case(NewDBterm(2)%Atomic_Number)
    !    case(1)  ! H atom
         !!!!!!!!!!!!!!!!!
    !    case(2)  ! He atom
         !!!!!!!!!!!!!!!!!
    !    case(3)
         !!!!!!!!!!!!!!!!!
         case(6)  ! C atom
         NNewatoms=2
         allocate (newatoms(NNewatoms))
          ! NEW ATOMS FOR THE TERMENAL
          ! Double bond (=CH2)
          CH_bond=2.03456411 ! Bohr
          Anglea=31.65 *0.0174533 ! Need Degree to RAD
          !H1
          newatoms(1)%x=0.0d0
          newatoms(1)%y=CH_bond*Cos(Anglea)
          newatoms(1)%z=CH_bond*Sin(Anglea)
          newatoms(1)%Atomic_Number=TerminalAtom
          newatoms(1)%element=element_symbols(TerminalAtom)
          newatoms(1)%ctype="=CH2"
          !H2
          newatoms(2)%x=0.0d0
          newatoms(2)%y=-CH_bond*Cos(Anglea)
```

```
          newatoms(2)%z= CH_bond*Sin(Anglea)
          newatoms(2)%Atomic_Number=TerminalAtom
          newatoms(2)%element=element_symbols(TerminalAtom)
          newatoms(2)%ctype="=CH2"


    !     case(8)  ! O atom
          !!!!!!!!!!!!!!!!!
    !     CASE DEFAULT
    !       NewDBterm(2)%Atomic_Number=TerminalAtom
    !       NewDBterm(2)%element=element_symbols(TerminalAtom)
        end select


       NewLoop=loop3
       do iloop=1, NNewatoms
           NewLoop=NewLoop+1
           NewDBterm(NewLoop)%x=newatoms(iloop)%x+DBterm(2)%x
           NewDBterm(NewLoop)%y=newatoms(iloop)%y+DBterm(2)%y
           NewDBterm(NewLoop)%z=newatoms(iloop)%z+DBterm(2)%z
           NewDBterm(NewLoop)%Atomic_Number=newatoms(iloop)%Atomic_Number
           NewDBterm(NewLoop)%element=newatoms(iloop)%element
           NewDBterm(NewLoop)%ctype=newatoms(iloop)%ctype
       end do

       if (allocated(newatoms)) deallocate(newatoms) ! if claUSE is necessary
       end SUBROUTINE getterm4

       SUBROUTINE getterm5 (DBterm,loop3,NewDBterm,NewLoop)
!*****************************************************************************
!      Date last modified:                                                     *
!      Author: Ibrahim Awad                                                    *
!      Description:                                                            *
!*****************************************************************************
!
       implicit none

       type (FragAtomInfo), dimension(:), allocatable, intent(in) :: DBterm
       integer, intent(in) :: loop3  ! +1 for new atoms
       type (FragAtomInfo), dimension(:), allocatable, intent(out) :: NewDBterm
       integer, intent(out) :: NewLoop  ! +1 for new atoms
       type (FragAtomInfo), dimension(:), allocatable  :: newatoms

       double precision :: CH_bond, Anglea
       integer :: NNewatoms, iloop

       allocate (NewDBterm(Natoms))

       NewDBterm=DBterm
       NNewatoms=0

    !  select case(NewDBterm(2)%Atomic_Number)
    !    case(1)  ! H atom
         !!!!!!!!!!!!!!!!!
    !    case(2)  ! He atom
         !!!!!!!!!!!!!!!!!
    !    case(3)
         !!!!!!!!!!!!!!!!!
    !    case(6)  ! C atom
         !!!!!!!!!!!!!!!!!
    !    case(8)  ! O atom
         !!!!!!!!!!!!!!!!!
    !    CASE DEFAULT
    !      NewDBterm(2)%Atomic_Number=TerminalAtom
```

```
    !       NewDBterm(2)%element=element_symbols(TerminalAtom)
    !   end select


      NewLoop=loop3
      do iloop=1, NNewatoms
          NewLoop=NewLoop+1
          NewDBterm(NewLoop)%x=newatoms(iloop)%x+DBterm(2)%x
          NewDBterm(NewLoop)%y=newatoms(iloop)%y+DBterm(2)%y
          NewDBterm(NewLoop)%z=newatoms(iloop)%z+DBterm(2)%z
          NewDBterm(NewLoop)%Atomic_Number=newatoms(iloop)%Atomic_Number
          NewDBterm(NewLoop)%element=newatoms(iloop)%element
          NewDBterm(NewLoop)%ctype=newatoms(iloop)%ctype
      end do

      if (allocated(newatoms)) deallocate(newatoms) ! if claUSE is necessary
      end SUBROUTINE getterm5

      SUBROUTINE getterm6 (DBterm,loop3,NewDBterm,NewLoop)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:                                                             *
!*****************************************************************************
!
      implicit none

      type (FragAtomInfo), dimension(:), allocatable, intent(in) :: DBterm
      integer, intent(in) :: loop3  ! +1 for new atoms
      type (FragAtomInfo), dimension(:), allocatable, intent(out) :: NewDBterm
      integer, intent(out) :: NewLoop  ! +1 for new atoms
      type (FragAtomInfo), dimension(:), allocatable  :: newatoms

      double precision :: CH_bond, Anglea
      integer :: NNewatoms, iloop

      allocate (NewDBterm(Natoms))

      NewDBterm=DBterm
      NNewatoms=0

    !  select case(NewDBterm(2)%Atomic_Number)
    !    case(1)  ! H atom
         !!!!!!!!!!!!!!!!!
    !    case(2)  ! He atom
         !!!!!!!!!!!!!!!!!
    !    case(3)
         !!!!!!!!!!!!!!!!!
    !    case(6)  ! C atom
         !!!!!!!!!!!!!!!!!
    !    case(8)  ! O atom
         !!!!!!!!!!!!!!!!!
    !    CASE DEFAULT
    !      NewDBterm(2)%Atomic_Number=TerminalAtom
    !      NewDBterm(2)%element=element_symbols(TerminalAtom)
    !  end select

      NewLoop=loop3
      do iloop=1, NNewatoms
          NewLoop=NewLoop+1
          NewDBterm(NewLoop)%x=newatoms(iloop)%x+DBterm(2)%x
          NewDBterm(NewLoop)%y=newatoms(iloop)%y+DBterm(2)%y
          NewDBterm(NewLoop)%z=newatoms(iloop)%z+DBterm(2)%z
          NewDBterm(NewLoop)%Atomic_Number=newatoms(iloop)%Atomic_Number
```

```
          NewDBterm(NewLoop)%element=newatoms(iloop)%element
          NewDBterm(NewLoop)%ctype=newatoms(iloop)%ctype
      end do

      if (allocated(newatoms)) deallocate(newatoms) ! if claUSE is necessary
      end SUBROUTINE getterm6
      end MODULE AddTerminal
```

```
      MODULE mod_build
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:                                                             *
!*****************************************************************************
!
      USE mod_rotation
      USE AIMDFT_Files
      USE program_constants

      implicit none

CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

      SUBROUTINE UpdateFragTypes (FragAtoms,FragInfo)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:  Update the types of fragments                              *
!*****************************************************************************
!
      USE GRAPH_objects

      implicit none
!
      type(FragAtomInfo),dimension(:,:), intent(inout), allocatable :: FragAtoms
      type(FragsInfo),dimension(:), intent(inout), allocatable :: FragInfo
      integer :: Ilist, Nlist, Iterm, MinZ, ZI
      integer :: Ivertex, Jvertex
      integer :: Itype, Jatom
      character(len=16) :: CAtype
      character(len=16) :: Ctype
!
      integer, dimension(:), allocatable :: Zlist    ! List of atomic numbers

      do Jatom=1, Natoms
          Nvertices=FragInfo(Jatom)%NAtoms
          allocate (Zlist(Nvertices))
          ! Now define atom types for each vertex
          do Ivertex=1, Nvertices
              ZI=FragAtoms(Jatom,Ivertex)%Atomic_Number
              FragAtoms(Jatom,Ivertex)%type=0
              if(ZI.le.0)cycle ! dummy atom
              Zlist(1:Nvertices)=999
              Nlist=0
              do Jvertex=1,Nvertices ! Find all atoms bonded to Ivertex
                  if((FragInfo(Jatom)%Connect(Ivertex,Jvertex)).ne.0)then
                      Nlist=Nlist+1
                      Zlist(Nlist)=FragAtoms(Jatom,Jvertex)%Atomic_Number
                  end if
              end do ! Jvertex!
              Iterm=0
              Itype=0
!             Define type for Ivertex, sum in order of increasing Z
              do while (Iterm.lt.Nlist) !
```

```
              MinZ=MINval(Zlist)
              do Ilist=1,Nlist
                  ZI=Zlist(Ilist)
                  if(ZI.le.MinZ)then
                      Iterm=Iterm+1
                      Itype=Itype+(ZI-2)*Iterm
                      Zlist(Ilist)=999
                  end if
              end do ! Ilist
          end do ! while
          FragAtoms(Jatom,Ivertex)%type=Itype
      end do ! Ivertex
      deallocate (Zlist)
      end do ! Jatom
      return
      end SUBROUTINE UpdateFragTypes
      SUBROUTINE CalcFragConnect(FragAtoms,FragInfo)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description: Calculate the connectivity matrix for the fragments       *
!*****************************************************************************
!

      USE type_elements

      implicit none

      type(FragAtomInfo),dimension(:,:), intent(inout), allocatable :: FragAtoms
      type(FragsInfo),dimension(:), intent(inout), allocatable :: FragInfo
      integer :: Jatom, IatomN

      ! In future you can USE differents type of radii
      call Frag_CONN (BS_RADII, NELEMENTS, BS_scalef, FragAtoms, FragInfo)
      return
      end SUBROUTINE CalcFragConnect
      SUBROUTINE Frag_CONN (RADII,   & ! Bragg-Slater atomic radii
                 ELMLEN,  & ! Length of data array RADII
                 FUDGE,   & ! Fudge factor for Connectivity determination
                 FragAtoms, &
                 FragInfo)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description: Calculate the connectivity matrix for the fragments       *
!                  using Bragg-Slater atomic radii                           *
!*****************************************************************************
!

      implicit none

      type (FragAtomInfo), dimension(:,:), intent(inout), allocatable :: FragAtoms
      type (FragsInfo), dimension(:), intent(inout), allocatable :: FragInfo

      integer, intent(in) :: ELMLEN
      double precision, intent(in) :: RADII(ELMLEN)
      double precision, intent(in) :: FUDGE

      integer :: Iatom, Jatom, Matom, IIAN, JIAN, IatomN
      double precision :: DisSep, FUDGE_save

      do Jatom=1, Natoms
         if (CARTESIAN(Jatom)%Atomic_Number.le.0) cycle
         IatomN=FragInfo(Jatom)%NAtoms
```

```
         if(IatomN.lt.2) cycle
         FragInfo(Jatom)%Connect=0
         do Iatom=1,IatomN
            IIAN=FragAtoms(Jatom,Iatom)%Atomic_Number
            if(IIAN.le.0)cycle ! Don't check if not 'real'.
            do Matom=Iatom+1,IatomN
               JIAN=FragAtoms(Jatom,Matom)%Atomic_Number
               if (JIAN.le.0) cycle ! Don't check if not 'real'.
               FUDGE_save=FUDGE
               ! Special case for HF molecule
               if((IIAN.eq.1.and.JIAN.eq.9).or.(IIAN.eq.9.and.JIAN.eq.1))then
                  FUDGE_save=1.273
               end if
               DisSep=dsqrt((FragAtoms(Jatom,Iatom)%x-&
                           FragAtoms(Jatom,Matom)%x)**2 + &
                         & (FragAtoms(Jatom,Iatom)%y-&
                           FragAtoms(Jatom,Matom)%y)**2 + &
                         & (FragAtoms(Jatom,Iatom)%z-&
                           FragAtoms(Jatom,Matom)%z)**2)
               if(bohr_to_angstrom*DisSep-FUDGE_save*&
                  (RADII(IIAN)+RADII(JIAN)).le.ZERO)then
                  FragInfo(Jatom)%Connect(Iatom,Matom)=1
               end if !
               FragInfo(Jatom)%Connect(Matom,Iatom)=&
               FragInfo(Jatom)%Connect(Iatom,Matom) ! Connectivity is symmetric.
            end do ! ! Matom
         end do ! ! Iatom
      end do ! ! Jatom
      return
      end SUBROUTINE Frag_CONN
      end MODULE mod_build
```

```
      MODULE AIMDFT_Files
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:                                                             *
!*****************************************************************************

      USE AIMDFT_type

      implicit none

CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

      SUBROUTINE INPUT_File_GRID (IndexNum, GridFilename)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description: Create a filename for Ploting data and get a unit.          *
!*****************************************************************************
! MODULEs:

      implicit none
!
      integer, intent(in) :: IndexNum
      character(len=:), allocatable, intent(out) :: GridFilename
      character(64) :: FileID, FilePOST

      character(15) :: Plot, CPlot
      integer :: lenstr
!
! Begin:
      FileID="GRID_"
```

```
      FilePOST=".dat"
      write(Plot,'(i4.4)') IndexNum
      call RemoveBlanks (Plot, CPlot, lenstr)
      GridFilename=trim(GridPath)//trim(FileID)//CPlot(1:lenstr)//trim(FilePOST)

      return
      end SUBROUTINE INPUT_File_GRID
      SUBROUTINE INPUT_File_FRAG (FileID, FilePOST, IndexNum, PlotFilename)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description: Create a filename for Ploting data and get a unit.          *
!*****************************************************************************
! MODULEs:

      implicit none
!
      integer, intent(in) :: IndexNum
      character(60), intent(out) :: PlotFilename
      character*(*), intent(in) :: FileID, FilePOST

      character(15) :: Plot, CPlot
      integer :: lenstr
!
! Begin:
      write(Plot,'(i4.4)') IndexNum
      call RemoveBlanks (Plot, CPlot, lenstr)
      PlotFilename=trim(FileID)//CPlot(1:lenstr)//trim(FilePOST)

      return
      end SUBROUTINE INPUT_File_FRAG
      end MODULE AIMDFT_Files
```

```
      MODULE mod_fragment
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:                                                             *
!*****************************************************************************
!
      USE program_files
      USE rotation_Cart
      USE type_elements !needed for element_symbols()
      USE GRAPH_objects ! needed for Connect()
      USE QM_objects ! needed for bond_order()
      USE mod_build
      USE GetDataBaseInfo
      USE GetMolecularProperties ! to use GetLevelConnect function

      implicit none

CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

      SUBROUTINE GenerateFragments(level,FragAtoms,FragInfo)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description: Partitioning the molecule to its fragments                  *
!*****************************************************************************
!
      implicit none

      integer, intent(in) :: level
      type (FragAtomInfo), dimension(:,:), intent(out), allocatable :: FragAtoms
```

```
type (FragsInfo), dimension(:), intent(out), allocatable :: FragInfo

integer :: Iatom
integer :: Jatom
integer :: ifound
integer :: ilevel
type (TwoDiArrInt), dimension(:), allocatable :: LevelConnect !
logical, dimension(:), allocatable :: found

allocate (FragAtoms(Natoms,Natoms))
allocate (FragInfo(Natoms))
allocate (LevelConnect(level))
allocate (found(Natoms))

CALL GetLevelConnect(level,LevelConnect)

do Jatom=1, NAtoms
    if (Cartesian(JAtom)%Atomic_Number.eq.0) cycle
    FragInfo(Jatom)%NAtoms=1
    FragInfo(Jatom)%NumNeighbourAtoms=0

    ! Defined the main atom (Jatom) element_symbols
    FragAtoms(Jatom,1)%level=0
    FragAtoms(Jatom,1)%MUNIdx=Jatom  ! the index of atom in MUNgauss
    FragAtoms(Jatom,1)%Atomic_Number=Cartesian(Jatom)%Atomic_Number
    FragAtoms(Jatom,1)%x=Cartesian(Jatom)%x
    FragAtoms(Jatom,1)%y=Cartesian(Jatom)%y
    FragAtoms(Jatom,1)%z=Cartesian(Jatom)%z
    FragAtoms(Jatom,1)%element=&
        element_symbols(Cartesian(Jatom)%Atomic_Number)
    FragAtoms(Jatom,1)%ctype=""
    FragAtoms(Jatom,1)%sorted= .true.

    ifound=1  ! index for each sub-atom found for Jatom

    do Iatom=1, NAtoms
        if (Cartesian(IAtom)%Atomic_Number.eq.0) cycle
        found=.false.
        do ilevel=1, level
            if ((LevelConnect(ilevel)%ijarray(Iatom,Jatom).ne.1).or.&
            (found(Iatom)).or.Iatom.eq.Jatom) cycle
            ifound=ifound+1
            found(Iatom)=.true.
            FragInfo(Jatom)%NAtoms=FragInfo(Jatom)%NAtoms+1
            if (ilevel.eq.1) FragInfo(Jatom)%NumNeighbourAtoms=&
            FragInfo(Jatom)%NumNeighbourAtoms+1
            FragAtoms(Jatom,ifound)%level=ilevel
            FragAtoms(Jatom,ifound)%MUNIdx=Iatom !  MUNgauss index
            FragAtoms(Jatom,ifound)%Atomic_Number=&
            Cartesian(IAtom)%Atomic_Number
            FragAtoms(Jatom,ifound)%x=Cartesian(IAtom)%x
            FragAtoms(Jatom,ifound)%y=Cartesian(IAtom)%y
            FragAtoms(Jatom,ifound)%z=Cartesian(IAtom)%z
            FragAtoms(Jatom,ifound)%element=&
            element_symbols(Cartesian(Iatom)%Atomic_Number)
            FragAtoms(Jatom,ifound)%ctype=""
            FragAtoms(Jatom,ifound)%sorted= .false.
        end do ! ilevel
    end do  ! Iatom
end do ! Jatom

CALL CalcFragConnect(FragAtoms,FragInfo)
CALL UpdateFragTypes(FragAtoms,FragInfo)
```

```
      deallocate(LevelConnect, found)
      return
      end SUBROUTINE GenerateFragments

      SUBROUTINE PRNTCART(FragAtoms,FragInfo)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:  Print the cartesian coordinate for the molecule fragments  *
!*****************************************************************************
!
      implicit none

      type (FragAtomInfo), dimension(:,:), intent(in), allocatable  :: FragAtoms
      type (FragsInfo), dimension(:), intent(in), allocatable :: FragInfo

      integer :: ifound, JAtom

      do JAtom=1, NAtoms
         if (Cartesian(JAtom)%Atomic_Number.eq.0) cycle
         write(UNIout,'(/A9,I3,A8,I3)') "Fragment#", JAtom, &
         ", Atoms#", FragInfo(Jatom)%NAtoms
         write(UNIout,'(A5,3A13,1X,4A9)') "!Atom ","X","Y","Z",&
         "Ctype","Ftype","Level","Sorted"
         do ifound=1, FragInfo(Jatom)%NAtoms
            write(UNIout,'(A5,3X,3F13.8,3X,A7,i5,i8,l8)') &
                      & FragAtoms(Jatom,ifound)%element,               &
                      & FragAtoms(Jatom,ifound)%x*CartPrnFactor,       &
                      & FragAtoms(Jatom,ifound)%y*CartPrnFactor,       &
                      & FragAtoms(Jatom,ifound)%z*CartPrnFactor,       &
                      & FragAtoms(Jatom,ifound)%Ctype,                 &
                      & FragAtoms(Jatom,ifound)%type,                  &
                    ! & FragAtoms(Jatom,ifound)%factor,                &
                      & FragAtoms(Jatom,ifound)%level,                 &
                      & FragAtoms(Jatom,ifound)%sorted
          end do
      end do
      end SUBROUTINE PRNTCART
      end MODULE mod_fragment
```

```
      MODULE GetMolecularProperties
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:                                                             *
!*****************************************************************************

      USE module_grid_points
      USE type_molecule
      USE program_constants
      USE AIMDFT_type
      USE N_integration

      implicit none

CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

      SUBROUTINE FindAdj(level,Jatom,atomAdj)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description: Partitioning the molecule to its fragments                  *
!*****************************************************************************
!
```

```
      implicit none

      integer, intent(in) :: level, Jatom
      integer, dimension(:), allocatable, intent(out) :: atomAdj
      integer :: Iatom, ilevel
      type (TwoDiArrInt), dimension(:), allocatable :: LevelConnect !
      logical, dimension(:), allocatable :: found

      allocate (LevelConnect(level))
      allocate (found(Natoms))
      allocate (atomAdj(Natoms))

      CALL GetLevelConnect(level,LevelConnect)
      atomAdj=0
          do Iatom=1, NAtoms
              if (Cartesian(IAtom)%Atomic_Number.eq.0) cycle
              found=.false.
              do ilevel=1, level
                  if ((LevelConnect(ilevel)%ijarray(Iatom,Jatom).ne.1).or.&
                  (found(Iatom)).or.Iatom.eq.Jatom) cycle
                  found(Iatom)=.true.
                  atomAdj(Iatom)=ilevel
              end do ! ilevel
          end do  ! Iatom

      deallocate(LevelConnect, found)
      return
      end SUBROUTINE FindAdj

      SUBROUTINE GetLevelConnect(level,LevCon)
!*********************************************************************
!     Date last modified:                                            *
!     Author: Ibrahim Awad                                           *
!     Description: Calculate the connectivity matrix for the molecule at       *
!                  different levels.                                 *
!*********************************************************************
!
      USE GRAPH_objects
      USE matrix_print

      implicit none

      integer, intent(in) :: level
      type (TwoDiArrInt), dimension(:), intent(out), allocatable :: LevCon
      integer :: ilevel

      allocate(LevCon(level))
      do ilevel=1, level
         allocate(LevCon(ilevel)%ijarray(NAtoms,NAtoms))
      end do

      CALL GET_object ('MOL', 'GRAPH', 'CONNECT')
      LevCon(1)%ijarray=CONNECT ! adjacency matrices with level order
     ! call PRT_matrix(CONNECT,NAtoms,NAtoms)
      do ilevel=2, level
      LevCon(ilevel)%ijarray=MATMUL(LevCon(ilevel-1)%ijarray,CONNECT)
      end do
      end SUBROUTINE GetLevelConnect
      SUBROUTINE CalAtomicEle(grid_points,NApts_atom,rho_Atom,Bweights,EleNum)
!*********************************************************************
!     Date last modified:                                            *
!     Author: Ibrahim Awad                                           *
!     Description:   Calculate Number of electrons  per Iatom        *
!*********************************************************************
```

```
! MODULEs:
      implicit none

      integer, INTENT(IN) :: NApts_atom
      type(type_grid_points),dimension(NApts_atom), INTENT(IN) :: grid_points
      double precision, dimension(NApts_atom), INTENT(IN) :: rho_Atom, Bweights
      double precision :: point_charge
      integer :: IApoint
      double precision, INTENT(OUT) :: EleNum

      EleNum=0.0d0
      do IApoint=1, NApts_atom
         point_charge=rho_Atom(IApoint)*grid_points(IApoint)%w*Bweights(IApoint)
         EleNum = EleNum+point_charge
      end do
      EleNum = FourPi*EleNum
      return
      end SUBROUTINE CalAtomicEle
      SUBROUTINE CalcVeeSelfA(grid_points,NApts_atom,rho_Atom,Bweights,Vee_A)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:   Calculate Number of electrons  per Iatom                  *
!*****************************************************************************
! MODULEs:
      implicit none

      integer, INTENT(IN) :: NApts_atom
      type(type_grid_points),dimension(NApts_atom), INTENT(IN) :: grid_points
      double precision, dimension(NApts_atom), INTENT(IN) :: rho_Atom, Bweights
      double precision :: pointI, pointJ, R_IJ
      integer :: IApoint, JApoint
      double precision, INTENT(OUT) :: Vee_A

      Vee_A=0.0d0
      do IApoint=1, NApts_atom
         do JApoint=IApoint+1, NApts_atom
            pointI=rho_Atom(IApoint)*grid_points(IApoint)%w*Bweights(IApoint)
            pointJ=rho_Atom(JApoint)*grid_points(JApoint)%w*Bweights(JApoint)
            R_IJ=DSQRT((grid_points(JApoint)%x-grid_points(IApoint)%x)**2+ &
                       (grid_points(JApoint)%y-grid_points(IApoint)%y)**2+ &
                       (grid_points(JApoint)%z-grid_points(IApoint)%z)**2)
            Vee_A = Vee_A + pointI*pointJ/R_IJ
         end do
      end do
      Vee_A = FourPi*FourPi*Vee_A
      return
      end SUBROUTINE CalcVeeSelfA
      subroutine get_Vne_Atom_Analytical(EV)
!*****************************************************************************
!     Date last modified: May 21, 2012                    Version 1.0          *
!     Author: R.A. Poirier & Jessica Besaw                                     *
!     Description: Calculate the atomic potential integrals (VAint) and        *
!                  the total potential integrals (VINT) in order to            *
!                  determine the atomic potential energies (EV)                *
!                  and the total potential energy (EVtotal)              *
!*****************************************************************************
! Modules:
      USE program_constants
      USE type_molecule
      USE type_basis_set
      USE QM_defaults
      USE QM_objects
      USE INT_objects
```

```
      USE matrix_print

      implicit none

! Local scalars:
      integer :: Ishell,Jshell,Ifrst,Jfrst,Ilast,Jlast
      integer :: Istart,Jstart,Iend,Jend
      integer :: Iatmshl,Jatmshl
      integer :: LAMAX,LBMAX
      integer :: Iatom
      integer :: Irange,Jrange
      integer :: Igauss,Jgauss
      integer :: Iaos,JaoS
      integer :: IGBGN,JGBGN,IGend,JGend
      integer :: LPMAX,I,IA,INTC,IX,IY,IZ,IZERO,J,JX,JY,JZ,LIM1DS,NZERO
      integer :: LENTQ
      double precision ABX,ABY,ABZ,ARABSQ,ARG,AS,ASXA,ASYA,ASZA,&
                       BS,COEF,CUT1,EP,EPI,EPIO2,PCX,PCY,PCZ,PEXP, &
                       PX,PY,PZ,RPCSQ,TWOASQ,TWOP,TWOPI,TWOPT2,XAP,XBP, &
                       YAP,YBP,ZAP,ZBP,ZCONST,ZT,ZTEMP
      double precision :: XA,XB,XC,YA,YB,YC,ZA,ZB,ZC,RABSQ,TRACLO

! Local arrays:
      double precision, dimension(:), allocatable :: VAint
      double precision, INTENT(OUT) :: EV
      integer INDIX(20),INDIY(20),INDIZ(20),INDJX(20),INDJY(20),INDJZ(20)
      double precision :: TP(7),WP(7)
      double precision A(45),CA(20),CB(20),CCX(192),CCY(192),CCZ(192), &
                       EEPB(100),EEPV(100), STWOCX(9), &
                       TWOCX(9),TWOCY(9),TWOCZ(9),XIP(80),YIP(80),ZIP(80)

      parameter (TWOPI=TWO*PI_VAL,CUT1=-75.0D0)

! Data related to Rys Polynomials
      DATA INDJX/1,2,1,1,3,1,1,2,2,1,4,1,1,2,3,3,2,1,1,2/, &
           INDJY/1,1,2,1,1,3,1,2,1,2,1,4,1,3,2,1,1,2,3,2/, &
           INDJZ/1,1,1,2,1,1,3,1,2,2,1,1,4,1,1,2,3,3,2,2/

! Obtaining the density matrix using HF theory
      call GET_object ('QM', 'DENSITY_1MATRIX', Wavefunction)

! Object:
      if(.not.allocated(VINT))then
        allocate (VAint((Basis%Nbasis*(Basis%Nbasis+1))/2))
        allocate (VINT((Basis%Nbasis*(Basis%Nbasis+1))/2))
      else
        if(Basis%Nbasis.ne.size(VINT,1))then
          deallocate (VINT)
          allocate (VAint((Basis%Nbasis*(Basis%Nbasis+1))/2))
          allocate (VINT((Basis%Nbasis*(Basis%Nbasis+1))/2))
        end if
      end if


! Initialize Sum to Zero
      VINT=0.

      call RYSSET

! Fill INDIX. (Related to Rys polynomials)
      do I=1,20
        INDIX(I)=4*(INDJX(I)-1)
        INDIY(I)=4*(INDJY(I)-1)
        INDIZ(I)=4*(INDJZ(I)-1)
```

```
      end do ! I

! Loop over atoms.
       Iatom=1

        IA=CARTESIAN(Iatom)%Atomic_number

! Perform calculatation on atoms, and not dummy atoms.
      IF(IA.GT.0)then
        XC=CARTESIAN(Iatom)%X
        YC=CARTESIAN(Iatom)%Y
        ZC=CARTESIAN(Iatom)%Z

! Loop over shells.
! Loop over Ishell.
      do Ishell=1,Basis%Nshells
        LAMAX=Basis%shell(Ishell)%Xtype+1
        Istart=Basis%shell(Ishell)%Xstart
        Iend=Basis%shell(Ishell)%Xend
        Irange=Iend-Istart+1
        IGBGN=Basis%shell(Ishell)%EXPBGN
        IGend=Basis%shell(Ishell)%EXPend
        Ifrst=Basis%shell(Ishell)%frstSHL

! Loop over Jshell.
      do Jshell=1,Ishell
        LBMAX=Basis%shell(Jshell)%Xtype+1
        Jstart=Basis%shell(Jshell)%Xstart
        Jend=Basis%shell(Jshell)%Xend
        Jrange=Jend-Jstart+1
        JGBGN=Basis%shell(Jshell)%EXPBGN
        JGend=Basis%shell(Jshell)%EXPend
        Jfrst=Basis%shell(Jshell)%frstSHL

        Ilast= Basis%shell(Ishell)%lastSHL
        Jlast= Basis%shell(Jshell)%lastSHL

! Loop over Iatmshl
      do Iatmshl=Ifrst,Ilast
        XA=CARTESIAN(Basis%atmshl(Iatmshl)%ATMLST)%X
        YA=CARTESIAN(Basis%atmshl(Iatmshl)%ATMLST)%Y
        ZA=CARTESIAN(Basis%atmshl(Iatmshl)%ATMLST)%Z
        Iaos=Basis%atmshl(Iatmshl)%frstAO-1

! Loop over Jatmshl
      IF(Ishell.EQ.Jshell)Jlast=Iatmshl

      do Jatmshl=Jfrst,Jlast
        XB=CARTESIAN(Basis%atmshl(Jatmshl)%ATMLST)%X
        YB=CARTESIAN(Basis%atmshl(Jatmshl)%ATMLST)%Y
        ZB=CARTESIAN(Basis%atmshl(Jatmshl)%ATMLST)%Z
        Jaos=Basis%atmshl(Jatmshl)%frstAO-1

        LPMAX=LAMAX+LBMAX-1
        LIM1DS=(LPMAX+3)/2
        LENTQ=Irange*Jrange
        NZERO=(LAMAX+LBMAX-2)/2+1
        ABX=XB-XA
        ABY=YB-YA
        ABZ=ZB-ZA
        RABSQ=ABX*ABX+ABY*ABY+ABZ*ABZ

! Zero accumulation area
      do I=1,LENTQ
```

```
        EEPV(I)=ZERO
      end do ! I

! Loop over primitive Gaussians.

      do Igauss=IGBGN,IGend
        AS=Basis%gaussian(Igauss)%exp
        TWOASQ=TWO*AS*AS
        ASXA=AS*XA
        ASYA=AS*YA
        ASZA=AS*ZA
        ARABSQ=AS*RABSQ

      call FILLCC (LAMAX, Basis%gaussian(Igauss)%CONTRC, CA)

      do Jgauss=JGBGN,JGend
        BS=Basis%gaussian(Jgauss)%exp
        call FILLCC (LBMAX, Basis%gaussian(Jgauss)%CONTRC, CB)
        EP=AS+BS
        EPI=ONE/EP
        EPIO2=PT5*EPI
        TWOP=EP+EP
        ARG=-BS*ARABSQ*EPI
        PEXP=ZERO
        IF(ARG.GT.CUT1)PEXP=DEXP(ARG)
        ZTEMP=TWOPI*EPI*PEXP
        PX=(ASXA+BS*XB)*EPI
        PY=(ASYA+BS*YB)*EPI
        PZ=(ASZA+BS*ZB)*EPI
        XAP=PX-XA
        XBP=PX-XB
        YAP=PY-YA
        YBP=PY-YB
        ZAP=PZ-ZA
        ZBP=PZ-ZB

      call GETCC1 (CCX, XAP, XBP, LAMAX+2, LBMAX+2)
      call GETCC1 (CCY, YAP, YBP, LAMAX+2, LBMAX+2)
      call GETCC1 (CCZ, ZAP, ZBP, LAMAX+2, LBMAX+2)

! Zero accumulation area.
! (I want to just say EEBV = 0., or is loop more cost effective?)
      do I=1,LENTQ
        EEPB(I)=ZERO
      end do ! I

! Calculate information about atom

      ZT=ZTEMP*DBLE(IA)
      PCX=XC-PX
      PCY=YC-PY
      PCZ=ZC-PZ
      RPCSQ=PCX*PCX+PCY*PCY+PCZ*PCZ
      ARG=EP*RPCSQ

      call RPOLX (NZERO, ARG, TP, WP)
      call GETA1 (A, EPIO2, 0, LPMAX)

! Loop over zeroes of Rys polynomial.
      do IZERO=1,NZERO
        TWOPT2=TWOP*TP(IZERO)
        ZCONST=ZT*WP(IZERO)
        call GET2C (TWOCX, PCX, ONE, A, TWOPT2, 0, LPMAX)
        call GET2C (TWOCY, PCY, ONE, A, TWOPT2, 0, LPMAX)
```

```
        call GET2C (TWOCZ, PCZ, ZCONST, A, TWOPT2, 0, LPMAX)
        call GET3C (XIP, 0, TWOCX, CCX, LAMAX, LBMAX)
        call GET3C (YIP, 0, TWOCY, CCY, LAMAX, LBMAX)
        call GET3C (ZIP, 0, TWOCZ, CCZ, LAMAX, LBMAX)

! Loop over atomic orbitals for potential energy integrals,

      INTC=0
      do I=Istart,Iend
        IX=INDIX(I)
        IY=INDIY(I)
        IZ=INDIZ(I)

      do J=Jstart,Jend
        JX=INDJX(J)
        JY=INDJY(J)
        JZ=INDJZ(J)
        INTC=INTC+1
        EEPB(INTC)=EEPB(INTC)+XIP(IX+JX)*YIP(IY+JY)*ZIP(IZ+JZ)
      end do ! J
      end do ! I

! End of AO loop.

      end do ! IZERO

! End of loop over Rys zeroes.

! Begin loop over atomic orbitals for potential energy integrals
! and apply the contraction coefficients.
! The potential integrals are already in EEPB.

      INTC=0
      do I=Istart,Iend
      do J=Jstart,Jend
        INTC=INTC+1
        COEF=CA(I)*CB(J)
        EEPV(INTC)=EEPV(INTC)-EEPB(INTC)*COEF
      end do ! J
      end do ! I

! End of loop over atomic orbitals

      end do ! Jgauss
      end do ! Igauss

! End of loop over Gaussians.

! FILMAT takes the atomic integrals in EEPB and stores them in
! their proper places in VAINT.

      call FILMAT (EEPV, VAINT, MATlen, Iend, Jend, &
                          Iatmshl, Jatmshl, Irange, Jrange, Iaos, JaoS)

      end do ! Jatmshl
      end do ! Iatmshl
      end do ! Jshell
      end do ! Ishell

! End of loop over shells.

! Calculate the atomic potential energies
```

```
  EV = TRACLO (VAint, PM0, Basis%Nbasis, MATlen)

! Print the atomic integrals
  !       write(UNIout,*)'Atomic Potential Integrals for atom ',
  !       &CARTESIAN(Iatom)%ELEMENT,'label', Iatom
  !       call PRT_matrix (VAint, MATlen, Basis%Nbasis)

! Calculate the total potential integral

      Vint = Vint+VAint

      end if ! IA

! End of loop over atoms.

! Write integrals, and atomic & total potential energy to file

  !   write(UNIout,*)'Total Potential Integrals'
  !   call PRT_matrix (VINT, MATlen, Basis%Nbasis)

      return
      end subroutine get_Vne_Atom_Analytical
      end MODULE GetMolecularProperties
```

```
      MODULE mod_rotation
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
!
      USE AIMDFT_type
      USE type_molecule

      implicit none

CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

      SUBROUTINE GetCentroidCart (A,n,T)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description: Calculate the centroid (Geometry center) of n coordinates *
!                  points                                                    *
!*****************************************************************************
! MODULEs:

      implicit none

      ! A=(?:1), Y=(?,2), Z=(?,3)
      double precision, dimension (:,:), intent(in), allocatable :: A
      double precision, dimension (:), intent(out), allocatable :: T
      integer, intent(in) :: n ! Number of coordinates.
      integer :: i, j

      allocate(T(1:3)) ! (X,Y,Z) values

      T=0.0d0;
      do i=1, n
         do j=1, 3 ! loop over X, Y and Z values
            T(j)=T(j)+A(i,j)
         end do
      end do
      do j=1, 3
```

```
      T(j)=T(j)/n
      end do
      return
      end SUBROUTINE GetCentroidCart
      SUBROUTINE GetRotMatrix (NewCart,CurrCart,numberpoints,R,T)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description: Return the rotation and translation matrices for a set of   *
!                  points using least square method for fitting (SVD) .      *
!*****************************************************************************
!

      implicit none

      double precision, dimension(:,:), intent(in), allocatable :: NewCart
      double precision, dimension(:,:), intent(in), allocatable :: CurrCart
      integer, intent(in) :: numberpoints

      double precision, dimension (3,3), intent(out) :: R
      double precision, dimension (3), intent(out) :: T

      double precision, dimension (:), allocatable :: Aver_A, Aver_B
      double precision, dimension (:,:), allocatable :: H_a, H_b ! large arrays

      double precision, dimension (1:3) :: temp_Ha, temp_Hb, SSS
      double precision, dimension (1:3,1:3) :: H_matrix,HH,AAA,UUU,VVV,VT,UT

      integer :: i,j,k,l

      allocate (H_a(1:numberpoints,1:3))
      allocate (H_b(1:numberpoints,1:3))
      allocate (Aver_A(1:3))
      allocate (Aver_B(1:3))

      ! calculate average of matrix A and matrix B
      call GetCentroidCart (NewCart, numberpoints, Aver_A)
      call GetCentroidCart (CurrCart, numberpoints, Aver_B)

      ! calculate H matrix
      H_matrix=0.0d0;  H_a=0.0d0 ; H_b=0.0d0;
      do i=1, numberpoints
         do j=1,3
            H_a(i,j)=NewCart(i,j)-Aver_A(j) !H_a(i,j)=A(i,j)-A(1,j)
            H_b(i,j)=CurrCart(i,j)-Aver_B(j) !H_b(i,j)=B(i,j)-B(1,j)
         end do
      end do
      do i=1, numberpoints
         temp_Ha=0.0d0; temp_Hb=0.0d0
         do j=1,3
            temp_Ha(j)=H_a(i,j)
            temp_Hb(j)=H_b(i,j)
         end do
         HH=0.0d0
         do k=1,3
            do l=1,3
               HH(k,l)=temp_Ha(k)*temp_Hb(l)
            end do
         end do
         H_matrix=H_matrix+ HH
      end do
      CALL SVD(H_matrix,UUU,SSS,VVV,3,3) !Calculate SVD
      VT=transpose(VVV)
      UT=transpose(UUU)
```

```
      R=matmul(VVV,UT) !Calculate Rotation matrix
      T=matmul(-R,Aver_A)+Aver_B
      deallocate(H_a, H_b, Aver_A, Aver_B)
      return
      end SUBROUTINE GetRotMatrix
      SUBROUTINE DoRotation (OldCart,NewCart,numberpoints,R,T)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description: Get the new coordinate after applying the rotation and      *
!                  the translation matrices.                                   *
!*****************************************************************************
!
      implicit none

      integer, intent(in) :: numberpoints
      double precision, dimension (:,:), intent(in), allocatable :: OldCart
      double precision, dimension (3,3), intent(in) :: R
      double precision, dimension (3), intent(in) :: T
      double precision, dimension (:,:), intent(out), allocatable :: NewCart

      double precision, dimension (1:3) :: NewPoint, point
      integer::i

      allocate(NewCart(1:numberpoints,3))

      do i=1, numberpoints
         point=(/OldCart(i,1),OldCart(i,2),OldCart(i,3)/)
         NewPoint= Matmul(R,point) + T
         NewCart(i,1)=NewPoint(1)    ! X points
         NewCart(i,2)=NewPoint(2)    ! Y points
         NewCart(i,3)=NewPoint(3)    ! Z points
      end do
      return
      end SUBROUTINE DoRotation
      SUBROUTINE GetOldPoints (OldCart,NewCart,numberpoints,R,T)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description: Get the old coordinate before applying the rotation and     *
!                  the translation matrices.                                   *
!*****************************************************************************
!
      USE mod_math

      implicit none

      double precision, dimension (:,:), intent(out), allocatable :: OldCart
      double precision, dimension (3,3), intent(in) :: R
      double precision, dimension (3), intent(in) :: T
      double precision, dimension(:,:), intent(in), allocatable :: NewCart
      integer, intent(in) :: numberpoints

      double precision, dimension (:,:), allocatable :: RI
      double precision, dimension (1:3) :: NewPoint, point
      integer::i

      allocate(OldCart(1:numberpoints,3))
      allocate(RI(3,3))

      call INV33(R,RI)
      do i=1, numberpoints
         point=(/NewCart(i,1),NewCart(i,2),NewCart(i,3)/)
         NewPoint=Matmul(RI,(point-T))
```

```
         OldCart(i,1)=NewPoint(1)
         OldCart(i,2)=NewPoint(2)
         OldCart(i,3)=NewPoint(3)
      end do
      return
      end SUBROUTINE GetOldPoints
      SUBROUTINE GetStoredRotMatrix (OldCart,numberpoints,R,T)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description: Return the rotation and translation matrices that needed    *
!                  to get the standard form.                                   *
!*****************************************************************************
!
      implicit none

      double precision, dimension (:,:), intent(in), allocatable :: OldCart
      double precision, dimension (3,3), intent(out) :: R
      double precision, dimension (3), intent(out) :: T
      integer, intent(in) :: numberpoints

      double precision, dimension (:,:), allocatable :: Tempcart

      double precision :: AB_side,AC_side,BC_side
      double precision :: AD_side,BD_side,S_ABC,S_ABD
      double precision :: CD_side,CG_side,DG_side,Y_along4C
      double precision :: Y_along3,Z_along3,signang_ABC,signang_ABD
      double precision :: Y_alongC4,Z_alongC4,Y_along4,Z_along4,EG_side
      double precision :: ABS_ABC,BCS_ABC,ACS_ABC,ABS_ABD,BDS_ABD
      double precision :: ADS_ABD,ss1,ss2,ss3,dd,X4,X4_1,X4_2,DG_sideC,CD_sideC
      integer::I,J,signa_ABC,signa_ABD,numberw,signa_CDG,signang_CDG

      allocate (Tempcart(1:numberpoints,1:3))

      numberw=1
      TempCart=0.0d0
      SELECT CASE (numberpoints)
      CASE (1)
         TempCart(1,1:3)=0.0d0
         numberw=1
      CASE (2)
         numberw=2
         TempCart(1:2,1:3)=0.0d0
         TempCart(2,3)=DSQRT((OldCart(2,1)-OldCart(1,1))**2+(OldCart(2,2)-&
                             OldCart(1,2))**2+(OldCart(2,3)-OldCart(1,3))**2)
         !TempCart(3,1)=0.0d0;TempCart(3,2)=0.0d0;TempCart(3,3)=0.0
      CASE (3:)
         numberw=3
         ! ABC triangle
         AB_side=DSQRT((OldCart(2,1)-OldCart(1,1))**2+(OldCart(2,2)-&
                       OldCart(1,2))**2+(OldCart(2,3)-OldCart(1,3))**2)
         AC_side=DSQRT((OldCart(3,1)-OldCart(1,1))**2+(OldCart(3,2)-&
                       OldCart(1,2))**2+(OldCart(3,3)-OldCart(1,3))**2)
         BC_side=DSQRT((OldCart(2,1)-OldCart(3,1))**2+(OldCart(2,2)-&
                       OldCart(3,2))**2+(OldCart(2,3)-OldCart(3,3))**2)
         S_ABC = (AB_side+BC_side+AC_side)/2.0
         ABS_ABC=S_ABC-AB_side
         ACS_ABC=S_ABC-AC_side
         BCS_ABC=S_ABC-BC_side
         signang_ABC=AB_side**2+AC_side**2-BC_side**2
         signa_ABC=1
         if (signang_ABC.le.0) signa_ABC=-1
         Y_along3=DSQRT((4.0*S_ABC*ABS_ABC*ACS_ABC*BCS_ABC)/(AB_side**2))
         Z_along3=DSQRT(AC_side**2-Y_along3**2)*signa_ABC
```

```
        TempCart=0.0d0
        TempCart(2,3)=AB_side ! Z-axes for second point
        TempCart(3,2)=Y_along3
        TempCart(3,3)=Z_along3
        if (numberpoints.ne.3) then
           numberw=4
           ! ABD triangle
           AD_side=DSQRT((OldCart(4,1)-OldCart(1,1))**2+(OldCart(4,2)-&
                         OldCart(1,2))**2+(OldCart(4,3)-OldCart(1,3))**2)
           BD_side=DSQRT((OldCart(2,1)-OldCart(4,1))**2+(OldCart(2,2)-&
                         OldCart(4,2))**2+(OldCart(2,3)-OldCart(4,3))**2)
           S_ABD = (AB_side+BD_side+AD_side)/2.0
           ABS_ABD=S_ABD-AB_side
           ADS_ABD=S_ABD-AD_side
           BDS_ABD=S_ABD-BD_side
           signang_ABD=AB_side**2+AD_side**2-BD_side**2
           signa_ABD=1
           if (signang_ABD.le.0) signa_ABD=-1
           ! perpendicular of triangle ABD
           Y_alongC4=DSQRT((4.0*S_ABD*ABS_ABD*ADS_ABD*BDS_ABD)/(AB_side**2))
           Z_along4=DSQRT(AD_side**2-Y_alongC4**2) *signa_ABD ! Z axes >>>OK<<<
           ! equation of plane ss1.X+ss2.Y+ss3.Z+dd=0 >>>
           ! >> Using Vectors to Describe a Plane
           ss1=(OldCart(2,2)-OldCart(1,2))*(OldCart(3,3)-OldCart(1,3))-&
               (OldCart(2,3)-OldCart(1,3))*(OldCart(3,2)-OldCart(1,2))
           ss2=(OldCart(2,3)-OldCart(1,3))*(OldCart(3,1)-OldCart(1,1))-&
               (OldCart(2,1)-OldCart(1,1))*(OldCart(3,3)-OldCart(1,3))
           ss3=(OldCart(2,1)-OldCart(1,1))*(OldCart(3,2)-OldCart(1,2))-&
               (OldCart(2,2)-OldCart(1,2))*(OldCart(3,1)-OldCart(1,1))
           dd=-(ss1*OldCart(1,1)+ss2*OldCart(1,2)+ss3*OldCart(1,3))
           ! Distance from a point to a plane
           X4_1=(ss1*OldCart(4,1)+ss2*OldCart(4,2)+ss3*OldCart(4,3)+dd)
           X4_2=DSQRT(ss1**2+ss2**2+ss3**2)
           if (abs(X4_2).lt.1.0d-06) then
              X4=0.0d0
           else
              X4=X4_1/X4_2
           end if
           !Sign Y4 ?????? CDG triangle
           CD_sideC=DSQRT((OldCart(3,1)-OldCart(4,1))**2+(OldCart(3,2)-&
                          OldCart(4,2))**2+(OldCart(3,3)-OldCart(4,3))**2)
           CD_side=DSQRT(CD_sideC**2-X4**2)
           CG_side=Y_along3
           !>>>>>>>>>By USing the new coordinate Just Y and Z
           EG_side=Z_along3-Z_along4
           Y_along4C=DSQRT(Y_alongC4**2-X4**2)  ! without sign
           DG_side=DSQRT(Y_along4C**2+EG_side**2)
          !DG_side=DSQRT((0-OldCart(4,1))**2+(0-OldCart(4,2))**2+(DG_sideC)**2)

           signang_CDG=CG_side**2+DG_side**2-CD_side**2
           signa_CDG=1
           if (signang_CDG.le.0) signa_CDG=-1
           Y_along4=Y_along4C*signa_CDG
           X4=abs(X4) ! we want X alwayes positive
           TempCart(4,1)=X4 ! Z-axes for second point
           TempCart(4,2)=Y_along4
           TempCart(4,3)=Z_along4
        end if
     end SELECT
     call GetRotMatrix (OldCart,TempCart,numberw,R,T)
     deallocate(TempCart)
     return
     end SUBROUTINE GetStoredRotMatrix
     end MODULE mod_rotation
```

```
      MODULE rotation_Cart
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:                                                             *
!*****************************************************************************
!
      USE mod_rotation

      implicit none

CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

      SUBROUTINE DoCartRotation(OldCarts,NewCarts,pointnumber,RR,TT)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description: Apply the rotation and translation matrices on fragment     *
!*****************************************************************************
!
      implicit none

      type (FragAtomInfo), dimension(:), intent(in), allocatable  :: OldCarts
      type (FragAtomInfo), dimension(:), intent(out), allocatable :: NewCarts
      double precision, dimension (3,3), intent(in) :: RR
      double precision, dimension (3), intent(in) :: TT
      integer, intent(in) :: pointnumber

      double precision, dimension (:,:), allocatable :: CurrCart,  NewCart
      integer :: ifound

      allocate(NewCarts(pointnumber))
      allocate (CurrCart(pointnumber,1:3))
      allocate (NewCart(pointnumber,1:3))

      NewCarts=OldCarts

      do ifound=1, pointnumber  !loop over cart
         CurrCart(ifound,1)=OldCarts(ifound)%x
         CurrCart(ifound,2)=OldCarts(ifound)%y
         CurrCart(ifound,3)=OldCarts(ifound)%z
      end do
      call DoRotation(CurrCart,NewCart,pointnumber,RR,TT)
      do ifound=1, pointnumber  !loop over cart
         NewCarts(ifound)%x=NewCart(ifound,1)
         NewCarts(ifound)%y=NewCart(ifound,2)
         NewCarts(ifound)%z=NewCart(ifound,3)
      end do
      deallocate(CurrCart, NewCart)
      return
      end SUBROUTINE DoCartRotation
      SUBROUTINE GetCartRotMatrix(OldCarts,NewCarts,pointnumber,RR,TT)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:  Get the the least square method rotation and translation   *
!                   matrices                                                   *
!*****************************************************************************
!
      implicit none

      type (FragAtomInfo), dimension(:), intent(in), allocatable  :: OldCarts
      type (FragAtomInfo), dimension(:), intent(in), allocatable :: NewCarts
      double precision, dimension (3,3), intent(out)  :: RR
```

```
      double precision, dimension (3), intent(out)  :: TT

      integer, intent(in) :: pointnumber

      double precision, dimension (:,:), allocatable :: CurrCart,  NewCart
      integer :: ifound, pointnumber2

      allocate (CurrCart(pointnumber,1:3))
      allocate (NewCart(pointnumber,1:3))

      do ifound=1, pointnumber  !loop over cart atoms
         CurrCart(ifound,1)=OldCarts(ifound)%x
         CurrCart(ifound,2)=OldCarts(ifound)%y
         CurrCart(ifound,3)=OldCarts(ifound)%z

         NewCart(ifound,1)=NewCarts(ifound)%x
         NewCart(ifound,2)=NewCarts(ifound)%y
         NewCart(ifound,3)=NewCarts(ifound)%z
      end do
      pointnumber2=pointnumber
      if (pointnumber.gt.3) pointnumber2=4
      call GetRotMatrix (CurrCart,NewCart,pointnumber2,RR,TT)

      deallocate(CurrCart, NewCart)
      return
      end SUBROUTINE GetCartRotMatrix
      SUBROUTINE GetCartRotMatrixW(OldCarts,NewCarts,pointnumber,RR,TT)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description: iGet the the least square method rotation and translation *
!                  matrices after applying weights                           *
!*****************************************************************************
!
      implicit none

      type (FragAtomInfo), dimension(:), intent(in), allocatable :: OldCarts
      type (FragAtomInfo), dimension(:), intent(in), allocatable :: NewCarts
      double precision, dimension (3,3), intent(out) :: RR
      double precision, dimension (3), intent(out) :: TT

      integer, intent(in) :: pointnumber

      double precision, dimension (:,:), allocatable :: CurrCart,  NewCart
      integer :: ifound,pointnumber2
      integer, parameter :: weight0=2, weight1=1, weight2=0
      integer :: Newpointnumber, Newifound, countfound

      ! calculate number of points
      Newpointnumber=0
      do ifound=1, pointnumber  !loop over cart atoms
         if (OldCarts(ifound)%level.eq.0) Newpointnumber=2000
         if (OldCarts(ifound)%level.eq.1) Newpointnumber=Newpointnumber+&
                              OldCarts(ifound)%Atomic_Number**(weight1+2)
         if (OldCarts(ifound)%level.eq.2) Newpointnumber=&
               Newpointnumber+OldCarts(ifound)%Atomic_Number**(weight2+2)
       ! if (OldCarts(ifound)%level.gt.2) Newpointnumber=&
       ! Newpointnumber+OldCarts(ifound)%Atomic_Number
      end do
      allocate (CurrCart(Newpointnumber,1:3))
      allocate (NewCart(Newpointnumber,1:3))
      countfound=0
      do ifound=1, pointnumber  !loop over cart atoms
         if (OldCarts(ifound)%level.eq.0) then
```

```
            do Newifound=1, 2000
               countfound=countfound+1
               CurrCart(countfound,1)=OldCarts(ifound)%x
               CurrCart(countfound,2)=OldCarts(ifound)%y
               CurrCart(countfound,3)=OldCarts(ifound)%z

               NewCart(countfound,1)=NewCarts(ifound)%x
               NewCart(countfound,2)=NewCarts(ifound)%y
               NewCart(countfound,3)=NewCarts(ifound)%z
            end do
         end if
         if (OldCarts(ifound)%level.eq.1) then
            do Newifound=1, OldCarts(ifound)%Atomic_Number**(weight1+2)
               countfound=countfound+1
               CurrCart(countfound,1)=OldCarts(ifound)%x
               CurrCart(countfound,2)=OldCarts(ifound)%y
               CurrCart(countfound,3)=OldCarts(ifound)%z

               NewCart(countfound,1)=NewCarts(ifound)%x
               NewCart(countfound,2)=NewCarts(ifound)%y
               NewCart(countfound,3)=NewCarts(ifound)%z
            end do
         end if
         if (OldCarts(ifound)%level.eq.2) then

            do Newifound=1, OldCarts(ifound)%Atomic_Number**(weight2+2)
               countfound=countfound+1
               CurrCart(countfound,1)=OldCarts(ifound)%x
               CurrCart(countfound,2)=OldCarts(ifound)%y
               CurrCart(countfound,3)=OldCarts(ifound)%z

               NewCart(countfound,1)=NewCarts(ifound)%x
               NewCart(countfound,2)=NewCarts(ifound)%y
               NewCart(countfound,3)=NewCarts(ifound)%z
            end do
         end if
      !  if (OldCarts(ifound)%level.gt.2) then
      !     do Newifound=1, OldCarts(ifound)%Atomic_Number**2
      !        countfound=countfound+1
      !        CurrCart(countfound,1)=OldCarts(ifound)%x
      !        CurrCart(countfound,2)=OldCarts(ifound)%y
      !        CurrCart(countfound,3)=OldCarts(ifound)%z
      !
      !        NewCart(countfound,1)=NewCarts(ifound)%x
      !        NewCart(countfound,2)=NewCarts(ifound)%y
      !        NewCart(countfound,3)=NewCarts(ifound)%z
      !     end do
      !  end if
      end do
      call GetRotMatrix (NewCart,CurrCart,countfound,RR,TT)
      deallocate(CurrCart, NewCart)
      return
      end SUBROUTINE GetCartRotMatrixW
      SUBROUTINE GetDBCart(OldCarts,NewCarts,pointnumber,RR,TT)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description: Get the database coordinates and the rotation and the     *
!                  translation matrics                                       *
!*****************************************************************************
!
      implicit none

      type (FragAtomInfo), dimension(:), intent(in), allocatable :: OldCarts
```

```
      type (FragAtomInfo), dimension(:), intent(out), allocatable :: NewCarts
      double precision, dimension (3,3), intent(out) :: RR
      double precision, dimension (3), intent(out) :: TT

      integer, intent(in) :: pointnumber

      double precision, dimension (:,:), allocatable :: CurrCart,  NewCart
      integer :: ifound

      allocate (NewCarts(pointnumber))
      allocate (CurrCart(pointnumber,1:3))
      allocate (NewCart(pointnumber,1:3))

      NewCarts=OldCarts

      do ifound=1, pointnumber  !loop over cart atoms
         CurrCart(ifound,1)=OldCarts(ifound)%x
         CurrCart(ifound,2)=OldCarts(ifound)%y
         CurrCart(ifound,3)=OldCarts(ifound)%z
      end do
      call GetStoredRotMatrix (CurrCart,pointnumber,RR,TT)
      call DoRotation (CurrCart,NewCart,pointnumber,RR,TT)
      do ifound=1, pointnumber  !loop over cart atoms
         NewCarts(ifound)%x=NewCart(ifound,1)
         NewCarts(ifound)%y=NewCart(ifound,2)
         NewCarts(ifound)%z=NewCart(ifound,3)
      end do
      deallocate(CurrCart, NewCart)
      return
      end SUBROUTINE GetDBCart
      SUBROUTINE GetDBFormFragments(FragAtoms,NewFragAtoms,FragInfo)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description: Get the database coordinates and the rotation and the     *
!                  translation matrics for the fragments                     *
!*****************************************************************************
!
      implicit none

      type(FragAtomInfo),dimension(:,:),intent(in), allocatable  :: FragAtoms
      type(FragAtomInfo),dimension(:,:),intent(out), allocatable :: NewFragAtoms
      type(FragsInfo),dimension(:),intent(inout), allocatable :: FragInfo

      double precision, dimension (3,3) :: RR
      double precision, dimension (3) :: TT

      double precision, dimension (:,:), allocatable :: CurrCart,  NewCart
      integer :: ifound,JAtom,pointnumbers

      allocate (NewFragAtoms(Natoms,Natoms))

      NewFragAtoms=FragAtoms

      do JAtom=1, NAtoms !loop over all atoms
         pointnumbers=FragInfo(JAtom)%NAtoms
         allocate (CurrCart(1:pointnumbers,1:3))
         allocate (NewCart(1:pointnumbers,1:3))
         do ifound=1, pointnumbers  !loop over cart
            CurrCart(ifound,1)=FragAtoms(Jatom,ifound)%x
            CurrCart(ifound,2)=FragAtoms(Jatom,ifound)%y
            CurrCart(ifound,3)=FragAtoms(Jatom,ifound)%z
         end do
         call GetStoredRotMatrix (CurrCart,pointnumbers,RR,TT)
```

```
         FragInfo(Jatom)%RMat=RR
         FragInfo(Jatom)%TMat=TT
         call DoRotation (CurrCart,NewCart,pointnumbers,RR,TT)
         do ifound=1, pointnumbers  !loop over cart
            NewFragAtoms(Jatom,ifound)%x=NewCart(ifound,1)
            NewFragAtoms(Jatom,ifound)%y=NewCart(ifound,2)
            NewFragAtoms(Jatom,ifound)%z=NewCart(ifound,3)
         end do
         deallocate(CurrCart, NewCart)
      end do !loop over all atoms
      return
      end SUBROUTINE GetDBFormFragments
      end MODULE rotation_Cart
```

```
      SUBROUTINE CalcMoleProperties()
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
! MODULEs:

      USE AIMDFT_type
      USE type_molecule

      implicit none

      interface
         SUBROUTINE Build_Molecule_DB(DatabaseInform,FragProp,FRgrids,FragAtoms)
          use aimdft_type
          implicit none
          type (IndexFile), dimension (:), allocatable :: DatabaseInform
          type (PropFile), dimension(:), allocatable :: FragProp
          type (GridsDataBase), dimension(:,:), allocatable :: FRgrids
          type (FragAtomInfo), dimension(:,:), allocatable :: FragAtoms
         end SUBROUTINE Build_Molecule_DB
         SUBROUTINE Build_Molecule_Direct(DatabaseInform,FragProp,FRgrids)
          use aimdft_type
          USE type_molecule
          implicit none
          type (IndexFile), dimension (:), allocatable :: DatabaseInform
          type (PropFile), dimension(:), allocatable :: FragProp
          type (GridsDataBase), dimension(:,:), allocatable :: FRgrids
         end SUBROUTINE Build_Molecule_Direct
         SUBROUTINE CalcMolePropertiesDB(DatabaseInform,FragProp,FRgrids)
          use aimdft_type
          implicit none
          type (IndexFile), dimension (:), allocatable :: DatabaseInform
          type (PropFile), dimension(:), allocatable :: FragProp
          type (GridsDataBase), dimension(:,:), allocatable :: FRgrids
         end SUBROUTINE CalcMolePropertiesDB
      end interface

      type (IndexFile), dimension (:), allocatable :: DatabaseInform
      type (PropFile), dimension(:), allocatable :: FragProp
      type (GridsDataBase), dimension(:,:), allocatable :: FRgrids
      type (FragAtomInfo), dimension(:,:), allocatable :: FragAtoms

      allocate (DatabaseInform(Natoms))
      allocate (FragProp(Natoms))
      allocate (FragAtoms(Natoms,Natoms))
      allocate (FRgrids(Natoms,MaxGridPnt))

      if(DirectMethod) then
```

```
         call Build_Molecule_Direct(DatabaseInform,FragProp,FRgrids)
      else
         call Build_Molecule_DB(DatabaseInform,FragProp,FRgrids,FragAtoms)
      end if

      call CalcMolePropertiesDB(DatabaseInform,FragProp,FRgrids)
      deallocate (DatabaseInform)
      deallocate (FragProp)
      deallocate (FragAtoms)
      deallocate (FRgrids)
      return
      end SUBROUTINE CalcMoleProperties
```

```
      SUBROUTINE FragmentsDisplay()
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description: Display the current and database cartesian coordinates      *
!                  for the molecule fragments                                  *
!*****************************************************************************
!
      USE Sorted_AIMDFT
      USE mod_math
      USE symbol_AIMDFT

      implicit none

      character (SymMax), dimension (:), allocatable :: symbols
      type (FragsInfo), dimension(:), allocatable :: FragInfo
      type (FragAtomInfo), dimension(:,:), allocatable :: FragAtoms, DBFragAtoms

      double precision, dimension (:,:), allocatable :: RR
      double precision, dimension (:), allocatable :: TT
      integer :: Iatom,JAtom,ifound

      !type (TwoDiArrInt), dimension(:), allocatable :: LevelConnect z

      allocate (FragInfo(Natoms))
      allocate (symbols(Natoms))
      allocate (FragAtoms(Natoms,Natoms))
      allocate (DBFragAtoms(Natoms,Natoms))
      allocate (RR(1:3,1:3))
      allocate (TT(1:3))
      !allocate (LevelConnect(Level_Number))

      CALL LOGO()

 !     CALL GetLevelConnect(Level_Number,LevelConnect)
 !     write(*,*) "*********************************"
 !     write(*,*) "          Adjacency matrix       "
 !     write(*,*) "*********************************"
 !     do i=1, Level_Number
 !         write(*,*) "for level # ", i
 !         write(*,*) "____________________"
 !         call PRT_matrix(LevelConnect(i)%ijarray,NAtoms,NAtoms)
 !     end do

      CALL GetSortCartTerm (Level_Number,FragAtoms,FragInfo)

      write(*,'(/A)') "*************************************"
      write(*,'(A)')  "    Current cartesian coordinates    "
      write(*,'(A)')  "*************************************"
      CALL PRNTCART(FragAtoms,FragInfo)
```

```
      CALL GetDBFormFragments(FragAtoms,DBFragAtoms,FragInfo)
      write(*,'(/A)') "************************************"
      write(*,'(A)')  "    Database cartesian coordinates   "
      write(*,'(A)')  "************************************"
      CALL PRNTCART(DBFragAtoms,FragInfo)

      CALL GetFragSymbols(Level_Number,symbols)
      write(*,'(/A)') "************************************"
      write(*,'(A)')  "     Molecule atoms symbols          "
      write(*,'(A)')  "************************************"
      do Iatom=1, Natoms
          if (Cartesian(IAtom)%Atomic_Number.eq.0) cycle
          write(*,'(I5,3X,A)') Iatom,trim(symbols(Iatom))
      end do

      deallocate (FragInfo)
      deallocate (symbols)
      deallocate (FragAtoms)
      deallocate (DBFragAtoms)
      deallocate (RR)
      deallocate (TT)
      return
      end SUBROUTINE FragmentsDisplay
```

```
      SUBROUTINE Build_FragFiles()
!*****e*****************************************************************************
!     Date last modified:                                                          *
!     Author: Ibrahim Awad                                                         *
!     Description:                                                                 *
!*****************************************************************************
! MODULEs:

      USE module_grid_points
      USE NI_defaults
      USE type_Weights
      USE QM_defaults
      USE Sorted_AIMDFT
      USE symbol_AIMDFT

      implicit none
!
! Local scalars:

      type (IndexFile), dimension(:), allocatable :: MolIdxInfo
      type (FragsInfo), dimension(:), allocatable :: FragInfo
      type (FragAtomInfo), dimension(:,:), allocatable :: SortedFragmentAtom
      type (FragAtomInfo), dimension(:,:), allocatable :: DBFragmentAtom

      character (SymMax), dimension (:), allocatable ::  symbols
      character (SymMax), dimension (:), allocatable ::  UniSym
      integer, dimension (:), allocatable :: MUNatom

      integer :: katom
      integer :: UniSymMum
      integer :: IndexFoundNum
      integer :: fileunit
      integer :: ifound
      integer :: newindex
      integer :: maxindex
      integer :: iSym

      character (4)  :: OPT_YN
      character (60) :: cmdd
      character (60) :: radline
```

```
      character (60) :: outfile
      character (60) :: inputfile
      logical :: Lerror

      allocate (MolIdxInfo(Natoms))
      allocate (FragInfo(Natoms))
      allocate (SortedFragmentAtom(Natoms,Natoms))
      allocate (DBFragmentAtom(Natoms,Natoms))
      allocate (symbols(Natoms))
      allocate (UniSym(Natoms))
      allocate (MUNatom(Natoms))

      CALL Check_DB_files()
      CALL GetSortCartTerm(Level_Number,SortedFragmentAtom,FragInfo)
      CALL GetDBFormFragments(SortedFragmentAtom,DBFragmentAtom,FragInfo)
      CALL GetFragSymbols(Level_Number,symbols)
      CALL GetUniSym (Level_Number,UniSym,MUNatom,UniSymMum)
      CALL GetMoleculeAtomsIndex (MolIdxInfo,UniSym,UniSymMum,IndexFoundNum)
      CALL GetMaxIndex (maxindex) ! Find Maxindex

      write(*,'(A)') "************************************"
      write(*,'(A)') " The unique Symbols in the molecule "
      write(*,'(A)') "************************************"
      do iSym=1, UniSymMum
        write (*,'(A,I4,A,A)') "SYM# ", iSym, " is ", trim(UniSym(iSym))
      end do
        write(*,'(A)') "AIMDFT-AIMDFT-AIMDFT-AIMDFT-AIMDFT-AIMDFT-AIMDFT-AIMDFT"
      !! WORKING WITH the nonavaliable fragments within database
      ! add the new symbols
      newindex=maxindex
      do ifound=1, UniSymMum
         if (MolIdxInfo(ifound)%Availability) cycle
         newindex=newindex+1
         ! generate input files
         CALL INPUT_File_FRAG ('INPUT_','.dat',newindex,inputfile)
         CALL INPUT_File_FRAG ('INPUT_','.out',newindex,outfile)
         CALL GET_unit (inputfile, fileunit, Lerror)
         open(UNIT=fileunit, file=inputfile, status='REPLACE')
         write(fileunit,'(A)') &
         "MOLECULE MUltiplicity =  1 CHarge =  0  UNit = Angstrom"
         write(fileunit,'(A)')'TITle = "'//UniSym(ifound)//'"'
         write(fileunit,'(A)')'CArtesian'
         do katom=1,FragInfo(MUNatom(ifound))%NAtoms
             write(fileunit,'(A5,3f26.20)') &
             &  DBFragmentAtom(MUNatom(ifound), &
             &  katom)%element, &
             &  DBFragmentAtom(MUNatom(ifound),katom)%x*Bohr_to_Angstrom, &
             &  DBFragmentAtom(MUNatom(ifound),katom)%y*Bohr_to_Angstrom, &
             &  DBFragmentAtom(MUNatom(ifound),katom)%z*Bohr_to_Angstrom
         end do
         write(fileunit,'(A)')'end ! CArtesian'
         write(fileunit,'(A/)')'end ! MOLECULE'
         write(fileunit,'(A12,A14,A4/)')'basis name= ',&
                                        trim(Basis_set_name),' end'

         write(fileunit,'(A25,I4.4,A9/)')'SET RUN NAME = "RUN_FRAG_',&
                                         newindex,'" end end'
      !   write(fileunit,'(A/)') &
      !   'GRID MESH ORigin= ( 0.0 -5.0 -2.0 ) &
      !    MESH = ( 0.02 0.02 0.02 ) NX=0 NY=500 NZ=350 end end'
         write(fileunit,'(A/)')'PLOT ATOM=false Molecule=false end'
         OPT_YN=""
         if (trim(OPT_status).eq.'OPT') then
            OPT_YN="OPT"
```

```
         end if
         write(fileunit,'(A20,I4,A8,I3,A2,A4,A4/)')'AIMDFT IndexNumber=',&
               & newindex,' Level=',Level_Number,' ',OPT_YN,'end'
         write(fileunit,'(A20,A10,A4/)') 'PARtitioning Scheme=',&
               & trim(DEN_Partitioning),' end'

         if (trim(RADIAL_grid).eq.'GILL') then
            write(radline,'(A20,I3,A10,I2,I3,I4,A10)') &
           &'RADial Gill RPoints=',NRPoints_Gill ,&
           &' APoints=(', NApoints_Gill(1),NApoints_Gill(2),NApoints_Gill(3),&
           &' ) end end'
         else
            write(radline,'(A22,A7,A8)') 'NUmercial RADial GRID=',&
                                         trim(RADIAL_grid),' end end'
         endif

         write(fileunit,'(A/)') radline

         write(fileunit,'(A/)')'output object=MOL:INTERNAL%ZMATRIX end'
         if (trim(OPT_status).eq.'OPT') then
            write(fileunit,'(A/)')'OPT ITER=100 RUN end'
         end if
        ! write(fileunit,'(A/)')'output object=QM:RADIAL_DENSITY%MESH end'
         write(fileunit,'(A/)')'output object=AIMDFT:FRAGCART%DBASE end'
         write(fileunit,'(A)')'stop'
         close(UNIT=fileunit)
         !write(cmdd,'(a7,a15,a1,a15)')"mgauss<",inputfile,">" ,outfile
         ! Running the input file
         write(*,'(///A,A/,A/)') "*** RUNing the inputfile :: ", inputfile
         write(*,'(/A)') "****************************************"
         write(*,*) "Working on symbol ",UniSym(ifound)
         write(*,*) "IndexNUM ", newindex
         write(*,'(A/)') "****************************************"
         write(cmdd,'(a7,a15)')"mgauss<",inputfile
         CALL SYSTEM(cmdd)
         write(cmdd,'(a3,a15)')"rm ",inputfile
         CALL SYSTEM(cmdd)
      end do
      write(*,'(A///)')
      deallocate (FragInfo)
      deallocate (SortedFragmentAtom)
      deallocate (symbols)
      deallocate (UniSym)
      deallocate (MUNatom)
      deallocate (MolIdxInfo)
      deallocate (DBFragmentAtom)
! end of routine DENSITY_MESH
!      call PRG_manager ('exit', 'DENSITY_MESH', 'UTILITY')
CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      SUBROUTINE Check_DB_files()  ! needed to add titles for database files
!*****e*****************************************************************************
!     Date last modified:                                                          *
!     Author: Ibrahim Awad                                                         *
!     Description:                                                                 *
!*****************************************************************************
! MODULEs

      implicit none

      logical :: index_exists,cart_exists,prop_exists
      integer :: File_unit
      logical :: Lerror

      ! file_exists will be TRUE
```

```
      INQUIRE(FILE=trim(IndexFilePath), EXIST=index_exists)
      INQUIRE(FILE=trim(CartFilePath), EXIST=cart_exists)
      INQUIRE(FILE=trim(PropFilePath), EXIST=prop_exists)
      if (.not.index_exists) then
      CALL GET_unit (trim(IndexFilePath), File_unit, Lerror)
      open(UNIT=File_unit,file=trim(IndexFilePath), &
           status='new',form='formatted')
      write(File_unit,fmt_index_title) "Index#",          &
                                       & "Symbol",    &
                                       & "Atom#",     &
                                       & "Grid#",     &
                                       & "Grid",      &
                                       & "Method",    &
                                       & "Basisset",  &
                                       & "Weight",    &
                                       & "CoreSize",  &
                                       & "Opt/NO-opt"
      close(unit=File_unit)
      end if

      if (.not.cart_exists) then
      CALL GET_unit (trim(CartFilePath), File_unit, Lerror)
      open(UNIT=File_unit,file=trim(CartFilePath), &
           status='new',form='formatted')
      write(File_unit,fmt_cart_title) "#","#","Sym","X","Y","Z"
      close(unit=File_unit)
      end if

      if (.not.prop_exists) then
      CALL GET_unit (trim(PropFilePath), File_unit, Lerror)
      open(UNIT=File_unit,file=trim(PropFilePath), &
           status='new',form='formatted')
      write(File_unit,fmt_prop_title) &
                                    & "#",            &
                                    & "Electrons",    &
                                    & "K",            &
                                    & "K_HF",         &
                                    & "T",            &
                                    & "Vne_Ana",      &
                                    & "Vne_Num",      &
                                    & "Vee(Ana/Num)", &
                                    & "J",            &
                                    & "J_HF",         &
                                    & "Jaa=Kaa",      &
                                    & "Vee_DFT_AtomA"


      close(unit=File_unit)
      end if

      end SUBROUTINE Check_DB_files
      end SUBROUTINE Build_FragFiles
```

```
      SUBROUTINE get_databaseform()
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description: Display the current and database cartesian coordinates    *
!                  for the molecule fragments                                *
!*****************************************************************************
!
      USE AIMDFT_type
      USE type_molecule
      USE mod_rotation
```

```
      implicit none

      double precision, dimension (:,:), allocatable :: OldCart
      double precision, dimension (:,:), allocatable :: ROTCart
      double precision, dimension (:,:), allocatable :: NewCart
      double precision, dimension (:,:), allocatable :: RR
      double precision, dimension (:), allocatable :: TT
      integer :: Iatom, JAtom, IROT

      allocate (RR(1:3,1:3))
      allocate (TT(1:3))
      allocate (ROTCart(NAtoms,3))
      allocate (OldCart(NAtoms,3))
      allocate (NewCart(NAtoms,3))

      if(NROTlist.eq.0) then  ! If no atoms specified, stop
      write(*,*) "Error: No NROTlist"
      call exit()
      end if

      OldCart(1:Natoms,1)=Cartesian(1:Natoms)%x
      OldCart(1:Natoms,2)=Cartesian(1:Natoms)%y
      OldCart(1:Natoms,3)=Cartesian(1:Natoms)%z

      do IROT=1,NROTlist
         Iatom=ROTlist(IROT)
         ROTCart(IROT,1:3) = OldCart(Iatom,1:3)
      end do

      call GetStoredRotMatrix (ROTCart,NROTlist,RR,TT)
      call DoRotation (OldCart,NewCart,Natoms,RR,TT)

      if (RotUpdate) then
          Cartesian(1:Natoms)%x=NewCart(1:Natoms,1)
          Cartesian(1:Natoms)%y=NewCart(1:Natoms,2)
          Cartesian(1:Natoms)%z=NewCart(1:Natoms,3)
          write(*,'(/A,/A)') &
          "RotUpdate is TRUE","The Cartesian coordinates were updated"
          write(*,'(A)')     "The new coordinates are: (Angstrom unit)"
          write(*,'(A4,3A14)') "Atom","X","Y","Z"
          do Iatom=1, Natoms
              write(*,'(A4,3F14.8)') Cartesian(Iatom)%element, &
                              & NewCart(Iatom,1)*CartPrnFactor, &
                              & NewCart(Iatom,2)*CartPrnFactor, &
                              & NewCart(Iatom,3)*CartPrnFactor
          end do
          write(*,*) "-------------------------------------------"
      end if

      deallocate (ROTCart,OldCart,NewCart)
      deallocate (RR)
      deallocate (TT)
      return
      end SUBROUTINE get_databaseform
```

```
      MODULE mod_math
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
!
      implicit none
```

```
CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

SUBROUTINE LOGO()
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:  The AIMDFT logo                                            *
!*****************************************************************************
!
      implicit none

      write(*,*) "      ___  _____ ___  _________ ______  _____    "
      write(*,*) "    / _ \|_   _||  \/  ||  _  \|  ___||_   _|   "
      write(*,*) "   / /_\ \ | |  | .  . || | | || |_      | |     "
      write(*,*) "   |  _  | | |  | |\/| || | | ||  _|     | |     "
      write(*,*) "   | | | |_| |_ | |  | || |/ / | |       | |     "
      write(*,*) "   \_| |_/\___/ \_|  |_/|___/  \_|       \_/     "
      write(*,*) "                                                   "

      end SUBROUTINE LOGO
      SUBROUTINE  SwapI(a, b)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:  Swap two integer values                                    *
!*****************************************************************************
!
      implicit none

      integer, intent(inout) :: a, b
      integer                :: Temp

      Temp=a ; a=b ; b=Temp
      end SUBROUTINE  SwapI
      SUBROUTINE  SwapR(a, b)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:  Swap two real values                                       *
!*****************************************************************************
!
      implicit none

      double precision, intent(inout) :: a, b
      double precision                :: Temp

      Temp=a ; a=b ; b=Temp
      end SUBROUTINE  SwapR
      double precision FUNCTION DET33(A)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description: Calculate the determinant of a matrix (3x3), real values.   *
!*****************************************************************************
!
      implicit none

      double precision, dimension(3,3), intent(in)  :: A

      DET33 = A(1,1)*A(2,2)*A(3,3)  &
            - A(1,1)*A(2,3)*A(3,2)  &
            - A(1,2)*A(2,1)*A(3,3)  &
            + A(1,2)*A(2,3)*A(3,1)  &
```

```
            + A(1,3)*A(2,1)*A(3,2)   &
            - A(1,3)*A(2,2)*A(3,1)
      RETURN
      end FUNCTION DET33
      double precision FUNCTION VectorsDistance(dim_num,v1,v2)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description: Calculate the distance between two vectors                  *
!*****************************************************************************
!
      implicit none

      integer, intent(in) :: dim_num
      double precision, dimension(dim_num), intent(in) :: v1, v2

      VectorsDistance=dsqrt(sum((v1(1:dim_num)-v2(1:dim_num))**2))

      RETURN
      end FUNCTION VectorsDistance
      SUBROUTINE INV33(A,T)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description: Calculate the inverse of a matrix (3x3), real values.       *
!*****************************************************************************
!
      implicit none

      double precision, dimension (3,3), intent(in) :: A
      double precision, dimension (3,3), intent(out) :: T

      T(1,1) =  (A(2,2)*A(3,3)-A(2,3)*A(3,2))/DET33(A)
      T(1,2) = -(A(1,2)*A(3,3)-A(1,3)*A(3,2))/DET33(A)
      T(1,3) =  (A(1,2)*A(2,3)-A(1,3)*A(2,2))/DET33(A)
      T(2,1) = -(A(2,1)*A(3,3)-A(2,3)*A(3,1))/DET33(A)
      T(2,2) =  (A(1,1)*A(3,3)-A(1,3)*A(3,1))/DET33(A)
      T(2,3) = -(A(1,1)*A(2,3)-A(1,3)*A(2,1))/DET33(A)
      T(3,1) =  (A(2,1)*A(3,2)-A(2,2)*A(3,1))/DET33(A)
      T(3,2) = -(A(1,1)*A(3,2)-A(1,2)*A(3,1))/DET33(A)
      T(3,3) =  (A(1,1)*A(2,2)-A(1,2)*A(2,1))/DET33(A)

      end SUBROUTINE INV33
      SUBROUTINE RANDSEED(R)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description: Return randum number depend on the seed time.               *
!*****************************************************************************
!
      implicit none

      integer, INTENT(OUT) :: R
      integer :: randnumI,seedsize,values(1:8)
      character(len=6) :: randnumS
      integer, allocatable :: seed(:)
      double precision :: randnum

      call date_and_time(values=values)
      call random_seed(size=seedsize)
      allocate(seed(seedsize))
      seed(:) = values(8)
      call random_seed(put=seed)
      call random_number(randnum)
```

```
      randnumI = FLOOR(100000*randnum)
!     write(randnumS,'(I6.6)') randnumI
      R = randnumI

      end SUBROUTINE RANDSEED
      end MODULE mod_math
```

```
      SUBROUTINE SVD(A,U,S,V,M,N)
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: sukhbinder Singh                                                 *
!     Modified by: Ibrahim Awad                                                *
!     Description: Program computes the matrix singular value decomposition.   *
!                  Using Lapack library.                                       *
!*****************************************************************************
! MODULEs:

      double precision, dimension(M,N) :: A
      double precision, dimension(M,M) :: U(M,M)
      double precision, dimension(N,N) :: VT, V
      double precision, dimension(N)   :: S
      double precision, dimension(:), ALLOCATABLE  :: WORK

      integer :: LDA, M, N, LWORK, LDVT, INFO, I, J
      character(1) :: JOBU, JOBVT

      JOBU='A'
      JOBVT='A'
      LDA=M
      LDU=M
      LDVT=N

      lwork=MAX(1,3*MIN(M,N)+MAX(M,N),5*MIN(M,N))

      allocate(WORK(lwork))

      CALL DGESVD(JOBU, JOBVT, M, N, A, LDA, S, U, LDU, VT, LDVT, &
      &                   WORK, LWORK, INFO )

      DO I=1, 3
         DO J=1, 3
            V(J,I)=VT(I,J)
         end do
      end do

      deallocate(WORK)

      end SUBROUTINE SVD
```

```
      MODULE Sorted_AIMDFT
!*****************************************************************************
!     Date last modified:                                                      *
!     Author: Ibrahim Awad                                                     *
!     Description:                                                             *
!*****************************************************************************
!

      USE AddTerminal

      implicit none

CONTAINS !>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
      SUBROUTINE GetSortCartTerm(level,SortFragAtoms,SortFragInfo)
!*****************************************************************************
```

```
!     Date last modified:                                                     *
!     Author: Ibrahim Awad                                                    *
!     Description:                                                            *
!*************************************************************************
!

      implicit none

      integer, intent(in) :: level
      type(FragAtomInfo),dimension(:,:),intent(out),allocatable::SortFragAtoms
      type (FragsInfo), dimension(:), intent(out), allocatable :: SortFragInfo

      allocate (SortFragAtoms(Natoms,Natoms))
      allocate (SortFragInfo(Natoms))

      CALL GenerateFragments(level,SortFragAtoms,SortFragInfo)
      CALL AddTerminalAtoms(level,SortFragAtoms,SortFragInfo)
      ! +1 >>> Also sort the new terminals
      CALL SortFType(level+1,SortFragAtoms,SortFragInfo)

      return
      end SUBROUTINE GetSortCartTerm
      SUBROUTINE SortLevel(level,SortFragAtoms,SortFragInfo)
!*************************************************************************
!     Date last modified:                                                     *
!     Author: Ibrahim Awad                                                    *
!     Description:                                                            *
!*************************************************************************
!
      implicit none

      integer, intent(in) :: level
      type(FragAtomInfo),dimension(:,:),intent(inout),allocatable::SortFragAtoms
      type(FragsInfo),dimension(:),intent(inout),allocatable :: SortFragInfo
      type(FragAtomInfo), dimension(:,:), allocatable :: TEMPFragAtoms
      integer :: currvalue, nextvalue, inext
      integer :: Iatom, Jatom, ilevel, NtempOrd, curr, next
      integer, dimension(:,:), allocatable :: countLevel
      integer, dimension(:), allocatable :: excludeLis
      integer, dimension(10) :: tempOrd
      character (SymMax) :: TempCopyChar
      logical :: swap

      allocate (TEMPFragAtoms(Natoms,Natoms))
      allocate (countLevel(Natoms,level))
      allocate (excludeLis(Natoms))

      ! FIRST :::: Sort atoms by levels
      ! loop over all fragments for Jatoms
      do Jatom=1, NAtoms
         ! exclude dummy atoms
         if (Cartesian(JAtom)%Atomic_Number.eq.0) cycle
         NtempOrd=SortFragInfo(Jatom)%NAtoms
         swap=.true.
         do while(swap)
            swap=.false.
            do curr=1,NtempOrd-1
               next=curr+1
               currvalue=SortFragAtoms(Jatom,curr)%level
               nextvalue=SortFragAtoms(Jatom,next)%level
               if(currvalue.gt.nextvalue)then
                  TEMPFragAtoms(Jatom,curr)=SortFragAtoms(Jatom,curr)
                  SortFragAtoms(Jatom,curr)=SortFragAtoms(Jatom,next)
                  SortFragAtoms(Jatom,next)=TEMPFragAtoms(Jatom,curr)
```

```
                swap=.true.
             end if
          end do ! do curr
       end do ! do while(swap)
    end do  ! all atoms (atom by atom)

    ! Check each level if there is a UNIQUE livel that has just one atom,
    ! then fix there positions
    countLevel=0
    do ilevel=1, level  ! level loop
       do Jatom=1, NAtoms ! fragments loop
          do Iatom=1, SortFragInfo(Jatom)%NAtoms ! atoms within the fragments
             if (SortFragAtoms(Jatom,Iatom)%level.ne.ilevel) cycle
             if (SortFragAtoms(Jatom,Iatom)%sorted) cycle
             countLevel(Jatom,ilevel)=countLevel(Jatom,ilevel)+1
          end do ! subatom within fragment
       end do  ! all atoms (atom by atom) (Fragments)
    end do  ! level by level
    ! Fixed the value for the UNIQUE atoms within ilevel
    do ilevel=1, level  ! level loop
       do Jatom=1, NAtoms ! fragments loop
          if (countLevel(Jatom, ilevel).eq.1) then
             do Iatom=1, SortFragInfo(Jatom)%NAtoms
                if (SortFragAtoms(Jatom,Iatom)%level.ne.ilevel) cycle
                if (SortFragAtoms(Jatom,Iatom)%sorted) cycle
                SortFragAtoms(Jatom,Iatom)%sorted=.true.
             end do ! subatom within fragment
          end if
       end do ! all atoms (atom by atom) (Fragments)
    end do  ! level by level
    deallocate (TEMPFragAtoms)
    deallocate (countLevel)
    deallocate (excludeLis)
    end SUBROUTINE SortLevel

    SUBROUTINE SortZNUM(level,SortFragAtoms,SortFragInfo)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
!
    implicit none

    integer, dimension(:), allocatable :: CountAtomicN, CountAtomicI
    integer, dimension(:), allocatable :: CountAtomicT,excludeLis
    integer :: NtempOrd, TempCopy, curr,next, iZ, Z_Vali, Z_Valj, excludeN
    integer, intent(in) :: level
    type(FragAtomInfo),dimension(:,:),intent(inout),allocatable::SortFragAtoms
    type(FragsInfo),dimension(:),intent(inout),allocatable::SortFragInfo
    type(FragAtomInfo), dimension(:,:), allocatable :: TEMPFragAtoms
    integer :: currvalue, nextvalue, inext, Iatom, Jatom, ilevel
    integer, dimension(:,:), allocatable :: countLevel
    integer, dimension(10) :: tempOrd
    character (SymMax) :: TempCopyChar
    logical :: swap

    allocate (CountAtomicN(Natoms))
    allocate (CountAtomicI(Natoms))
    allocate (CountAtomicT(Natoms))
    allocate (excludeLis(Natoms))
    allocate (TEMPFragAtoms(Natoms,Natoms))
    allocate (countLevel(Natoms,level))
```

```
      CALL SortLevel(level,SortFragAtoms,SortFragInfo)

      ! SECOND :::: Sort atoms by ATOMIC NUMBER and keep the level
      ! Now loop within each level for fragment Jatom (fixed the level)
      do ilevel=1, level  ! level loop
          do Jatom=1, NAtoms
              ! exclude dummy atoms
              if (Cartesian(JAtom)%Atomic_Number.eq.0) cycle

              NtempOrd=SortFragInfo(Jatom)%NAtoms
              swap=.true.
              do while(swap)
                  !if (FragAtoms(Jatom,ifound)%level.ne.ilevel) cycle
                  swap=.false.
                  do curr=1,NtempOrd-1
                      !Just  ilevel
                      if (SortFragAtoms(Jatom,curr)%level.ne.ilevel) cycle
                      if (SortFragAtoms(Jatom,curr)%sorted) cycle
                      ! set curr subatom
                      currvalue=SortFragAtoms(Jatom,curr)%Atomic_Number

                      ! search for the next subatom in the same level,
                      ! then compare
                      next=-1
                      do inext=curr+1, NtempOrd
                         if (SortFragAtoms(Jatom,inext)%sorted) cycle
                         if (SortFragAtoms(Jatom,inext)%level.eq.ilevel) then
                            nextvalue=SortFragAtoms(Jatom,inext)%Atomic_Number
                            next=inext
                            ! Found subatom in the same level with the curr atom
                            exit
                         end if
                      end do
                      ! There just one atom at that level (NO next atom)
                      if (next.eq.-1) cycle

                      ! compare
                      if(currvalue.lt.nextvalue)then
                          TEMPFragAtoms(Jatom,curr)=SortFragAtoms(Jatom,curr)
                          SortFragAtoms(Jatom,curr)=SortFragAtoms(Jatom,next)
                          SortFragAtoms(Jatom,next)=TEMPFragAtoms(Jatom,curr)
                          swap=.true.
                      endif
                  end do ! do curr
                  !write(*,*) currvalue, "<", nextvalue, v1,v2
              end do ! do while(swap)
          end do  ! all atoms (atom by atom)
      end do  ! level by level

      ! NEXT step is to Check each level if there is a UNIQUE atom with ....
      ! ... (no other atoms have same Z), then fix its position
      do Jatom=1, NAtoms ! fragments loop
          do ilevel=1, level  ! level loop
              iZ=0
              CountAtomicN=0
              do Iatom=1, SortFragInfo(Jatom)%NAtoms !atoms within the fragments
                  if (SortFragAtoms(Jatom,Iatom)%level.ne.ilevel) cycle
                  if (SortFragAtoms(Jatom,Iatom)%sorted) cycle
                  iZ=iZ+1
                  CountAtomicN(iZ)=SortFragAtoms(Jatom,Iatom)%Atomic_Number
                  CountAtomicI(iZ)=Iatom
              end do ! subatom within fragment
              ! now check and put the value within array
              excludeN=0
```

```
              excludeLis=0
              do Z_Vali=1, iZ
                  do Z_Valj=Z_Vali+1,iZ
                      if (Z_Vali.eq.Z_Valj) cycle
                      if(CountAtomicN(Z_Vali).eq.CountAtomicN(Z_ValJ)) then
                          if (.not.ANY( excludeLis==CountAtomicI(Z_Vali))) then
                              excludeN=excludeN+1
                              excludeLis(excludeN)=CountAtomicI(Z_Vali)
                           end if
                          if (.not.ANY( excludeLis==CountAtomicI(Z_ValJ))) then
                              excludeN=excludeN+1
                              excludeLis(excludeN)=CountAtomicI(Z_ValJ)
                           end if
                      end if
                  end do
              end do
              do Iatom=1, SortFragInfo(Jatom)%NAtoms !atoms within the fragments
                  if (SortFragAtoms(Jatom,Iatom)%level.ne.ilevel) cycle
                  if (SortFragAtoms(Jatom,Iatom)%sorted) cycle
                  if (.not.ANY( excludeLis==Iatom)) &
                  SortFragAtoms(Jatom,Iatom)%sorted=.true.
              end do ! subatom within fragment
          end do  ! level by level
      end do ! all atoms (atom by atom) (Fragments)
      deallocate (CountAtomicN)
      deallocate (CountAtomicI)
      deallocate (CountAtomicT)
      deallocate (excludeLis)
      deallocate (TEMPFragAtoms)
      deallocate (countLevel)
      return
      end SUBROUTINE SortZNUM

      SUBROUTINE SortFType(level,SortFragAtoms,SortFragInfo)
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
!
      implicit none

      integer, dimension(:), allocatable :: CountAtomicN, CountAtomicI
      integer, dimension(:), allocatable :: CountAtomicT,excludeLis
      integer :: NtempOrd, TempCopy, curr,next, iZ, Z_Vali, Z_Valj, excludeN
      integer, intent(in) :: level
      type(FragAtomInfo),dimension(:,:),intent(inout),allocatable::SortFragAtoms
      type(FragsInfo),dimension(:),intent(inout),allocatable::SortFragInfo
      type(FragAtomInfo),dimension(:,:),allocatable::TEMPFragAtoms
      integer :: currvalue, nextvalue, inext, Iatom, Jatom, ilevel
      integer, dimension(:,:), allocatable :: countLevel
      integer, dimension(10) :: tempOrd
      character (SymMax) :: TempCopyChar
      logical :: swap

      allocate (CountAtomicN(Natoms))
      allocate (CountAtomicI(Natoms))
      allocate (CountAtomicT(Natoms))
      allocate (excludeLis(Natoms))
      allocate (TEMPFragAtoms(Natoms,Natoms))
      allocate (countLevel(Natoms,level))

      CALL SortLevel(level,SortFragAtoms,SortFragInfo)
      CALL SortZNUM(level,SortFragAtoms,SortFragInfo)
```

```
! Third :::: Sort atoms by ATOMIC TYPE for the fragment
! without the NEW termenal and keep the level
! Now loop within each level for fragment Jatom (fixed the level)
do ilevel=1, level  ! level loop
    do Jatom=1, NAtoms
        ! exclude dummy atoms
        if (Cartesian(JAtom)%Atomic_Number.eq.0) cycle

        NtempOrd=SortFragInfo(Jatom)%NAtoms
        swap=.true.
        do while(swap)
        !if (FragAtoms(Jatom,ifound)%level.ne.ilevel) cycle !Just  ilevel
            swap=.false.
            do curr=1,NtempOrd-1
                if (SortFragAtoms(Jatom,curr)%level.ne.ilevel) cycle
                if (SortFragAtoms(Jatom,curr)%sorted) cycle
                ! set curr subatom
                currvalue=SortFragAtoms(Jatom,curr)%type

                ! search for the next subatom in the same level,
                ! then compare
                next=-1
                do inext=curr+1, NtempOrd
                    ! exclude T sorted
                    if (SortFragAtoms(Jatom,inext)%sorted) cycle
                    ! same atomic number
                    if (SortFragAtoms(Jatom,curr)%Atomic_Number.ne.&
                    SortFragAtoms(Jatom,inext)%Atomic_Number) cycle
                    if (SortFragAtoms(Jatom,inext)%level.eq.ilevel) then
                        nextvalue=SortFragAtoms(Jatom,inext)%type
                        next=inext
                        exit ! Found subatom in the same level and same
                             ! atomic number with the curr atom
                    end if
                end do
                if (next.eq.-1) cycle ! There just one atom at that level

                ! compare
                if(currvalue.lt.nextvalue)then
                    TEMPFragAtoms(Jatom,curr)=SortFragAtoms(Jatom,curr)
                    SortFragAtoms(Jatom,curr)=SortFragAtoms(Jatom,next)
                    SortFragAtoms(Jatom,next)=TEMPFragAtoms(Jatom,curr)
                    swap=.true.
                endif
            end do ! do curr
            !write(*,*) currvalue, "<", nextvalue, v1,v2
        end do ! do while(swap)
    end do  ! all atoms (atom by atom)
end do  ! level by level

! NEXT step is to Check each level if there is a UNIQUE atom with ....
! ... (no other atoms have same Z and type), then fix its position
do Jatom=1, NAtoms ! fragments loop
    do ilevel=1, level  ! level loop
        iZ=0
        CountAtomicN=0
        do Iatom=1, SortFragInfo(Jatom)%NAtoms !atoms within the fragments
            if (SortFragAtoms(Jatom,Iatom)%level.ne.ilevel) cycle
            if (SortFragAtoms(Jatom,Iatom)%sorted) cycle
            iZ=iZ+1
            CountAtomicT(iZ)=SortFragAtoms(Jatom,Iatom)%type
            CountAtomicN(iZ)=SortFragAtoms(Jatom,Iatom)%Atomic_Number
            CountAtomicI(iZ)=Iatom
```

```
              end do ! subatom within fragment
              ! now check and put the value within array
              excludeN=0
              excludeLis=0
              do Z_Vali=1, iZ
                  do Z_Valj=Z_Vali+1,iZ
                      if (Z_Vali.eq.Z_Valj) cycle
                      if(CountAtomicN(Z_Vali).eq.CountAtomicN(Z_ValJ).and.&
                      CountAtomicT(Z_Vali).eq.CountAtomicT(Z_ValJ)) then
                          if (.not.ANY( excludeLis==CountAtomicI(Z_Vali))) then
                              excludeN=excludeN+1
                              excludeLis(excludeN)=CountAtomicI(Z_Vali)
                           end if
                          if (.not.ANY( excludeLis==CountAtomicI(Z_ValJ))) then
                              excludeN=excludeN+1
                              excludeLis(excludeN)=CountAtomicI(Z_ValJ)
                           end if
                      end if
                  end do
              end do
              do Iatom=1, SortFragInfo(Jatom)%NAtoms !atoms within the fragments
                  if (SortFragAtoms(Jatom,Iatom)%level.ne.ilevel) cycle
                  if (SortFragAtoms(Jatom,Iatom)%sorted) cycle
                  if (.not.ANY( excludeLis==Iatom))&
                  SortFragAtoms(Jatom,Iatom)%sorted=.true.
              end do ! subatom within fragment
          end do  ! level by level
      end do ! all atoms (atom by atom) (Fragments)
      deallocate (CountAtomicN)
      deallocate (CountAtomicI)
      deallocate (CountAtomicT)
      deallocate (excludeLis)
      deallocate (TEMPFragAtoms)
      deallocate (countLevel)
      return
      end SUBROUTINE SortFType
      SUBROUTINE GetSortCart(level,SortFragAtoms,SortFragInfo)   ! not completed
!*****************************************************************************
!     Date last modified:                                                    *
!     Author: Ibrahim Awad                                                   *
!     Description:                                                           *
!*****************************************************************************
!
      implicit none

      integer, intent(in) :: level
      type(FragAtomInfo),dimension(:,:),intent(out),allocatable :: SortFragAtoms
      type(FragsInfo),dimension(:),intent(out),allocatable :: SortFragInfo
      type(FragAtomInfo),dimension(:,:),allocatable :: FragAtoms,TEMPFragAtoms
      type(FragsInfo),dimension(:),allocatable :: FragInfo  ! (main atom)
      integer, dimension(:), allocatable:: sortindex,sorttype
      integer :: Iatom,Jatom,ifound,ilevel,kfound,mfound,ifoundlevel
      integer :: startilevel,nosimilar,nsim,munatom
      integer :: currvalue, nextvalue, inext
      integer, dimension(:,:), allocatable :: countLevel
      integer, dimension(:), allocatable :: CountAtomicT,excludeLis
      integer, dimension(:), allocatable :: CountAtomicN, CountAtomicI
      double precision, dimension(:), allocatable :: SumZ, SumZ_x
      double precision, dimension(:), allocatable :: SumZ_y, SumZ_z
      double precision, dimension(:,:), allocatable :: Z_x ,Z_y ,Z_z
      double precision, dimension(:,:), allocatable :: Cent_x, Cent_y, Cent_z
      double precision, dimension(:,:), allocatable :: Cent_R
      integer, dimension(10) :: tempOrd
      integer :: NtempOrd, TempCopy, curr,next, iZ, Z_Vali, Z_Valj
```

```
      integer :: excludeN, testi
      character (SymMax) :: TempCopyChar
      logical :: swap

      !>>>>>>>>>>>>>>Begin >>>>>>>>>>>>>>>>>>>
      allocate (SortFragAtoms(Natoms,Natoms))
      allocate (SortFragInfo(Natoms))
      allocate (FragAtoms(Natoms,Natoms))
      allocate (TEMPFragAtoms(Natoms,Natoms))
      allocate (FragInfo(Natoms))
      allocate (countLevel(Natoms,level))
      allocate (CountAtomicN(Natoms))
      allocate (CountAtomicI(Natoms))
      allocate (CountAtomicT(Natoms))
      allocate (excludeLis(Natoms))
      allocate (SumZ(Natoms))
      allocate (SumZ_x(Natoms))
      allocate (SumZ_y(Natoms))
      allocate (SumZ_z(Natoms))
      allocate (Z_x(Natoms,Natoms))
      allocate (Z_y(Natoms,Natoms))
      allocate (Z_z(Natoms,Natoms))
      allocate (Cent_x(Natoms,Natoms))
      allocate (Cent_y(Natoms,Natoms))
      allocate (Cent_z(Natoms,Natoms))
      allocate (Cent_R(Natoms,Natoms))


      CALL GenerateFragments(level,SortFragAtoms,SortFragInfo)
      CALL SortLevel(level,SortFragAtoms,SortFragInfo)
      CALL SortZNUM(level,SortFragAtoms,SortFragInfo)
      CALL SortFType(level,SortFragAtoms,SortFragInfo)


      if(.false.) then
      ! Forth :::: Sort atoms by their far from center ......
      !            of mass (Atomic Number) .....
      !         ... for each fragment without the NEW termenal ...
      !         ... and keep all the pevious conditions

      ! Now calculate the center of mass (used Z instead of atomic mass)
      ! Forth-a:::: calculate total Z for each fragment without the termenals
      do Jatom=1, NAtoms ! fragments loop
          SumZ(Jatom)=0.0d0
          SumZ_x(Jatom)=0.0d0; SumZ_y(Jatom)=0.0d0 ; SumZ_z(Jatom)=0.0d0
          do Iatom=1, FragInfo(Jatom)%NAtoms ! atoms within the fragments
              ! Calculate Z*x,Z*y, and Z*z for each atom
              Z_x(Jatom,Iatom)=SortFragAtoms(Jatom,Iatom)%Atomic_Number*&
                               SortFragAtoms(Jatom,Iatom)%x
              Z_y(Jatom,Iatom)=SortFragAtoms(Jatom,Iatom)%Atomic_Number*&
                               SortFragAtoms(Jatom,Iatom)%y
              Z_z(Jatom,Iatom)=SortFragAtoms(Jatom,Iatom)%Atomic_Number*&
                               SortFragAtoms(Jatom,Iatom)%z
              ! Find the sum each of Z*x,Z*y, and Z*z
              SumZ_x(Jatom)=SumZ_x(Jatom)+Z_x(Jatom,Iatom)
              SumZ_y(Jatom)=SumZ_y(Jatom)+Z_y(Jatom,Iatom)
              SumZ_z(Jatom)=SumZ_z(Jatom)+Z_z(Jatom,Iatom)
              ! calculate Total sum of Z
              SumZ(Jatom)=SumZ(Jatom)+&
              dble(SortFragAtoms(Jatom,Iatom)%Atomic_Number)
          end do ! subatom within fragment
        ! calculate center of mass \[X = \frac{{\sum {Z \cdot x} }}{{\sum Z }}\]
          do Iatom=1, FragInfo(Jatom)%NAtoms ! atoms within the fragments
              Cent_x(Jatom,Iatom)=(Z_x(Jatom,Iatom)-SumZ_x(Jatom))/SumZ(Jatom)
```

```
            Cent_y(Jatom,Iatom)=(Z_y(Jatom,Iatom)-SumZ_y(Jatom))/SumZ(Jatom)
            Cent_z(Jatom,Iatom)=(Z_z(Jatom,Iatom)-SumZ_z(Jatom))/SumZ(Jatom)
            Cent_R(Jatom,Iatom)=dsqrt( Cent_x(Jatom,Iatom)**2 + &
                                     & Cent_y(Jatom,Iatom)**2 + &
                                     & Cent_z(Jatom,Iatom)**2)

            write(*,*) Jatom, Iatom, Cent_R(Jatom,Iatom),&
                       SortFragAtoms(Jatom,Iatom)%element
         end do ! subatom within fragment
         !SumZ_x(Jatom)=SumZ_x(Jatom)/SumZ(Jatom)
         !SumZ_y(Jatom)=SumZ_y(Jatom)/SumZ(Jatom)
         !SumZ_z(Jatom)=SumZ_z(Jatom)/SumZ(Jatom)
        !write(*,'(I3,4F16.8)') Jatom, SumZ(Jatom), &
        !                       SumZ_x(Jatom), SumZ_y(Jatom), SumZ_z(Jatom)
    end do ! all atoms (atom by atom) (Fragments)
    end if ! just comment

    CALL CalcFragConnect(SortFragAtoms,SortFragInfo)
    deallocate(FragAtoms,TEMPFragAtoms, FragInfo, countLevel)
    deallocate(CountAtomicN,CountAtomicI,excludeLis,CountAtomicT)
    deallocate(SumZ,SumZ_x,SumZ_y,SumZ_z,Z_x ,Z_y ,Z_z,)
    deallocate(Cent_x,Cent_y,Cent_z,Cent_R)
    end SUBROUTINE GetSortCart
    end MODULE Sorted_AIMDFT
```

Bibliography