DEPARTMENT OF PHYSICS

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

ROORKEE – 247667

Solving the Dirac Hamiltonian for Many Electron Atoms using the Dirac Hartree Fock Self Consistent Field Method in MATLAB

A Thesis
Submitted by
LAKSH ARORA
20615013

For the award of the degree
Of
MASTER OF SCIENCE
May 2022

DEPARTMENT OF PHYSICS

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

ROORKEE – 247667

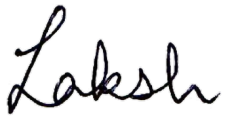
Solving the Dirac Hamiltonian for Many Electron Atoms using the Dirac Hartree Fock Self Consistent Field Method in MATLAB

A Thesis
Submitted by
LAKSH ARORA
20615013

For the award of the degree
Of
MASTER OF SCIENCE
May 2022

THESIS CERTIFICATE

This is to undertake that the Thesis titled **SOLVING THE DIRAC HAMILTONIAN FOR MANY ELECTRON ATOMS USING THE DIRAC HARTREE FOCK SELF CONSISTENT FIELD METHOD IN MATLAB**, submitted by me to the Indian Institute of Technology Roorkee, for the award of **Master of Science**, is a bona fide record of the research work done by me under the supervision of **Dr. H. S. Nataraj**. The contents of this Thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.



Laksh Arora
20615013

Roorkee 247667
Date: 02 May 2022

Dr. H. S. Nataraj
Project Supervisor
Assistant Professor
Department of Physics
IIT Roorkee

dissertation file

ORIGINALITY REPORT

14%	9%	9%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Indian Institute of Technology, Madras Student Paper	2%
2	Ajaya K. Mohanty. "A Dirac-Fock self-consistent field method for closed-shell molecules including Breit interaction", International Journal of Quantum Chemistry, 05/20/1992 Publication	1%
3	gim.unex.es Internet Source	1%
4	Pedro Antonio Soares de Alcântara. "On symbol correspondences for spin and quark systems", Universidade de Sao Paulo, Agencia USP de Gestao da Informacao Academica (AGUIA), 2022 Publication	1%
5	www.mathworks.com Internet Source	1%
6	Submitted to Facultad Latinoamericana de Ciencias Sociales (FLACSO) - Sede Ecuador	1%

Student Paper

7	cfile25.uf.tistory.com Internet Source	1 %
8	Hans Shih-Han Wang, Cory Cornelius, Brandon Edwards, Jason Martin. "Toward Few-step Adversarial Training from a Frequency Perspective", Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence, 2020 Publication	<1 %
9	Rezaei, G.. "Nonlinear optical properties of a hydrogenic impurity in an ellipsoidal finite potential quantum dot", Current Applied Physics, 201103 Publication	<1 %
10	fismed.ciemat.es Internet Source	<1 %
11	www.osti.gov Internet Source	<1 %
12	Nan Wang, Zhen Qin, Xuanhui Wang, Hongning Wang. "Non-Clicks Mean Irrelevant? Propensity Ratio Scoring As a Correction", Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021 Publication	<1 %

epdf.tips

13	Internet Source	<1%
14	Submitted to Indian Institute of Technology Guwahati Student Paper	<1%
15	deepblue.lib.umich.edu Internet Source	<1%
16	www.ee.iitm.ac.in Internet Source	<1%
17	Submitted to Pennsylvania State System of Higher Education Student Paper	<1%
18	Submitted to Indian Institute of Technology, Kanpur Student Paper	<1%
19	Submitted to The University of Manchester Student Paper	<1%
20	Fei Song, Zhengyang Ai, Haowei Zhang, Ilsun You, Shiyong Li. "Smart Collaborative Balancing for Dependable Network Components in Cyber-Physical Systems", IEEE Transactions on Industrial Informatics, 2020 Publication	<1%
21	Submitted to Australian National University Student Paper	<1%

22	Luis Serrano-Andrés, Juan José Serrano-Pérez. "Chapter 14 Calculation of Excited States: Molecular Photophysics and Photochemistry on Display", Springer Science and Business Media LLC, 2012 Publication	<1 %
23	Submitted to KTH - The Royal Institute of Technology Student Paper	<1 %
24	Mukhopadhyay, Atashi Basu. "Theoretical Investigation of Static and Dynamic Properties of Zeolite ZSM-5 Based Amorphous Material", Kölner UniversitätsPublikationsServer, 2011. Publication	<1 %
25	web.archive.org Internet Source	<1 %
26	Jakob Sternby. "Frame Deformation Energy Matching of On-Line Handwritten Characters", Lecture Notes in Computer Science, 2005 Publication	<1 %
27	Jingtao Yao, Chew Lim Tan. "A case study on using neural networks to perform technical forecasting of forex", Neurocomputing, 2000 Publication	<1 %
28	Svetlana Kotochigova, Zachary H. Levine, Eric L. Shirley, M. D. Stiles, Charles W. Clark.	<1 %

"Local-density-functional calculations of the energy of atoms", Physical Review A, 1997
Publication

29	arxiv.org Internet Source	<1%
30	epdf.pub Internet Source	<1%
31	epljournal.edpsciences.org Internet Source	<1%
32	researchbank.rmit.edu.au Internet Source	<1%
33	www.ideals.illinois.edu Internet Source	<1%

Exclude quotes	Off	Exclude matches	< 10 words
Exclude bibliography	Off		

Dr H S Nataraj
03/05/2022
Project Supervisor
Assistant Professor
Department of Physics
IIT Roorkee

CONTENTS

CHAPTER 1

DIRAC HARTREE FOCK THEORY

This chapter is a summary of Szabo and Ostlund (1996) and Nataraj (2021*b*).

1.1 DIRAC HAMILTONIAN

The total Dirac-Breit Hamiltonian for a many electron system has two terms, a one electron part consisting of kinetic energy, mass energy and nuclear attraction given by

$$h(i) = c\boldsymbol{\alpha} \cdot \boldsymbol{p_i} + c^2(\beta - 1) + V_{nuc}(r_i)$$

where we have subtracted the rest mass energy. For a point nucleus,

$$V_{nuc}(r_i) = -\frac{z}{r_i} \tag{1.1}$$

The second term is a two electron term consisting of a Coulomb repulsion term and the Breit interaction,

$$H_{12} = \sum_{ij,i<j} \frac{1}{r_{ij}} + \sum_{ij,i<j} B_{ij}$$

In the low frequency limit,

$$B_{12} = -\frac{1}{2r_{12}} \left\{ \boldsymbol{\alpha_1} \cdot \boldsymbol{\alpha_2} + \frac{(\boldsymbol{\alpha_1} \cdot \boldsymbol{r_{12}})(\boldsymbol{\alpha_2} \cdot \boldsymbol{r_{12}})}{r_{12}^2} \right\}$$

where Dirac operators $\boldsymbol{\alpha}$ and β are expressed by matrices,

$$\boldsymbol{\alpha} = \begin{bmatrix} 0 & \boldsymbol{\sigma} \\ \boldsymbol{\sigma} & 0 \end{bmatrix} \text{ and } \quad \beta = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$$

where $\boldsymbol{\sigma}$ are the three Pauli matrices and I is the 2×2 unit matrix.

Atomic units ($e = \hbar = m = 1$) will be used throughout this dissertation unless explicitly mentioned otherwise. We will not be considering Breit Interaction throughout this discussion.

1.2 HARTREE FOCK EQUATIONS

We are interested in finding a set of spin orbitals $\{\chi_a\}$ such that the Slater determinant formed from these spin orbitals,

$$|\Psi_0\rangle = |\chi_1\chi_2\cdots\chi_a\chi_b\cdots\chi_n\rangle$$

is the best approximation to the ground state of the system.

The total energy is given by

$$E_0 = \langle\Psi_0|H|\Psi_0\rangle = \sum_a \langle a|h|a\rangle + \frac{1}{2}\sum_{ab}\langle ab||ab\rangle \tag{1.2}$$

where,

$$\langle i|h|j\rangle = \langle\chi_i|h|\chi_j\rangle = \int \boldsymbol{dx_1}\chi_i^*(\boldsymbol{x_1})h(\boldsymbol{r_1})\chi_j(\boldsymbol{x_1})$$

$$\langle ij|kl\rangle = \langle\chi_i\chi_j|\chi_k\chi_l\rangle = \int \boldsymbol{dx_1}\,\boldsymbol{dx_2}\chi_i^*(\boldsymbol{x_1})\chi_j^*(\boldsymbol{x_2})r_{12}^{-1}\chi_k(\boldsymbol{x_1})\chi_l(\boldsymbol{x_2})$$

$$\langle ij||kl\rangle = \langle ij|kl\rangle - \langle ij|lk\rangle$$

We can now vary the spin orbitals $\{\chi_a\}$, such that they remain orthonormal,

$$\langle\chi_a|\chi_b\rangle = \delta_{ab}$$

until the energy E_0 is minimum. The corresponding spin orbitals are the best approximation to the ground state of the system according to the variational principle.

The resulting equation is called the Hartree-Fock integro-differential equation,

$$h(1)\chi_a(1) + \sum_{b\neq a}\left[\int \boldsymbol{dx_2}\,|\chi_b(2)|^2\,r_{12}^{-1}\right]\chi_a(1) - \sum_{b\neq a}\left[\int \boldsymbol{dx_2}\chi_b^*(2)\chi_a(2)r_{12}^{-1}\right]\chi_b(1) = \epsilon_a\chi_a(1) \tag{1.3}$$

where h is the one electron operator.

The two summation terms in equation 1.3 represent electron electron interaction.

The first is the Direct Coulomb term which can be interpreted as the average potential acting on the electron in χ_a from the $(N-1)$ other electrons.

$$J_b(1) = \int \boldsymbol{dx_2}\, |\chi_b(2)|^2\, r_{12}^{-1}$$

is defined as the coulomb operator.

The second term is called the Exchange term which arises from the antisymmetric nature of the Slater determinant. It is due to a quantum phenomenon and therefore does not have a simple classical interpretation.

None the less, we can write the Hartree-Fock integro-differential equation as an eigenvalue equation,

$$\left[h(1) + \sum_{b\neq a} J_b(1) - \sum_{b\neq a} K_b(1)\right]\chi_a(1) = \epsilon_a \chi_a(1) \tag{1.4}$$

provided that

$$K_b(1)\chi_a(1) = \left[\int \boldsymbol{dx_2}\chi_b^*(2) r_{12}^{-1}\chi_a(2)\right]\chi_b(1)$$

which involves an "exchange" of electron 1 and 2 compared to

$$J_b(1)\chi_a(1) = \left[\int \boldsymbol{dx_2}\chi_b^*(2) r_{12}^{-1}\chi_b(2)\right]\chi_a(1)$$

Since $[J_a(1) - K_a(1)]\,\chi_a(1) = 0$, we can add this to equation 1.4 to obtain the usual form of the Hartree-Fock equation,

$$f\,|\chi_a\rangle = \epsilon_a\,|\chi_a\rangle$$

where

$$f(1) = h(1) + \sum_b [J_b(1) - K_b(1)]$$

1.3 INTERPRETATION OF SOLUTIONS OF THE HARTREE-FOCK EQUATIONS

The orbital energies can be expressed as

$$\begin{aligned}\epsilon_i &= \langle \chi_i | f | \chi_i \rangle \\ &= \langle \chi_i | h + \sum_b (J_b - K_b) | \chi_i \rangle \\ &= \langle i|h|i\rangle + \sum_b \langle ib||ib\rangle\end{aligned}$$

If we add up all the orbital energies, we get,

$$\sum_a \epsilon_a = \sum_a \langle a|h|a\rangle + \sum_a \sum_b \langle ab||ab\rangle$$

Comparing this to equation 1.2, we can see that

$$E_0 \neq \sum_a \epsilon_a$$

Now, given a single determinant $|^N\Psi_0\rangle$, the ionization potential to produce an (N-1) electron single determinant $|^{N-1}\Psi_a\rangle$ by removing an electron from χ_a is given by,

$$\begin{aligned}IP &= {}^{N-1}E_a - {}^{N}E_0 \\ &= \sum_{c\neq a} \langle c|h|c\rangle + \frac{1}{2}\sum_{c\neq a}\sum_{b\neq a} \langle cb||cb\rangle - \sum_c \langle c|h|c\rangle - \frac{1}{2}\sum_c \sum_b \langle cb||cb\rangle \\ &= -\langle a|h|a\rangle - \frac{1}{2}\sum_c \langle ca||ca\rangle - \frac{1}{2}\sum_b \langle ab||ab\rangle \\ &= -\langle a|h|a\rangle - \sum_b \langle ab||ab\rangle \\ &= -\epsilon_a\end{aligned}$$

Therefore, the orbital energies are the negative of ionisation potentials to remove the corresponding electrons (Koopman's Theorem).

This difference is because we have not solved the exact electronic Schrodinger equation,

$$H\left|\Phi_0\right\rangle = E_0\left|\Phi_0\right\rangle$$

but rather we have used the variational principle to find an approximate solution which instead are exact solutions of a Hamiltonian given by,

$$H_0 = \sum_{i=1}^{N} f(i)$$

1.4 RESTRICTED CLOSED SHELL HARTREE FOCK

For the rest of this thesis, we are only concerned with restricted closed shell spin orbitals, that is, we are only allowed to have an even number of electrons with all electrons paired such that all spatial orbitals are doubly occupied.

The restricted set of spin orbitals have the form,

$$\chi_i(\boldsymbol{x}) = \begin{cases} \psi_j(r)\alpha(\omega) \\ \psi_j(r)\beta(\omega) \end{cases}$$

Using the above, we can integrate out the spin functions to give the following closed shell equations:

$$f(1)\psi_j(1) = \epsilon_j\psi_j(1) \tag{1.5}$$

$$f(1) = h(1) + \sum_{a}^{N/2} 2J_a(1) - K_a(1) \tag{1.6}$$

$$J_a(1) = \int dr_2 \psi_a^*(2) r_{12}^{-1} \psi_a(2)$$

$$K_a(1)\psi_i(1) = \left[\int dr_2 \psi_a^*(2) r_{12}^{-1} \psi_i(2)\right] \psi_a(1)$$

$$E_0 = 2\sum_a h_{aa} + \sum_a \sum_b 2J_{ab} - K_{ab}$$

1.5 INTRODUCTION OF BASIS: ROOTHAN EQUATIONS

Introduce K basis functions $\{\phi_\mu(r), \mu = 1, 2, \ldots, K\}$

such that,

$$\psi_i = \sum_{\mu=1}^{K} C_{\mu i}\phi_\mu \tag{1.7}$$

Substituting this into equation 1.5 gives

$$f(1)\sum_\nu C_{\nu i}\phi_\nu(1) = \epsilon_i \sum_\nu C_{\nu i}\phi_\nu(1)$$

Multiplying $\phi^*_\mu(1)$ on the left and integrating, we get,

$$\sum_\nu C_{\nu i}\int dr_1\phi^*_\mu(1)f(1)\phi_\nu(1) = \epsilon_i \sum_\nu C_{\nu i}\int dr_1\phi^*_\mu(1)\phi_\nu(1) \tag{1.8}$$

We define,

1. $$S_{\mu\nu} = \int dr_1\phi^*_\mu(1)\phi_\nu(1)$$

   called the overlap matrix. It is a $K \times K$ Hermitian matrix with diagonal elements unity and $0 \leq |S_{\mu\nu}| \leq 1$, since the basis functions are generally normalised and linearly independent but not orthogonal.

2. $$F_{\mu\nu} = \int dr_1\phi^*_\mu(1)f(1)\phi_\nu(1)$$

   called the Fock matrix.

With these, equation 1.8 can be written as

$$\sum_\nu F_{\mu\nu}C_{\nu i} = \epsilon_i \sum_\nu S_{\mu\nu}C_{\nu i},$$

$$i = 1, 2, \ldots, K.$$

or more compactly,

$$FC = SC\epsilon \tag{1.9}$$

These are called the Roothan equations.

Where C is a $K \times K$ matrix of expansion coefficients $C_{\mu i}$.

$$C = \begin{bmatrix} C_{11} & C_{12} & \cdots & \cdots & C_{1K} \\ C_{21} & C_{22} & \cdots & \cdots & C_{2K} \\ \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ C_{K1} & C_{K2} & \cdots & \cdots & C_{KK} \end{bmatrix}$$

and ϵ is a diagonal matrix of orbital energies ϵ_i,

$$\epsilon = \begin{bmatrix} \epsilon_1 & 0 & 0 & \cdots & 0 \\ 0 & \epsilon_2 & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ 0 & 0 & \cdots & 0 & \epsilon_K \end{bmatrix}$$

1.6 CHARGE DENSITY

The probability of finding an electron in a volume element dr at point r is $|\psi_a(r)|^2\, dr$.

Therefore, the total charge density,

$$\rho(r) = 2 \sum_{a=1}^{N/2} |\psi_a(r)|^2$$

and

$$\begin{aligned} \int \rho(r)\, dr &= 2 \sum_{a}^{N/2} \int dr |\psi_a(r)|^2 \\ &= 2 \sum_{a}^{N/2} 1 \\ &= N \end{aligned}$$

Inserting the basis functions 1.7 into the above gives,

$$\rho(r) = \sum_{\mu\nu} P_{\mu\nu}\phi_\mu(r)\phi^*_\nu(r)$$

where,

$$P_{\mu\nu} = 2\sum_{a}^{N/2} C^*_{\mu a}C^*_{\nu a}$$

is called the density matrix.

Since P is directly related to the expansion coefficients C, the spatial orbitals can be characterised by either C or P.

1.7 ORTHOGONALISATION OF BASIS: SOLVING THE ROOTHAN EQUATIONS

If the basis were orthogonal, S would be an identity matrix and the Roothan equations 1.9 would become,

$$FC = C\epsilon$$

which is just a usual matrix eigenvalue problem.

Because of the non-orthogonality, we need a way to solve the pseudo-eigenvalue problem,

$$FC = SC\epsilon$$

This is done by orthogonalising the basis functions.

We need to find a transformation matrix X, such that,

$$\phi'_\mu = \sum_\nu X_{\nu\mu}\phi_\nu,$$

$$\mu = 1, 2, \ldots, K.$$

forms an orthonormal set.

i.e.,

$$\int dr \phi'^{*}_{\mu}(r)\phi'_{\nu}(r) = \delta_{\mu\nu}$$

Now,

$$\begin{aligned}\int dr \phi'^{*}_{\mu}(r)\phi'_{\nu}(r) &= \int dr \left[\sum_{\lambda} X^{*}_{\lambda\mu}\phi^{*}_{\lambda}(r)\right]\left[\sum_{\sigma} X_{\sigma\nu}\phi_{\sigma}(r)\right] \\ &= \sum_{\lambda}\sum_{\sigma} X^{*}_{\lambda\mu}\int dr \phi^{*}_{\lambda}(r)\phi_{\sigma}(r)X_{\sigma\nu} \\ &= \sum_{\lambda}\sum_{\sigma} X^{*}_{\lambda\mu}S_{\lambda\sigma}X_{\sigma\nu} \\ &= \delta_{\mu\nu}\end{aligned}$$

or,

$$X^{\dagger}SX = \mathbb{1} \tag{1.10}$$

Since S is Hermitian, it can be diagonalised by a unitary matrix U,

$$U^{\dagger}SU = s$$

where, s is the diagonal matrix of eigenvalues of S.

There are two common ways of orthogonalising the basis set:

1. Symmetric orthogonalisation

   $$X \equiv S^{-1/2} = Us^{-1/2}U^{\dagger}$$

   This leads to a problem for basis sets with linear dependence and hence, is not used.

2. Canonical orthogonalisation

   $$X \equiv Us^{-1/2}$$

   that is, the coulumns of U are divided by the square root of the corresponding eigenvalues.

   Now, we can eliminate columns corresponding to eigenvalues equal to zero and circumvent the problem discussed in symmetric orthogonalisation.

Now, considering

$$C' = X^{-1}C$$
$$C = XC' \tag{1.11}$$

$$FC = SC\epsilon$$

becomes

$$FXC' = SXC'\epsilon$$

Multiplying by $X^{\dagger}$ on the left, we get

$$(X^{\dagger}FX)C' = (X^{\dagger}SX)C'\epsilon$$

Defining $F' = X^{\dagger}FX$ and using equation 1.10, we get,

$$F'C' = C'\epsilon$$

These are the transformed Roothan equations.

These can be solved by simply diagonalising F'. Once C' is obtained, C can be calculated using equation 1.11.

1.8 THE SELF CONSISTENT FIELD PROCEDURE

Since the Fock operator itself depends on the expansion coefficients, the Roothan equations are non-linear and have to solved iteratively till there is self-consistency.

The SCF procedure can be depicted by the following flowchart:

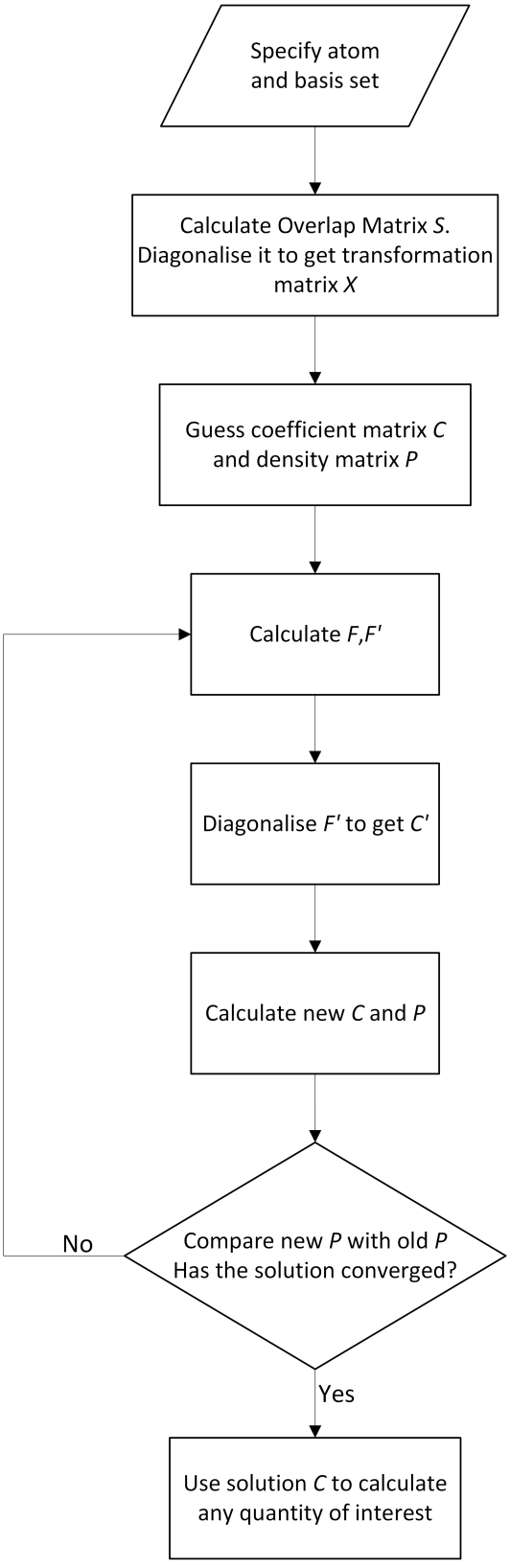


Figure 1.1: The SCF Procedure

The density matrices can be compared by calculating the standard deviation of successive density matrix elements.

1.9 TOTAL ENERGY

$$E_0 = 2\sum_a h_{aa} + \sum_a \sum_b 2J_{ab} - K_{ab}$$

Using equation 1.6, we get,

$$E_0 = \sum_a^{N/2} (h_{aa} + f_{aa})$$

Substituting the basis function expansion from equation 1.7, we get,

$$E_0 = \frac{1}{2}\sum_\mu \sum_\nu P_{\nu\mu}(h_{\mu\nu} + F_{\mu\nu}) \tag{1.12}$$

CHAPTER 2

GAUSSIAN BASIS SET EXPRESSIONS

2.1 BASIS

In the central field approximation, the solution to the Dirac equation is given by four component spinors

$$\Psi_{n\kappa m}(r,\theta,\phi) = r^{-1}\begin{pmatrix} P_{n\kappa}(r) & \chi_{\kappa m}(\theta,\phi) \\ iQ_{n\kappa}(r) & \chi_{-\kappa m}(\theta,\phi)\end{pmatrix}$$

where $P_{n\kappa}(r)$ and $Q_{n\kappa}(r)$ are the large and small components of the radial wave function. They satisfy the orthonormality condition,

$$\int_0^\infty [P_{n\kappa}(r)P_{n'\kappa}(r) + Q_{n\kappa}(r)Q_{n'\kappa}(r)]\, dr = \delta_{nn'}$$

The spinor $\chi_{\kappa m}(\theta,\phi)$ is given by,

$$\chi_{\kappa m} = \sum_\sigma Y_{l,m-\sigma}(\theta,\phi)\phi^\sigma c(l,1/2,j;m-\sigma,\sigma)$$

The angular state is characterised by the quantum number κ given by

$$\kappa = -2(j-l)(j+1/2)$$

where l is the orbital quantum number and $j = l \pm 1/2$ is the total angular momentum number, c are the Clebsch-Gordon coefficients and ϕ^σ are the two component spinors,

$$\phi^{1/2} = \begin{pmatrix}1\\0\end{pmatrix}, \phi^{-1/2} = \begin{pmatrix}0\\1\end{pmatrix}$$

The radial functions are expanded in terms of the N basis functions as

$$P_{n\kappa}(r) = \sum_p^N C^L_{\kappa p} g^L_{\kappa p}(r)$$

and

$$Q_{n\kappa}(r) = \sum_{p}^{N} C_{\kappa p}^{S} g_{\kappa p}^{S}(r)$$

For the large component, we have taken Gaussian Type Orbitals (GTOs) of the form:

$$g_{\kappa i}^{L}(r) = C_{N_{\kappa i}}^{L} r^{n_\kappa} e^{-\alpha_i r^2}$$

where

Table 2.1: Values of n_κ and κ for different symmetries

Symmetry	$s_{1/2}$	$p_{1/2}$	$p_{3/2}$	$d_{3/2}$	$d_{5/2}$
n_κ	1	2	2	3	3
κ	-1	$+1$	-2	$+2$	-3

The normalisation constants are

$$C_{N_{\kappa i}}^{L} = \left[\frac{\sqrt{\pi}}{2^{2n_\kappa+3/2}} \frac{(2n_\kappa - 1)!!}{\alpha_i^{\frac{2n_\kappa+1}{2}}} \right]^{-1/2}$$

Using the constraint on the small component using the kinetic balance condition to prevent variational collapse, we get,

$$g_{\kappa i}^{S}(r) = C_{N_{\kappa i}}^{S} \left(\frac{d}{dr} + \frac{\kappa}{r} \right) g_{\kappa i}^{L}(r)$$

where,

$$C_{N_{\kappa i}}^{S} = \left\{ \frac{\alpha_i}{2n_\kappa - 1} \left[(4\kappa^2 - 1) + 4(n_\kappa + \kappa) \right] \right\}^{-1/2}$$

The exponents $\alpha_i' s$ are generated according to

$$\alpha_i = \alpha_0 \beta^{i-1}, \qquad i = 1, 2, \ldots, N.$$

where α_0 and β are input parameters.

2.2 RELATIVISTIC ROOTHAN EQUATIONS

The Fock matrix is of $2N_\kappa \times 2N_\kappa$ dimension. The matrices are block diagonalised according to the symmetry type κ.

$$F_\kappa C_\kappa = \epsilon_\kappa S_\kappa C_\kappa$$

where

$$C_\kappa = \begin{bmatrix} C_\kappa^L \\ C_\kappa^S \end{bmatrix}$$

$$S_\kappa = \begin{bmatrix} S_\kappa^{LL} & 0 \\ 0 & \S_\kappa^{SS} \end{bmatrix}$$

Let the fock operator be $F_\kappa = h_\kappa + g_\kappa$, where

$$h_\kappa = \begin{bmatrix} V_\kappa^{LL} & c\Pi_\kappa^{LS} \\ c\Pi_\kappa^{SL} & V_\kappa^{SS} - 2c^2 S_\kappa^{SS} \end{bmatrix}$$

is the one electron operator, c is the speed of light and

$$g_\kappa = \begin{bmatrix} J_\kappa^{LL} - K_\kappa^{LL} & -K_\kappa^{LS} \\ -K_\kappa^{SL} & J_\kappa^{SS} - K_\kappa^{SS} \end{bmatrix}$$

is the two electron operator.

The various terms in the above matrices are given by the following integrals. I make no attempts to derive these in this dissertation. Refer to Clementi (1991) for derivations and further reading.

The one electron matrix has the elements,

$$S_{\kappa ij}^{TT} = \int_0^\infty g_{\kappa i}^T(r) g_{\kappa j}^T(r)\, dr$$

$$V_{\kappa ij}^{TT} = \int_0^\infty g_{\kappa i}^T(r) V_{nuc}(r) g_{\kappa j}^T(r)\, dr$$

where $V_{nuc}(r)$ is given by equation 1.1 and TT indicates LL or SS.

$$\Pi_{\kappa ij}^{SL} = \int_0^\infty g_{\kappa i}^S(r) \left[\frac{d}{dr} + \frac{\kappa}{r}\right] g_{\kappa j}^L(r)\, dr$$

and

$$\Pi^{LS}_{\kappa ij} = \Pi^{SL\dagger}_{\kappa ij}$$

The two electron matrix has the elements,

$$J^{TT}_{\kappa pq} = \sum_{\kappa' rs} \left[D^{TT}_{\kappa' rs} J^{0TTTT}_{\kappa pq,\kappa' rs} + D^{\bar{T}\bar{T}}_{\kappa' rs} J^{0TT\bar{T}\bar{T}}_{\kappa pq,\kappa' rs} \right]$$

and

$$K^{TT'}_{\kappa pq} = \sum_{\nu} \sum_{\kappa' rs} \left[\bar{b}^{c}_{\nu}(j j') \right] D^{TT'}_{\kappa' rs} K^{\nu TT'TT'}_{\kappa pq,\kappa' rs}$$

where $T\bar{T}$ indicates LS or SL and T, T' are either L or S.

The density matrices are defined as,

$$D^{TT'}_{\kappa pq} = N_{\kappa} C^{T}_{\kappa p} C^{T}_{\kappa q}{}'$$

where $N_{\kappa} = 2j + 1$ for closed shells.

ν is in the range

$$|j_p - j_q| \leq \nu \leq j_p + j_q$$

and the coefficient $\bar{b}^{c}_{\nu}(j j')$ is given by,

$$\bar{b}^{c}_{\nu}(j j') = \begin{bmatrix} j_p & \nu & j_q \\ 1/2 & 0 & -1/2 \end{bmatrix}^2$$

where

$$\begin{bmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{bmatrix}$$

is the Wigner-3j symbol.

The Coulomb integrals are given by,

$$J^{\nu TT,T'T'}_{\kappa pq,\kappa' rs} = \int_0^{\infty} \int_0^{\infty} g^{T}_{\kappa p}(r_1) g^{T}_{\kappa q}(r_1) U_{\nu}(r_1, r_2) g^{T'}_{\kappa' r}(r_2) g^{T'}_{\kappa' s}(r_2)\, dr_1\, dr_2$$

$$K^{\nu TT',TT'}_{\kappa pq,\kappa' rs} = \int_0^{\infty} \int_0^{\infty} g^{T}_{\kappa p}(r_1) g^{T}_{\kappa' r}(r_1) U_{\nu}(r_1, r_2) g^{T'}_{\kappa q}(r_2) g^{T'}_{\kappa' s}(r_2)\, dr_1\, dr_2$$

where,

$$U_\nu(r_1, r_2) = \begin{cases} \frac{r_1^\nu}{r_2^{\nu+1}}, & \text{if } r_1 < r_2 \\ \frac{r_2^\nu}{r_1^{\nu+1}}, & \text{if } r_1 > r_2 \end{cases}$$

CHAPTER 3

CODE CONSIDERATIONS

3.1 INPUT PARAMETERS

1. Atomic number, Atomic mass.
2. Number of symmetries, number of basis and occupied orbitals for each symmetry.
3. Parameters α_0 and β for the Gaussian Type Orbitals.
4. Maximum number of iterations, critical value to establish convergence.

3.2 GRID

We choose a grid which is nearly linear near the origin and exponentially increasing at large r,

$$r(i) = a(e^{h(i-1)} - 1),$$
$$i = 1, 2, \ldots, 740.$$

where $h = 0.03$ and $a = \frac{e^{-65/16}}{z}$.

This is done because such a grid gives the optimum placement of mesh points in the sense of minimizing the error in energy for a given number of degrees of freedom. For more details and derivation, refer to Levine and Wilkins (1991).

For use in calculating various integrals, we also store the following:

$$\Delta r = a e^{h(i-1)}$$

and

$$\frac{\Delta r}{r} = \frac{e^{h(i-1)}}{e^{h(i-1)} - 1}$$

3.3 INTEGRATION

All integrations are done using the 11-point Newton-Cotes method. The grid is divided into 11-point segments and area of each segment is calculated using

$$\int_{x_1}^{x_{11}} f(x)\,dx = \frac{5}{299376}h\,[16067(f_1 + f_{11}) + 106300(f_2 + f_{10}) - 48525(f_3 + f_9) \\ +272400(f_4 + f_8) - 260550(f_5 + f_7) + 427368f_6]$$

where h is the distance between two consecutive points.

3.4 WIGNER-3J

I am using Sovkov (2022) to calculate the Wigner-3j symbol. It is freely available to download at Matlab File Exchange.

3.5 SCF PROCEDURE

The first guess for the density matrix is generally taken to be $P = 0$. This corresponds to neglecting the electron electron interaction and is therefore a logical guess to make. This also means that there is no need to calculate the two electron operator during the first iteration.

The difference between consecutive density matrices is calculated as

$$\delta = \left[K^{-2}\sum_{\mu}\sum_{\nu}\left[P_{\mu\nu}^{(i)} - P_{\mu\nu}^{(i-1)}\right]^2\right]^{1/2}$$

where $K = 2$.

It is generally seen according to Szabo and Ostlund (1996) that a value of $\delta = 10^{-4}$ will usually give an error in the energy of less than 10^{-6} Hartrees.

If the maximum number of iterations are exceeded without convergence, the code stops. At the end of the code, I only try to calculate the total energy as given by equation 1.12 to no avail however, as discussed in the next chapter.

The complete Matlab code is attached in Appendix A.

CHAPTER 4

FAILURE AND SUGGESTIONS

I have been unable to get the SCF Procedure to converge. Therefore, I didn't add any more features to the code as planned earlier.

A second problem is that calculation of the double integrals takes too long. Some ways of approximating the inner integral at the cost of accuracy should be added as in Nataraj (2021*a*).

Some suggestions for anyone who builds upon my work are as follows:

1. Some working code should be used to generate a solution very close to convergence. This should then be used as an initial guess to check the code.

2. If that works, some advanced methods of accelerating convergence, for example in Schlegel (1991), may be looked at.

3. If the speed of calculating double integrals is increased, atoms with high atomic number should be used. In this case, our approximate solutions are closer to the exact solution as the approximation of an average potential due to all the electrons is more valid.

APPENDIX A

CODE

A.1 MAIN

```
% Input Values
%
%   (a) atom : symbol of atom in two capital letters
%
%   (b) nsym:  no. of symmetry to be used
%
%   (c) nbas(i) :  no. of basis for each symmetry
%
%   (d) nocorb(i) : no. of occupied orbitals for each symmetry
%
%   (e) alpha and beta : values for generating basis
%
%   (f) maxit : max. no. of iterations it should check for
 ↪  convergence
%
%   (g) npower : tolerence parameter for 10^(-npower)
%
%   (h) amass : mass number
%
%   (i) z : atomic number

%-----------------------------------------------------------------------
% the output files are:
 ↪
%
 ↪
%     scfout.out:  contains all information of orbital energies
 ↪
%     wfn.dat  :   binary file containing df wave functions
 ↪
%
 ↪
%-----------------------------------------------------------------------


% this is the tiny number to check density consistency in each
 ↪  iteration
global tiny
```

```
tiny = 1.0e-12;

readinp();
setgrd();
setbasis();
setmatrix();
scfiter();
```

A.2 READ INPUT

```
function readinp
% This subroutine reads input from scfinp.dat
global atom stdout stdin iwfn nsym nbas nocorb alpha0 beta maxit
 ↪  npower ...
    orbj kap amass z nbasis nocc mbasis crit h n rnt rrms rho

stdout = fopen('scfout.out','w');
stdin = fopen('scfinp.dat','r');
iwfn = fopen('wfn.dat','w');

% read atom
atom = fgetl(stdin);

% read number of symmetries
nsym = str2double(fgetl(stdin));

% read number of basis for each symmetry in the form of space
 ↪  separated values
nbas = str2num(fgetl (stdin));

% read number of occupied orbitals for each symmetry in the form
 ↪  of space separated values
nocorb = str2num(fgetl (stdin));

% total number of basis
nbasis=sum(nbas);

mbasis=2*nbasis;

% read constants to generate exponents
alpha0 = str2double(fgetl (stdin));
beta = str2double(fgetl (stdin));

% read maximum number of iterations
maxit = str2double(fgetl (stdin));
% read tolerance
```

```
npower = str2double(fgetl (stdin));
% read atomic mass
amass = str2double(fgetl (stdin));
% read atomic number
z = str2double(fgetl (stdin));
% total number of occupied orbitals
nocc=sum(nocorb);
% values of j for each symmetry
orbj = [0.5 0.5 1.5 1.5 2.5 2.5 3.5 3.5 4.5 4.5 5.5 5.5 6.5 6.5
 ↪  7.5];
% values of kappa for each symmetry
kap = [-1 1 -2 2 -3 3 -4 4 -5 5 -6 6 -7 7 -8];
% number of values in the grid along r
n = 740;
% value of h to generate exponential grid
h = 3e-2;
%%h = 739*0.03/(n-1)
% critical value to end iteration procedure
crit=(0.1)^npower;

cp=2.2677*1e-5*amass^(1/3);
rrms=sqrt(3/5)*cp;

rnt= exp(-65/16)/z;
rho(1:n) = z; % value for point nucleus

fprintf(stdout,'                        ^^^^*
 ↪  \n');
fprintf(stdout,'<<<<<<<<<<<<<<<<<<<<<<<<o>>>>>>>>>>>>>>>>>>>>>>>>>\n');
fprintf(stdout,'                        ^^^^*
 ↪  \n');
fprintf(stdout,'scf run for df wave function\n');
fprintf(stdout,atom);
fprintf(stdout,'\n');
fprintf(stdout,'^^^^^^^^^^^^^^^^^^\n');
fprintf(stdout,'*                                   *\n');
fprintf(stdout,'*                                   *\n');
fprintf(stdout,'*                                   *\n');
fprintf(stdout,'*       relativistic scf run        *\n');
fprintf(stdout,'*                                   *\n');
fprintf(stdout,'*           written by              *\n');
fprintf(stdout,'*                                   *\n');
fprintf(stdout,'*          Laksh Arora              *\n');
fprintf(stdout,'*                                   *\n');
fprintf(stdout,'*                                   *\n');
fprintf(stdout,'*                                   *\n');
```

```
fprintf(stdout,'^^^^^^^^^^^^^^^^^^^\n');
fprintf(stdout,' gaussian type of orbitals chosen\n');
fprintf(stdout,' point nucleus is considered          \n');
fprintf(stdout,' neutral atom has been considered     \n');
fprintf(stdout,'\n');
fprintf(stdout,'>>>>>>>>>>>>>>> input data
 ↪  >>>>>>>>>>>>>>>>>>>>>>>>\n');
fprintf(stdout,'number of symmetries = %2d\n',nsym);
fprintf(stdout,'nbas -->');
fprintf(stdout,char(strjoin(string(nbas))));
fprintf(stdout,'\nocc orb  --> ');
fprintf(stdout,char(strjoin(string(nocorb))));
fprintf(stdout,'\n');
fprintf(stdout,'basis type :Universal Basis\n');
fprintf(stdout,'alpha0 = %10.4f beta = %10.4f\n',alpha0,beta);
fprintf(stdout,'max iter = %d conv. = %10.4f\n',maxit, crit);
fprintf(stdout,'atomic number %10.4f and mass =
 ↪  %10.4f\n',z,amass);
fprintf(stdout,'<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<\n');
fprintf(stdout,'nuclear r0 = %f  rrms = %f',rnt,rrms);

end
```

A.3 SET GRID

```
function setgrd
%ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
%cc  this subroutine defines the grids
 ↪  cc
%ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

global stdout rp rpor rnt h n r

eph = exp(h);

%c
 ↪  ----------------------------------------------------------------------
%c   set up the arrays r, rp, rpor
%c
 ↪  ----------------------------------------------------------------------

ett = eph.^(0:n-1);
ettm1 = ett - 1;
r = rnt*ettm1;
rp = rnt*ett;
rpor = ett./ettm1;
```

```
rpor(1) = 0;

fprintf(stdout,'stepsize=%f total grids= %d\n',h,n);
fprintf(stdout,'maximum radius r(n) = %f\n',r(n));

end
```

A.4 FACTORIAL

```
%c ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
%c function idbf
%c ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
%c     calculates factorial
%c-------------------------------------------------------------------------
function fac = idbf(x,i)
k = 1;
for m = x
    if(m<-1)
        fprintf(stdout,'factorial problem: m<-1 %d',m);
        return

    else

        if(m==0 || m==-1)

            if(m==0)
                tmp=1;
            end

            if(m==-1)
                if(i==2)
                    tmp=1;
                else
                    fprintf('factorial problem: m<0 %d',m);
                    return
                end
            end

        else

            tmp=1;

            for j=m:-i:1
                tmp=j*tmp;
            end
```

```
        end

    end
fac(k) = tmp;
k = k+1;
end
end
```

A.5 SET BASIS

```
function setbasis
% This subroutine calculates the GTOs
global stdout nsym nbas nocorb alpha1 beta1 orbj kap r nbasis gl
 ↪  gs dgl ...
    dg1 dg2 kappe orbje kappc orbjc iqe iqc lorba lorbb nskipe
 ↪  nskipc alpha

lorba=[0 1 1 2 2 3 3 4 4 5 5 6 6 7 7];
lorbb=[1 0 2 1 3 2 4 3 5 4 6 5 7 6 8];
nk = lorba+1;
nskipe = [0 cumsum(nbas)];

for isym=1:nsym
    ii = nskipe(isym);
    nn = nbas(isym);
    if (nn~=0)
        kappe(ii+1:ii+nn)=kap(isym);
        orbje(ii+1:ii+nn)=orbj(isym);
        nke(ii+1:ii+nn)=nk(isym);
        iqe(ii+1:ii+nn) = (-1)^(isym+1);
        alpha(ii+1:ii+nn) = alpha1.*(beta1.^(0:(nbas(isym)-1)));
    end
end

nskipc = [0 cumsum(nocorb)];

for isym=1:nsym
    ii = nskipc(isym);
    nn=nocorb(isym);
    if(nn~=0)
        kappc(ii+1:ii+nn)=kap(isym);
        orbjc(ii+1:ii+nn)=orbj(isym);
        iqc(ii+1:ii+nn) = (-1)^(isym+1);
    end
end
```

```
fprintf(stdout,' total number of basis function= %d',nbasis);
fprintf(stdout,'symmetry   no. of occ orbits   nskipc');

for isym=1:nsym
    fprintf(stdout,'%d            %d
↪  %d',isym,nocorb(isym),nskipc(isym));
end

fprintf(stdout,'symmetry   no. of basis function   nskipe');
for isym=1:nsym
    fprintf(stdout,'%d            %d
↪  %d',isym,nbas(isym),nskipe(isym));
end

ifact=idbf(2.*nke-1,2);
power=(2.*nke+1)./2;
cnl=sqrt((2.^(2.*nke+1.5)).*(alpha.^power)./(ifact.*sqrt(pi)));
fact1=4.*alpha.*((nke+kappe).^2)./(2.*nke-1);
fact2=(2.*nke+1).*alpha;
fact3=-4.*alpha.*(nke+kappe);
cns=sqrt(1./(fact1+fact2+fact3));

expon= exp(-(alpha.').*r.*r);

gl=(cnl.').*expon.*(r.^(nke.'));
gs=(cns.').*gl.*((nke.'+kappe.')./r - 2.*(alpha.').*r);
gs(:,1)=0;
dgl=gl.*((nke.'+kappe.')./r - 2.*(alpha.').*r);
dgl(:,1)=0;
dg1=((nke.')./r) - (2.*(alpha.').*r);
dg1(:,1)=0;
dg2=-(((nke.'+kappe.')./(r.^2))+2.*(alpha.'))./...
    ((nke.'+kappe.')./r-2.*(alpha.').*r);
dg2(:,1)=0;

end
```

A.6 INTEGRAL

```
function res = valint(fx)
%c ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
%c    this is the integration routine using 11-point Newton-Cotes
%c ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
global h n
```

```
res=0;

for i=n+1:n+10
    fx(i)=0;
end

for i=1:10:n
    res=res+(16067*(fx(i)+fx(i+10))+427368*fx(i+5) ...
        +106300*(fx(i+1)+fx(i+9))-48525*(fx(i+2)+fx(i+8)) ...
        +272400*(fx(i+3)+fx(i+7))-260550*(fx(i+4)+fx(i+6)));
end

res=5.0*h*(res/299376.0);

end
```

A.7 ONE ELECTRON OPERATOR

```
function setmatrix
%c ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
%c  this subroutine calculates the one electron Fock operator
%c
 ↪  ----------------------------------------------------------------------
global   rho ...
    nsym nbas mbasis...
    rp rpor  ...
     c ...
    gl gs dgl nskipe ...
    S df_single
Sll(mbasis,mbasis)=0;
Sss(mbasis,mbasis)=0;
PI(mbasis,mbasis)=0;
V(mbasis,mbasis)=0;

for isym=1:nsym
    for ia=1:nbas(isym)
        ja=ia+nskipe(isym);
        ka=ia+2*nskipe(isym);
        index1=ia+nbas(isym)+2*nskipe(isym);
        for ib=1:nbas(isym)
            jb=ib+nskipe(isym);
            kb=ib+2*nskipe(isym);
            index2=ib+nbas(isym)+2*nskipe(isym);

            Sll(ka,kb)=valint(gl(ja,:).*gl(jb,:).*rp);
            Sss(index1,index2)=valint(gs(ja,:).*gs(jb,:).*rp);
```

```
            PI(index1,kb)=valint(gs(ja,:).*dgl(jb,:).*rp);

            V(ka,kb)=valint(-gl(ja,:).*gl(jb,:).*rpor.*rho);

 ↪  V(index1,index2)=valint(-gs(ja,:).*gs(jb,:).*rpor.*rho);
        end
    end
end

S = Sll + Sss;
PI = PI + PI';

df_single = V + c*PI - 2*c*c*Sss;
```

A.8 INNER INTEGRAL

```
function result = valint2(v,f,g)
% This calculates the inner integral in double integrals
global n r rp

result(n) = 0;
u(n) = 0;
for i = 1:n
    r1 = r(i);
    for j =1:n
        r2 = r(j);
        if (j<i)
            u(j) = r2^v/r1^(v+1);
        elseif (j==i)
            u(j) = 1/r1;
        else
            u(j) = r1^v/r2^(v+1);
        end
    end
    result(i) = valint(u.*f.*g.*rp);
end
```

A.9 TWO ELECTRON OPERATOR

```
function dfpotiter()
%c ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
%c this evaluates the two electron fock operator.
%c p(i,j) is the density matrix.
%c-------------------------------------------------------------------
global nsym nbas nocorb ...
```

```
    nskipe nskipc ...
    df_two orbjc vec mbasis gl gs orbje rp

Jll(mbasis,mbasis)=0;
Jss(mbasis,mbasis)=0;
Kll(mbasis,mbasis)=0;
Kss(mbasis,mbasis)=0;
Kls(mbasis,mbasis)=0;
Ksl(mbasis,mbasis)=0;

for isym=1:nsym
    for ip=1:nbas(isym)

        jp=ip+nskipe(isym);
        kp=ip+2*nskipe(isym);
        indexp=ip+nbas(isym)+2*nskipe(isym);
        j1 = orbje(jp);

        for iq=1:nbas(isym)

            jq=iq+nskipe(isym);
            kq=iq+2*nskipe(isym);
            indexq=iq+nbas(isym)+2*nskipe(isym);
            j2 = orbje(jq);

            for jsym=1:nsym
                for ik=1:nocorb(jsym)

                    jk=ik+nskipc(jsym);
                    kk=ik+2*nskipe(jsym);

                    Nk = 2*orbjc(jk)+1;

                    for ir=1:nbas(jsym)

                        jr=ir+nskipe(jsym);
                        kr=ir+2*nskipe(jsym);
                        indexr=ir+nbas(jsym)+2*nskipe(jsym);

                        for is=1:nbas(jsym)

                            js=is+nskipe(jsym);
                            ks=is+2*nskipe(jsym);
                            indexs=is+nbas(jsym)+2*nskipe(jsym);

                            Dll = Nk*vec(kr,kk)*vec(ks,kk);
```

```
                            Dss =
↪ Nk*vec(indexr,kk)*vec(indexs,kk);
                            Dls = Nk*vec(kr,kk)*vec(indexs,kk);
                            Dsl = Nk*vec(indexr,kk)*vec(ks,kk);

                            Jllll =
↪ valint(gl(jp,:).*gl(jq,:).*valint2(0,gl(jr,:),gl(js,:)).*rp);
                            Jllss =
↪ valint(gl(jp,:).*gl(jq,:).*valint2(0,gs(jr,:),gs(js,:)).*rp);
                            Jssss =
↪ valint(gs(jp,:).*gs(jq,:).*valint2(0,gs(jr,:),gs(js,:)).*rp);
                            Jssll =
↪ valint(gs(jp,:).*gs(jq,:).*valint2(0,gl(jr,:),gl(js,:)).*rp);


↪ Jll(kp,kq)=Jll(kp,kq)+Dll*Jllll+Dss*Jllss;

↪ Jss(indexp,indexq)=Jss(indexp,indexq)+Dss*Jssss+Dll*Jssll;

                            for v=abs(j1-j2):(j1+j2)

                                factor =
↪ Wigner3j(j1,v,j2,0.5,0,-0.5)^2;
                                %Vladimir Sovkov (2022). Wigner
↪ 3j-6j-9j ...

↪ %https://www.mathworks.com/matlabcentral/fileexchange/74069-wigner-3j-6j-9j

                                Kllll =
↪ valint(gl(jp,:).*gl(jr,:).*valint2(v,gl(jq,:),gl(js,:)).*rp);
                                Kssss =
↪ valint(gs(jp,:).*gs(jr,:).*valint2(v,gs(jq,:),gs(js,:)).*rp);
                                Klsls =
↪ valint(gl(jp,:).*gl(jr,:).*valint2(v,gs(jq,:),gs(js,:)).*rp);
                                Kslsl =
↪ valint(gs(jp,:).*gs(jr,:).*valint2(v,gl(jq,:),gl(js,:)).*rp);


↪ Kll(kp,kq)=Kll(kp,kq)+factor*Dll*Kllll;

↪ Kss(indexp,indexq)=Kss(indexp,indexq)+factor*Dss*Kssss;

↪ Kls(kp,indexq)=Kls(kp,indexq)+factor*Dls*Klsls;

↪ Ksl(indexp,kq)=Ksl(indexp,kq)+factor*Dsl*Kslsl;
```

```
                        end
                    end
                end
            end
        end
    end
end
J = Jll + Jss;
K = Kll + Kss + Ksl + Kls;
df_two = J-K;
```

A.10 SCF PROCEDURE

```
%c ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
%c subroutine scfiter
%c ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
%c     this performs the scf iteration.
%c--------------------------------------------------------------------
function scfiter

global stdout     ...
    tiny      ...
    nsym nbas nocorb  delta  ...
    crit maxit npower orbj  ...
    mbasis ...
    nskipe  ...
    S df_single df_two ...
    en sa drs vec

nh_i = [' S ', ' P-', ' P ', ' D-', ' D ', ' F-', ' F ' ,' G-' ,'
↪  G ', ' H-', ' H ', ...
    ' I-', ' I ', ' K-',  ' K '];
oe(mbasis,nsym)=0;

iter=0;
p1(mbasis,mbasis)=0;

fprintf(stdout,['convergence data\n maximum no. of iteration= %6d
↪  ...' ...
    'convergence criterion   = 1.0d-%2d\n'],maxit,npower);
fprintf(stdout,'cycle      density conv\n');

while(iter<maxit)
    iter=iter+1;
    if(iter~=1)
```

```
        dfpotiter();
    end
    for isym=1:nsym
        ndim=nbas(isym);
        ndim2=2*ndim;
        skip = 2*nskipe(isym);
        stemp = S(1+skip:ndim2+skip,1+skip:ndim2+skip);

        drs = stemp;

        [drs,temp] = eig(drs);
        [sa,ind] = sort(diag(temp),'descend');
        drs = drs(:,ind);


↪   %c--------------------------------------------------------------------
        %c       canonical transformation

↪   %c--------------------------------------------------------------------

        if (min(abs(sa))>tiny)
            can = drs./sqrt(abs(sa'))
        else
            fprintf('there is diagonalisation problem
↪   %f\n\n',min(abs(sa)));
            return
        end

        if(iter~=1)
            a=df_single(1+skip:ndim2+skip,1+skip:ndim2+skip)+...
                df_two(1+skip:ndim2+skip,1+skip:ndim2+skip)
        else
            a=df_single(1+skip:ndim2+skip,1+skip:ndim2+skip)
        end

        df_mat = can'*a*can

        [df_mat,temp] = eig(df_mat)
        [eng,ind] = sort(diag(temp),'ascend')
        df_mat = df_mat(:,ind)

        drs = df_mat

        eigv = can*drs

        oe(1:ndim2,isym)=eng
```

```
        pold(1+skip:ndim2+skip,1+skip:ndim2+skip) =
↪   p1(1+skip:ndim2+skip,1+skip:ndim2+skip);

        nn=nocorb(isym);
        if(nn~=0)
            p1(1+skip:ndim2+skip,1+skip:ndim2+skip) =
↪   (2*orbj(isym)+1)*eigv(:,1:nn)*eigv(:,1:nn)'
        end

        vec(1+skip:ndim2+skip,1+skip:ndim2+skip)=eigv

    end
    delta=sum(sum((p1-pold).^2))

    delta=sqrt(delta/4.0)

    fprintf(stdout,'%3d   %20.7f\n',iter,delta);
    fprintf('%3d   %20.7f\n',iter,delta);
    if(delta<crit)
        break
    end
end
if (iter>=maxit)
    fprintf(stdout,'\n\n scf fails to converges at cycle
↪   %4d\n',iter);
    return
end

en = 0.5*sum(sum(p1.*(2*df_single+df_two)))

fprintf(stdout,'\n\nscf converges at cycle, %4d\n',iter);
fprintf(stdout,'\n\nscf electronic energy=, %25.15f\n',en);

fprintf(stdout,'\n\norbital energies (+ve and -ve)\n');

for isym=1:nsym
    for ibas=1:nbas(jsym)
        fprintf(stdout,['%2d %s %25.15f
↪   %25.15f\n',ibas,nh_i(isym), ...
            oe(ibas,isym),oe(ibas+nbas(jsym),isym)]);
    end
end

end
```

A.11 SAMPLE INPUT

```
RB
11
8 6 6 4 4 3 3 2 2 1 1
4 3 3 1 1 0 0 0 0 0 0
0.00625
2.63
25
4
85.4678
37
```

REFERENCES

1. **Clementi, E.** (1991). Modern techniques in computational chemistry: Motecc-91.

2. **Levine, Z. H.** and **J. W. Wilkins** (1991). An energy-minimizing mesh for the schrödinger equation. *Journal of Computational Physics*, **83**, 361–372. ISSN 0021-9991. URL https://www.sciencedirect.com/science/article/pii/0021999189901241.

3. **Nataraj, H. S.** (2021*a*). Fortran code df_scf.f90.

4. **Nataraj, H. S.** (2021*b*). Unit-1 many electron atoms. PHN-626 Advanced Atomic and Molecular Physics, IIT Roorkee.

5. **Schlegel, J., H.B.and McDouall** (1991). Do you have scf stability and convergence problems? *Computational Advances in Organic Chemistry: Molecular Structure and Reactivity*, **330**. URL https://link.springer.com/chapter/10.1007/978-94-011-3262-6_2.

6. **Sovkov, V.** (2022). Wigner 3j-6j-9j. URL https://www.mathworks.com/matlabcentral/fileexchange/74069-wigner-3j-6j-9j.

7. **Szabo, A.** and **N. S. Ostlund**, *Modern quantum chemistry: Introduction to advanced electronic structure theory*. Dover Publications, inc., 1996. ISBN 9780486691862.