國立交通大學

應用數學系

碩 士 論 文

氫分子和水分子基態能量的 Ab Initio 計算

Ab Initio computation for ground state energies of H_2 and H_2O

研 究 生：陳相宇

指導教授：葉立明　教授

中 華 民 國 九 十 八 年 七 月

氫分子和水分子基態能量的 Ab Initio 計算

Ab Initio computation for ground state energies of H_2 and H_2O

研 究 生：陳相宇　　Student：Shiang-Yen Chen

指導教授：葉立明　　Advisor：Li-Ming Yeh

國 立 交 通 大 學

應 用 數 學 系

碩 士 論 文

A Thesis

Submitted to Department of Applied Mathematics

College of Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Applied Mathematics

July 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年七月

氫分子和水分子基態能量的 Ab Initio 計算

學生：陳相宇　　指導教授：葉立明　教授

國立交通大學應用數學學系（研究所）碩士班

摘　　要

以 Ab initio 方法來計算氫分子和水分子的基態能量。我們計算能量的理論基礎是密度泛函理論(DFT)和局部密度近似法(LDA)，我們首先以原子軌道為基底函數，並對他們做正交歸一化，接著再求每一項能量密度泛函的值，最後運用 Lagrange multipliers 對總密度能量泛函做最小化，再以最速下降法求得基態總能。

Ab Initio computation for ground state energies of H_2 and H_2O

student：Shiang-Yen Chen　　Advisors：Dr. Li-Ming Yeh

Department of Applied Mathematics

National Chiao Tung University

ABSTRACT

Computation of the ground-state energies of H_2 and H_2O is concerned. Based on the density functional theory (DFT) the ground state energy of a many-electron system can be expressed by electronic density and is the minimum value of a density functional. So the computation can be done by finding a minimum value of a total energy density functional. Firstly, we use linear combination of atomic orbitals (LCAO) to construct the molecular orbital and calculate each term of the density functional, for example, kinetic energy, electron-electron Coulomb energy, electron-core Coulomb energy, and exchange-correlation energy. Next, we take an initial guess and minimize the total density energy functional by using Largrange multipliers method. Finally by steepest descent method, the ground state total energy is obtained. In some special care is needed for the computation of Coulomb potential to avoid the singular points.

誌　　謝

首先我要感謝我的指導教授-葉立明老師。感謝老師每次都細心的為我解釋程式上或是推導上不懂的地方，並且給我許多建議和解決問題的方法，甚至之後為了讓我趕的上口試時間，還在回國的當天晚上特地回學校和我討論程式結果，並且花了好多時間幫我修正我的論文，感謝老師煞費苦心的幫助我，讓我能順利完成碩士學位。我還要感謝清大物理系-許貞雄老師，感謝許老師開了許多物理課程，而且還在課後抽空替我解答問題，使我能對我要做的問題有更深的了解。

接著我要感謝我的爸爸媽媽，他們資助我讓我在生活上不用擔憂，感謝我的哥哥在我感到無聊時陪我聊天，並且幫我修正我的論文中的英文語法。還要感謝我的家人們，他們支持我，讓我在大學畢業後能繼續念研究所，而且沒有給我任何壓力。

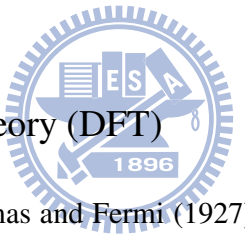
我還要感謝我的朋友們，感謝廖茜婷在我趕論文的時候陪我熬夜並且幫我修正論文中的英文文法。感謝林浙于和楊長銘在這段時間裡，和我一起讀書、吃飯和打電動、並且讓我接觸到基金和股票，使我對理財方面有更深的認識。感謝陳廷彰在預官考試還有論文寫作上提供協助，沒有你們，我的研究所生活一定過的相當乏味。

Contents

Figures

1. Physics

Nearly all physical properties are related to total energies or to the differences between total energies. There is an advantage of Ab Initio total energy calculations. While just one piece of given data is needed to calculate all the physical properties that are related to total energy. The ground-state energy is meaningful in physics which is a stationary state with the lowest energy of a system and it is the most often state of a stable system. The object of this study is to find the ground state total energies of H_2 and H_2O by Ab Initio method.



1.1 Density-functional theory (DFT)

The first mentioned by Thomas and Fermi (1927), they submit a theory said we can use electron density to express energy. But it is coarse of the era even there are some problems in molecular system. The most famous people are Hohenberg and Kohn (1964), they submit the Hohenberg - Kohn (H-K) theorem which contains two main statements of a many-electron system. One is that the ground state properties of a many-electron system are uniquely determined by an electronic density $n(\vec{r})$. And the other is that a energy functional for the system can be defined and it can be proved that the ground state electronic density minimizes this energy functional. Therefore, we can put any guess density $n(\vec{r})$ into the total energy functional $E[n(\vec{r})]$ until generate the minimum value which guarantees the ground-state total energy.

Unfortunately, Density-functional theory (DFT) only proves the existence of the ground-state total energy functional $\mathrm{E_G}[n(\vec{r})]$ but it does not provide the exact form of the total energy functional. In general, the ground-state total energy functional of a many-electron system can be expressed by

$$\mathrm{E_G}[n(\vec{r})] = E_{ec}[n(\vec{r})] + T_m[n(\vec{r})] + E_H[n(\vec{r})]$$

where $E_{ec}[n(\vec{r})] = \int v_{ext}(\vec{r})n(\vec{r})dr$ and $v_{ext}(\vec{r})$ is the external potential (the Coulomb potential is due to the nuclear charges), $\mathrm{T}_m[n(\vec{r})]$ is the kinetic energy and $\mathrm{E}_H[n(\vec{r})]$ is the energy of electron-electron interaction. The last two terms of the density functional are unknown.

One year later, Kohn-Sham (1965) provided a method to derive exact forms for $\mathrm{T}_m[n(\vec{r})]$ and $\mathrm{E}_H[n(\vec{r})]$. They transform the intractable many-body problem of interacting electrons in external potential to a non-interacting electrons in an effective potential. Thus they can rewrite the total energy functional as the sum of the kinetic energy of single-particle $T_0[n(\vec{r})]$, the Coulomb energy between electrons $E_{ee}[n(\vec{r})]$, and the rest (called exchange-correlation energy $E_{xc}[n(\vec{r})]$). Therefore, we have the form of the total energy functional

$$\mathrm{E_G}[n(\vec{r})] = E_{ee}[n(\vec{r})] + T_0[n(\vec{r})] + E_{ec}[n(\vec{r})] + E_{xc}[n(\vec{r})].$$

1.2 Local density approximation (LDA)

The wave functions of electronic orbitals of atomic system are written as :

$$\phi_i(\vec{r}) = R_{pl}(r)Y_{lm}(\theta,\phi)$$

represented by the index *i={p,l,m}*, the main quantum number *p*, the angular momentum quantum number *l*, and the magnetic quantum number *m*. The electron density is

$$n(\vec{r}) = \sum_{i\in occupied} f_i \left|\phi_i(\vec{r})\right|^2$$

For example, O-atom has atomic number of Z=8 and occupies=1S,2S,2P, the density of O-atom is

$$n(\vec{r}) = f_{1S}\left|\phi_{1S}(\vec{r})\right|^2 + f_{2S}\left|\phi_{2S}(\vec{r})\right|^2 + f_{2P}\left|\phi_{2P}(\vec{r})\right|^2 = 2\left|\phi_{1S}(\vec{r})\right|^2 + 2\left|\phi_{2S}(\vec{r})\right|^2 + 4\left|\phi_{2P}(\vec{r})\right|^2$$

In this thesis, we use the atomic unit with $\hbar = m = e = 1$.

Kohn and Sham (1965) provided a simplest method to treatment $\mathrm{E_G}[n(\vec{r})]$. The total energy functional is

$$\begin{aligned}\mathrm{E_G}[n(\vec{r})] &= E_{ee}[n(\vec{r})] + T_0[n(\vec{r})] + E_{ec}[n(\vec{r})] + E_{xc}[n(\vec{r})] \\ &= \frac{1}{2}\iint \frac{n(\vec{r})n(\vec{r})}{\left|\vec{r}-\vec{r}\,'\right|} d\vec{r}d\vec{r}\,' + T_0[n(\vec{r})] + E_{ec}[n(\vec{r})] + E_{xc}[n(\vec{r})] \\ &= \frac{1}{2}\int v_{ee}(\vec{r})n(\vec{r})dr + T_0[n(\vec{r})] + E_{ec}[n(\vec{r})] + E_{xc}[n(\vec{r})] \qquad (1.1)\end{aligned}$$

Here, the electron density satisfies

$$\mathrm{N} = \int n(\vec{r})dr, \qquad (1.2)$$

where N is the total electron number in this system. $E_{ee}[n(\vec{r})]$ is the electron-electron Coulomb energy functional and $v_{ee}(\vec{r})$ is the electron-electron Coulomb potential

$$v_{ee}(\vec{r}) = \int \frac{n(\vec{r}\,')}{\left|\vec{r}-\vec{r}\,'\right|} d\vec{r}\,'.$$

$T_0[n(\vec{r})]$ is the kinetic energy functional expressed in terms of a system of non-interacting electrons (under the assumption of independent particles)

$$T_0[n(\vec{r})] = \sum_{i\in occupied} f_i \int \phi_i^*(\vec{r}) \left(-\frac{\nabla^2}{2} \right) \phi_i(\vec{r}) d\vec{r} = \frac{1}{2} \sum_{i\in occupied} f_i \int \nabla \phi_i^*(\vec{r}) \cdot \nabla \phi_i(\vec{r}) d\vec{r}$$

$E_{ec}[n(\vec{r})]$ is the electron-nuclear Coulomb energy functional

$$E_{ec}[n(\vec{r})] = \int V_{ext}(\vec{r}) n(\vec{r}) d\vec{r}$$

where $v_{ext}(\vec{r}) = -\frac{Z}{\vec{r}}$ is the external potential (Z is an atomic number and the Coulomb potential is due to the nuclear charges) . $E_{xc}([n])$ is the exchange-correlation energy functional

$$E_{xc}([n]) = \int \varepsilon_{xc}(n(\vec{r}))\, n(\vec{r}) d\vec{r} \,,$$

where $\varepsilon_{xc}(n(\vec{r}))$ is the exchange-correlation energy per unit density (more detail of $\varepsilon_{xc}(n(\vec{r}))$ will be explained in section 2.5).

To compute molecular system, we should add one more term E_{cc} which is the core-core Coulomb energy given by

$$E_{cc} = \frac{1}{2} \sum_{I,J} \frac{Z_I Z_J}{|\tau_I - \tau_J|} \,.$$

1.3 Linear combination of atomic orbitals (LCAO)

Linear combination of atomic orbitals is a technique to calculate molecular orbitals.

A mathematical description is

$$\psi = C_1\phi_1 + C_2\phi_2 + \ldots C_n\phi_n = \sum_i C_i\phi_i,$$

where ψ is a molecular orbital, ϕ_i , $i = 1,\ldots,n$, are the atomic orbitals, C_i , $i = 1,\ldots,n,$ are the coefficients corresponding to ϕ_i . Roughly speaking, we shall use atom's wave functions to construct the molecular wave function.

.

2. Process of study

In this study, we want to find the ground-state total energies of H_2 and H_2O. We take the atomic orbitals as the basis functions to construct the molecular orbital. Because these basis functions are at different sites, we have to orthonormalize of them firstly. Next, we take these orthonormal functions to calculate each term of the density functional, for example of kinetic energy, electron-electron Coulomb energy, electron-core energy, and exchange-correlation energy. Then, we write the total energy functional as $F(\mathrm{C})$ where $\mathrm{C} = (C_1, C_2, ..., C_n)$ are the coefficients which corresponded to the basis functions. Finally, we take an initial guess and minimize the total energy functional $F(\mathrm{C})$ by using Lagrange multipliers method and steepest descent method to get the ground state total energy.

2.1 Orthonormalize basis functions

We take the radial wave function $R_{pl}(r)$ of Hydrogen-like atom to be construct our basis function. For example, H_2 has two H-atoms with distance 0.74Å. In Cartesian coordinate, we put the first H atom on the origin and the other H atom on point (0.74 , 0 , 0). Each of them takes five orbital basis functions which are written as $\phi_{(1,0,0)}(\vec{r})$, $\phi_{(2,0,0)}(\vec{r})$, $\phi_{(2,1,-1)}(\vec{r})$, $\phi_{(2,1,0)}(\vec{r})$, $\phi_{(2,1,1)}(\vec{r})$, therefore, we use ten basis functions to compute H_2. In the same way, H_2O has one O-atom and two H-atoms (the position as illustrated in Figure 1.) O-atom takes six orbital basis functions and each of H-atoms takes two orbital basis functions, therefore, we use ten basis functions to compute H_2O .

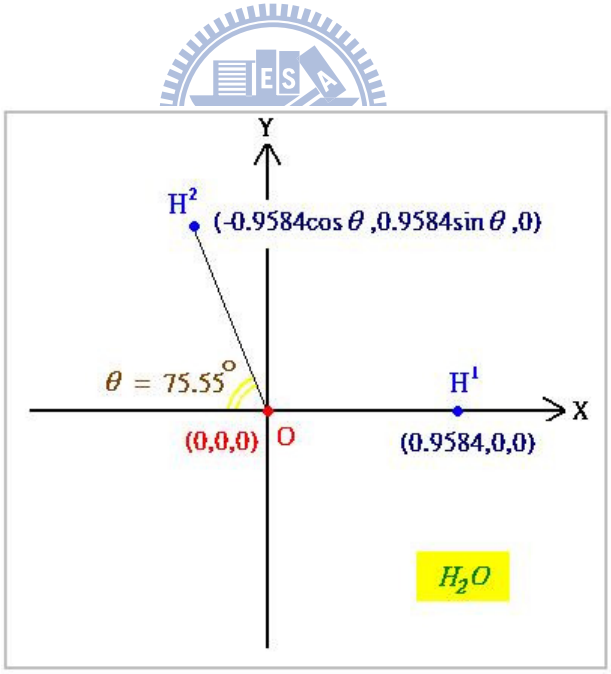


Figure 1 : The position of atoms in H_2O

Because the atoms are at different sites, we use Gram-Schmidt method to these basis functions. By the Gram-Schmidt method,

$$\phi_i'(\vec{r}) = \phi_i(\vec{r}) - \sum_{j=1}^{i-1} \langle \phi_j(\vec{r}) \mid \phi_i(\vec{r}) \rangle \, \phi_j(\vec{r})$$

$$\phi_i''(\vec{r}) = \frac{\phi_i'}{\|\phi_i'\|} \quad \text{where} \quad \|\phi_i'\| = \langle \phi_i' \mid \phi_i' \rangle^{\frac{1}{2}}$$

For convenience, the orthonormalized basis functions are denoted by

$$\phi_1(\vec{r}), \phi_2(\vec{r}), \ldots, \phi_{10}(\vec{r}) .$$

Then, by linear combination of the above orthonormal basis functions, we can construct the wave function by

$$\psi(\vec{r}) = \sum_i C_i \phi_i(\vec{r}) \tag{2.1}$$

The density of molecule is

$$n(\vec{r}) = |\psi(\vec{r})|^2 = \left| \sum_i C_i \phi_i(\vec{r}) \right|^2 = \sum_{i,j} C_i C_j \, \phi_i^*(\vec{r}) \phi_j(\vec{r}) \tag{2.2}$$

2.2 Kinetic energy functional $E_{kin}[n(\vec{r})]$

$$E_{kin} = \langle \psi(\vec{r}) \mid \frac{-\nabla^2}{2} \mid \psi(\vec{r}) \rangle$$

$$= \sum_{i,j} C_i^* C_j \langle \phi_i(\vec{r}) \mid \frac{-\nabla^2}{2} \mid \phi_j(\vec{r}) \rangle$$

$$= \frac{1}{2} \sum_{i,j} C_i^* C_j \int \nabla \phi_i^*(\vec{r}) \cdot \nabla \phi_j(\vec{r}) d\vec{r} .$$

Vector $\vec{r} = (x, y, z)$ in rectangular coordinate

.

Then $\nabla \phi_i^*(x, y, z) = (\frac{\partial \phi_i^*}{\partial x}, \frac{\partial \phi_i^*}{\partial y}, \frac{\partial \phi_i^*}{\partial z})$

$$\nabla \phi_j(x,y,z) = (\frac{\partial \phi_j}{\partial x}, \frac{\partial \phi_j}{\partial y}, \frac{\partial \phi_j}{\partial z})$$

$$\nabla \phi_i^*(x,y,z) \cdot \nabla \phi_j(x,y,z) = \frac{\partial \phi_i^*}{\partial x}\frac{\partial \phi_j}{\partial x} + \frac{\partial \phi_i^*}{\partial y}\frac{\partial \phi_j}{\partial y} + \frac{\partial \phi_i^*}{\partial z}\frac{\partial \phi_j}{\partial z}$$

We use the following space discretization :

$$\frac{\partial \phi_i^*}{\partial x} = \frac{\phi_i^*(x+h,y,z) - \phi_i^*(x-h,y,z)}{2h},$$

$$\frac{\partial \phi_j}{\partial x} = \frac{\phi_j(x+h,y,z) - \phi_j(x-h,y,z)}{2h},$$

$$\frac{\partial \phi_i^*}{\partial y} = \frac{\phi_i^*(x,y+h,z) - \phi_i^*(x,y-h,z)}{2h},$$

$$\frac{\partial \phi_j}{\partial y} = \frac{\phi_j(x,y+h,z) - \phi_j(x,y-h,z)}{2h},$$

$$\frac{\partial \phi_i^*}{\partial z} = \frac{\phi_i^*(x,y,z+h) - \phi_i^*(x,y,z-h)}{2h},$$

$$\frac{\partial \phi_j}{\partial z} = \frac{\phi_j(x,y,z+h) - \phi_j(x,y,z-h)}{2h}.$$

The computation domain is $[a,b]^3$, $h = \frac{b-a}{m}$, m is the partition number on [a,b], (m must be even, since we use the Simpson's rule).

2.3 Electron-electron Coulomb energy functional $E_{ee}[n]$

$$E_{ee}[n(\vec{r})] = \frac{1}{2}\iint \frac{n(\vec{r}\,')n(\vec{r})}{|\vec{r} - \vec{r}\,'|} d\vec{r}\,' d\vec{r}$$

By (2.2),

$$v_{ee}(n(\vec{r})) = \int \frac{n(\vec{r}\,')}{|\vec{r} - \vec{r}\,'|} d\vec{r}\,' = \sum_{i,j} C_i C_j \int \frac{\phi_i^*(\vec{r}\,')\phi_j(\vec{r}\,')}{|\vec{r} - \vec{r}\,'|} d\vec{r}\,'$$

$$E_{ee}[n(\vec{r})] = \frac{1}{2}\iint \frac{(\sum_i C_i\phi_i(\vec{r}\,'))^2(\sum_k C_k\phi_k(\vec{r}))^2}{|\vec{r}-\vec{r}\,'|}d\vec{r}\,'d\vec{r}$$

$$= \sum_{i,j,k,h} C_iC_jC_kC_h \iint \frac{\phi_i^*(\vec{r}\,')\phi_j(\vec{r}\,')\phi_k^*(\vec{r})\phi_h(\vec{r})}{|\vec{r}-\vec{r}\,'|}d\vec{r}\,'d\vec{r}$$

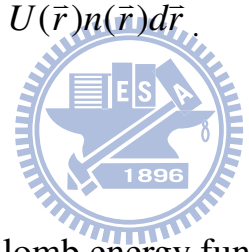
Now we need to calculate

$$\iint \frac{\phi_i^*(\vec{r}\,')\phi_j(\vec{r}\,')\phi_k^*(\vec{r})\phi_h(\vec{r})}{|\vec{r}-\vec{r}\,'|}d\vec{r}\,'d\vec{r} = \left\langle \phi_i(\vec{r}\,')\phi_j(\vec{r}\,') \left| \frac{1}{|\vec{r}-\vec{r}\,'|} \right| \phi_k(r)\phi_h(r) \right\rangle \qquad (2.3)$$

and that is a 6-dimension integral. By solving the Poisson's equation :

$$\begin{cases} \Delta U(\vec{r}) = -\phi_i(\vec{r})\phi_j(\vec{r}) & on\ \ \Omega \\ U(\vec{r}) = 0 & on\ \ \partial\Omega \end{cases},$$

for Ω large enough, (2.3) is $\int U(\vec{r})n(\vec{r})d\vec{r}$.



2.4 Electron-nuclear Coulomb energy functional $E_{ec}[n]$

$$E_{ec}[n(\vec{r})] = \int v_{ext}(\vec{r})\, n(\vec{r})d\vec{r}$$

In molecule, $v_{ext}(\vec{r}) = \sum_I \frac{-Z_I}{|\vec{r}-\tau_I|}$, where Z_I is the I-th atom's atomic number and τ_I is its atomic site.

$$E_{ec}[n(\vec{r})] = \int v_{ext}(\vec{r})(\sum_i C_i\phi_i(\vec{r}))^2 d\vec{r}$$

$$= \sum_{i,j} C_iC_j \int v_{ext}(\vec{r})\phi_i^*(\vec{r})\phi_j(\vec{r})d\vec{r}$$

Thus, the electron-nuclear Coulomb energy corresponding to each orbital can be written by

$$\langle \phi_i \,|\, v_{ext}(\vec{r}) \,|\, \phi_j \rangle \quad \text{for} \quad i,j = 1,2,\ldots,n$$

Because of the singular points τ_I, we must be careful to take the integral range otherwise

some grid points may be close to τ_I and this results in computational inaccuracy.

We divide the integral range into two parts : A ; B. Part-A contains the singular point and the rest is part-B. After partitioning the integral range $[a,b]$, finding the minimum interval which contains the singular point named $[c_0,d_0]$. Next, we extend interval $[c_0,d_0]$ for N intervals as $[c_N,d_N]$ namely part-A. Then, taking a large partition number to part-A for getting a more precise result. Finally, we integrate part-A and part-B and sum them up.

For example, the extend interval N equals to 1 and 2 as illustrated in Figure 2 and Figure 3.

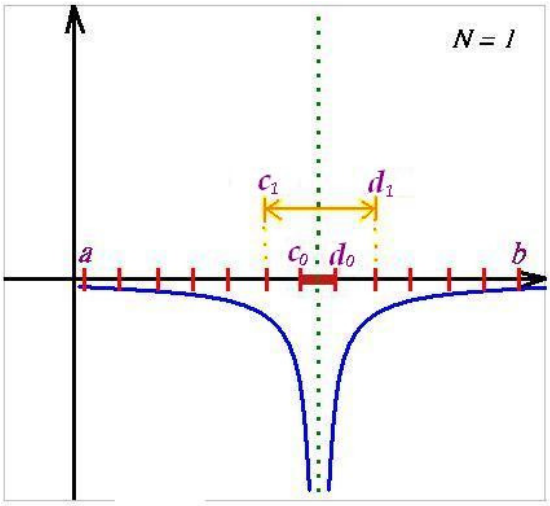


Figure 2 : Extended interval $N = 1$

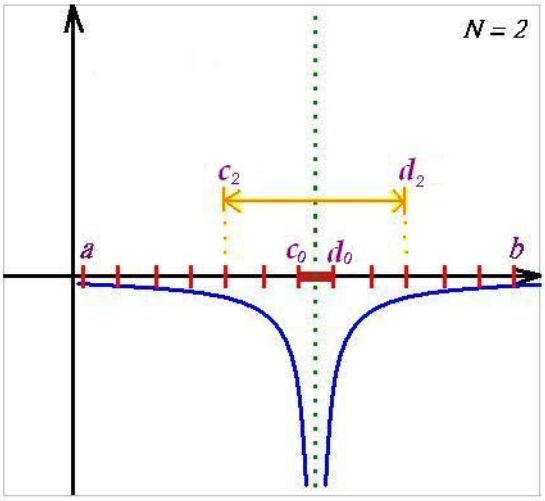


Figure 3 : Extended interval $N = 2$

Then $\langle \phi_i | v_{ext}(\vec{r}) | \phi_j \rangle$ can be expressed by

$$\int_{[a,b]\backslash \text{A-}part} v_{ext}(\vec{r})\phi_i^*(\vec{r})\phi_j(\vec{r})d\vec{r} + \int_{\text{A-}part} v_{ext}(\vec{r})\phi_i^*(\vec{r})\phi_j(\vec{r})d\vec{r}$$

2.5 Exchange-correlation energy functional $E_{xc}[n]$

$$E_{xc}([n]) = \int \varepsilon_{xc}(n(\vec{r}))\, n(\vec{r}) d\vec{r}$$

$$= \sum_{i,j} C_i C_j \int \varepsilon_{xc}(n(\vec{r}))\, \phi_i^*(\vec{r}) \phi_j(\vec{r}) d\vec{r}$$

The exchange-correlation energy per unit density $\varepsilon_{xc}(n(\vec{r}))$ depends only on the density $n(\vec{r})$ at the position $\vec{r}$. $\varepsilon_{xc}(n)$ can be decomposed into a sum of the exchange energy $\varepsilon_x(n)$ and the correlation energy $\varepsilon_c(n)$.

$$\varepsilon_{xc}(n) = \varepsilon_x(n) + \varepsilon_c(n)$$

They are obtained from a system of uniform electron gas with the corresponding density n and they are frequently expressed in terms of the radius r_s of a unit charge defined by

$$r_s = \left(\frac{3}{4\pi n}\right)^{1/3}.$$

The exchange energy is $\varepsilon_x(n) = -\frac{0.458}{r_s}$ and the correlation energy is

$$\varepsilon_c(n) = \begin{cases} -\gamma / (1 + \beta_1 \sqrt{r_s} + \beta_2 r_s) \ ; for \geqq 1 \\ A \ln r_s + B + C r_s \ln r_s + D r_s \ ; for < 1 \end{cases}$$

where

$$\gamma = 0.1423 \ ; \ \beta_1 = 1.0529 \ ; \ \beta_2 = 0.3334$$

$$A = 0.311 \ ; B = -0.048 \ ; C = 0.002 \ ; D = -0.0116 \ .$$

We can use them to calculate $\langle \phi_i \mid \varepsilon_{xc}(n) \mid \phi_j \rangle$.

3. The iterative procedure

Because of the total energy density functional is

$$E_{tol}[n(\vec{r})] = E_{kin}[n(\vec{r})] + E_{ee}[n(\vec{r})] + E_{ec}[n(\vec{r})] + E_{xc}[n(\vec{r})]$$

and because the density $n(\vec{r})$ can be represented by $n(\vec{r}) = \sum_{i,j} C_i C_j \phi_i^*(\vec{r}) \phi_j(\vec{r})$,

we can write the total energy functional as $F(\mathrm{C})$ where $\mathrm{C} = (C_1, C_2 \ldots C_{10})$. Next, we minimize the total energy functional $F(\mathrm{C})$ under the constraints :

$$\mathrm{N} = \int n(\vec{r})\, d\vec{r} = \sum_{i,j} C_i C_j \int \phi_i^* \phi_j \, d\vec{r} \ .$$

Because the basis functions $\{\phi_i\}_{i=1,\ldots,10}$ are orthonormal, the constraints can be written as $C_1^2 + C_2^2 + \ldots + C_{10}^2 = \mathrm{N}$.

Then the problem becomes

$\min F(C_1, C_2 \ldots C_{10})$ under the constraints $C_1^2 + C_2^2 + \ldots + C_{10}^2 = \mathrm{N}$

Define $G(C_1, C_2, \ldots, C_{10}) = C_1^2 + C_2^2 + \ldots + C_{10}^2 - \mathrm{N} = 0$ by Largrange multipliers, we need to solve the problem

$$\begin{cases} \nabla F = \lambda \nabla G \\ C_1^2 + C_2^2 + \ldots + C_{10}^2 = \mathrm{N} \end{cases}$$

or

$$\begin{cases} \nabla F = \lambda (2C_1, 2C_2 \ldots 2C_{10}) \\ C_1^2 + C_2^2 + \ldots + C_{10}^2 = \mathrm{N} \end{cases}$$

Therefore, we have

$$\begin{cases} \dfrac{\partial F}{\partial C_1} = \lambda 2C_1 \\ \dfrac{\partial F}{\partial C_2} = \lambda 2C_2 \\ \vdots \\ \dfrac{\partial F}{\partial C_{10}} = \lambda 2C_{10} \\ C_1^{\,2} + C_2^{\,2} + \ldots + C_{10}^{\,2} = \mathrm{N} \end{cases} \tag{3.1}$$

3.1 The gradient of the total energy functional $F(\mathrm{C})$

The total energy functional $F(C_1, C_2, \ldots, C_{10})$ can be rewrite by

$$F(\mathrm{C}) = E_{kin}(\mathrm{C}) + E_{ec}(\mathrm{C}) + E_{ee}(\mathrm{C}) + E_{xc}(\mathrm{C}) \,. \tag{3.2}$$

Thus $\dfrac{\partial F}{\partial C_i} = \dfrac{\partial}{\partial C_i}(E_{kin}(\mathrm{C}) + E_{ec}(\mathrm{C}) + E_{ee}(\mathrm{C}) + E_{xc}(\mathrm{C}))$.

From section 2.2~2.5, we can get

$$\frac{\partial E_{kin}(\mathrm{C})}{\partial C_i} = \frac{1}{2}\left(\sum_{j=1}^{10} C_j \left(\langle \phi_i \,|\, -\nabla^2 \,|\, \phi_j \rangle + \langle \phi_j \,|\, -\nabla^2 \,|\, \phi_i \rangle\right)\right) = \sum_{j=1}^{10} C_j \langle \phi_i \,|\, -\nabla^2 \,|\, \phi_j \rangle \tag{3.3}$$

$$\frac{\partial E_{ec}(\mathrm{C})}{\partial C_i} = \left(\sum_{j=1}^{10} C_j \left(\langle \phi_i \,|\, v_{ext} \,|\, \phi_j \rangle + \langle \phi_j \,|\, v_{ext} \,|\, \phi_i \rangle\right)\right) = 2\sum_{j=1}^{10} C_j \langle \phi_i \,|\, v_{ext} \,|\, \phi_j \rangle \tag{3.4}$$

we note $\left\langle \phi_i \phi_j \left| \dfrac{1}{|\vec{r} - \vec{r}\,'|} \right| \phi_k \phi_h \right\rangle = \left\langle \langle \phi_k \phi_h \left| \dfrac{1}{|\vec{r} - \vec{r}\,'|} \right| \phi_i \phi_j \rangle \right\rangle$

$$\frac{\partial E_{ee}(\mathrm{C})}{\partial C_i}=\frac{1}{2}\left(\sum_{i,j,k=1}^{10}C_iC_jC_k\left\langle\phi_i\phi_j\left|\frac{1}{|\vec{r}-\vec{r}\,'|}\right|\phi_k\phi_h\right\rangle+\sum_{i,j,h=1}^{10}C_iC_jC_h\left\langle\phi_i\phi_j\left|\frac{1}{|\vec{r}-\vec{r}\,'|}\right|\phi_k\phi_h\right\rangle\right.$$
$$\left.+\sum_{i,k,h=1}^{10}C_iC_kC_h\left\langle\phi_i\phi_j\left|\frac{1}{|\vec{r}-\vec{r}\,'|}\right|\phi_k\phi_h\right\rangle+\sum_{j,k,h=1}^{10}C_jC_kC_h\left\langle\phi_i\phi_j\left|\frac{1}{|\vec{r}-\vec{r}\,'|}\right|\phi_k\phi_h\right\rangle\right)$$

$$=\left(\sum_{i,j,k=1}^{10}C_iC_jC_k\left\langle\phi_i\phi_j\left|\frac{1}{|\vec{r}-\vec{r}\,'|}\right|\phi_k\phi_h\right\rangle+\sum_{i,j,h=1}^{10}C_iC_jC_h\left\langle\phi_i\phi_j\left|\frac{1}{|\vec{r}-\vec{r}\,'|}\right|\phi_k\phi_h\right\rangle\right)$$

$$=2\sum_{i,j,k=1}^{10}C_iC_jC_k\left\langle\phi_i\phi_j\left|\frac{1}{|\vec{r}-\vec{r}\,'|}\right|\phi_k\phi_h\right\rangle \qquad (3.5)$$

If an initial guess $(C_1,C_2,\ldots,C_{10})$ is given, then we can compute $\langle\phi_i\,|\,v_{ee}\,|\,\phi_j\rangle$ and (3.5)

becomes $2\sum_{j=1}^{10}C_j\langle\phi_i\,|\,v_{ee}\,|\,\phi_j\rangle.$ (3.6)

$$\frac{\partial E_{xc}(\mathrm{C})}{\partial C_i}=\left(\sum_{j=1}^{10}C_j\left(\langle\phi_i\,|\,\varepsilon_{xc}\,|\,\phi_j\rangle+\langle\phi_j\,|\,\varepsilon_{xc}\,|\,\phi_i\rangle\right)\right)=2\sum_{j=1}^{10}C_j\langle\phi_i\,|\,\varepsilon_{xc}\,|\,\phi_j\rangle \qquad (3.7)$$

To calculate (3.7), we need insert initial guess to decide ε_{xc}.

So we have

$$\frac{\partial F(\mathrm{C})}{\partial C_i}=\sum_{j=1}^{10}C_j\left(\langle\phi_i\,|\,-\nabla^2\,|\,\phi_j\rangle+2\langle\phi_i\,|\,v_{ext}\,|\,\phi_j\rangle+2\langle\phi_i\,|\,v_{ee}\,|\,\phi_j\rangle+2\langle\phi_i\,|\,\varepsilon_{xc}\,|\,\phi_j\rangle\right).$$

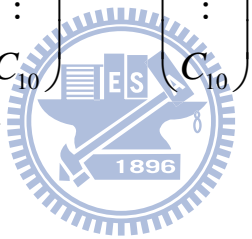
Denote it by : $\frac{\partial F(\mathrm{C})}{\partial C_i}=\sum_{j=1}^{10}C_j\mathrm{A}_{\mathrm{j}}^{\mathrm{i}}$

After linearization, (3.1) can be represented by

$$\begin{cases} \sum_{j=1}^{10} C_j \mathrm{A}_{\mathrm{j}}^{1} = \lambda 2 C_1 \\ \sum_{j=1}^{10} C_j \mathrm{A}_{\mathrm{j}}^{2} = \lambda 2 C_2 \\ \vdots \\ \sum_{j=1}^{10} C_j \mathrm{A}_{\mathrm{j}}^{10} = \lambda 2 C_{10} \\ C_1^{\,2} + C_2^{\,2} + \ldots + C_{10}^{\,2} = \mathrm{N} \end{cases} \tag{3.8}$$

In matrix form (3.8) is

$$\begin{cases} \begin{pmatrix} \mathrm{A}_1^1 & \mathrm{A}_2^1 & \cdots & \mathrm{A}_{10}^1 \\ \mathrm{A}_1^2 & \mathrm{A}_2^2 & \cdots & \mathrm{A}_{10}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{A}_1^{10} & \mathrm{A}_2^{10} & \cdots & \mathrm{A}_{10}^{10} \end{pmatrix} \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_{10} \end{pmatrix} = 2\lambda \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_{10} \end{pmatrix} \\ C_1^{\,2} + C_2^{\,2} + \ldots + C_{10}^{\,2} = \mathrm{N} \end{cases} \tag{3.9}$$



3.2 The procedure of Ab Initio self-consistency

For a given C^{old} guess, we solve the eigenproblem (3.9), find the eigenvector corresponding to the minimum eigenvalue, and let the eigenvector satisfy $C_1^{\,2} + C_2^{\,2} + \ldots + C_{10}^{\,2} = \mathrm{N}$. In this way, we obtain new coefficients $\mathrm{C}^{new} = (C_1, C_2, \ldots, C_{10})$.

Compare new coefficients C^{new} with the previous one C^{old} $\left(\dfrac{\left\| \mathrm{C}^{New} - \mathrm{C}^{Old} \right\|}{\left\| \mathrm{C}^{New} \right\|} < \text{1E-6} \right)$.

If they are close enough, then we say the iterative procedure converges and then use them to

compute the total energy functional of coefficients F to get total energy (adding E_{cc}). Otherwise we replace C^{old} by C^{new} as the initial guess and repeat the above procedure , as illustrated in Figure 4,

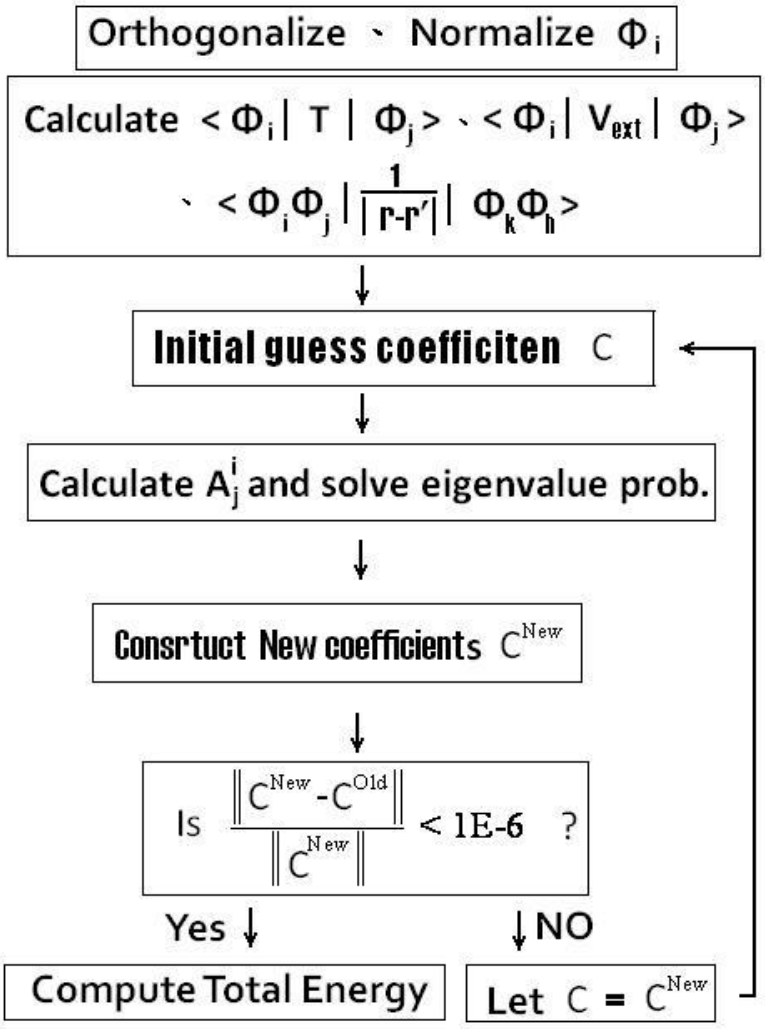


Figure 4 : The procedure of computation

3.3 Numerical results of H_2

Result. 1		Not include Exc											
Integral range of Ekin		(-3.5^3.5)											
Eee		(-3.5^3.5)											
Eec (1)		(-3.0^2.5)											
(2)		(-2.5^3.0)											
Extend interval N for Eec		1			2			3			4		
Each time select eigenvalue		min	max		min	max		min	max		min	max	
Iteration number		4	6		4	6		4	6		5	6	
First Energy	Ekin	2.630753045316 88			2.630753045316 88			2.630753045316 88			2.630753045316 88		
	Eec	-92.59876165095 41			-58.84963378881 33			-44.44216812864 97			-36.37382695168 60		
	Eee	0.3884135087398 24			0.3884135087398 24			0.3884135087398 24			0.3884135087398 24		
	Eec	1.3513513803482 1			1.3513513803482 1			1.3513513803482 1			1.3513513803482 1		
Converge Energy	Ekin	9.823730642993 81	2.432680214920 10		11.09281130863 58	3.132195653092 68		11.70401883792 80	3.134312567944 86		11.91193600322 46	3.452909708241 38	
	Eec	-288.6072374198 15	-74.70186902568 86		-183.6414426100 21	-49.18655598124 22		-134.8549899735 62	-36.28188750231 71		-106.8205937997 55	-29.06466855352 88	
	Eee	1.740631302550 53	0.2055332526210 82		1.903526215026 01	0.3399854424608 63		1.899596140715 77	0.3548840980318 45		1.819450658463 80	0.3631171189129 42	
Total Energy		-275.6915240939 22	-70.71230417779 93		-169.2937537060 11	-44.36302350534 04		-119.9000236145 70	-31.44133945599 22		-91.73785575771 87	-23.89729034602 62	

Result. 2		Not include Exc							
Integral range of Ekin		(-3.5~3.5)							
Eee		(-3.5~3.5)							
Eec (1) (2)		(-3.0~2.5) (-2.5~3.0)							
Extend interval N for Eec		5		6		7		8	
Each time select eigenvalue		min	max	min	max	min	max	min	max
Iteration number		5	6	5	6	5	7	5	6
First Energy	Ekin	2.630753045316 88		2.630753045316 88		2.630753045316 88		2.630753045316 88	
	Eec	-31.1159083260929		-27.3726048485679		-37.0725652752236		-22.3096105908597	
	Eee	0.388413508739824		0.388413508739824		0.388413508739824		0.388413508739824	
	Eec	1.351351380348 21		1.351351380348 21		1.351351380348 21		1.351351380348 21	
Converge Energy	Ekin	11.8674957522893	4.13245224397013	11.6746110311624	5.13728217362051	11.3952752092710	6.20008205621818	11.0685145954059	7.09812920420943
	Eec	-87.8230775563068	-24.5907456981600	-74.7673496736649	-21.6543828736774	-65.1322944156424	-19.5455389955781	-57.7715480297510	-17.8800923195504
	Eee	1.70842757065049	0.391924696542322	1.58827001640209	0.447855241429800	1.46766400449033	0.517019375206792	1.35389619834623	0.58314746599798
Total Energy		-72.8958028530188	-18.7150173772994	-60.1531172457522	-14.7178940782789	-50.9180038215330	-11.4770861838049	-43.9977858556506	-8.84746426899480

Result.3		Not include Exc					
Integral range of Ekin		(-3.5~3.5)					
Eee		(-3.5~3.5)					
Eec (1)		(-3.0~2.5)					
(2)		(-2.5~3.0)					
Extend interval N for Eec		12		13		15	
Each time select eigenvalue		min	max	min	max	min	max
Iteration number		6	7	6	7	6	8
First Energy	Ekin	2.63075304531688		2.63075304531688		2.63075304531688	
	Eec	-16.5504571893976		-15.6200812713936		-13.9290273463978	
	Eee	0.388413508739824		0.388413508739824		0.388413508739824	
Converge Energy	Ekin	9.71190883243779	9.46487361471365	9.45914175821986	9.85448088438887	8.91162870831761	10.5347517195961
	Eec	-39.8007093243827	-13.5857952473565	-37.2375851623085	-12.8784467569777	-32.2107288472586	-11.7633181365192
	Eee	1.00395566686832	0.826635215235297	0.936632144423734	0.882495850112858	0.852109081648014	0.990557062036701
Total Energy		-27.7334934447284	-1.94293503705933	-25.4904598793167	-0.790118642127806	-21.0956396769448	1.11334202546178

Result.4		Eec do not deal with the singular point Not include Exc	
Integral range of Ekin		(-3.5~3.5)	
Eee		(-3.5~3.5)	
Eec		(-3.5~2.5)	
Each time select eigenvalue		min	max
Iteration number		20	15
First Energy	Ekin	2.6307530453168 8	
	Eec	-3.1556909217606 6	
	Eee	0.3884135087398 24	
	Eec	1.3513513803482 1	
Converge Energy	Ekin	3.3502714483940 2	14.588934857490 6
	Eec	-6.4444333148934 7	-7.2317828949651 8
	Eee	1.7787038748989 7	1.1036706572192 6
Total Energy		0.0358933887477 2641	9.8121740000928 9

(experimental value of energy for H_2 ≒ -1.154948 ～ -1.175482)

The tables above do not include Exc. After observing the results given above and comparing the total energies with each other, we can discover that the energy of Eec is the main reason leading to an un-precise outcome. It shows that we could not get a more accurate total energy, even though we have an more exact calculation around the singular points.

Result. 5		Change the range of Ekin & Eee					
Integral range of Ekin		(-2.5~2.5)					
Eee		(-2.5~2.5)					
Exc		(-2.5~2.5)					
Eec (1)		(-3.0~2.5)					
(2)		(-2.5~3.0)					
Extend interval N for Eec		1		2		3	
Each time select eigenvalue		min	max	min	max	min	max
Iteration number		4	4	4	6	4	6
First Energy	Ekin	4.83649041186860		4.83649041186860		4.83649041186860	
	Eec	-92.5987616509541		-58.8496337888133		-44.4421681286497	
	Eee	0.388413508739824		0.388413508739824		0.388413508739824	
	Ecc	1.35135138034821		1.35135138034821		1.35135138034821	
Converge Energy	Ekin	11.5819267245436	5.56901965983781	13.0820840752875	5.82128213425869	13.9706948366152	5.51433395443284
	Eec	-288.614439446010	-74.7190633856251	-183.645178320542	-49.2472447506139	-134.856755295844	-36.4519225916829
	Eee	1.75076259226451	0.201161227935734	1.92095529062591	0.283745232422369	1.92843887897648	0.328950642004402
	Exc	-1.32766845510565	-0.576274290305866	-1.42621280874378	-0.644444665233445	-1.45266089004600	-0.629887786665072
Total Energy		-275.258067203959	-68.1738054078092	-168.717000383024	-42.4353106688180	-119.058931089950	-29.8871744015625

Result. 6		Eec do not deal with singular point		
Integral range of Ekin		(-2.5~2.5)		
Eee		(-2.5~2.5)		
Exc		(-2.5~2.5)		
Eec		(-3.0~2.5)		
Each time select eigenvalue		min		max
Iteration number		8		7
First Energy	Ekin	4.836490411868 60		
	Eec	-3.155690921760 66		
	Eee	0.388413508739824		
Converge Energy	Ekin	3.46006528750612		14.094813084 0825
	Eec	-6.39359064664345		-3.855817701 67025
	Eee	1.76562959516415		0.239814322 145208
	Exc	-1.12062669138637		-0.711678836 743809
	Ecc	1.35135138034821		1.351351380 34821
Total Energy		-0.937171075011342		11.118482248 1618

This time, we change the integral range of Ekin and Eee and also add Exc. By observing the values of total energies of result.1 and result.5, we can find that these total energies do not change too much. By observing result.6, we can know that we get a even more precise outcome, namely being more close to the experimental value, through doing nothing to the singular points. Here, we got the best result : -0.9371.

Result. 7		Change the range of Eec					
Integral range of Ekin		(-2.5~2.5)					
Eee		(-2.5~2.5)					
Exc		(-2.5~2.5)					
Eec (1)		(-3.5~3.0)					
(2)		(-3.0~3.5)					
Extend interval N for Eec		1		2		3	
Each time select eigenvalue		min	max	min	max	min	max
Iteration number		4	6	4	9	4	9
First Energy	Ekin	4.83649041186860		4.83649041186860		4.83649041186860	
	Eec	-83.2457412756039		-51.3523571370950		-38.1874302581489	
	Eee	0.388413508739824		0.388413508739824		0.388413508739824	
	Ecc	1.35135138034821		1.35135138034821		1.35135138034821	
Converge Energy	Ekin	11.7527227753179	2.12934613781941	13.3422102160653	3.31386343870899	14.3751354567239	3.95660073571880
	Eec	-231.132091338661	-67.0500577196363	-148.657395368680	-42.4581665320255	-110.060892630419	-31.2265233056505
	Eee	2.26397961332533	0.384665501129649	-3.48447153968849	0.250541137349189	2.31133980073240	0.229674241831632
	Exc	-3.31271521938211	-1.16987522333386	1.35135138034821	-1.11530551140087	-3.50758215873747	-1.14698745036981
Total Energy		-219.076752789051	-64.3545699236729	-135.084989021262	-38.6577160870200	-34.3876437895086	-26.8358843981217

Result. 8		Eec do not deal with singular point	
Integral range of Ekin		(-2.5~2.5)	
Eee		(-2.5~2.5)	
Exc		(-2.5~2.5)	
Eec		(-3.5~3.0)	
Each time select eigenvalue		min	max
Iteration number		8	10
First Energy	Ekin	4.836490411868 60	
	Eec	-2.952488301913 63	
	Eee	0.388413508739824	
	Ecc	1.351351380348 21	
Converge Energy	Ekin	3.541318216607 83	13.884741651451 2
	Eec	-6.350826057167 57	-3.303743576431 05
	Eee	1.781064619470 23	0.225407797910 953
	Exc	-2.478678553472 63	-1.538837732166 74
	Ecc	1.351351380348 21	1.351351380348 21
Total Energy		-2.155770394213 93	10.618919521112 6

Final, we change the integral range of Eec and then observe the results. By comparing with result.1, result.5 and result.7, we can discover that values of total energies have a big change after changing the range of Eec. And also we can find that the total energies still do not accurate enough.

3.4 Numerical results of H_2O

Result. 9		Let these parts consist of singular points have the same integral range					
Integral range of Ekin		(-10.0~10.0)					
Eee		(-10.0~10.0)					
Exc		(-10.0~10.0)					
Eec		(-9.5~10.0)					
Extend interval N for Eec		1		2			
Each time select eigenvalue		min	max	min		max	
Iteration number		7	10	8		8	
First Energy	Ekin	280.738962227609		280.738962227609			
	Eec	-1482.5933405050 3		-1238.837291818 93			
	Eee	135.450420273308		135.450420273308			
Converge Energy	Ekin	238.220278576702	795.811613792587	251.790168696975		841.868137114063	
	Eec	-3793.105986807 06	-1318.490414617 85	-2799.917386598 96		-1077.765586513 16	
	Eee	81.4233910064685	1169.739844873 04	93.3910970709406		1193.219952795 97	
	Exc	-13.961982646010 4	-86.967136302404 9	-15.791691727491 4		-88.651398803 5572	
	Ecc	17.354520897121 7	17.354520897121 7	17.354520897121 7		17.354520897121 7	
Total Energy		-3470.069778972 78	577.548428642492	-2453.173291661 41		886.025625490431	

Result. 10		Let these parts consist of singular points have different integral ranges			
Integral range of Ekin		(-10.0~10.0)			
Eee		(-10.0~10.0)			
Exc		(-10.0~10.0)			
Eec	(1)	(-9.5~9.0)			
	(2)	(-9.0~9.5)			
	(3)	(-8.5~10.0)			
Extend interval N for Eec		1		2	
Each time select eigenvalue		min	max	min	max
Iteration number		7	10	8	8
First Energy	Ekin	280.738962227609		280.738962227609	
	Eec	-1357.98490254325		-1228.83309098572	
	Eee	135.450420273308		135.450420273308	
Converge Energy	Ekin	245.788962694488	802.278136689815	253.014841426987	841.783946861435
	Eec	-3839.83543974324	-1365.65325500952	-2860.43858439752	-1087.51063810256
	Eee	87.2715694156726	1182.94185913991	93.3480711889756	1192.74225561562
	Exc	-14.5794078481103	-87.6407566908131	-15.7169574956897	-88.6284827269076
	Ecc	17.3545208971217	17.3545208971217	17.3545208971217	17.3545208971217
Total Energy		-3503.99979458407	549.280505026509	-2512.43810838012	875.741602544707

Result. 11		Eec do not deal with singular point and change the integral range of Eec					
Integral range of Ekin		(-10.0~10.0)					
Eee		(-10.0~10.0)					
Exc		(-10.0~10.0)					
Eec		(-9.5~10.0)		(-9.999~10.0)		(-9.9999~10.0)	
Each time select eigenvalue		min	max	min	max	min	max
Iteration number		Not convergence	9	7	8	3	Not convergence
First Energy	Ekin	280.738962227609		280.738962227609		280.738962227609	
	Eec	-288.941185193061		-288.941185193061		-73820.9666817395	
	Eee	135.450420273308		135.450420273308		135.450420273308	
Converge Energy	Ekin		1000.61978145311	244.341781946018	836.713701224497	997.200350315808	2.50589422693925
	Eec		-864.101903019618	-3546.67467029387	-1034.47423132429	-401649.836482543	-27.6094632423374
	Eee		1452.84785377292	89.7958752244207	1138.17466288120	1504.02756088824	8.12031005919380
	Exc		-100.522300545342	-15.4367749706148	-85.8830533778353	-103.410786218784	-2.51916638459076
	Ecc		17.3545208971217	17.3545208971217	17.3545208971217	17.3545208971217	17.3545208971217
Total Energy		-52.4597622023997 Jump between of them 745.027071825483	1506.19795255819	-3210.61926719693	871.885600300693	-399234.664836661	-2.14790534973144531 25

Result. 12			Eec do not deal with singular point and Let these singular points have different integral ranges		
Integral range of Ekin			(-10.0~10.0)		
Eee			(-10.0~10.0)		
Exc			(-10.0~10.0)		
Eec (1) (2) (3)			(-9.5~9.0) (-9.0~9.5) (-8.5~10.0)		
Each time select eigenvalue			min	max	
Iteration number			Not convergence	23	
First Energy	Ekin		280.738962227609		
	Eec		-348.220196023354		
	Eee		135.450420273308		
Converge Energy	Ekin				2.50628844283582
	Eec				-27.6134601597648
	Eee				8.12022789232073
	Exc				-2.5191475035673 3
	Ecc				17.3545208971217
Total Energy			-44.943093580850 2 Jump between of them 870.434582839281		-2.1515704310539 2

(experimental value of energy for $H_2O \fallingdotseq$ -75.58596)

Through observing the results above, we can discover that they even do not converge. From the results of H_2 and H_2O, we could know that it's necessary to avoid the singular points while calculating Eec, otherwise the outcome will not be precise even there is a bigger partition number. It represents that we need to use other methods to calculate Eec. For example, pseudo potential that might be a better choice.

3.5 Idea of pseudo potential

The pseudo potential is used to represent the complicated effects of the movements of an atom's electrons and nucleus with other effective potentials. We should know Kohn-Sham equation first. Kohn-Sham equation is obtained from the energy functional (1.1) by using the variational principle with respect to $\phi^*(\vec{r})$ under the constraint $\mathrm{N} = \int n(\vec{r})dr$,

$$\left\{-\frac{1}{2}\nabla^2 + v_{ext}(\vec{r}) + v_{ee}(n(\vec{r})) + v_{xc}(n(\vec{r}))\right\}\phi_i(\vec{r}) = \varepsilon_i(\vec{r})\phi_i(\vec{r})$$

where ε_i is a Lagrange multiplier and eigenvalue corresponding to $\phi_i(\vec{r})$.

Next, we need to construct a pseudo wave function which has the same graph as real wave function while $r > r_c$ (r_c is a cutoff radius) and is a smooth nodeless function to 0 as $r < r_c$. When the pseudo wave function have the same eigenvalue as real wave function, the potential is pseudo potential. Because the pseudo wave function is a smooth nodeless function as $r < r_c$, there is no such kind of problems about singular point by using pseudo potential.

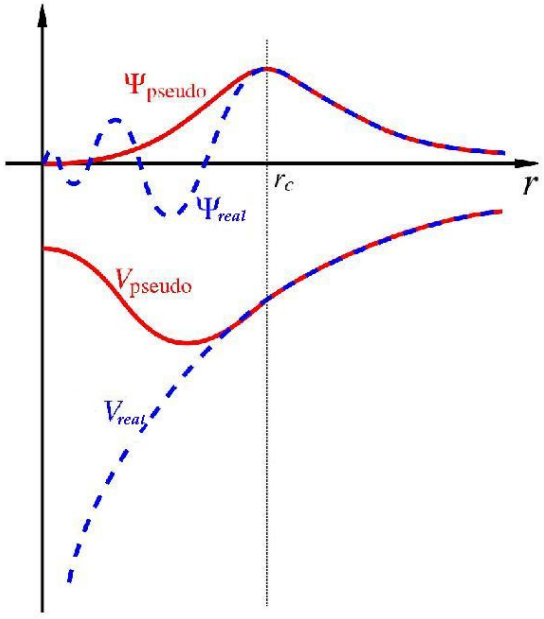


Figure 5 : The graph of pseudo potential and pseudo wave function

A Appendix

A.1 The contents of the main program

```
!-------------------------------------------------------------------!
!              Ylm   →  動能:complex  其他:real
!              積分  →  辛普森積分
!------------宣告常數-----------------------------------------------!
module global
   implicit none
         real(kind=8),parameter :: a0 = 0.529189379          ! Bohr radius  單位:Å
         real(kind=8),parameter :: pi = 3.141592653589     ！ 圓周率π
         real(kind=8) :: x_max   ！ 積分範圍  x_min→x_max
         real(kind=8) :: y_max   ！ 積分範圍  y_min→y_max
         real(kind=8) :: z_max   ！ 積分範圍  z_min→z_max
         real(kind=8) :: x_min   ！ 積分範圍  x_min→x_max
         real(kind=8) :: y_min   ！ 積分範圍  y_min→y_max
         real(kind=8) :: z_min   ！ 積分範圍  z_min→z_max
         real(kind=8) :: EecX_min,EecY_min,EecZ_min
         real(kind=8) :: EecX_max,EecY_max,EecZ_max
         real(kind=8) :: Eec,Ecc,Eee,Eexc,E_kin,E_tol     ！ 各項能量
         integer,parameter :: inn_m_parti=200     ！ 內積積分-辛普森法  切割數(必須爲偶數)
         integer,parameter :: basics = 10           ！ 基底函數  數量
         real(kind=8) :: C(basics)                    ！ 基底函數  係數
         real(kind=8) :: C_new(basics),norm2,norm3        ！ 基底函數  係數
         complex(kind=8) :: singleEkin(basics,basics)
         complex(kind=8) :: singleEec(basics,basics)
         complex(kind=8) :: singleEec2(6,basics,basics)
         complex(kind=8) :: ijkh_Eee(basics,basics,basics,basics)
         complex(kind=8) :: singleEee(basics,basics)
         complex(kind=8) :: singleExc(basics,basics)
         complex(kind=8) :: Exc_N
         real(kind=8) :: Exc_N2
         real(kind=8)::xi(inn_m_parti+1),yi(inn_m_parti+1),zi(inn_m_parti+1)
         complex(kind=8),allocatable:: wave(:,:,:,:) ! Φi,i=1,10
         complex(kind=8),allocatable:: grad_phi(:,:,:),grad_phiA(:,:,:),grad_phiB(:,:,:)
```

```
        real(kind=8),allocatable :: vext(:,:,:)
        real(kind=8),allocatable :: Exc(:,:,:)
        integer,parameter    :: CA_M =(1+BASICS)*BASICS/2            ! 計算上三角的總數
        integer                  :: IC(CA_M),JC(CA_M)                ! 計算上三角
end module

!==============================主程式==============================!
program main
   use IMSL
   use global
   implicit none
!--------------宣告變數&常數--------------!
        real(kind=8) :: r,x1,y1,z1,x2,y2,z2          ! 座標變數
        real(kind=8) :: theta,phi
        real(kind=8),external ::       Rpl,vextF
        complex(kind=8),external :: Ylm
        complex(kind=8),external :: basis_func
        integer          :: i,j,k,h,h2,counter,counter2
        complex(kind=8) :: temA,temB,temC,temD
        real(kind=8) :: F_diffC(basics,basics)                    ! F_diffC(basics)爲 F 對 C 微分
        real(kind=8)       :: ANS
        complex(kind=8) :: ANS2,ANS3,ANS4,ANS5
        integer,parameter::nn=basics ! 計算  eigenvalue
        integer,parameter::nm=basics
      integer is1,is2,ierr,matz
      double precision a(nm,nn),wr(nn),wi(nn),z(nm,nn),fv1(nn),extremaN
      integer iv1(nn),select,extrema
        allocate(wave(basics,inn_m_parti+1,inn_m_parti+1,inn_m_parti+1))
        allocate(grad_phi(inn_m_parti+1,inn_m_parti+1,inn_m_parti+1))
        allocate(grad_phiA(inn_m_parti+3,inn_m_parti+3,inn_m_parti+3))
        allocate(grad_phiB(inn_m_parti+3,inn_m_parti+3,inn_m_parti+3))
        allocate(vext(inn_m_parti+1,inn_m_parti+1,inn_m_parti+1))
        allocate(Exc(inn_m_parti+1,inn_m_parti+1,inn_m_parti+1))
        !-----把輸出結果 印在 txt 檔上
        open(unit=8,file='10wave_function.txt',status='old')          ! 同檔案 10wave_function.txt
        open(unit=60,file='new_10single_Ekin.txt',status='old')     ! 此檔案同 Ekin.txt
```

```
        open(unit=61,file='new_10single_Eec.txt',status='old')      ! 此檔案同 Eec.txt
        open(unit=62,file='new_10single_Eee.txt',status='old')      ! 此檔案同 Eee.txt
        open(unit=90,file='10TOTAL_ENERGY.txt')       ! 最後收斂後的 Total Energy
!-前置作業 1--------------------!
!----FIRST   START    要先算一次 各基底函數對應的值    ----------------!
!!! 給定積分範圍   Exc 跌代需要先給訂積分範圍
          x_max = 10.0              ! 積分範圍 x_min→x_max
          y_max = 10.0              ! 積分範圍 y_min→y_max
          z_max = 10.0              ! 積分範圍 z_min→z_max
          x_min = -10.0             ! 積分範圍 x_min→x_max
          y_min = -10.0             ! 積分範圍 y_min→y_max
          z_min = -10.0             ! 積分範圍 z_min→z_max
!----(0) 需用到 正交歸一的波函數 下列方法 2 選一        ----------------!
!-------(i) 對波函數正交歸一化
!                       call othonormal()
!-------(ii)讀取先前正交歸一的波函數檔案
                        do h = 1,basics
                         do i = 1,inn_m_parti+1
                          do j = 1,inn_m_parti+1
                           do k = 1,inn_m_parti+1
                                read(8,*) temD
                                wave(h,i,j,k) = temD
                           end do
                          end do
                         end do
                        end do
!----(1)動能 部分 :  計算每個 <Φi α |T| Φj β)>                  ------!
!       call calEkin()                                                  !
!----(2)電-核 庫倫能部分 : 計算每個 <Φi α |Vext| Φj β)>       ------!
!       call calEec(2) !!給定 奇異點 區間擴展數
!----(3)電-電 庫倫能部分 : 讀取 2 檔案 存成<i,jl lk,h>              ------!
!       call calEee()
!       GOTO 115                                                        !
!-----------------------------------------------------------------!

!----  讀檔案   給定 singleEkin、singleEec 值              ----------------!
```

```
      do i = 1,basics
       do j = 1,basics
            read(60,'(TR8,(E30.20,E30.20))')temA     ! TR8:從第 8 個位置開始讀
            read(61,'(TR8,(E30.20,E30.20))')temB
            singleEkin(i,j) = temA              ! <Φj｜  T  ｜Φi)>
            singleEec(i,j)  = temB              ! <Φj｜Vext｜Φi)>
            do k = 1,basics
             do h = 1,basics
                  read(62,'(TR13,(E30.20,E30.20))')temC
                  ijkh_Eee(i,j,k,h) = temC            ! <Φj｜  T  ｜Φi)>
             end do
            end do
       end do
      end do

! 給定數值:(1)選取最小特徵值  (2)  選取最大特徵值
      do select = 1,2
            if(select==1)then
                  write(90,*)"(1)選取  最小特徵值:"
                  write(90,*)
            elseif(select==2)then
                  write(90,*)
                  write(90,*)"(2)選取  最大特徵值:"
                  write(90,*)
            else
                  write(*,*)"select 錯誤!"
                  GOTO 115
            endif

!--     給定  初始係數 C(i)----------------------------------------!
      C(1) =   1.0
      C(2) =   1.0
      C(3) =   1.0
      C(4) =   1.0
      C(5) =   1.0
      C(6) =   1.0
```

```
        C(7) =    1.0
        C(8) =    1.0
        C(9) =    1.0
        C(10)=    1.0

!------------第一次迭代前的能量----------------!
                call calExc() ! 計算  Exc-------
                call calTol() ! 列出第一次的 total energy
                write(90,*) "第一次迭代前的能量!"
                write(90,*) "Ekin ", E_kin
                write(90,*) "Eec " , Eec
                write(90,*) "Eee " , Eee
                write(90,*) "Exc " , Eexc
                write(90,*) "Total Energy ",E_tol
                write(90,*)

        counter = 1             ! 計算迭代係數 C 的次數
!-------       計算<Φi｜Vee｜Φj)>  存到 singleEee(:,:)，共 17*17 項     ---!
113 singleEee(:,:) = (0.0,0.0)    ! 113  代表迭代係數從這裡開始
        do i = 1,basics
          do j = 1,basics
            do k = 1,basics
              do h = 1,basics
        ! <Φi｜Vee｜ Φj)> = Σi Σk C i C j <Φk(r')Φh(r')｜Vee｜ Φi(r)Φj(r))>
                  singleEee(i,j) = singleEee(i,j) + C(k)*C(h)*ijkh_Eee(k,h,i,j)
              end do
            end do
          end do
        end do

    if(counter==1)GOTO 114
        call calExc() ! 計算  <Φi｜ ε xc(n)｜ Φj)>

!-----   給定  F_diffC(i,j)   實數
114     F_diffC(:,:) = 0.0
        do i =1,basics
```

```
        do j =1,basics
         ! F_diffC(i,j) =   <Φi｜T + 2Vec + 2Vee + 2 ε xc｜Φj)>
              F_diffC(i,j) = singleEkin(i,j)+2.0*singleEec(i,j) +2.0*singleEee(i,j)+2.0*singleExc(i,j)
        end do
       end do

!----解 eigenvalue prob-------!
       matz = 1   ！給值 1  輸出特徵值&特徵向量

       do i =1,basics
        do j =1,basics
             a(i,j) = F_diffC(i,j)
        end do
       end do

       call rg(nm,nn,a,wr,wi,matz,z,iv1,fv1,ierr)

!      if (counter<=2)then     ！列出前 2 次跌代的 特徵值與特徵向量
!                    write(90,*) counter
!             do j=1,nn
!                    write(90,*)
!                    write(90,*) "eigenvalne:",wr(j),wi(j)！列出特徵值 實部 虛部
!                    write(90,*) "eigenvector:"
!              do i=1,nn
!                    write(90,*) "                ",z(i,j)          ！列出相對應的特徵向量
!               end do
!             end do
!      end if

       extremaN = wr(1)
       do j =1,nn
       if(select==1)then          ！取最小特徵值 所對應的特徵向量
             if(wr(j) <= extremaN )then
                    extremaN = wr(j)
                    extrema = j
             end if
```

```
elseif(select==2)then            ! 取最大特徵值 所對應的特徵向量
        if(wr(j) >= extremaN )then
                extremaN = wr(j)
                extrema = j
        end if
else
        write(*,*)"選取特徵值 出錯!"
end if
end do

!       !------取特徵值 對應的特徵向量 整理後令爲新係數 C_new
        norm2 = 0.0
        do i =1,basics
                norm2 = norm2 + z(i,extrema)*z(i,extrema)
        end do
        do i =1,basics
                C_new(i) = (z(i,extrema)*sqrt(10.0))/sqrt(norm2)
        end do


        !-------判停---------!
        norm2 = 0.0
        norm3 = 0.0
        do i =1,basics
                norm2 = norm2 + (C_new(i)-C(i))*(C_new(i)-C(i))
                norm3 = norm3 + C_new(i)*C_new(i)
        end do


        if(sqrt(norm2/norm3)<1E-6)then
                write(90,'(A2,I5,A8)') "第",counter,"次收斂!"
                C(:) = C_new(:)
                do i =1,basics
                        write(90,*)"                                ", C(i)
                end do
                call calTol()     ! 計算  total energy
        else
                call calTol()     ! 計算  total energy
```

```
            write(90,*) counter,E_tol   ！列出每一次的 total energy
            do i =1,basics
                write(90,*)"                                    ", C(i)
            end do
            counter = counter+1
            C(:) = C_new(:)

            if(counter>100)then
                write(90,*) "  跌代次數超過 100  還不會收斂!"
                GOTO 115
            else
                GOTO 113
            end if
        end if

!----          計算   個別的能量                    --------------------!
        write(90,*)
        write(90,*) "Ekin   = " , E_kin
        write(90,*)
        write(90,*) "Eec    = " , Eec
        write(90,*)
        write(90,*) "Eee    = " , Eee
        write(90,*)
        write(90,*) "Exc    = " , Eexc
        write(90,*)

!----   計算 Ecc 項                                  ----------------------!
!----                  1          Z I  Z J          ----------------------!
!----      Ecc   =   ——  Σ Σ  ————————            ----------------------!
!----                  2      I J   | τ I- τ J |    ----------------------!
        Ecc =   2.0*(8.0/0.9584) + 1.0/(0.9584*sqrt(2.0+2.0*cos(75.55*pi/180.0)))
        write(90,*) "Ecc    = " , Ecc
        write(90,*)

!------ 列出 Ekin + Eee + Eec + Ecc 總合----------------------------!
        write(90,*) "Total Energy =",E_kin + Eec + Eee + Eexc + Ecc
```

```
        write(90,*)

        end do   ! ← select

115     DEALLOCATE(wave)   ! 釋放可變動陣列記憶體
        DEALLOCATE(grad_phi)
        DEALLOCATE(grad_phiA)
        DEALLOCATE(grad_phiB)
        DEALLOCATE(vext,Exc)
   stop
end
!=================          主程式 結尾   ============================!


!-----------     radial function        Rpl(r)        ----------------------!
real(kind=8) function Rpl(atom,p,l,r)
   use global
   implicit none
        integer          :: p,l             ! p  主量子數 ;l  角動量子數;atom  原子序
        real(kind=8) :: r,atom
                if (p==1 .AND. l==0)then            ! R10
                        Rpl = 2.0*((atom/a0)**(1.5))*exp(-(atom/a0)*r)
                else if (p==2 .AND. l==0)then     ! R20
                        Rpl = ((atom/(2.0*a0))**(1.5))*(2.0-(atom/a0)*r)*exp(-(atom/(2.0*a0))*r)
                else if (p==2 .AND.l==1)then       ! R21
                        Rpl = (1.0/sqrt(3.0))*((atom/(2.0*a0))**(1.5))*((atom/a0)*r)*exp(-(atom/(2.0*a0))*r)
                else if (p==3 .AND. l==0)then     ! R30
                        Rpl = (2.0/27.0)*((atom/(3.0*a0))**(1.5))*(27.0-18.0*((atom/a0)*r)+ &
                                & (2.0*atom*atom/(a0*a0))*r*r)*exp(-(atom/(3.0*a0))*r)
                else
                        write(*,*)" Rpl 超過 R30 !"
                end if
   return
end

!-----------     spherical harmonic function    Ylm(θ,φ)        ---------!
```

```
!-----------     爲計算上方便 省略 phi 部分 → 實函數          ----------!
complex(kind=8) function Ylm(l,m,theta,phi)
   use global
   implicit none
        real(kind=8) :: theta,phi          ! θ,φ
        integer :: l,m                              ! l 角動量子數 ; m 磁量子數
                if (l==0 .AND. m==0)then              ! Y00
                        Ylm = 1.0/(sqrt(4.0*pi))
                else if(l==1 .AND. m==-1)then    ! Y1-1
                        Ylm = (sqrt(3.0/(8.0*pi)))*sin(theta)*(cos(phi)-csqrt((-1.0,0.0))*sin(phi))
                else if(l==1 .AND. m==0)then      ! Y10
                        Ylm = (sqrt(3.0/(4.0*pi)))*cos(theta)
                else if(l==1 .AND. m==1)then      ! Y11
                        Ylm = -(sqrt(3.0/(8.0*pi)))*sin(theta)*(cos(phi)+csqrt((-1.0,0.0))*sin(phi))
                else if(l==2 .AND. m==-2)then    ! Y2-2
                      Ylm = (sqrt(15.0/(32.0*pi)))*((sin(theta))**(2.0))*exp(-2.0*csqrt((-1.0,0.0))*phi)
                else if(l==2 .AND. m==-1)then    ! Y2-1
                        Ylm = (sqrt(15.0/(8.0*pi)))*(sin(theta))*(cos(theta))*exp(-csqrt((-1.0,0.0))*phi)
                else if(l==2 .AND. m==0)then      ! Y20
                        Ylm = (sqrt(5.0/(16.0*pi)))*(3.0*(cos(theta))**(2.0)-1.0)
                else if(l==2 .AND. m==1)then      ! Y21
                        Ylm = -(sqrt(15.0/(8.0*pi)))*(sin(theta))*(cos(theta))*exp(csqrt((-1.0,0.0))*phi)
                else if(l==2 .AND. m==2)then      ! Y22
                        Ylm = (sqrt(15.0/(32.0*pi)))*((sin(theta))**(2.0))*exp(2.0*csqrt((-1.0,0.0))*phi)
                end if
   return
end

!-----   原始基底函數  Φ1～Φ10   -----(  Φ  = Rpl*Ylm )-------------------!
complex(kind=8) function basis_func(i,x1,y1,z1)
   use global
   implicit none
        integer :: i
        real(kind=8) :: x1,y1,z1,atom                        ! x1,y1,z1 爲直角坐標上變數
        real(kind=8) :: r,r1,r2,r3                             ! r1 中心爲 H1    r2 中心爲第一個 H2
        real(kind=8) :: theta,theta1,theta2,theta3          ! 角度  θ :theta
```

```
real(kind=8) :: phi,phi1,phi2,phi3                    ! 角度 φ:phi
real(kind=8),external :: Rpl                          ! 函數 Rpl
complex(kind=8),external ::Ylm                        ! 函數 Ylm

r1 = sqrt(x1**2.0+y1**2.0+z1**2.0)                    ! r 換成直角坐標 x,y,z  中心爲 O
r2 = sqrt((x1-0.9584)**2.0+y1**2.0+z1**2.0)           !                       中心爲 H1
r3 = sqrt((x1+0.9584*cos(75.55*pi/180.0))**2.0+(y1-0.9584*sin(75.55*pi/180.0))**2.0+z1**2.0)   !
中心爲 H2
theta1 = atan2(sqrt(x1**2.0+y1**2.0),z1)              ! ← arctangent(a/b)
theta2 = atan2(sqrt((x1-0.9584)**2.0+y1**2.0),z1)
theta3 = atan2(sqrt((x1+0.9584*cos(75.55*pi/180.0))**2.0+(y1-0.9584*sin(75.55*pi/180.0))**2.0),z1)
phi1 =  atan2(y1,x1)
phi2 =  atan2(y1,x1-0.9584)
phi3 =  atan2(y1-0.9584*sin(75.55*pi/180.0),x1+0.9584*cos(75.55*pi/180.0))

      if(i<=6)then                                    ! 波函數共 10 項，前 6 項中心 O
            atom = 8d0
            r    = r1
            theta = theta1
            phi  = phi1
      elseif(i==7.OR.i==8)then                        ! 第 7,8 項中心 H-1
            atom  = 1d0
            r    = r2
            theta = theta2
            phi  = phi2
      elseif(i==9.OR.i==10)then                       ! 第 9,10 項中心 H-2
            atom  = 1d0
            r    = r3
            theta = theta3
            phi  = phi3
      else
            write(*,*)"wave function 出問題"
      end if

      if(i==1)then
```

```
            basis_func = Rpl(atom,1,0,r) * Ylm(0,0,theta,phi)        ! O 軌道 1S
        elseif(i==2)then
            basis_func = Rpl(atom,2,0,r) * Ylm(0,0,theta,phi)        ! O 軌道 2S
        elseif(i==3)then
            basis_func = Rpl(atom,2,1,r) * Ylm(1,-1,theta,phi)       ! O 軌道 2P
        elseif(i==4)then
            basis_func = Rpl(atom,2,1,r) * Ylm(1,0,theta,phi)        ! O 軌道 2P
        elseif(i==5)then
            basis_func = Rpl(atom,2,1,r) * Ylm(1,1,theta,phi)        ! O 軌道 2P
        elseif(i==6)then
            basis_func = Rpl(atom,3,0,r) * Ylm(0,0,theta,phi)        ! O 軌道 3S
        elseif(i==7)then
            basis_func = Rpl(atom,1,0,r) * Ylm(0,0,theta,phi)        ! H1 軌道 1S
        elseif(i==8)then
            basis_func = Rpl(atom,2,0,r) * Ylm(0,0,theta,phi)       ! H1 軌道 2S
        elseif(i==9)then
            basis_func = Rpl(atom,1,0,r) * Ylm(0,0,theta,phi)        ! H2 軌道 1S
        elseif(i==10)then
            basis_func = Rpl(atom,2,0,r) * Ylm(0,0,theta,phi)        ! H2 軌道 2S
        else
            write(*,*)"basis_func 超過 10!!"
        end if
    return
end

!---------------------------------------------------------------------!
!----                                - Z I                            ----------!
!----      Vext( r )   =     Σ   ————————                            ----------!
!----                          I   |  r - τ I |                        ----------!
!---------------------------------------------------------------------
real(kind=8) function vext_H1F(x1,y1,z1)
    use global
    implicit none
        real(kind=8) :: x1,y1,z1
            vext_H1F = -(8.0/(sqrt(x1*x1 + y1*y1 + z1*z1)))
return
```

```
end function
!-----------------------------
real(kind=8) function vext_H2F(x1,y1,z1)
   use global
   implicit none
        real(kind=8) :: x1,y1,z1
                vext_H2F = - (1.0/(sqrt((x1-0.9584)**2.0 + y1*y1 + z1*z1)))
return
end function
!-----------------------------
real(kind=8) function vext_H3F(x1,y1,z1)
   use global
   implicit none
        real(kind=8) :: x1,y1,z1
                vext_H3F = - (1.0/(sqrt((x1+0.9584*cos(75.55*pi/180.0))**2.0+(y1-0.9584*sin(75.55*pi/180.0))**2.0+z1**2.0)))
return
end function

!----給定陣列    wave(i,:,:,:)值,i=1~10          ----------!
subroutine sub1()
   use global
   implicit none
        integer :: h,i,j,k
        complex(kind=8),external::basis_func

        !---給定定義域  x,y,z 格子點值
        do i = 1,inn_m_parti+1
                xi(i) = x_min + ((x_max-x_min)/inn_m_parti)* (i-1)
                yi(i) = y_min + ((y_max-y_min)/inn_m_parti)* (i-1)
                zi(i) = z_min + ((z_max-z_min)/inn_m_parti)* (i-1)
        end do

        !--  給定陣列    wave(i,:,:,:)=Φ(i,:,:,:)
        do h=1,basics
                do i =1,inn_m_parti+1
                  do j =1,inn_m_parti+1
```

```
                do k =1,inn_m_parti+1
                    wave(h,i,j,k) = basis_func(h,xi(i),yi(j),zi(k))
                end do
               end do
              end do
        end do
   return
end subroutine

!----計算 <ΦilΦj>   3-D 辛普森積分----------!
subroutine sub2(h1,h2,ANS2)
use global
implicit none
        integer :: i,j,k,h1,h2
        real(kind=8)::C_i,C_j,C_k
        complex(kind=8):: ANS,ANS2

        ANS =0.0
        ! 辛普森三重積分 ∫∫∫Φi・Φj*  ; i,j=1～10
        do i=1,inn_m_parti+1
            if(i==1 .OR. i==inn_m_parti+1)then    ! 首項、末項點 係數 1
                C_i = 1.0
            else if(mod(i,2)==0)then              ! 偶數項點 係數 4
                C_i = 4.0
            else                                  ! 奇數項點 係數 2
                C_i= 2.0
            end if
          do j=1,inn_m_parti+1
            if(j==1 .OR. j==inn_m_parti+1)then ! 首項、末項點 係數 1
                C_j = 1.0
            else if(mod(j,2)==0)then          ! 偶數項點 係數 4
                C_j = 4.0
            else                              ! 奇數項點 係數 2
                C_j= 2.0
            end if
           do k=1,inn_m_parti+1
```

```
                    if(k==1 .OR. k==inn_m_parti+1)then   ! 首項、末項點 係數 1
                        C_k = 1.0
                    else if(mod(k,2)==0)then            ! 偶數項點 係數 4
                        C_k = 4.0
                    else                                ! 奇數項點 係數 2
                        C_k= 2.0
                    end if

                    ANS = ANS+ C_i*C_j*C_k *conjg(wave(h1,i,j,k))*wave(h2,i,j,k)
                end do
              end do
            end do

            ANS2 = ANS*((x_max-x_min)*(y_max-y_min)*(z_max-z_min))/((3.0*inn_m_parti)**(3.0))
!           write(*,*)h1,h2,ANS2

  return
end subroutine

!----計算動能 grad_phi(i,j,k)= ▽Φ* x▽Φ ----------!
subroutine sub5(h1,h2)
  use global
  implicit none
    integer :: i,j,k,h1,h2
    real(kind=8)::delt1,delt2,delt3 ! 格子點間的距離

    delt1 = (x_max-x_min)/inn_m_parti
    delt2 = (y_max-y_min)/inn_m_parti
    delt3 = (z_max-z_min)/inn_m_parti

            do i =2,inn_m_parti+2
              do j =2,inn_m_parti+2
                do k =2,inn_m_parti+2
                    grad_phiA(i,j,k) = conjg(wave(h1,i-1,j-1,k-1))
                    grad_phiB(i,j,k) = wave(h2,i-1,j-1,k-1)
                end do
```

```
                end do
              end do


!       !-- 給定陣列   othonormal_wave(i,:,:,:)=NEW_Φ(i,:,:,:)
              do i =1,inn_m_parti+1
               do j =1,inn_m_parti+1
                do k =1,inn_m_parti+1
                    grad_phi(i,j,k) = &
                & (grad_phiA(i+2,j+1,k+1)-grad_phiA(i,j+1,k+1))*&
                                                &(grad_phiB(i+2,j+1,k+1)-grad_phiB(i,j+1,k+1))/(4.0*delt1*delt1) + &
                & (grad_phiA(i+1,j+2,k+1)-grad_phiA(i+1,j,k+1))*&
                                                &(grad_phiB(i+1,j+2,k+1)-grad_phiB(i+1,j,k+1))/(4.0*delt2*delt2) + &
                & (grad_phiA(i+1,j+1,k+2)-grad_phiA(i+1,j+1,k))*&
                                                &(grad_phiB(i+1,j+1,k+2)-grad_phiB(i+1,j+1,k))/(4.0*delt3*delt3)
                end do
               end do
              end do
  return
end subroutine

!----計算 <Φil T lΦj>   3-D 辛普森積分----------!
subroutine sub6(ANS2)
use global
implicit none
      integer :: i,j,k
      real(kind=8)::C_i,C_j,C_k
      complex(kind=8):: ANS,ANS2

      ANS =(0.0,0.0)
      ! 辛普森三重積分 ∫∫∫Φi・Φj*   ; i,j=1～10
      do i=1,inn_m_parti+1
            if(i==1 .OR. i==inn_m_parti+1)then    ! 首項、末項點 係數 1
                  C_i = 1.0
            else if(mod(i,2)==0)then              ! 偶數項點 係數 4
                  C_i = 4.0
```

```
                else                                    ! 奇數項點 係數 2
                    C_i= 2.0
                end if
            do j=1,inn_m_parti+1
                if(j==1 .OR. j==inn_m_parti+1)then ! 首項、末項點 係數 1
                    C_j = 1.0
                else if(mod(j,2)==0)then          ! 偶數項點 係數 4
                    C_j = 4.0
                else                                    ! 奇數項點 係數 2
                    C_j= 2.0
                end if
              do k=1,inn_m_parti+1
                    if(k==1 .OR. k==inn_m_parti+1)then   ! 首項、末項點 係數 1
                        C_k = 1.0
                    else if(mod(k,2)==0)then            ! 偶數項點 係數 4
                        C_k = 4.0
                    else                                ! 奇數項點 係數 2
                        C_k= 2.0
                    end if

                    ANS = ANS+ C_i*C_j*C_k *grad_phi(i,j,k)
              end do
            end do
          end do

                ! 動能係數乘 0.5
                ANS2 = 0.5*ANS*((x_max-x_min)*(y_max-y_min)*(z_max-z_min))/((3.0*inn_m_parti)**(3.0))
!               write(*,*)h1,h2,ANS2

    return
end subroutine

!----把 VextF 寫成陣列 ----------!
subroutine sub7(h,w)
    use global
    implicit none
```

```
integer :: i,j,k,h,w,a1,b1,a2,b2,a3,b3
real(kind=8) ::i1,i2,i3,i4
real(kind=8),external::vext_H1F,vext_H2F,vext_H3F



i1 = (x_max-(inn_m_parti+1)*x_min)/(x_max-x_min)                    !  0.0 的位置
i2 = (inn_m_parti*(0.9584-x_min)+(x_max-x_min))/(x_max-x_min)     !  0.9584 的位置
i3 = (-inn_m_parti*(x_min+0.9584*cos(75.55*pi/180.0))+(x_max-x_min))/(x_max-x_min)
i4 = (x_max-(inn_m_parti+1)*x_min+inn_m_parti*0.9584*sin(75.55*pi/180.0))/(x_max-x_min)

a1 = CEILING(i1)+w
b1 = FLOOR(i1)-w

if(h==1)then
        EecX_max = xi(a1)
        EecX_min = xi(b1)
        EecY_max = yi(a1)
        EecY_min = yi(b1)
        EecZ_max = zi(a1)
        EecZ_min = zi(b1)

        !-- 給定陣列   vext(:,:,:)
        do i =1,inn_m_parti+1
          do j =1,inn_m_parti+1
            do k =1,inn_m_parti+1
                if((i>=(b1-1).and.i<=(a1-1)).and.(j>=(b1-1).and.j<=(a1-1)).and.(k>=(b1-1).and.k<=(a1-1)))then
                        vext(i,j,k) =0.0
                else
                        vext(i,j,k) = vext_H1F(xi(i),yi(j),zi(k))
                endif
            end do
          end do
        end do
elseif(h==2)then
        a2 = CEILING(i2)+w
        b2 = FLOOR(i2)-w
```

```
        EecX_max = xi(a2)
        EecX_min = xi(b2)
        EecY_max = yi(a1)
        EecY_min = yi(b1)
        EecZ_max = zi(a1)
        EecZ_min = zi(b1)

        !-- 給定陣列   vext(:,:,:)
        do i =1,inn_m_parti+1
          do j =1,inn_m_parti+1
            do k =1,inn_m_parti+1
                if((i>=(b2-1).and.i<=(a2-1)).and.(j>=(b1-1).and.j<=(a1-1)).and.(k>=(b1-1).and.k<=(a1-1)))then
                    vext(i,j,k) =0.0
                else
                    vext(i,j,k) = vext_H2F(xi(i),yi(j),zi(k))
                endif
            end do
          end do
        end do
elseif(h==3)then
        a2 = CEILING(i3)+w
        b2 = FLOOR(i3)-w
        a3 = CEILING(i4)+w
        b3 = FLOOR(i4)-w
        EecX_max = xi(a2)
        EecX_min = xi(b2)
        EecY_max = yi(a3)
        EecY_min = yi(b3)
        EecZ_max = zi(a1)
        EecZ_min = zi(b1)

        !-- 給定陣列   vext(:,:,:)
        do i =1,inn_m_parti+1
          do j =1,inn_m_parti+1
            do k =1,inn_m_parti+1
                if((i>=(b2-1).and.i<=(a2-1)).and.(j>=(b3-1).and.j<=(a3-1)).and.(k>=(b1-1).and.k<=(a1-1)))then
```

```
                    vext(i,j,k) =0.0
                else
                    vext(i,j,k) = vext_H3F(xi(i),yi(j),zi(k))
                endif
              end do
             end do
            end do
        else
            write(*,*)"Vext  範圍給定有錯誤!!"
        end if

   return
end subroutine

!----計算  <Φil Vext lΦj>   3-D 辛普森積分  -----!
subroutine sub8(h1,h2,ANS2)
use global
implicit none
        integer :: i,j,k,h1,h2
        real(kind=8)::C_i,C_j,C_k
        complex(kind=8):: ANS,ANS2

        ANS =(0.0,0.0)
        ! 辛普森三重積分  ∫∫∫Φi・Φj*   ; i,j=1～10
        do i=1,inn_m_parti+1
            if(i==1 .OR. i==inn_m_parti+1)then    ! 首項、末項點  係數  1
                C_i = 1.0
            else if(mod(i,2)==0)then              ! 偶數項點  係數  4
                C_i = 4.0
            else                                  ! 奇數項點  係數  2
                C_i= 2.0
            end if
          do j=1,inn_m_parti+1
              if(j==1 .OR. j==inn_m_parti+1)then ! 首項、末項點  係數  1
                C_j = 1.0
              else if(mod(j,2)==0)then           ! 偶數項點  係數  4
```

```
                C_j = 4.0
            else                                    ! 奇數項點 係數 2
                C_j= 2.0
            end if
        do k=1,inn_m_parti+1
            if(k==1 .OR. k==inn_m_parti+1)then   ! 首項、末項點 係數 1
                C_k = 1.0
            else if(mod(k,2)==0)then             ! 偶數項點 係數 4
                C_k = 4.0
            else                                    ! 奇數項點 係數 2
                C_k= 2.0
            end if

            ANS = ANS + C_i*C_j*C_k *vext(i,j,k)*conjg(wave(h1,i,j,k))*wave(h2,i,j,k)
        end do
      end do
    end do

        ANS2 = ANS*((x_max-x_min)*(y_max-y_min)*(z_max-z_min))/((3.0*inn_m_parti)**(3.0))

  return
end subroutine

!----------------給定 奇異點部分的 Vext 陣列值----------------------!
subroutine sub9(h)
  use global
  implicit none
    integer :: h,i,j,k
    real(kind=8),external::vext_H1F,vext_H2F,vext_H3F

    !-- 給定陣列   vextH1(:,:,:),vextH2(:,:,:)
    if(h==1)then
        do i =1,inn_m_parti+1
          do j =1,inn_m_parti+1
            do k =1,inn_m_parti+1
                vext(i,j,k) = vext_H1F(xi(i),yi(j),zi(k))
```

```
                end do
              end do
            end do
        elseif(h==2)then
            do i =1,inn_m_parti+1
              do j =1,inn_m_parti+1
                do k =1,inn_m_parti+1
                    vext(i,j,k) = vext_H2F(xi(i),yi(j),zi(k))
                end do
              end do
            end do
        elseif(h==3)then
            do i =1,inn_m_parti+1
              do j =1,inn_m_parti+1
                do k =1,inn_m_parti+1
                    vext(i,j,k) = vext_H3F(xi(i),yi(j),zi(k))
                end do
              end do
            end do
        else
            write(*,*)"給定 Vext 奇異點陣列 有錯!!"
        endif

    return
end subroutine

!----------計算 ε xc(n)----------------
subroutine sub12()
    use global
    implicit none
        real(kind=8):: ANS
        integer :: i,j,k,h1,h2

        do i =1,inn_m_parti+1
          do j =1,inn_m_parti+1
            do k =1,inn_m_parti+1
```

```
                Exc_N = (0.0,0.0)
                 do h1 = 1,basics
                  do h2 = 1,basics
                        Exc_N = Exc_N + C(h1)*C(h2)*conjg(wave(h1,i,j,k))*wave(h2,i,j,k)
                  end do
                 end do
                Exc_N2 = Exc_N
                call sub13(Exc_N2,ANS)
                Exc(i,j,k) = ANS
            end do
          end do
        end do
    return
end subroutine

!----------計算 ε xc(n)  輸入密度  決定 ε xc ----------------
subroutine sub13(input,ANS)
    use global
    implicit none
        real(kind=8):: input,ANS,ANS2,ANS3,rs
        integer :: i,j,k,h1,h2
        real(kind=8)::corR,corB1,corB2,corA,corB,corC,corD
                rs = (3.0/(4.0*pi*input))**(1.0/3.0)
                corR    = 0.1423
                corB1 = 1.0529
                corB2 = 0.3334
                corA =    0.0311
                corB = -0.0480
                corC =    0.0020
                corD = -0.0116

                if(rs>=1.0)then
                        ANS2 = -0.458/rs
                        ANS3 = -corR/(1.0+corB1*sqrt(rs)+corB2*rs)
                elseif(rs<1.0 .and. rs>0.0)then
                        ANS2 = -0.458/rs
```

```
                ANS3 = corA*log(rs)+corB+corC*rs*log(rs)+corD*rs
            elseif(rs==0.0 )then
                write(*,*)"發生錯誤: Exc 項  rs=0 ! "
            else
                write(*,*)"發生錯誤: Exc 項  rs<0 ! "
            end if
            ANS = ANS2+ANS3
   return
end subroutine

!----計算 <Φil  ε xc(n) lΦj>   3-D 辛普森積分  -----!
subroutine sub14(h1,h2,ANS2)
use global
implicit none
        integer :: i,j,k,h1,h2
        real(kind=8)::C_i,C_j,C_k
        complex(kind=8):: ANS,ANS2

        ANS =(0.0,0.0)
        ! 辛普森三重積分  ∫ ∫ ∫ Φi · Φj*   ; i,j=1～10
        do i=1,inn_m_parti+1
            if(i==1 .OR. i==inn_m_parti+1 )then    ! 首項、末項點  係數  1
                C_i = 1.0
            else if(mod(i,2)==0)then              ! 偶數項點  係數  4
                C_i = 4.0
            else                                  ! 奇數項點  係數  2
                C_i= 2.0
            end if
          do j=1,inn_m_parti+1
              if(j==1 .OR. j==inn_m_parti+1 )then ! 首項、末項點  係數  1
                C_j = 1.0
              else if(mod(j,2)==0)then            ! 偶數項點  係數  4
                C_j = 4.0
              else                                ! 奇數項點  係數  2
                C_j= 2.0
              end if
```

```
            do k=1,inn_m_parti+1
                if(k==1 .OR. k==inn_m_parti+1 )then   ! 首項、末項點 係數 1
                    C_k = 1.0
                else if(mod(k,2)==0)then              ! 偶數項點 係數 4
                    C_k = 4.0
                else                                  ! 奇數項點 係數 2
                    C_k= 2.0
                end if

                ANS = ANS + C_i*C_j*C_k *Exc(i,j,k)*conjg(wave(h1,i,j,k))*wave(h2,i,j,k)
            end do
          end do
        end do

            ANS2 = ANS*((x_max-x_min)*(y_max-y_min)*(z_max-z_min))/((3.0*inn_m_parti)**(3.0))
  return
end subroutine

!---- 正交歸一化  波函數陣列 ----------!
subroutine othonormal()
  use global
  implicit none
      real(kind=8):: ANS
      complex(kind=8):: ANS2
      integer :: i,j,k,h1,h2
!       open(unit=5,file='10orthonomal.txt')
!       open(unit=7,file='10wave_function.txt')

      call sub1()   ! 把 10 個波函數 寫成陣列
      call sub2(1,1,ANS2)   !   <Φ1|Φ1>
      ANS = ANS2
      wave(1,:,:,:) = wave(1,:,:,:)/sqrt(ANS)     ! 把Φ1 做 normalize 得到 NEWΦ1

      do h1 = 2,basics
        do h2 = 1,h1-1
            call sub2(h2,h1,ANS2)   !   <Φi|Φj>
```

```
                    do i =1,inn_m_parti+1
                     do j =1,inn_m_parti+1
                      do k =1,inn_m_parti+1
                            wave(h1,i,j,k) = wave(h1,i,j,k) -ANS2*wave(h2,i,j,k)
                      end do
                     end do
                    end do
            end do
                  call sub2(h1,h1,ANS2)   !   <ΦilΦi>
                  ANS = ANS2
                  wave(h1,:,:,:) = wave(h1,:,:,:)/sqrt(ANS)
          end do

!!!-----列出<Φi｜Φj> 結果!!
!         write(5,*) "測試 波函數是否滿足正交歸一!"
!         write(5,*) "Φi    Φj            <Φi｜Φj> 實部                          虛部 "
!                 do h1=1,basics
!                  do h2=1,basics
!                         if(h1>h2)cycle     ！共軛，所以只需算上三角
!                                 call sub2(h1,h2,ANS2)
!                                 write(5,'(I3,I5,E30.20,E30.20)') h1,h2,ANS2
!                  end do
!                 end do

!-----把離散化的波函數存入檔案
!         do h1 = 1,basics
!          do i =1,inn_m_parti+1
!           do j =1,inn_m_parti+1
!            do k =1,inn_m_parti+1
!                    write(7,*) wave(h1,i,j,k)
!            end do
!           end do
!          end do
!         end do

   return
```

```
end subroutine


!----計算 動能 ------------------------------!
subroutine calEkin()
   use global
   implicit none
        complex(kind=8):: ANS,ANS2,temA
        integer :: i,j,k,h1,h2
           open(unit=10,file='new_10Ekin.txt')        ! 最初先計算 基底函數對應的值 <Φi α | T | Φj β)>
        !   給訂積分範圍
        x_min = -10.0         ! 積分範圍 x_min→x_max
        x_max =    10.0       ! 積分範圍 x_min→x_max
        y_min = -10.0         ! 積分範圍 y_min→y_max
        y_max =    10.0       ! 積分範圍 y_min→y_max
        z_min = -10.0         ! 積分範圍 z_min→z_max
        z_max =    10.0       ! 積分範圍 z_min→z_max
        call othonormal()


        do h1 = 1,basics
         do h2 = 1,basics
              if(h1<=h2)then
                   call sub5(h1,h2)
                   call sub6(singleEkin(h1,h2))
              else
                   singleEkin(h1,h2) = conjg(singleEkin(h2,h1))
              end if
              !!!-----   列出動能<Φi | T | Φj>
              write(10,'(I2,I3,(E30.20,E30.20))') h1,h2,singleEkin(h1,h2)
          end do
        end do


  return
end subroutine



!----計算 Eec ------------------------------!
```

```
subroutine calEec(w)
  use global
  implicit none
      integer :: i,j,k,h1,h2,counter,counter2,w
      open(unit=20,file='new_10Eec.txt')        ! 最初先計算 基底函數對應的值 <Φi α | Vext | Φj β>
      open(unit=21,file='new_10EecH_A.txt')
      open(unit=22,file='new_10EecH_B.txt')
      open(unit=23,file='new_10EecH_C.txt')
      open(unit=24,file='new_10EecH_D.txt')
      open(unit=25,file='new_10EecH_E.txt')
      open(unit=26,file='new_10EecH_F.txt')

      counter2 = 1
      do counter=1,3
          if(counter==1)then
          !  給訂 H1-Vext 積分範圍
              x_max = 10.0          ! 積分範圍 x_min→x_max
              y_max = 10.0          ! 積分範圍 y_min→y_max
              z_max = 10.0          ! 積分範圍 z_min→z_max
              x_min = -9.999        ! 積分範圍 x_min→x_max
              y_min = -9.999        ! 積分範圍 y_min→y_max
              z_min = -9.999        ! 積分範圍 z_min→z_max
          elseif(counter==2)then
       !  給訂 H2-Vext 積分範圍
              x_max = 10.0          ! 積分範圍 x_min→x_max
              y_max = 10.0          ! 積分範圍 y_min→y_max
              z_max = 10.0          ! 積分範圍 z_min→z_max
              x_min = -9.999        ! 積分範圍 x_min→x_max
              y_min = -9.999        ! 積分範圍 y_min→y_max
              z_min = -9.999        ! 積分範圍 z_min→z_max
          else
      !  給訂 H3-Vext 積分範圍
              x_max = 10.0          ! 積分範圍 x_min→x_max
              y_max = 10.0          ! 積分範圍 y_min→y_max
              z_max = 10.0          ! 積分範圍 z_min→z_max
              x_min = -9.999        ! 積分範圍 x_min→x_max
```

```
            y_min = -9.999          ! 積分範圍 y_min→y_max
            z_min = -9.999          ! 積分範圍 z_min→z_max
        endif
 call othonormal()
 call sub7(counter,w)  ! 把 Vext 函數 寫成陣列

        do h1 = 1,basics
         do h2 = 1,basics
            if(h1<=h2)then      ! 只計算上三角
                call sub8(h1,h2,singleEec2(counter2,h1,h2))
            else
                singleEec2(counter2,h1,h2) = conjg(singleEec2(counter2,h2,h1))
            end if
         end do
        end do
            counter2=counter2+1

!   給訂奇異點積分範圍
x_max = EecX_max            ! 積分範圍 x_min→x_max
y_max = EecY_max            ! 積分範圍 y_min→y_max
z_max = EecZ_max            ! 積分範圍 z_min→z_max
x_min = EecX_min            ! 積分範圍 x_min→x_max
y_min = EecY_min            ! 積分範圍 y_min→y_max
z_min = EecZ_min            ! 積分範圍 z_min→z_max
call othonormal()
call sub9(counter)

        do h1 = 1,basics
         do h2 = 1,basics
            if(h1<=h2)then      ! 只計算上三角
                call sub8(h1,h2,singleEec2(counter2,h1,h2))
            else
                singleEec2(counter2,h1,h2) = conjg(singleEec2(counter2,h2,h1))
            end if
         end do
        end do
```

```
                counter2 = counter2+1
        end do

!!!-----   列出 Eec     <Φi｜Vext｜Φj>  存成檔案
        do h1=1,basics
         do h2=1,basics
                singleEec(h1,h2) = singleEec2(1,h1,h2)+singleEec2(2,h1,h2)+singleEec2(3,h1,h2)&
                        & + singleEec2(4,h1,h2)+singleEec2(5,h1,h2)+singleEec2(6,h1,h2)
                write(20,'(I2,I3,(E30.20,E30.20))') h1,h2,singleEec(h1,h2)
                write(21,'(I2,I3,(E30.20,E30.20))') h1,h2,singleEec2(1,h1,h2)
                write(22,'(I2,I3,(E30.20,E30.20))') h1,h2,singleEec2(2,h1,h2)
                write(23,'(I2,I3,(E30.20,E30.20))') h1,h2,singleEec2(3,h1,h2)
                write(24,'(I2,I3,(E30.20,E30.20))') h1,h2,singleEec2(4,h1,h2)
                write(25,'(I2,I3,(E30.20,E30.20))') h1,h2,singleEec2(5,h1,h2)
                write(26,'(I2,I3,(E30.20,E30.20))') h1,h2,singleEec2(6,h1,h2)
         end do
        end do

 return
end subroutine

!----計算  交換相關能  ------------------------------!
subroutine calExc()
   use global
   implicit none
        complex(kind=8):: ANS
        integer :: i,j,k,h1,h2

                call sub12()
        do h1 = 1,basics
         do h2 = 1,basics
                if(h1<=h2)then
                        call sub14(h1,h2,singleExc(h1,h2))
                else
                        singleExc(h1,h2) = conjg(singleExc(h2,h1))
                end if
```

```
      end do
    end do

  return
end subroutine

!----讀取 Eee 檔案  寫成<i,kl lj,h>-----!
subroutine calEee()
  use global
  implicit none
    integer :: i,j,k,h,counter,cou,cou2
    real(kind=8)::m1(BASICS,BASICS,BASICS,BASICS),m2(BASICS,BASICS,BASICS,BASICS),temA,temB

    open(unit=54,file='new_10Eee.txt') ! 讀取 <ΦhΦk｜Vee｜ΦjΦi)>
    open(unit=55,file='new_10EeeAC.txt',status='old') ! 讀取 <ΦhΦk｜Vee｜ΦjΦi)>
    open(unit=56,file='new_10EeeBD.txt',status='old') ! 讀取 <ΦhΦk｜Vee｜ΦjΦi)>
    !--------------計算上三角部分所需----------------
    counter=1
    do i =1,BASICS
      do j =i,BASICS
        IC(counter) =i
        JC(counter) =j
        counter =counter+1
      end do
    end do

    do cou    = 1,CA_M
      do cou2 = cou,CA_M
        read(55,'(TR13,E30.20)')temA     ! TR8:從第 8 個位置開始讀
        read(56,'(TR13,E30.20)')temB
          m1(IC(cou),JC(cou),IC(cou2),JC(cou2)) = temA - temB
          m2(IC(cou),JC(cou),IC(cou2),JC(cou2)) = temA + temB
      end do
    end do

    do i = 1,basics
```

```
do j = 1,basics
  do k = 1,basics
    do h = 1,basics
        if(min(i,j)<min(k,h))then
            if(i<=j.and.k<=h)then
                ijkh_Eee(i,j,k,h) = m1(i,j,k,h)
            elseif(i>j.and.k>h)then
                ijkh_Eee(i,j,k,h) = m1(j,i,h,k)
            elseif(i<=j.and.k>h)then
                ijkh_Eee(i,j,k,h) = m2(i,j,h,k)
            elseif(i>j.and.k<=h)then
                ijkh_Eee(i,j,k,h) = m2(j,i,k,h)
            else
                write(*,*) "Eee 讀取第一部分出錯!"
            endif
        elseif(min(i,j)==min(k,h))then
            if(max(i,j)<=max(k,h))then
                if(i<=j.and.k<=h)then
                    ijkh_Eee(i,j,k,h) = m1(i,j,k,h)
                elseif(i>j.and.k>h)then
                    ijkh_Eee(i,j,k,h) = m1(j,i,h,k)
                elseif(i<=j.and.k>h)then
                    ijkh_Eee(i,j,k,h) = m2(i,j,h,k)
                elseif(i>j.and.k<=h)then
                    ijkh_Eee(i,j,k,h) = m2(j,i,k,h)
                else
                    write(*,*) "Eee 讀取第二部分出錯!"
                endif
            else
                if(i<=j.and.k<=h)then
                    ijkh_Eee(i,j,k,h) = m1(k,h,i,j)
                elseif(i>j.and.k>h)then
                    ijkh_Eee(i,j,k,h) = m1(h,k,j,i)
                elseif(i<=j.and.k>h)then
                    ijkh_Eee(i,j,k,h) = m2(h,k,i,j)
                elseif(i>j.and.k<=h)then
```

```
                    ijkh_Eee(i,j,k,h) = m2(k,h,j,i)
                else
                    write(*,*) "Eee 讀取第三部分出錯!"
                endif
            end if
        else
            if(i<=j.and.k<=h)then
                ijkh_Eee(i,j,k,h) = m1(k,h,i,j)
            elseif(i>j.and.k>h)then
                ijkh_Eee(i,j,k,h) = m1(h,k,j,i)
            elseif(i<=j.and.k>h)then
                ijkh_Eee(i,j,k,h) = m2(h,k,i,j)
            elseif(i>j.and.k<=h)then
                ijkh_Eee(i,j,k,h) = m2(k,h,j,i)
            else
                write(*,*) "Eee 讀取第四部分出錯!"
            endif
        end if

        write(54,'(I2,I3,I3,I3,E30.20,E30.20)') i,j,k,h,ijkh_Eee(i,j,k,h) ! 存成檔案
      end do
    end do
  end do
 end do


  return
end subroutine
!----計算  total energy-----!
subroutine calTol()
  use global
  implicit none
    integer :: i,j,k,h


    E_tol = 0.0
    E_kin = 0.0
    Eec   = 0.0
```

```
          Eee     = 0.0
          Eexc    = 0.0

          do i = 1,basics
           do j = 1,basics
                E_kin = E_kin + C(i)*C(j)*singleEkin(i,j)
                Eec     = Eec     + C(i)*C(j)*singleEec(i,j)
                Eexc    = Eexc    + C(i)*C(j)*singleExc(i,j)
             do k = 1,basics
              do h = 1,basics
                Eee     = Eee     +     C(i)*C(j)*C(h)*C(k)*ijkh_Eee(k,h,i,j)
              end do
             end do
            end do
          end do
                E_tol = E_kin + Eec + Eexc + Eee
   return
end subroutine
```

```
!!!!!-----計算特徵值 特徵向量的副程式-----!!!
! rg ------------------------------------------!
         subroutine rg(nm,n,a,wr,wi,matz,z,iv1,fv1,ierr)
          integer n,nm,is1,is2,ierr,matz
          double precision a(nm,n),wr(n),wi(n),z(nm,n),fv1(n)
          integer iv1(n)

          if (n .le. nm) go to 10
          ierr = 10 * n
          go to 50
       10 call    balanc(nm,n,a,is1,is2,fv1)
          call    elmhes(nm,n,is1,is2,a,iv1)
          if (matz .ne. 0) go to 20
!         ......... find eigenvalues only .........
```

```
      call    hqr(nm,n,is1,is2,a,wr,wi,ierr)
      go to 50
!     ..... find both eigenvalues and eigenvectors .....
   20 call    eltran(nm,n,is1,is2,a,iv1,z)
      call    hqr2(nm,n,is1,is2,a,wr,wi,z,ierr)
      if (ierr .ne. 0) go to 50
      call    balbak(nm,n,is1,is2,fv1,n,z)
   50 return
      end
!hqr-----------------------------------------------------------
      subroutine hqr(nm,n,low,igh,h,wr,wi,ierr)
!    RESTORED CORRECT INDICES OF LOOPS (200,210,230,240). (9/29/89 BSG)
!
      integer i,j,k,l,m,n,en,ll,mm,na,nm,igh,itn,its,low,mp2,enm2,ierr
      double precision h(nm,n),wr(n),wi(n)
      double precision p,q,r,s,t,w,x,y,zz,norm,tst1,tst2
      logical notlas

      ierr = 0
      norm = 0.0d0
      k = 1
!     ..... store roots isolated by balanc
!                     and compute matrix norm .....
      do 50 i = 1, n
         do 40 j = k, n
   40    norm = norm + dabs(h(i,j))
         k = i
         if (i .ge. low .and. i .le. igh) go to 50
         wr(i) = h(i,i)
         wi(i) = 0.0d0
   50 continue
      en = igh
      t = 0.0d0
      itn = 30*n
!     ..... search for next eigenvalues .....
   60 if (en .lt. low) go to 1001
```

```
      its = 0
      na = en - 1
      enm2 = na - 1
!     ..... look for single small sub-diagonal element
!                  for l=en step -1 until low do -- .....
   70 do 80 ll = low, en
         l = en + low - ll
         if (l .eq. low) go to 100
         s = dabs(h(l-1,l-1)) + dabs(h(l,l))
         if (s .eq. 0.0d0) s = norm
         tst1 = s
         tst2 = tst1 + dabs(h(l,l-1))
         if (tst2 .eq. tst1) go to 100
   80 continue
!     ..... form shift .....
  100 x = h(en,en)
      if (l .eq. en) go to 270
      y = h(na,na)
      w = h(en,na) * h(na,en)
      if (l .eq. na) go to 280
      if (itn .eq. 0) go to 1000
      if (its .ne. 10 .and. its .ne. 20) go to 130
!     ..... form exceptional shift .....
      t = t + x
      do 120 i = low, en
  120 h(i,i) = h(i,i) - x
!
      s = dabs(h(en,na)) + dabs(h(na,enm2))
      x = 0.75d0 * s
      y = x
      w = -0.4375d0 * s * s
  130 its = its + 1
      itn = itn - 1
!     ..... look for two consecutive small
!                  sub-diagonal elements.
!                  for m=en-2 step -1 until l do -- .....
```

```
      do 140 mm = l, enm2
         m = enm2 + l - mm
         zz = h(m,m)
         r = x - zz
         s = y - zz
         p = (r * s - w) / h(m+1,m) + h(m,m+1)
         q = h(m+1,m+1) - zz - r - s
         r = h(m+2,m+1)
         s = dabs(p) + dabs(q) + dabs(r)
         p = p / s
         q = q / s
         r = r / s
         if (m .eq. l) go to 150
         tst1 = dabs(p)*(dabs(h(m-1,m-1)) + dabs(zz) + dabs(h(m+1,m+1)))
         tst2 = tst1 + dabs(h(m,m-1))*(dabs(q) + dabs(r))
         if (tst2 .eq. tst1) go to 150
  140 continue
  150 mp2 = m + 2
      do 160 i = mp2, en
         h(i,i-2) = 0.0d0
         if (i .eq. mp2) go to 160
         h(i,i-3) = 0.0d0
  160 continue
!     ..... double qr step involving rows l to en and
!                    columns m to en .....
      do 260 k = m, na
         notlas = k .ne. na
         if (k .eq. m) go to 170
         p = h(k,k-1)
         q = h(k+1,k-1)
         r = 0.0d0
         if (notlas) r = h(k+2,k-1)
         x = dabs(p) + dabs(q) + dabs(r)
         if (x .eq. 0.0d0) go to 260
         p = p / x
         q = q / x
```

```
            r = r / x
  170       s = dsign(dsqrt(p*p+q*q+r*r),p)
            if (k .eq. m) go to 180
            h(k,k-1) = -s * x
            go to 190
  180       if (l .ne. m) h(k,k-1) = -h(k,k-1)
  190       p = p + s
            x = p / s
            y = q / s
            zz = r / s
            q = q / p
            r = r / p
            if (notlas) go to 225
!           ..... row modification .....
            do 200 j = k, EN
               p = h(k,j) + q * h(k+1,j)
               h(k,j) = h(k,j) - p * x
               h(k+1,j) = h(k+1,j) - p * y
  200       continue
            j = min0(en,k+3)
!           ..... column modification .....
            do 210 i = L, j
               p = x * h(i,k) + y * h(i,k+1)
               h(i,k) = h(i,k) - p
               h(i,k+1) = h(i,k+1) - p * q
  210       continue
            go to 255
  225       continue
!           ..... row modification .....
            do 230 j = k, EN
               p = h(k,j) + q * h(k+1,j) + r * h(k+2,j)
               h(k,j) = h(k,j) - p * x
               h(k+1,j) = h(k+1,j) - p * y
               h(k+2,j) = h(k+2,j) - p * zz
  230       continue
            j = min0(en,k+3)
```

```
!        ..... column modification .....
               do 240 i = L, j
                  p = x * h(i,k) + y * h(i,k+1) + zz * h(i,k+2)
                  h(i,k) = h(i,k) - p
                  h(i,k+1) = h(i,k+1) - p * q
                  h(i,k+2) = h(i,k+2) - p * r
   240         continue
   255         continue
   260 continue
            go to 70
!        ..... one root found .....
   270 wr(en) = x + t
            wi(en) = 0.0d0
            en = na
            go to 60
!        ..... two roots found .....
   280 p = (y - x) / 2.0d0
            q = p * p + w
            zz = dsqrt(dabs(q))
            x = x + t
            if (q .lt. 0.0d0) go to 320
!        ..... real pair .....
            zz = p + dsign(zz,p)
            wr(na) = x + zz
            wr(en) = wr(na)
            if (zz .ne. 0.0d0) wr(en) = x - w / zz
            wi(na) = 0.0d0
            wi(en) = 0.0d0
            go to 330
!        ..... complex pair .....
   320 wr(na) = x + p
            wr(en) = x + p
            wi(na) = zz
            wi(en) = -zz
   330 en = enm2
            go to 60
```

```
!          ..... set error -- all eigenvalues have not
!                              converged after 30*n iterations .....
  1000 ierr = en
  1001 return
            end
!hqr2----------------------------------------------
            subroutine hqr2(nm,n,low,igh,h,wr,wi,z,ierr)
            integer i,j,k,l,m,n,en,ii,jj,ll,mm,na,nm,nn,&
           &                igh,itn,its,low,mp2,enm2,ierr
            double precision h(nm,n),wr(n),wi(n),z(nm,n)
            double precision p,q,r,s,t,w,x,y,ra,sa,vi,vr,zz,norm,tst1,tst2
            logical notlas
! eltran        ------------------------------------------------------------------
            ierr = 0
            norm = 0.0d0
            k = 1
!           ..... store roots isolated by balanc
!                              and compute matrix norm .....
            do 50 i = 1, n
               do 40 j = k, n
      40       norm = norm + dabs(h(i,j))
               k = i
               if (i .ge. low .and. i .le. igh) go to 50
               wr(i) = h(i,i)
               wi(i) = 0.0d0
      50    continue
            en = igh
            t = 0.0d0
            itn = 30*n
!           ..... search for next eigenvalues .....
      60    if (en .lt. low) go to 340
            its = 0
            na = en - 1
            enm2 = na - 1
!           ..... look for single small sub-diagonal element
!                              for l=en step -1 until low do -- .....
```

```
   70 do 80 ll = low, en
         l = en + low - ll
         if (l .eq. low) go to 100
         s = dabs(h(l-1,l-1)) + dabs(h(l,l))
         if (s .eq. 0.0d0) s = norm
         tst1 = s
         tst2 = tst1 + dabs(h(l,l-1))
         if (tst2 .eq. tst1) go to 100
   80 continue
!        ..... form shift .....
  100 x = h(en,en)
         if (l .eq. en) go to 270
         y = h(na,na)
         w = h(en,na) * h(na,en)
         if (l .eq. na) go to 280
         if (itn .eq. 0) go to 1000
         if (its .ne. 10 .and. its .ne. 20) go to 130
!        ..... form exceptional shift .....
         t = t + x
         do 120 i = low, en
  120 h(i,i) = h(i,i) - x
         s = dabs(h(en,na)) + dabs(h(na,enm2))
         x = 0.75d0 * s
         y = x
         w = -0.4375d0 * s * s
  130 its = its + 1
         itn = itn - 1
!        ..... look for two consecutive small
!                    sub-diagonal elements.
!                    for m=en-2 step -1 until l do -- .....
         do 140 mm = l, enm2
            m = enm2 + l - mm
            zz = h(m,m)
            r = x - zz
            s = y - zz
            p = (r * s - w) / h(m+1,m) + h(m,m+1)
```

```
            q = h(m+1,m+1) - zz - r - s
            r = h(m+2,m+1)
            s = dabs(p) + dabs(q) + dabs(r)
            p = p / s
            q = q / s
            r = r / s
            if (m .eq. l) go to 150
            tst1 = dabs(p)*(dabs(h(m-1,m-1)) + dabs(zz) + dabs(h(m+1,m+1)))
            tst2 = tst1 + dabs(h(m,m-1))*(dabs(q) + dabs(r))
            if (tst2 .eq. tst1) go to 150
  140 continue
  150 mp2 = m + 2
        do 160 i = mp2, en
            h(i,i-2) = 0.0d0
            if (i .eq. mp2) go to 160
            h(i,i-3) = 0.0d0
  160 continue
!       ..... double qr step involving rows l to en and
!                       columns m to en .....
        do 260 k = m, na
            notlas = k .ne. na
            if (k .eq. m) go to 170
            p = h(k,k-1)
            q = h(k+1,k-1)
            r = 0.0d0
            if (notlas) r = h(k+2,k-1)
            x = dabs(p) + dabs(q) + dabs(r)
            if (x .eq. 0.0d0) go to 260
            p = p / x
            q = q / x
            r = r / x
  170       s = dsign(dsqrt(p*p+q*q+r*r),p)
            if (k .eq. m) go to 180
            h(k,k-1) = -s * x
            go to 190
  180       if (l .ne. m) h(k,k-1) = -h(k,k-1)
```

```
  190     p = p + s
          x = p / s
          y = q / s
          zz = r / s
          q = q / p
          r = r / p
          if (notlas) go to 225
!         ..... row modification .....
          do 200 j = k, n
             p = h(k,j) + q * h(k+1,j)
             h(k,j) = h(k,j) - p * x
             h(k+1,j) = h(k+1,j) - p * y
  200     continue
          j = min0(en,k+3)
!         ..... column modification .....
          do 210 i = 1, j
             p = x * h(i,k) + y * h(i,k+1)
             h(i,k) = h(i,k) - p
             h(i,k+1) = h(i,k+1) - p * q
  210     continue
!         ..... accumulate transformations .....
          do 220 i = low, igh
             p = x * z(i,k) + y * z(i,k+1)
             z(i,k) = z(i,k) - p
             z(i,k+1) = z(i,k+1) - p * q
  220     continue
          go to 255
  225     continue
!         ..... row modification .....
          do 230 j = k, n
             p = h(k,j) + q * h(k+1,j) + r * h(k+2,j)
             h(k,j) = h(k,j) - p * x
             h(k+1,j) = h(k+1,j) - p * y
             h(k+2,j) = h(k+2,j) - p * zz
  230     continue
          j = min0(en,k+3)
```

```
!         ..... column modification .....
              do 240 i = 1, j
                 p = x * h(i,k) + y * h(i,k+1) + zz * h(i,k+2)
                 h(i,k) = h(i,k) - p
                 h(i,k+1) = h(i,k+1) - p * q
                 h(i,k+2) = h(i,k+2) - p * r
  240         continue
!         ..... accumulate transformations .....
              do 250 i = low, igh
                 p = x * z(i,k) + y * z(i,k+1) + zz * z(i,k+2)
                 z(i,k) = z(i,k) - p
                 z(i,k+1) = z(i,k+1) - p * q
                 z(i,k+2) = z(i,k+2) - p * r
  250         continue
  255         continue
  260 continue
              go to 70
!         ..... one root found .....
  270 h(en,en) = x + t
              wr(en) = h(en,en)
              wi(en) = 0.0d0
              en = na
              go to 60
!         ..... two roots found .....
  280 p = (y - x) / 2.0d0
              q = p * p + w
              zz = dsqrt(dabs(q))
              h(en,en) = x + t
              x = h(en,en)
              h(na,na) = y + t
              if (q .lt. 0.0d0) go to 320
!         ..... real pair .....
              zz = p + dsign(zz,p)
              wr(na) = x + zz
              wr(en) = wr(na)
              if (zz .ne. 0.0d0) wr(en) = x - w / zz
```

```
            wi(na) = 0.0d0
            wi(en) = 0.0d0
            x = h(en,na)
            s = dabs(x) + dabs(zz)
            p = x / s
            q = zz / s
            r = dsqrt(p*p+q*q)
            p = p / r
            q = q / r
!           ..... row modification .....
            do 290 j = na, n
               zz = h(na,j)
               h(na,j) = q * zz + p * h(en,j)
               h(en,j) = q * h(en,j) - p * zz
  290 continue
!           ..... column modification .....
            do 300 i = 1, en
               zz = h(i,na)
               h(i,na) = q * zz + p * h(i,en)
               h(i,en) = q * h(i,en) - p * zz
  300 continue
!           ..... accumulate transformations .....
            do 310 i = low, igh
               zz = z(i,na)
               z(i,na) = q * zz + p * z(i,en)
               z(i,en) = q * z(i,en) - p * zz
  310 continue
            go to 330
!           ..... complex pair .....
  320 wr(na) = x + p
            wr(en) = x + p
            wi(na) = zz
            wi(en) = -zz
  330 en = enm2
            go to 60
!           ..... all roots found.    backsubstitute to find
```

```
!                       vectors of upper triangular form .....
  340 if (norm .eq. 0.0d0) go to 1001
!       ..... for en=n step -1 until 1 do -- .....
        do 800 nn = 1, n
           en = n + 1 - nn
           p = wr(en)
           q = wi(en)
           na = en - 1
           if (q) 710, 600, 800
!       ..... real vector .....
  600      m = en
           h(en,en) = 1.0d0
           if (na .eq. 0) go to 800
!       ..... for i=en-1 step -1 until 1 do -- .....
           do 700 ii = 1, na
              i = en - ii
              w = h(i,i) - p
              r = 0.0d0
              do 610 j = m, en
  610         r = r + h(i,j) * h(j,en)
              if (wi(i) .ge. 0.0d0) go to 630
              zz = w
              s = r
              go to 700
  630         m = i
              if (wi(i) .ne. 0.0d0) go to 640
              t = w
              if (t .ne. 0.0d0) go to 635
                 tst1 = norm
                 t = tst1
  632            t = 0.01d0 * t
                 tst2 = norm + t
                 if (tst2 .gt. tst1) go to 632
  635         h(i,en) = -r / t
              go to 680
!       ..... solve real equations .....
```

```
  640           x = h(i,i+1)
                y = h(i+1,i)
                q = (wr(i) - p) * (wr(i) - p) + wi(i) * wi(i)
                t = (x * s - zz * r) / q
                h(i,en) = t
                if (dabs(x) .le. dabs(zz)) go to 650
                h(i+1,en) = (-r - w * t) / x
                go to 680
  650           h(i+1,en) = (-s - y * t) / zz
!       ..... overflow control .....
  680           t = dabs(h(i,en))
                if (t .eq. 0.0d0) go to 700
                tst1 = t
                tst2 = tst1 + 1.0d0/tst1
                if (tst2 .gt. tst1) go to 700
                do 690 j = i, en
                   h(j,en) = h(j,en)/t
  690           continue
  700       continue
!       ..... end real vector .....
            go to 800
!       ..... complex vector .....
  710       m = na
!       ..... last vector component chosen imaginary so that
!                       eigenvector matrix is triangular .....
            if (dabs(h(en,na)) .le. dabs(h(na,en))) go to 720
            h(na,na) = q / h(en,na)
            h(na,en) = -(h(en,en) - p) / h(en,na)
            go to 730
  720       call cdiv(0.0d0,-h(na,en),h(na,na)-p,q,h(na,na),h(na,en))
  730       h(en,na) = 0.0d0
            h(en,en) = 1.0d0
            enm2 = na - 1
            if (enm2 .eq. 0) go to 800
!       ..... for i=en-2 step -1 until 1 do -- .....
            do 795 ii = 1, enm2
```

```
                i = na - ii
                w = h(i,i) - p
                ra = 0.0d0
                sa = 0.0d0
                do 760 j = m, en
                    ra = ra + h(i,j) * h(j,na)
                    sa = sa + h(i,j) * h(j,en)
  760           continue
                if (wi(i) .ge. 0.0d0) go to 770
                zz = w
                r = ra
                s = sa
                go to 795
  770           m = i
                if (wi(i) .ne. 0.0d0) go to 780
                call cdiv(-ra,-sa,w,q,h(i,na),h(i,en))
                go to 790
!       ..... solve complex equations .....
  780           x = h(i,i+1)
                y = h(i+1,i)
                vr = (wr(i) - p) * (wr(i) - p) + wi(i) * wi(i) - q * q
                vi = (wr(i) - p) * 2.0d0 * q
                if (vr .ne. 0.0d0 .or. vi .ne. 0.0d0) go to 784
                    tst1 = norm * (dabs(w) + dabs(q) + dabs(x) &
     &                                 + dabs(y) + dabs(zz))
                    vr = tst1
  783               vr = 0.01d0 * vr
                    tst2 = tst1 + vr
                    if (tst2 .gt. tst1) go to 783
  784           call cdiv(x*r-zz*ra+q*sa,x*s-zz*sa-q*ra,vr,vi,&
     &                        h(i,na),h(i,en))
                if (dabs(x) .le. dabs(zz) + dabs(q)) go to 785
                h(i+1,na) = (-ra - w * h(i,na) + q * h(i,en)) / x
                h(i+1,en) = (-sa - w * h(i,en) - q * h(i,na)) / x
                go to 790
  785           call cdiv(-r-y*h(i,na),-s-y*h(i,en),zz,q,&
```

```
     &                    h(i+1,na),h(i+1,en))
!       ..... overflow control .....
  790       t = dmax1(dabs(h(i,na)), dabs(h(i,en)))
            if (t .eq. 0.0d0) go to 795
            tst1 = t
            tst2 = tst1 + 1.0d0/tst1
            if (tst2 .gt. tst1) go to 795
            do 792 j = i, en
               h(j,na) = h(j,na)/t
               h(j,en) = h(j,en)/t
  792       continue
  795    continue
!       ..... end complex vector .....
  800 continue
!       ..... end back substitution.
!                    vectors of isolated roots .....
       do 840 i = 1, n
          if (i .ge. low .and. i .le. igh) go to 840
          do 820 j = i, n
  820     z(i,j) = h(i,j)
  840 continue
!       ..... multiply by transformation matrix to give
!                    vectors of original full matrix.
!                    for j=n step -1 until low do -- .....
       do 880 jj = low, n
          j = n + low - jj
          m = min0(j,igh)
          do 880 i = low, igh
             zz = 0.0d0
             do 860 k = low, m
  860        zz = zz + z(i,k) * h(k,j)
             z(i,j) = zz
  880 continue
       go to 1001
!       ..... set error -- all eigenvalues have not
!                    converged after 30*n iterations .....
```

```
 1000 ierr = en
 1001 return
      end
!---------------------------
      subroutine eltran(nm,n,low,igh,a,int,z)
      integer i,j,n,kl,mm,mp,nm,igh,low,mp1
      double precision a(nm,igh),z(nm,n)
      integer int(igh)
!     ..... initialize z to identity matrix .....
      do 80 j = 1, n
         do 60 i = 1, n
   60    z(i,j) = 0.0d0
!
         z(j,j) = 1.0d0
   80 continue
      kl = igh - low - 1
      if (kl .lt. 1) go to 200
!     ..... for mp=igh-1 step -1 until low+1 do -- .....
      do 140 mm = 1, kl
         mp = igh - mm
         mp1 = mp + 1
         do 100 i = mp1, igh
  100    z(i,mp) = a(i,mp-1)
         i = int(mp)
         if (i .eq. mp) go to 140
         do 130 j = mp, igh
            z(mp,j) = z(i,j)
            z(i,j) = 0.0d0
  130    continue
         z(i,mp) = 1.0d0
  140 continue
  200 return
      end
! elmhes--------------------
      subroutine elmhes(nm,n,low,igh,a,int)
      integer i,j,m,n,la,nm,igh,kp1,low,mm1,mp1
```

```
      double precision a(nm,n)
      double precision x,y
      integer int(igh)

      la = igh - 1
      kp1 = low + 1
      if (la .lt. kp1) go to 200
!
      do 180 m = kp1, la
         mm1 = m - 1
         x = 0.0d0
         i = m
         do 100 j = m, igh
            if (dabs(a(j,mm1)) .le. dabs(x)) go to 100
            x = a(j,mm1)
            i = j
  100    continue
         int(m) = i
         if (i .eq. m) go to 130
!        ..... interchange rows and columns of a .....
         do 110 j = mm1, n
            y = a(i,j)
            a(i,j) = a(m,j)
            a(m,j) = y
  110    continue
         do 120 j = 1, igh
            y = a(j,i)
            a(j,i) = a(j,m)
            a(j,m) = y
  120    continue
!        ..... end interchange .....
  130    if (x .eq. 0.0d0) go to 180
         mp1 = m + 1
         do 160 i = mp1, igh
            y = a(i,mm1)
            if (y .eq. 0.0d0) go to 160
```

```
                y = y / x
                a(i,mm1) = y
                do 140 j = m, n
  140           a(i,j) = a(i,j) - y * a(m,j)
                do 150 j = 1, igh
  150           a(j,m) = a(j,m) + y * a(j,i)
  160       continue
  180 continue
  200 return
        end
! cdiv--------------------------
        subroutine cdiv(ar,ai,br,bi,cr,ci)
        double precision ar,ai,br,bi,cr,ci
!       complex division, (cr,ci) = (ar,ai)/(br,bi)
        double precision s,ars,ais,brs,bis
        s = dabs(br) + dabs(bi)
        ars = ar/s
        ais = ai/s
        brs = br/s
        bis = bi/s
        s = brs**2 + bis**2
        cr = (ars*brs + ais*bis)/s
        ci = (ais*brs - ars*bis)/s
        return
        end
! balbak-------------------------
        subroutine balbak(nm,n,low,igh,scale,m,z)
        integer i,j,k,m,n,ii,nm,igh,low
        double precision scale(n),z(nm,m)
        double precision s

        if (m .eq. 0) go to 200
        if (igh .eq. low) go to 120
        do 110 i = low, igh
           s = scale(i)
!       ..... left hand eigenvectors are back transformed
```

```
!                       if the foregoing statement is replaced by
!                       s=1.0d0/scale(i). .........
            do 100 j = 1, m
  100       z(i,j) = z(i,j) * s
  110 continue
!       ......... for i=low-1 step -1 until 1,
!                       igh+1 step 1 until n do -- .........
  120 do 140 ii = 1, n
            i = ii
            if (i .ge. low .and. i .le. igh) go to 140
            if (i .lt. low) i = low - ii
            k = scale(i)
            if (k .eq. i) go to 140
            do 130 j = 1, m
               s = z(i,j)
               z(i,j) = z(k,j)
               z(k,j) = s
  130       continue
  140 continue
  200 return
        end
! balanc-------------------------
        subroutine balanc(nm,n,a,low,igh,scale)
        integer i,j,k,l,m,n,jj,nm,igh,low,iexc
        double precision a(nm,n),scale(n)
        double precision c,f,g,r,s,b2,radix
        logical noconv

        radix = 16.0d0
        b2 = radix * radix
        k = 1
        l = n
        go to 100
!       ......... in-line procedure for row and
!                       column exchange .........
     20 scale(m) = j
```

```
            if (j .eq. m) go to 50
            do 30 i = 1, l
                f = a(i,j)
                a(i,j) = a(i,m)
                a(i,m) = f
         30 continue
            do 40 i = k, n
                f = a(j,i)
                a(j,i) = a(m,i)
                a(m,i) = f
         40 continue
         50 go to (80,130), iexc
!           ..... search for rows isolating an eigenvalue
!                       and push them down .....
         80 if (l .eq. 1) go to 280
            l = l - 1
!           ..... for j=l step -1 until 1 do -- .....
        100 do 120 jj = 1, l
                j = l + 1 - jj
                do 110 i = 1, l
                    if (i .eq. j) go to 110
                    if (a(j,i) .ne. 0.0d0) go to 120
        110         continue
                m = l
                iexc = 1
                go to 20
        120 continue
            go to 140
!           ..... search for columns isolating an eigenvalue
!                       and push them left .....
        130 k = k + 1
        140 do 170 j = k, l
                do 150 i = k, l
                    if (i .eq. j) go to 150
                    if (a(i,j) .ne. 0.0d0) go to 170
        150         continue
```

```
            m = k
            iexc = 2
            go to 20
  170 continue
!         ..... now balance the submatrix in rows k to l .....
          do 180 i = k, l
  180 scale(i) = 1.0d0
!         ..... iterative loop for norm reduction .....
  190 noconv = .false.
          do 270 i = k, l
              c = 0.0d0
              r = 0.0d0
              do 200 j = k, l
                  if (j .eq. i) go to 200
                  c = c + dabs(a(j,i))
                  r = r + dabs(a(i,j))
  200         continue
!         ..... guard against zero c or r due to underflow .....
              if (c .eq. 0.0d0 .or. r .eq. 0.0d0) go to 270
              g = r / radix
              f = 1.0d0
              s = c + r
  210         if (c .ge. g) go to 220
              f = f * radix
              c = c * b2
              go to 210
  220         g = r * radix
  230         if (c .lt. g) go to 240
              f = f / radix
              c = c / b2
              go to 230
!         ..... now balance .....
  240         if ((c + r) / f .ge. 0.95d0 * s) go to 270
              g = 1.0d0 / f
              scale(i) = scale(i) * f
              noconv = .true.
```

```
            do 250 j = k, n
250         a(i,j) = a(i,j) * g
            do 260 j = 1, l
260         a(j,i) = a(j,i) * f
270 continue
        if (noconv) go to 190
280 low = k
        igh = l
        return
        end
```

A.2 The program for solving Poisson's equation

```
!-------------------------------------------------------------------!
!                解 poission 方程  ΔU=Φi・Φj   ; i,j=1...17          !
!                                                                   !
! 即 (D/DX)(DU/DX)+(D/DY)(DU/DY)+(D/DZ)(DU/DZ)+λ*U = F(X,Y,Z)       !
!-------------------------------------------------------------------!

module global
  implicit none
        real,parameter :: a0 = 0.529189379          ! Bohr radius  單位:Å
        real,parameter :: pi = 3.141592653589      ! 圓周率π
        real,parameter :: x_max =    10.0           ! 積分範圍  x_min→x_max
        real,parameter :: y_max =    10.0           ! 積分範圍  y_min→y_max
        real,parameter :: z_max =    10.0           ! 積分範圍  z_min→z_max
        real,parameter :: x_min = -10.0             ! 積分範圍  x_min→x_max
        real,parameter :: y_min = -10.0             ! 積分範圍  y_min→y_max
        real,parameter :: z_min = -10.0             ! 積分範圍  z_min→z_max
        integer,parameter :: inn_m_parti=200      ! 內積積分-辛普森法  切割數(必須爲偶數)
        integer,parameter :: basics = 10           ! 基底函數  數量
        complex       :: ijkh_Eee(basics,basics,basics,basics)
        real::xi(inn_m_parti+1),yi(inn_m_parti+1),zi(inn_m_parti+1)
```

```
        complex,allocatable::wave(:,:,:,:) ! Φi,i=1,10
        real,allocatable::real_wave(:,:,:,:) ! Φi,i=1,10
        real,allocatable::image_wave(:,:,:,:) ! Φi,i=1,10
end module

!------------------------主程式------------------------------
program Sol_Poisson
use global
implicit none

integer,parameter   :: L=inn_m_parti,M=inn_m_parti, N=inn_m_parti    ! 分別爲 X,Y,Z 範圍的分割數目
integer,parameter   :: ELMBDA =0   ! Poisson 方程中的λ係數
integer             :: LBDCND,MBDCND,NBDCND ! 分別爲在 X,Y,Z 邊界點的"條件"
integer             :: I,J,K,H,LDIMF,MDIMF,IERROR
real        :: DX,DY,DZ,PERTRB      ! 前三者分別爲 X,Y,Z 上 每個區間長
real,parameter   :: XS=x_min,XF=x_max,YS=y_min,YF=y_max,ZS=z_min,ZF=z_max ! 此爲 X,Y,Z 範圍[S,F]
real,allocatable:: F(:,:,:),BDXS(:,:),BDXF(:,:),BDYS(:,:),BDYF(:,:),BDZS(:,:),BDZF(:,:) ! F 值、邊界點的值
real,allocatable:: W(:),X(:),Y(:),Z(:)
real,allocatable:: U(:,:,:),single_Eee(:,:,:,:)
integer             :: cou,cou2,cou3,cou4,counter
integer,parameter   :: CA_M =(1+BASICS)*BASICS/2        ! 計算上三角的總數
integer             :: IC(CA_M),JC(CA_M)                ! 計算上三角
real        :: Eee,Ci,Cj,Ck,r,theta,phi         ! Ci,Cj,Ck 爲辛普森積分的係數
complex   ::      temA

! 給定下列陣列大小
allocate(F(L+1,M+1,N+1),BDXS(M+1,N+1),BDXF(M+1,N+1),BDYS(L+1,N+1),BDYF(L+1,N+1),BDZS(L+1,M+1),BDZF(L+1,M+1))
allocate(X(L+1),Y(M+1),Z(N+1),U(L+1,M+1,N+1),single_Eee(BASICS,BASICS,BASICS,BASICS))
allocate(wave(basics,inn_m_parti+1,inn_m_parti+1,inn_m_parti+1))
allocate(real_wave(basics,inn_m_parti+1,inn_m_parti+1,inn_m_parti+1))
allocate(image_wave(basics,inn_m_parti+1,inn_m_parti+1,inn_m_parti+1))

I =30+L+M+5*N+MAX(L,M,N)+7*(INT((L+1)/2))+INT((M+1)/2)
allocate(W(I))   ! 1-D work space
!-----------------------------------------------------------------!
```

```
open(unit=10,file='10wave_function.txt',status='old')
open(unit=20,file='10new_10Poi_sol.txt')        ! U 乘函數 F                    存到此檔案
open(unit=30,file='10new_10Eee_TRI.txt') ! 對 U*F 做 3 維積分          存到此檔案
open(unit=40,file='10new_10Eee.txt')

DX = (XF-XS)/dble(L) ! [XS,XF]  分割 L 份後  每個區間長
DY = (YF-YS)/dble(M) ! [YS,YF]  分割 M 份後  每個區間長
DZ = (ZF-ZS)/dble(N) ! [ZS,ZF]  分割 N 份後  每個區間長

LBDCND = 1 ! 給值 1  表示 U 在 XS,XF 各有特定值
MBDCND = 1 ! 給值 1  表示 U 在 YS,YF 各有特定值
NBDCND = 1 ! 給值 1  表示 U 在 ZS,ZF 各有特定值

PERTRB = 0.0 ! 讓程式 能找到有意義的解
LDIMF = L+1
MDIMF = M+1

!--------------給訂 Domain 上  格子點的值
do I=1,L+1
do J=1,M+1
do K=1,N+1
        X(I) = XS + (I-1)*DX ! X 方向分割後  每個格子點的值
      Y(J) = YS + (J-1)*DY ! Y 方向分割後  每個格子點的值
        Z(K) = ZS + (K-1)*DZ ! Z 方向分割後  每個格子點的值
end do
end do
end do
!-------------------------------------------------!
!         call othonormal()     ! 正交歸一化  波函數

!-----讀取已經  寫好的 wave function  值  (分割數=100)----!
        do h = 1,basics
                do i = 1,inn_m_parti+1
                do j = 1,inn_m_parti+1
                do k = 1,inn_m_parti+1
                        read(10,*)temA
```

```
                    wave(h,i,j,k) = temA
                end do
                end do
                end do
        end do

        do i= 1,basics
                real_wave(i,:,:,:) = wave(i,:,:,:)
                image_wave(i,:,:,:) = aimag(wave(i,:,:,:))
        end do

!--------------計算上三角部分所需----------------
        counter=1
        do i =1,BASICS
        do j =i,BASICS
                IC(counter) =i
                JC(counter) =j
                counter =counter+1
        end do
        end do

!-----給定邊界上的值   (邊界  = 0)
do J=1,M+1
do K=1,N+1
        F(1,J,K)    = 0.0
        F(L+1,J,K)= 0.0
end do
end do

do I=1,L+1
do J=1,M+1
        F(I,J,1)    = 0.0
        F(I,J,N+1)= 0.0
end do
end do
```

```
do I=1,L+1
do K=1,N+1
        F(I,1,K)    = 0.0
        F(I,M+1,K)= 0.0
end do
end do
!--------解 Poisson 方程之前 先給定等號右邊的函數 F
        do cou    = 1,CA_M
          do I=2,L
                do J=2,M
                        do K =2,N
!       Eee 拆實部&虛部   : (A) R(i)R(k)+I(i)I(k)
        F(I,J,K) =-4.0*pi*(real_wave(IC(cou),I,J,K)*real_wave(JC(cou),I,J,K)&
                        & + image_wave(IC(cou),I,J,K)*image_wave(JC(cou),I,J,K))
!                               (B) R(i)I(k)-I(i)R(k)
!       F(I,J,K) =-4.0*pi*(real_wave(IC(cou),I,J,K)*image_wave(JC(cou),I,J,K)&
!                       & - image_wave(IC(cou),I,J,K)*real_wave(JC(cou),I,J,K))
                        end do
                end do
          end do

! call HW3CRT 解 poission 方程， U = ∫∫∫Φi・Φj/｜r-r'｜
call HW3CRT(XS,XF,L,LBDCND,BDXS,BDXF,YS,YF,M,MBDCND,BDYS,BDYF,&
                & ZS,ZF,N,NBDCND,BDZS,BDZF,ELMBDA,LDIMF,MDIMF,F,PERTRB,IERROR,W)

do cou2 = cou,CA_M
! 把上面得到的 U 再乘上函數Φk、Φh
        U(:,:,:) = 0.0
        do I=2,L
                do J=2,M
                        do K =2,N
!     Eee 拆實部&虛部   : (C) R(j)R(h)+I(j)I(h)
                        U(I,J,K) = F(I,J,K)*(real_wave(IC(cou2),I,J,K)*real_wave(JC(cou2),I,J,K)&
                                                & + image_wave(IC(cou2),I,J,K)*image_wave(JC(cou2),I,J,K))
!                               (D) R(j)I(h)-I(j)R(h)
!                       U(I,J,K) = F(I,J,K)*(real_wave(IC(cou2),I,J,K)*image_wave(JC(cou2),I,J,K)&
```

```
!                                    & - image_wave(IC(cou2),I,J,K)*real_wave(JC(cou2),I,J,K))
                        end do
                end do
        end do
! write(20,*)"U= F*basis ", U ! 把 Poisson 方程的解 U 乘上Φi、Φj 存到 Poi_solxF.txt 檔
!                                    ! 上面即爲 ∫Vee · Φi · Φj = ∫n(r) · Φi · Φj/｜r-r'｜

!---------- (3 維)辛普森積分法 算 Eee 項
Eee = 0.0
do I=1,L+1
                if(I==1 .OR. I==L+1)then        ! 首項和末項 係數 1
                Ci = 1.0
                else if(mod(I,2)==0)then        ! 偶數項 係數 4
                Ci = 4.0
                else                                    ! 奇數項 係數 2
                Ci= 2.0
                end if
    do J=1,M+1
                        if(J==1 .OR. J==M+1)then    ! 首項和末項 係數 1
                        Cj = 1.0
                        else if(mod(J,2)==0)then    ! 偶數項 係數 4
                        Cj = 4.0
                        else                                ! 奇數項 係數 2
                        Cj= 2.0
                        end if
                do K =1,N+1
                                if(K==1 .OR. K==N+1)then   ! 首項和末項 係數 1
                                Ck = 1.0
                                else if(mod(K,2)==0)then   ! 偶數項 係數 4
                                Ck = 4.0
                                else                               ! 奇數項 係數 2
                                Ck= 2.0
                                end if
                        Eee = Ci*Cj*Ck*U(I,J,K) + Eee
                end do
        end do
```

```
end do

! 把∫∫Φk・Φh・Φi・Φj/｜r-r'｜存到 Poi_solxF_int.txt 檔， XF-XS=YF-YS=ZF-ZS=40.0
 write(30,'(I2,I3,I3,I3,E30.20)') IC(cou),JC(cou),IC(cou2),JC(cou2),&
                                        & 0.5*Eee*((XF-XS)/(3.0*L))*((YF-YS)/(3.0*M))*((ZF-ZS)/(3.0*N))

        !--------存入陣列 single_Eee------------------!
!       single_Eee(cou,cou2,cou3,cou4) = 0.5*Eee*((XF-XS)/(3.0*L))*((YF-YS)/(3.0*M))*((ZF-ZS)/(3.0*N))
!       write(40,'(I2,I3,I3,I3,E30.20)') cou,cou2,cou3,cou4,single_Eee(cou,cou2,cou3,cou4)

        end do
        end do

 stop
end program

!----------     radial function      Rpl(r)      ----------!
real function Rpl(atom,p,l,r)
  use global
  implicit none
        integer         :: p,l          ! p 主量子數 ; l 角動量子數;atom 原子序
        real :: r,atom
                if (p==1 .AND. l==0)then            ! R10
                        Rpl = 2.0*((atom/a0)**(1.5))*exp(-(atom*r)/a0)
                else if (p==2 .AND. l==0)then       ! R20
                        Rpl = ((atom/(2.0*a0))**(1.5))*(2.0-(atom*r)/a0)*exp(-(atom*r)/(2.0*a0))
                else if (p==2 .AND.l==1)then        ! R21
                        Rpl = (1.0/sqrt(3.0))*((atom/(2.0*a0))**(1.5))*((atom*r)/a0)*exp(-(atom*r)/(2.0*a0))
                else if (p==3 .AND. l==0)then       ! R30
                        Rpl = (2.0/27.0)*((atom/(3.0*a0))**(1.5))*(27.0-18.0*((atom*r)/a0)+ &
                                & (2.0*atom*atom*r*r)/(a0*a0))*exp(-(atom*r)/(3.0*a0))
                else if (p==3 .AND. l==1)then       ! R31
                        Rpl = (4.0*sqrt(2.0)/54.0)*((atom/(3.0*a0))**(1.5))*(atom*r/a0)*(6.0-(atom*r/a0))*exp(-(atom*r/(3.0*a0)))
```

```
                else
                        write(*,*)" Rpl 超過 R31 !"
                end if
    return
end

!-----------     spherical harmonic function    Ylm(θ,ϕ)        ---------!
!-----------     爲計算上方便 省略 phi 部分 → 實函數        ----------!
complex function Ylm(l,m,theta,phi)
    use global
    implicit none
        real :: theta,phi        ! θ,ϕ
        integer :: l,m                          ! l 角動量子數 ; m 磁量子數
                if (l==0 .AND. m==0)then            ! Y00
                        Ylm = 1.0/(sqrt(4.0*pi))
                else if(l==1 .AND. m==-1)then   ! Y1-1
                        Ylm = (sqrt(3.0/(8.0*pi)))*sin(theta)*(cos(phi)-csqrt((-1.0,0.0))*sin(phi))
                else if(l==1 .AND. m==0)then    ! Y10
                        Ylm = (sqrt(3.0/(4.0*pi)))*cos(theta)
                else if(l==1 .AND. m==1)then    ! Y11
                        Ylm = -(sqrt(3.0/(8.0*pi)))*sin(theta)*(cos(phi)+csqrt((-1.0,0.0))*sin(phi))
                else if(l==2 .AND. m==-2)then   ! Y2-2
                      Ylm = (sqrt(15.0/(32.0*pi)))*((sin(theta))**(2.0))*exp(-2.0*csqrt((-1.0,0.0))*phi)
                else if(l==2 .AND. m==-1)then   ! Y2-1
                        Ylm = (sqrt(15.0/(8.0*pi)))*(sin(theta))*(cos(theta))*exp(-csqrt((-1.0,0.0))*phi)
                else if(l==2 .AND. m==0)then    ! Y20
                        Ylm = (sqrt(5.0/(16.0*pi)))*(3.0*(cos(theta))**(2.0)-1.0)
                else if(l==2 .AND. m==1)then    ! Y21
                        Ylm = -(sqrt(15.0/(8.0*pi)))*(sin(theta))*(cos(theta))*exp(csqrt((-1.0,0.0))*phi)
                else if(l==2 .AND. m==2)then    ! Y22
                        Ylm = (sqrt(15.0/(32.0*pi)))*((sin(theta))**(2.0))*exp(2.0*csqrt((-1.0,0.0))*phi)
                end if
    return
end

!-----   原始基底函數  Φ1～Φ10   -----(  Φ  = Rpl*Ylm )-------------------!
```

```
complex function basis_func(i,x1,y1,z1)
   use global
   implicit none
        integer :: i
        real :: x1,y1,z1,atom                    ! x1,y1,z1 爲直角坐標上變數
        real :: r,r1,r2                          ! r1 中心爲 H1   r2 中心爲第一個 H2
        real:: theta,theta1,theta2               ! 角度 θ :theta
        real :: phi,phi1,phi2                    ! 角度 φ :phi
        real,external :: Rpl                     ! 函數 Rpl
        complex,external ::Ylm                   ! 函數 Ylm

        r1 = sqrt(x1**2.0+y1**2.0+z1**2.0)              ! r 換成直角坐標 x,y,z   中心爲 H1
        r2 = sqrt((x1-0.74)**2.0+y1**2.0+z1**2.0)       !                        中心爲 H2
        theta1 = atan2(sqrt(x1**2.0+y1**2.0),z1)        ! ← arctangent(a/b)
        theta2 = atan2(sqrt((x1-0.74)**2.0+y1**2.0),z1)
        phi1 =   atan2(y1,x1)
        phi2 =   atan2(y1,x1-0.74)

            if(i<=5)then                         ! 波函數共 10 項，前 6 項中心 O
                atom = 1.0
                r    = r1
                theta = theta1
                phi   = phi1
            elseif(i>=6.OR.i<=10)then            ! 第 7,8 項中心 H-1
                atom  = 1.0
                r     = r2
                theta = theta2
                phi   = phi2
            else
                write(*,*)"wave function  出問題"
            end if

            if(i==1)then
                basis_func = Rpl(atom,1,0,r) * Ylm(0,0,theta,phi)      ! H1 軌道 1S
            elseif(i==2)then
```

```
            basis_func = Rpl(atom,2,0,r) * Ylm(0,0,theta,phi)        ! H1 軌道 2S
        elseif(i==3)then
            basis_func = Rpl(atom,2,1,r) * Ylm(1,-1,theta,phi)       ! H1 軌道 2P
        elseif(i==4)then
            basis_func = Rpl(atom,2,1,r) * Ylm(1,0,theta,phi)        ! H1 軌道 2P
        elseif(i==5)then
            basis_func = Rpl(atom,2,1,r) * Ylm(1,1,theta,phi)        ! H1 軌道 2P
        elseif(i==6)then
            basis_func = Rpl(atom,1,0,r) * Ylm(0,0,theta,phi)        ! H2 軌道 1S
        elseif(i==7)then
            basis_func = Rpl(atom,2,0,r) * Ylm(0,0,theta,phi)        ! H2 軌道 2S
        elseif(i==8)then
            basis_func = Rpl(atom,2,1,r) * Ylm(1,-1,theta,phi)       ! H2 軌道 2P
        elseif(i==9)then
            basis_func = Rpl(atom,2,1,r) * Ylm(1,0,theta,phi)        ! H2 軌道 2P
        elseif(i==10)then
            basis_func = Rpl(atom,2,1,r) * Ylm(1,1,theta,phi)        ! H2 軌道 2P
        else
            write(*,*)"basis_func 超過 10!!"
        end if
    return
end

!----給定陣列   wave(i,:,:,:)值,i=1~10        ------------------!
subroutine sub1()
    use global
    implicit none
        integer :: h,i,j,k
        complex,external::basis_func

        !---給定定義域 x,y,z 格子點值
        do i = 1,inn_m_parti+1
            xi(i) = x_min + ((x_max-x_min)/inn_m_parti)* (i-1)
            yi(i) = y_min + ((y_max-y_min)/inn_m_parti)* (i-1)
            zi(i) = z_min + ((z_max-z_min)/inn_m_parti)* (i-1)
        end do
```

```
        !-- 給定陣列   wave(i,:,:,:)=Φ(i,:,:,:)
        do h=1,basics
                do i =1,inn_m_parti+1
                  do j =1,inn_m_parti+1
                    do k =1,inn_m_parti+1
                          wave(h,i,j,k) = basis_func(h,xi(i),yi(j),zi(k))
                    end do
                  end do
                end do
        end do
   return
end subroutine

!----計算 <Φi| Φj>   3-D 辛普森積分----------!
subroutine sub2(h1,h2,ANS2)
use global
implicit none
        integer :: i,j,k,h1,h2
        real::C_i,C_j,C_k
        complex:: ANS,ANS2

        ANS =0.0
        ! 辛普森三重積分 ∫∫∫Φi・Φj*   ; i,j=1～10
        do i=1,inn_m_parti+1
                if(i==1 .OR. i==inn_m_parti+1)then    ! 首項、末項點 係數 1
                        C_i = 1.0
                else if(mod(i,2)==0)then              ! 偶數項點 係數 4
                        C_i = 4.0
                else                                  ! 奇數項點 係數 2
                        C_i= 2.0
                end if
           do j=1,inn_m_parti+1
                  if(j==1 .OR. j==inn_m_parti+1)then ! 首項、末項點 係數 1
                        C_j = 1.0
                  else if(mod(j,2)==0)then          ! 偶數項點 係數 4
```

```
                C_j = 4.0
            else                                    ! 奇數項點 係數 2
                C_j= 2.0
            end if
        do k=1,inn_m_parti+1
            if(k==1 .OR. k==inn_m_parti+1)then   ! 首項、末項點 係數 1
                C_k = 1.0
            else if(mod(k,2)==0)then            ! 偶數項點 係數 4
                C_k = 4.0
            else                                    ! 奇數項點 係數 2
                C_k= 2.0
            end if

            ANS = ANS+ C_i*C_j*C_k *conjg(wave(h1,i,j,k))*wave(h2,i,j,k)
        end do
      end do
    end do

        ANS2 = ANS*(x_max-x_min)*(y_max-y_min)*(z_max-z_min)/((3.0*inn_m_parti)**(3.0))
!       write(*,*)h1,h2,ANS2

  return
end subroutine

!----計算 <Φi| Φi>  3-D 辛普森積分----------!
subroutine sub3(h1,ANS2)
use global
implicit none
    integer :: i,j,k,h1
    real::C_i,C_j,C_k,ANS,ANS2

    ANS =0.0
    ! 辛普森三重積分 ∫∫∫Φi・Φi*  ; i,j=1～10
    do i=1,inn_m_parti+1
        if(i==1 .OR. i==inn_m_parti+1)then   ! 首項、末項點 係數 1
            C_i = 1.0
```

```
            else if(mod(i,2)==0)then              ! 偶數項點 係數 4
                C_i = 4.0
            else                                  ! 奇數項點 係數 2
                C_i= 2.0
            end if
        do j=1,inn_m_parti+1
            if(j==1 .OR. j==inn_m_parti+1)then ! 首項、末項點 係數 1
                C_j = 1.0
            else if(mod(j,2)==0)then          ! 偶數項點 係數 4
                C_j = 4.0
            else                                  ! 奇數項點 係數 2
                C_j= 2.0
            end if
          do k=1,inn_m_parti+1
                if(k==1 .OR. k==inn_m_parti+1)then   ! 首項、末項點 係數 1
                    C_k = 1.0
                else if(mod(k,2)==0)then          ! 偶數項點 係數 4
                    C_k = 4.0
                else                                  ! 奇數項點 係數 2
                    C_k= 2.0
                end if

                ANS = ANS+ C_i*C_j*C_k *conjg(wave(h1,i,j,k))*wave(h1,i,j,k)
          end do
        end do
      end do

            ANS2 = ANS*(x_max-x_min)*(y_max-y_min)*(z_max-z_min)/((3.0*inn_m_parti)**(3.0))
!           write(*,*)h1,ANS2

   return
end subroutine

!----歸一化(單位化)波函數       othonormal_wave(h,i,j,k) --------------------!
subroutine sub4(h1,input)
   use global
```

```
   implicit none
        real:: input
        integer :: i,j,k,h1
              wave(h1,:,:,:) = wave(h1,:,:,:)/sqrt(input)

   return
end subroutine

!---- 正交歸一化  波函數陣列 ------------------------------!
subroutine othonormal()
   use global
   implicit none
        real:: ANS
        complex:: ANS2
        integer :: i,j,k,h1,h2

        call sub1()  ! 把 10 個波函數 寫成陣列
        call sub3(1,ANS)  !  <Φi|Φi>
        call sub4(1,ANS)  ! 把Φ1 做 normalize 得到 NEWΦ1

        do h1 = 2,basics
        do h2 = 1,h1-1
              call sub2(h2,h1,ANS2)
                 do i =1,inn_m_parti+1
                 do j =1,inn_m_parti+1
                 do k =1,inn_m_parti+1
                       wave(h1,i,j,k) = wave(h1,i,j,k) -ANS2*wave(h2,i,j,k)
                 end do
                 end do
                 end do
        end do
              call sub3(h1,ANS)
              call sub4(h1,ANS)
        end do
```

```
   return
end subroutine

      SUBROUTINE COSQB1 (N,X,W,XH)
      DIMENSION       X(1)       ,W(1)       ,XH(1)
      NS2 = (N+1)/2
      NP2 = N+2
      DO 101 I=3,N,2
         XIM1 = X(I-1)+X(I)
         X(I) = X(I)-X(I-1)
         X(I-1) = XIM1
  101 CONTINUE
      X(1) = X(1)+X(1)
      MODN = MOD(N,2)
      IF (MODN .EQ. 0) X(N) = X(N)+X(N)
      CALL RFFTB (N,X,XH)
      DO 102 K=2,NS2
         KC = NP2-K
         XH(K) = W(K-1)*X(KC)+W(KC-1)*X(K)
         XH(KC) = W(K-1)*X(K)-W(KC-1)*X(KC)
  102 CONTINUE
      IF (MODN .EQ. 0) X(NS2+1) = W(NS2)*(X(NS2+1)+X(NS2+1))
      DO 103 K=2,NS2
         KC = NP2-K
         X(K) = XH(K)+XH(KC)
         X(KC) = XH(K)-XH(KC)
  103 CONTINUE
      X(1) = X(1)+X(1)
      RETURN
      END

      SUBROUTINE COSQB (N,X,WSAVE)
      DIMENSION       X(1)       ,WSAVE(1)
      DATA TSQRT2 /2.82842712474619/
      IF (N-2) 101,102,103
  101 X(1) = 4.*X(1)
```

```
      RETURN
102 X1 = 4.*(X(1)+X(2))
      X(2) = TSQRT2*(X(1)-X(2))
      X(1) = X1
      RETURN
103 CALL COSQB1 (N,X,WSAVE,WSAVE(N+1))
      RETURN
      END

    SUBROUTINE COSQF1 (N,X,W,XH)
      DIMENSION       X(1)       ,W(1)       ,XH(1)
      NS2 = (N+1)/2
      NP2 = N+2
      DO 101 K=2,NS2
         KC = NP2-K
         XH(K) = X(K)+X(KC)
         XH(KC) = X(K)-X(KC)
101 CONTINUE
      MODN = MOD(N,2)
      IF (MODN .EQ. 0) XH(NS2+1) = X(NS2+1)+X(NS2+1)
      DO 102 K=2,NS2
         KC = NP2-K
         X(K) = W(K-1)*XH(KC)+W(KC-1)*XH(K)
         X(KC) = W(K-1)*XH(K)-W(KC-1)*XH(KC)
102 CONTINUE
      IF (MODN .EQ. 0) X(NS2+1) = W(NS2)*XH(NS2+1)
      CALL RFFTF (N,X,XH)
      DO 103 I=3,N,2
         XIM1 = X(I-1)-X(I)
         X(I) = X(I-1)+X(I)
         X(I-1) = XIM1
103 CONTINUE
      RETURN
      END

      SUBROUTINE COSQF (N,X,WSAVE)
```

```
      DIMENSION         X(1)         ,WSAVE(1)
      DATA SQRT2 /1.4142135623731/
      IF (N-2) 102,101,103
101 TSQX = SQRT2*X(2)
      X(2) = X(1)-TSQX
      X(1) = X(1)+TSQX
102 RETURN
103 CALL COSQF1 (N,X,WSAVE,WSAVE(N+1))
      RETURN
      END

      SUBROUTINE COSQI (N,WSAVE)
      DIMENSION         WSAVE(1)
      DATA PIH /1.57079632679491/
      DT = PIH/FLOAT(N)
      FK = 0.
      DO 101 K=1,N
         FK = FK+1.
         WSAVE(K) = COS(FK*DT)
101 CONTINUE
      CALL RFFTI (N,WSAVE(N+1))
      RETURN
      END

      SUBROUTINE COST (N,X,WSAVE)
      DIMENSION         X(1)         ,WSAVE(1)
      NM1 = N-1
      NP1 = N+1
      NS2 = N/2
      IF (N-2) 106,101,102
101 X1H = X(1)+X(2)
      X(2) = X(1)-X(2)
      X(1) = X1H
      RETURN
102 IF (N .GT. 3) GO TO 103
      X1P3 = X(1)+X(3)
```

```
      TX2 = X(2)+X(2)
      X(2) = X(1)-X(3)
      X(1) = X1P3+TX2
      X(3) = X1P3-TX2
      RETURN
103 C1 = X(1)-X(N)
      X(1) = X(1)+X(N)
      DO 104 K=2,NS2
         KC = NP1-K
         T1 = X(K)+X(KC)
         T2 = X(K)-X(KC)
         C1 = C1+WSAVE(KC)*T2
         T2 = WSAVE(K)*T2
         X(K) = T1-T2
         X(KC) = T1+T2
104 CONTINUE
      MODN = MOD(N,2)
      IF (MODN .NE. 0) X(NS2+1) = X(NS2+1)+X(NS2+1)
      CALL RFFTF (NM1,X,WSAVE(N+1))
      XIM2 = X(2)
      X(2) = C1
      DO 105 I=4,N,2
         XI = X(I)
         X(I) = X(I-2)-X(I-1)
         X(I-1) = XIM2
         XIM2 = XI
105 CONTINUE
      IF (MODN .NE. 0) X(N) = XIM2
106 RETURN
      END

      SUBROUTINE COSTI (N,WSAVE)
      DIMENSION       WSAVE(1)
      DATA PI /3.14159265358979/
      IF (N .LE. 3) RETURN
      NM1 = N-1
```

```
      NP1 = N+1
      NS2 = N/2
      DT = PI/FLOAT(NM1)
      FK = 0.
      DO 101 K=2,NS2
         KC = NP1-K
         FK = FK+1.
         WSAVE(K) = 2.*SIN(FK*DT)
         WSAVE(KC) = 2.*COS(FK*DT)
  101 CONTINUE
      CALL RFFTI (NM1,WSAVE(N+1))
      RETURN
      END

! FISHPK12 FROM PORTLIB                                    12/30/83
      SUBROUTINE HW3CRT (XS,XF,L,LBDCND,BDXS,BDXF,YS,YF,M,MBDCND,BDYS, &
     &                   BDYF,ZS,ZF,N,NBDCND,BDZS,BDZF,ELMBDA,LDIMF, &
     &                   MDIMF,F,PERTRB,IERROR,W)
!
!
!     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!     *                                                               *
!     *                        F I S H P A K                          *
!     *                                                               *
!     *                                                               *
!     *     A PACKAGE OF FORTRAN SUBPROGRAMS FOR THE SOLUTION OF      *
!     *                                                               *
!     *      SEPARABLE ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS        *
!     *                                                               *
!     *                  (VERSION 3.1 , OCTOBER 1980)                 *
!     *                                                               *
!     *                             BY                                *
!     *                                                               *
!     *        JOHN ADAMS, PAUL SWARZTRAUBER AND ROLAND SWEET         *
!     *                                                               *
!     *                             OF                                *
```

```
!     *                                                               *
!     *             THE NATIONAL CENTER FOR ATMOSPHERIC RESEARCH      *
!     *                                                               *
!     *                  BOULDER, COLORADO  (80307)  U.S.A.           *
!     *                                                               *
!     *                      WHICH IS SPONSORED BY                    *
!     *                                                               *
!     *                THE NATIONAL SCIENCE FOUNDATION                *
!     *                                                               *
!     *****************************************
!
!
!    **********  PURPOSE  ********************
!
!          SUBROUTINE HW3CRT SOLVES THE STANDARD SEVEN-POINT FINITE
!     DIFFERENCE APPROXIMATION TO THE HELMHOLTZ EQUATION IN CARTESIAN
!     COORDINATES:
!
!          (D/DX)(DU/DX) + (D/DY)(DU/DY) + (D/DZ)(DU/DZ)
!
!                        + LAMBDA*U = F(X,Y,Z) .
!
!    *********************************************
!
!
!    ********  PARAMETER DESCRIPTION  **********
!
!
!            *******  ON INPUT  *******
!
!     XS,XF
!          THE RANGE OF X, I.E. XS .LE. X .LE. XF .
!          XS MUST BE LESS THAN XF. (XS < X < XF)
!
!     L
!          THE NUMBER OF PANELS INTO WHICH THE INTERVAL (XS,XF) IS
```

```
!         SUBDIVIDED.    HENCE, THERE WILL BE L+1 GRID POINTS IN THE
!         X-DIRECTION GIVEN BY X(I) = XS+(I-1)DX FOR I=1,2,...,L+1,
!         WHERE DX = (XF-XS)/L IS THE PANEL WIDTH.    L MUST BE AT
!         LEAST 5 .
!
!     LBDCND
!         INDICATES THE TYPE OF BOUNDARY CONDITIONS AT X = XS AND X = XF.
!
!         = 0    IF THE SOLUTION IS PERIODIC IN X, I.E.
!                U(L+I,J,K) = U(I,J,K).
!         = 1    IF THE SOLUTION IS SPECIFIED AT X = XS AND X = XF.
!         = 2    IF THE SOLUTION IS SPECIFIED AT X = XS AND THE DERIVATIVE
!                OF THE SOLUTION WITH RESPECT TO X IS SPECIFIED AT X = XF.
!         = 3    IF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO X IS
!                SPECIFIED AT X = XS AND X = XF.
!         = 4    IF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO X IS
!                SPECIFIED AT X = XS AND THE SOLUTION IS SPECIFIED AT X=XF.
!
!     BDXS
!         A TWO-DIMENSIONAL ARRAY THAT SPECIFIES THE VALUES OF THE
!         DERIVATIVE OF THE SOLUTION WITH RESPECT TO X AT X = XS.
!         WHEN LBDCND = 3 OR 4,
!
!                BDXS(J,K) = (D/DX)U(XS,Y(J),Z(K)), J=1,2,...,M+1,
!                                                   K=1,2,...,N+1.
!
!         WHEN LBDCND HAS ANY OTHER VALUE, BDXS IS A DUMMY VARIABLE.
!         BDXS MUST BE DIMENSIONED AT LEAST (M+1)*(N+1).
!
!     BDXF
!         A TWO-DIMENSIONAL ARRAY THAT SPECIFIES THE VALUES OF THE
!         DERIVATIVE OF THE SOLUTION WITH RESPECT TO X AT X = XF.
!         WHEN LBDCND = 2 OR 3,
!
!                BDXF(J,K) = (D/DX)U(XF,Y(J),Z(K)), J=1,2,...,M+1,
!                                                   K=1,2,...,N+1.
```

```
!
!          WHEN LBDCND HAS ANY OTHER VALUE, BDXF IS A DUMMY VARIABLE.
!          BDXF MUST BE DIMENSIONED AT LEAST (M+1)*(N+1).
!
!     YS,YF
!          THE RANGE OF Y, I.E. YS .LE. Y .LE. YF.
!          YS MUST BE LESS THAN YF.
!
!     M
!          THE NUMBER OF PANELS INTO WHICH THE INTERVAL (YS,YF) IS
!          SUBDIVIDED.    HENCE, THERE WILL BE M+1 GRID POINTS IN THE
!          Y-DIRECTION GIVEN BY Y(J) = YS+(J-1)DY FOR J=1,2,...,M+1,
!          WHERE DY = (YF-YS)/M IS THE PANEL WIDTH.    M MUST BE AT
!          LEAST 5 .
!
!     MBDCND
!          INDICATES THE TYPE OF BOUNDARY CONDITIONS AT Y = YS AND Y = YF.
!
!          = 0    IF THE SOLUTION IS PERIODIC IN Y, I.E.
!                 U(I,M+J,K) = U(I,J,K).
!          = 1    IF THE SOLUTION IS SPECIFIED AT Y = YS AND Y = YF.
!          = 2    IF THE SOLUTION IS SPECIFIED AT Y = YS AND THE DERIVATIVE
!                 OF THE SOLUTION WITH RESPECT TO Y IS SPECIFIED AT Y = YF.
!          = 3    IF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO Y IS
!                 SPECIFIED AT Y = YS AND Y = YF.
!          = 4    IF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO Y IS
!                 SPECIFIED AT Y = YS AND THE SOLUTION IS SPECIFIED AT Y=YF.
!
!     BDYS
!          A TWO-DIMENSIONAL ARRAY THAT SPECIFIES THE VALUES OF THE
!          DERIVATIVE OF THE SOLUTION WITH RESPECT TO Y AT Y = YS.
!          WHEN MBDCND = 3 OR 4,
!
!                 BDYS(I,K) = (D/DY)U(X(I),YS,Z(K)), I=1,2,...,L+1,
!                                                        K=1,2,...,N+1.
!
```

```
!         WHEN MBDCND HAS ANY OTHER VALUE, BDYS IS A DUMMY VARIABLE.
!         BDYS MUST BE DIMENSIONED AT LEAST (L+1)*(N+1).
!
!     BDYF
!         A TWO-DIMENSIONAL ARRAY THAT SPECIFIES THE VALUES OF THE
!         DERIVATIVE OF THE SOLUTION WITH RESPECT TO Y AT Y = YF.
!         WHEN MBDCND = 2 OR 3,
!
!              BDYF(I,K) = (D/DY)U(X(I),YF,Z(K)), I=1,2,...,L+1,
!                                                 K=1,2,...,N+1.
!
!         WHEN MBDCND HAS ANY OTHER VALUE, BDYF IS A DUMMY VARIABLE.
!         BDYF MUST BE DIMENSIONED AT LEAST (L+1)*(N+1).
!
!     ZS,ZF
!         THE RANGE OF Z, I.E. ZS .LE. Z .LE. ZF.
!         ZS MUST BE LESS THAN ZF.
!
!     N
!         THE NUMBER OF PANELS INTO WHICH THE INTERVAL (ZS,ZF) IS
!         SUBDIVIDED.    HENCE, THERE WILL BE N+1 GRID POINTS IN THE
!         Z-DIRECTION GIVEN BY Z(K) = ZS+(K-1)DZ FOR K=1,2,...,N+1,
!         WHERE DZ = (ZF-ZS)/N IS THE PANEL WIDTH.    N MUST BE AT LEAST 5.
!
!     NBDCND
!         INDICATES THE TYPE OF BOUNDARY CONDITIONS AT Z = ZS AND Z = ZF.
!
!         = 0    IF THE SOLUTION IS PERIODIC IN Z, I.E.
!                U(I,J,N+K) = U(I,J,K).
!         = 1    IF THE SOLUTION IS SPECIFIED AT Z = ZS AND Z = ZF.
!         = 2    IF THE SOLUTION IS SPECIFIED AT Z = ZS AND THE DERIVATIVE
!                OF THE SOLUTION WITH RESPECT TO Z IS SPECIFIED AT Z = ZF.
!         = 3    IF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO Z IS
!                SPECIFIED AT Z = ZS AND Z = ZF.
!         = 4    IF THE DERIVATIVE OF THE SOLUTION WITH RESPECT TO Z IS
!                SPECIFIED AT Z = ZS AND THE SOLUTION IS SPECIFIED AT Z=ZF.
```

```
!
!     BDZS
!          A TWO-DIMENSIONAL ARRAY THAT SPECIFIES THE VALUES OF THE
!          DERIVATIVE OF THE SOLUTION WITH RESPECT TO Z AT Z = ZS.
!          WHEN NBDCND = 3 OR 4,
!
!               BDZS(I,J) = (D/DZ)U(X(I),Y(J),ZS), I=1,2,...,L+1,
!                                                  J=1,2,...,M+1.
!
!          WHEN NBDCND HAS ANY OTHER VALUE, BDZS IS A DUMMY VARIABLE.
!          BDZS MUST BE DIMENSIONED AT LEAST (L+1)*(M+1).
!
!     BDZF
!          A TWO-DIMENSIONAL ARRAY THAT SPECIFIES THE VALUES OF THE
!          DERIVATIVE OF THE SOLUTION WITH RESPECT TO Z AT Z = ZF.
!          WHEN NBDCND = 2 OR 3,
!
!               BDZF(I,J) = (D/DZ)U(X(I),Y(J),ZF), I=1,2,...,L+1,
!                                                  J=1,2,...,M+1.
!
!          WHEN NBDCND HAS ANY OTHER VALUE, BDZF IS A DUMMY VARIABLE.
!          BDZF MUST BE DIMENSIONED AT LEAST (L+1)*(M+1).
!
!     ELMBDA
!          THE CONSTANT LAMBDA IN THE HELMHOLTZ EQUATION. IF
!          LAMBDA .GT. 0, A SOLUTION MAY NOT EXIST.  HOWEVER, HW3CRT WILL
!          ATTEMPT TO FIND A SOLUTION.
!
!     F
!          A THREE-DIMENSIONAL ARRAY THAT SPECIFIES THE VALUES OF THE
!          RIGHT SIDE OF THE HELMHOLTZ EQUATION AND BOUNDARY VALUES (IF
!          ANY).  FOR I=2,3,...,L, J=2,3,...,M, AND K=2,3,...,N
!
!                    F(I,J,K) = F(X(I),Y(J),Z(K)).
!
!          ON THE BOUNDARIES F IS DEFINED BY
```

```
!
!       LBDCND          F(1,J,K)              F(L+1,J,K)
!       ------    ---------------     ---------------
!
!         0         F(XS,Y(J),Z(K))     F(XS,Y(J),Z(K))
!         1         U(XS,Y(J),Z(K))     U(XF,Y(J),Z(K))
!         2         U(XS,Y(J),Z(K))     F(XF,Y(J),Z(K))     J=1,2,...,M+1
!         3         F(XS,Y(J),Z(K))     F(XF,Y(J),Z(K))     K=1,2,...,N+1
!         4         F(XS,Y(J),Z(K))     U(XF,Y(J),Z(K))
!
!       MBDCND          F(I,1,K)              F(I,M+1,K)
!       ------    ---------------     ---------------
!
!         0         F(X(I),YS,Z(K))     F(X(I),YS,Z(K))
!         1         U(X(I),YS,Z(K))     U(X(I),YF,Z(K))
!         2         U(X(I),YS,Z(K))     F(X(I),YF,Z(K))     I=1,2,...,L+1
!         3         F(X(I),YS,Z(K))     F(X(I),YF,Z(K))     K=1,2,...,N+1
!         4         F(X(I),YS,Z(K))     U(X(I),YF,Z(K))
!
!       NBDCND          F(I,J,1)              F(I,J,N+1)
!       ------    ---------------     ---------------
!
!         0         F(X(I),Y(J),ZS)     F(X(I),Y(J),ZS)
!         1         U(X(I),Y(J),ZS)     U(X(I),Y(J),ZF)
!         2         U(X(I),Y(J),ZS)     F(X(I),Y(J),ZF)     I=1,2,...,L+1
!         3         F(X(I),Y(J),ZS)     F(X(I),Y(J),ZF)     J=1,2,...,M+1
!         4         F(X(I),Y(J),ZS)     U(X(I),Y(J),ZF)
!
!       F MUST BE DIMENSIONED AT LEAST (L+1)*(M+1)*(N+1).
!
!       NOTE:
!
!       IF THE TABLE CALLS FOR BOTH THE SOLUTION U AND THE RIGHT SIDE F
!       ON A BOUNDARY, THEN THE SOLUTION MUST BE SPECIFIED.
!
!     LDIMF
```

```
!         THE ROW (OR FIRST) DIMENSION OF THE ARRAYS F,BDYS,BDYF,BDZS,
!         AND BDZF AS IT APPEARS IN THE PROGRAM CALLING HW3CRT. THIS
!         PARAMETER IS USED TO SPECIFY THE VARIABLE DIMENSION OF THESE
!         ARRAYS.  LDIMF MUST BE AT LEAST L+1.
!
!    MDIMF
!         THE COLUMN (OR SECOND) DIMENSION OF THE ARRAY F AND THE ROW (OR
!         FIRST) DIMENSION OF THE ARRAYS BDXS AND BDXF AS IT APPEARS IN
!         THE PROGRAM CALLING HW3CRT.  THIS PARAMETER IS USED TO SPECIFY
!         THE VARIABLE DIMENSION OF THESE ARRAYS.
!         MDIMF MUST BE AT LEAST M+1.
!
!    W
!         A ONE-DIMENSIONAL ARRAY THAT MUST BE PROVIDED BY THE USER FOR
!         WORK SPACE.  THE LENGTH OF W MUST BE AT LEAST 30 + L + M + 5*N
!         + MAX(L,M,N) + 7*(INT((L+1)/2) + INT((M+1)/2))
!
!
!              * * * * * *    ON OUTPUT    * * * * * *
!
!    F
!         CONTAINS THE SOLUTION U(I,J,K) OF THE FINITE DIFFERENCE
!         APPROXIMATION FOR THE GRID POINT (X(I),Y(J),Z(K)) FOR
!         I=1,2,...,L+1, J=1,2,...,M+1, AND K=1,2,...,N+1.
!
!    PERTRB
!         IF A COMBINATION OF PERIODIC OR DERIVATIVE BOUNDARY CONDITIONS
!         IS SPECIFIED FOR A POISSON EQUATION (LAMBDA = 0), A SOLUTION
!         MAY NOT EXIST.  PERTRB IS A CONSTANT, CALCULATED AND SUBTRACTED
!         FROM F, WHICH ENSURES THAT A SOLUTION EXISTS.  PWSCRT THEN
!         COMPUTES THIS SOLUTION, WHICH IS A LEAST SQUARES SOLUTION TO
!         THE ORIGINAL APPROXIMATION.  THIS SOLUTION IS NOT UNIQUE AND IS
!         UNNORMALIZED.  THE VALUE OF PERTRB SHOULD BE SMALL COMPARED TO
!         THE RIGHT SIDE F.  OTHERWISE, A SOLUTION IS OBTAINED TO AN
!         ESSENTIALLY DIFFERENT PROBLEM.  THIS COMPARISON SHOULD ALWAYS
!         BE MADE TO INSURE THAT A MEANINGFUL SOLUTION HAS BEEN OBTAINED.
```

```
!
!     IERROR
!         AN ERROR FLAG THAT INDICATES INVALID INPUT PARAMETERS.  EXCEPT
!         FOR NUMBERS 0 AND 12, A SOLUTION IS NOT ATTEMPTED.
!
!         =  0  NO ERROR
!         =  1  XS .GE. XF
!         =  2  L .LT. 5
!         =  3  LBDCND .LT. 0 .OR. LBDCND .GT. 4
!         =  4  YS .GE. YF
!         =  5  M .LT. 5
!         =  6  MBDCND .LT. 0 .OR. MBDCND .GT. 4
!         =  7  ZS .GE. ZF
!         =  8  N .LT. 5
!         =  9  NBDCND .LT. 0 .OR. NBDCND .GT. 4
!         = 10  LDIMF .LT. L+1
!         = 11  MDIMF .LT. M+1
!         = 12  LAMBDA .GT. 0
!
!         SINCE THIS IS THE ONLY MEANS OF INDICATING A POSSIBLY INCORRECT
!         CALL TO HW3CRT, THE USER SHOULD TEST IERROR AFTER THE CALL.
!
!
!    * * * * * * *    PROGRAM SPECIFICATIONS    * * * * * * * * * * * *
!
!     DIMENSION OF    BDXS(MDIMF,N+1),BDXF(MDIMF,N+1),BDYS(LDIMF,N+1),
!     ARGUMENTS       BDYF(LDIMF,N+1),BDZS(LDIMF,M+1),BDZF(LDIMF,M+1),
!                     F(LDIMF,MDIMF,N+1),W(SEE ARGUMENT LIST)
!
!     LATEST          DECEMBER 1, 1978
!     REVISION
!
!     SUBPROGRAMS     HW3CRT,POIS3D,POS3D1,TRID,RFFTI,RFFTF,RFFTF1,
!     REQUIRED        RFFTB,RFFTB1,COSTI,COST,SINTI,SINT,COSQI,COSQF,
!                     COSQF1,COSQB,COSQB1,SINQI,SINQF,SINQB,CFFTI,
!                     CFFTI1,CFFTB,CFFTB1,PASSB2,PASSB3,PASSB4,PASSB,
```

```
!                       CFFTF,CFFTF1,PASSF1,PASSF2,PASSF3,PASSF4,PASSF,
!                       PIMACH
!
!     SPECIAL           NONE
!     CONDITIONS
!
!     COMMON              VALUE
!     BLOCKS
!
!     I/O               NONE
!
!     PRECISION         SINGLE
!
!     SPECIALIST        ROLAND SWEET
!
!     LANGUAGE            FORTRAN
!
!     HISTORY           WRITTEN BY ROLAND SWEET AT NCAR IN JULY,1977
!
!     ALGORITHM          THIS SUBROUTINE DEFINES THE FINITE DIFFERENCE
!                       EQUATIONS, INCORPORATES BOUNDARY DATA, AND
!                       ADJUSTS THE RIGHT SIDE OF SINGULAR SYSTEMS AND
!                       THEN CALLS POIS3D TO SOLVE THE SYSTEM.
!
!     SPACE             7862(DECIMAL) = 17300(OCTAL) LOCATIONS ON THE
!     REQUIRED           NCAR CONTROL DATA 7600
!
!     TIMING AND              THE EXECUTION TIME T ON THE NCAR CONTROL DATA
!     ACCURACY           7600 FOR SUBROUTINE HW3CRT IS ROUGHLY PROPORTIONAL
!                       TO L*M*N*(LOG2(L)+LOG2(M)+5), BUT ALSO DEPENDS ON
!                       INPUT PARAMETERS LBDCND AND MBDCND.    SOME TYPICAL
!                       VALUES ARE LISTED IN THE TABLE BELOW.
!                            THE SOLUTION PROCESS EMPLOYED RESULTS IN A LOSS
!                       OF NO MORE THAN THREE SIGNIFICANT DIGITS FOR L,M AN
!                       N AS LARGE AS 32.    MORE DETAILED INFORMATION ABOUT
!                       ACCURACY CAN BE FOUND IN THE DOCUMENTATION FOR
```

```
!                       SUBROUTINE POIS3D WHICH IS THE ROUTINE THAT ACTUALL
!                       SOLVES THE FINITE DIFFERENCE EQUATIONS.
!
!
!                          L(=M=N)        LBDCND(=MBDCND=NBDCND)         T(MSECS)
!                          -------        ----------------------         --------
!
!                            16                     0                      300
!                            16                     1                      302
!                            16                     3                      348
!                            32                     0                     1925
!                            32                     1                     1929
!                            32                     3                     2109
!
!     PORTABILITY       AMERICAN NATIONAL STANDARDS INSTITUTE FORTRAN.
!                       THE MACHINE DEPENDENT CONSTANT PI IS DEFINED IN
!                       FUNCTION PIMACH.
!
!     REQUIRED          COS,SIN,ATAN
!     RESIDENT
!     ROUTINES
!
!     REFERENCE         NONE
!
!     REQUIRED            COS,SIN,ATAN
!     RESIDENT
!     ROUTINES
!
!     REFERENCE           NONE
!
!    ***********************************
!
      DIMENSION         BDXS(MDIMF,1)             ,BDXF(MDIMF,1)             , &
     &                  BDYS(LDIMF,1)             ,BDYF(LDIMF,1)             , &
     &                  BDZS(LDIMF,1)             ,BDZF(LDIMF,1)             , &
     &                  F(LDIMF,MDIMF,1)          ,W(1)
```

```
!
!       CHECK FOR INVALID INPUT.
!
        IERROR = 0
        IF (XF .LE. XS) IERROR = 1
        IF (L .LT. 5) IERROR = 2
        IF (LBDCND.LT.0 .OR. LBDCND.GT.4) IERROR = 3
        IF (YF .LE. YS) IERROR = 4
        IF (M .LT. 5) IERROR = 5
        IF (MBDCND.LT.0 .OR. MBDCND.GT.4) IERROR = 6
        IF (ZF .LE. ZS) IERROR = 7
        IF (N .LT. 5) IERROR = 8
        IF (NBDCND.LT.0 .OR. NBDCND.GT.4) IERROR = 9
        IF (LDIMF .LT. L+1) IERROR = 10
        IF (MDIMF .LT. M+1) IERROR = 11
        IF (IERROR .NE. 0) GO TO 188
        DY = (YF-YS)/FLOAT(M)
        TWBYDY = 2./DY
        C2 = 1./(DY**2)
        MSTART = 1
        MSTOP = M
        MP1 = M+1
        MP = MBDCND+1
        GO TO (104,101,101,102,102),MP
  101 MSTART = 2
  102 GO TO (104,104,103,103,104),MP
  103 MSTOP = MP1
  104 MUNK = MSTOP-MSTART+1
        DZ = (ZF-ZS)/FLOAT(N)
        TWBYDZ = 2./DZ
        NP = NBDCND+1
        C3 = 1./(DZ**2)
        NP1 = N+1
        NSTART = 1
        NSTOP = N
        GO TO (108,105,105,106,106),NP
```

```
  105 NSTART = 2
  106 GO TO (108,108,107,107,108),NP
  107 NSTOP = NP1
  108 NUNK = NSTOP-NSTART+1
      LP1 = L+1
      DX = (XF-XS)/FLOAT(L)
      C1 = 1./(DX**2)
      TWBYDX = 2./DX
      LP = LBDCND+1
      LSTART = 1
      LSTOP = L
!
!     ENTER BOUNDARY DATA FOR X-BOUNDARIES.
!
      GO TO (122,109,109,112,112),LP
  109 LSTART = 2
      DO 111 J=MSTART,MSTOP
         DO 110 K=NSTART,NSTOP
            F(2,J,K) = F(2,J,K)-C1*F(1,J,K)
  110    CONTINUE
  111 CONTINUE
      GO TO 115
  112 DO 114 J=MSTART,MSTOP
         DO 113 K=NSTART,NSTOP
            F(1,J,K) = F(1,J,K)+TWBYDX*BDXS(J,K)
  113    CONTINUE
  114 CONTINUE
  115 GO TO (122,116,119,119,116),LP
  116 DO 118 J=MSTART,MSTOP
         DO 117 K=NSTART,NSTOP
            F(L,J,K) = F(L,J,K)-C1*F(LP1,J,K)
  117    CONTINUE
  118 CONTINUE
      GO TO 122
  119 LSTOP = LP1
      DO 121 J=MSTART,MSTOP
```

```
            DO 120 K=NSTART,NSTOP
                F(LP1,J,K) = F(LP1,J,K)-TWBYDX*BDXF(J,K)
  120       CONTINUE
  121 CONTINUE
  122 LUNK = LSTOP-LSTART+1
!
!       ENTER BOUNDARY DATA FOR Y-BOUNDARIES.
!
        GO TO (136,123,123,126,126),MP
  123 DO 125 I=LSTART,LSTOP
            DO 124 K=NSTART,NSTOP
                F(I,2,K) = F(I,2,K)-C2*F(I,1,K)
  124       CONTINUE
  125 CONTINUE
        GO TO 129
  126 DO 128 I=LSTART,LSTOP
            DO 127 K=NSTART,NSTOP
                F(I,1,K) = F(I,1,K)+TWBYDY*BDYS(I,K)
  127       CONTINUE
  128 CONTINUE
  129 GO TO (136,130,133,133,130),MP
  130 DO 132 I=LSTART,LSTOP
            DO 131 K=NSTART,NSTOP
                F(I,M,K) = F(I,M,K)-C2*F(I,MP1,K)
  131       CONTINUE
  132 CONTINUE
        GO TO 136
  133 DO 135 I=LSTART,LSTOP
            DO 134 K=NSTART,NSTOP
                F(I,MP1,K) = F(I,MP1,K)-TWBYDY*BDYF(I,K)
  134       CONTINUE
  135 CONTINUE
  136 CONTINUE
!
!       ENTER BOUNDARY DATA FOR Z-BOUNDARIES.
!
```

```
      GO TO (150,137,137,140,140),NP
  137 DO 139 I=LSTART,LSTOP
          DO 138 J=MSTART,MSTOP
              F(I,J,2) = F(I,J,2)-C3*F(I,J,1)
  138     CONTINUE
  139 CONTINUE
      GO TO 143
  140 DO 142 I=LSTART,LSTOP
          DO 141 J=MSTART,MSTOP
              F(I,J,1) = F(I,J,1)+TWBYDZ*BDZS(I,J)
  141     CONTINUE
  142 CONTINUE
  143 GO TO (150,144,147,147,144),NP
  144 DO 146 I=LSTART,LSTOP
          DO 145 J=MSTART,MSTOP
              F(I,J,N) = F(I,J,N)-C3*F(I,J,NP1)
  145     CONTINUE
  146 CONTINUE
      GO TO 150
  147 DO 149 I=LSTART,LSTOP
          DO 148 J=MSTART,MSTOP
              F(I,J,NP1) = F(I,J,NP1)-TWBYDZ*BDZF(I,J)
  148     CONTINUE
  149 CONTINUE
!
!     DEFINE A,B,C COEFFICIENTS IN W-ARRAY.
!
  150 CONTINUE
      IWB = NUNK+1
      IWC = IWB+NUNK
      IWW = IWC+NUNK
      DO 151 K=1,NUNK
          I = IWC+K-1
          W(K) = C3
          W(I) = C3
          I = IWB+K-1
```

```
            W(I) = -2.*C3+ELMBDA
  151 CONTINUE
        GO TO (155,155,153,152,152),NP
  152 W(IWC) = 2.*C3
  153 GO TO (155,155,154,154,155),NP
  154 W(IWB-1) = 2.*C3
  155 CONTINUE
        PERTRB = 0.
!
!       FOR SINGULAR PROBLEMS ADJUST DATA TO INSURE A SOLUTION WILL EXIST.
!
        GO TO (156,172,172,156,172),LP
  156 GO TO (157,172,172,157,172),MP
  157 GO TO (158,172,172,158,172),NP
  158 IF (ELMBDA) 172,160,159
  159 IERROR = 12
        GO TO 172
  160 CONTINUE
        MSTPM1 = MSTOP-1
        LSTPM1 = LSTOP-1
        NSTPM1 = NSTOP-1
        XLP = (2+LP)/3
        YLP = (2+MP)/3
        ZLP = (2+NP)/3
        S1 = 0.
        DO 164 K=2,NSTPM1
           DO 162 J=2,MSTPM1
              DO 161 I=2,LSTPM1
                 S1 = S1+F(I,J,K)
  161         CONTINUE
              S1 = S1+(F(1,J,K)+F(LSTOP,J,K))/XLP
  162      CONTINUE
           S2 = 0.
           DO 163 I=2,LSTPM1
              S2 = S2+F(I,1,K)+F(I,MSTOP,K)
  163      CONTINUE
```

```
      S2 = (S2+(F(1,1,K)+F(1,MSTOP,K)+F(LSTOP,1,K)+F(LSTOP,MSTOP,K))/ &
     &                                                              XLP)/YLP
      S1 = S1+S2
  164 CONTINUE
      S = (F(1,1,1)+F(LSTOP,1,1)+F(1,1,NSTOP)+F(LSTOP,1,NSTOP)+ &
     &    F(1,MSTOP,1)+F(LSTOP,MSTOP,1)+F(1,MSTOP,NSTOP)+ &
     &                                    F(LSTOP,MSTOP,NSTOP))/(XLP*YLP)
      DO 166 J=2,MSTPM1
         DO 165 I=2,LSTPM1
            S = S+F(I,J,1)+F(I,J,NSTOP)
  165    CONTINUE
  166 CONTINUE
      S2 = 0.
      DO 167 I=2,LSTPM1
         S2 = S2+F(I,1,1)+F(I,1,NSTOP)+F(I,MSTOP,1)+F(I,MSTOP,NSTOP)
  167 CONTINUE
      S = S2/YLP+S
      S2 = 0.
      DO 168 J=2,MSTPM1
         S2 = S2+F(1,J,1)+F(1,J,NSTOP)+F(LSTOP,J,1)+F(LSTOP,J,NSTOP)
  168 CONTINUE
      S = S2/XLP+S
      PERTRB = (S/ZLP+S1)/((FLOAT(LUNK+1)-XLP)*(FLOAT(MUNK+1)-YLP)* &
     &                                              (FLOAT(NUNK+1)-ZLP))
      DO 171 I=1,LUNK
         DO 170 J=1,MUNK
            DO 169 K=1,NUNK
               F(I,J,K) = F(I,J,K)-PERTRB
  169       CONTINUE
  170    CONTINUE
  171 CONTINUE
  172 CONTINUE
      NPEROD = 0
      IF (NBDCND .EQ. 0) GO TO 173
      NPEROD = 1
      W(1) = 0.
```

```
      W(IWW-1) = 0.
  173 CONTINUE
      CALL POIS3D (LBDCND,LUNK,C1,MBDCND,MUNK,C2,NPEROD,NUNK,W,W(IWB), &
     &             W(IWC),LDIMF,MDIMF,F(LSTART,MSTART,NSTART),IR,W(IWW))
!
!     FILL IN SIDES FOR PERIODIC BOUNDARY CONDITIONS.
!
      IF (LP .NE. 1) GO TO 180
      IF (MP .NE. 1) GO TO 175
      DO 174 K=NSTART,NSTOP
         F(1,MP1,K) = F(1,1,K)
  174 CONTINUE
      MSTOP = MP1
  175 IF (NP .NE. 1) GO TO 177
      DO 176 J=MSTART,MSTOP
         F(1,J,NP1) = F(1,J,1)
  176 CONTINUE
      NSTOP = NP1
  177 DO 179 J=MSTART,MSTOP
         DO 178 K=NSTART,NSTOP
            F(LP1,J,K) = F(1,J,K)
  178    CONTINUE
  179 CONTINUE
  180 CONTINUE
      IF (MP .NE. 1) GO TO 185
      IF (NP .NE. 1) GO TO 182
      DO 181 I=LSTART,LSTOP
         F(I,1,NP1) = F(I,1,1)
  181 CONTINUE
      NSTOP = NP1
  182 DO 184 I=LSTART,LSTOP
         DO 183 K=NSTART,NSTOP
            F(I,MP1,K) = F(I,1,K)
  183    CONTINUE
  184 CONTINUE
  185 CONTINUE
```

```
      IF (NP .NE. 1) GO TO 188
      DO 187 I=LSTART,LSTOP
         DO 186 J=MSTART,MSTOP
            F(I,J,NP1) = F(I,J,1)
  186    CONTINUE
  187 CONTINUE
  188 CONTINUE
      RETURN
      END

      FUNCTION PIMACH (DUM)
!
!     THIS SUBPROGRAM SUPPLIES THE VALUE OF THE CONSTANT PI CORRECT TO
!     MACHINE PRECISION WHERE
!
!     PI=3.14159265358979323846264338327950288419716939937510582097494 46
!
      PIMACH = 3.14159265358979
      RETURN
      END

      SUBROUTINE POIS3D (LPEROD,L,C1,MPEROD,M,C2,NPEROD,N,A,B,C,LDIMF, &
     &                   MDIMF,F,IERROR,W)
!
!
!     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!     *                                                               *
!     *                        F I S H P A K                          *
!     *                                                               *
!     *                                                               *
!     *     A PACKAGE OF FORTRAN SUBPROGRAMS FOR THE SOLUTION OF      *
!     *                                                               *
!     *      SEPARABLE ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS        *
!     *                                                               *
!     *                     (VERSION 3.1 , OCTOBER 1980)              *
!     *                                                               *
```

```
!     *                               BY                                *
!     *                                                                 *
!     *          JOHN ADAMS, PAUL SWARZTRAUBER AND ROLAND SWEET         *
!     *                                                                 *
!     *                               OF                                *
!     *                                                                 *
!     *          THE NATIONAL CENTER FOR ATMOSPHERIC RESEARCH           *
!     *                                                                 *
!     *                BOULDER, COLORADO  (80307)  U.S.A.               *
!     *                                                                 *
!     *                        WHICH IS SPONSORED BY                    *
!     *                                                                 *
!     *                THE NATIONAL SCIENCE FOUNDATION                  *
!     *                                                                 *
!     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
!
!     * * * * * * * * * *  PURPOSE  * * * * * * * * * * * * * * * * * *
!
!     SUBROUTINE POIS3D SOLVES THE LINEAR SYSTEM OF EQUATIONS
!
!       C1*(X(I-1,J,K)-2.*X(I,J,K)+X(I+1,J,K))
!     + C2*(X(I,J-1,K)-2.*X(I,J,K)+X(I,J+1,K))
!     + A(K)*X(I,J,K-1)+B(K)*X(I,J,K)+C(K)*X(I,J,K+1) = F(I,J,K)
!
!     FOR  I=1,2,...,L , J=1,2,...,M , AND K=1,2,...,N .
!
!     THE INDICES K-1 AND K+1 ARE EVALUATED MODULO N, I.E.
!     X(I,J,0) = X(I,J,N) AND X(I,J,N+1) = X(I,J,1). THE UNKNOWNS
!     X(0,J,K), X(L+1,J,K), X(I,0,K), AND X(I,M+1,K) ARE ASSUMED TO TAKE
!     ON CERTAIN PRESCRIBED VALUES DESCRIBED BELOW.
!
!     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
!
!     * * * * * * * *  PARAMETER DESCRIPTION  * * * * * * * * * *
```

```
!
!
!              * * * * * *    ON INPUT      * * * * * *
!
!     LPEROD     INDICATES THE VALUES THAT X(0,J,K) AND X(L+1,J,K) ARE
!                ASSUMED TO HAVE.
!
!                = 0    IF X(0,J,K) = X(L,J,K) AND X(L+1,J,K) = X(1,J,K).
!                = 1    IF X(0,J,K) = X(L+1,J,K) = 0.
!                = 2    IF X(0,J,K) = 0    AND X(L+1,J,K) = X(L-1,J,K).
!                = 3    IF X(0,J,K) = X(2,J,K) AND X(L+1,J,K) = X(L-1,J,K).
!                = 4    IF X(0,J,K) = X(2,J,K) AND X(L+1,J,K) = 0.
!
!     L          THE NUMBER OF UNKNOWNS IN THE I-DIRECTION. L MUST BE AT
!                LEAST 3.
!
!     C1         THE REAL CONSTANT THAT APPEARS IN THE ABOVE EQUATION.
!
!     MPEROD     INDICATES THE VALUES THAT X(I,0,K) AND X(I,M+1,K) ARE
!                ASSUMED TO HAVE.
!
!                = 0    IF X(I,0,K) = X(I,M,K) AND X(I,M+1,K) = X(I,1,K).
!                = 1    IF X(I,0,K) = X(I,M+1,K) = 0.
!                = 2    IF X(I,0,K) = 0 AND X(I,M+1,K) = X(I,M-1,K).
!                = 3    IF X(I,0,K) = X(I,2,K) AND X(I,M+1,K) = X(I,M-1,K).
!                = 4    IF X(I,0,K) = X(I,2,K) AND X(I,M+1,K) = 0.
!
!     M          THE NUMBER OF UNKNOWNS IN THE J-DIRECTION. M MUST BE AT
!                LEAST 3.
!
!     C2         THE REAL CONSTANT WHICH APPEARS IN THE ABOVE EQUATION.
!
!     NPEROD     = 0    IF A(1) AND C(N) ARE NOT ZERO.
!                = 1    IF A(1) = C(N) = 0.
!
!     N          THE NUMBER OF UNKNOWNS IN THE K-DIRECTION. N MUST BE AT
```

```
!                   LEAST 3.
!
!
!     A,B,C       ONE-DIMENSIONAL ARRAYS OF LENGTH N THAT SPECIFY THE
!                 COEFFICIENTS IN THE LINEAR EQUATIONS GIVEN ABOVE.
!
!                 IF NPEROD = 0 THE ARRAY ELEMENTS MUST NOT DEPEND UPON THE
!                 INDEX K, BUT MUST BE CONSTANT.    SPECIFICALLY,THE
!                 SUBROUTINE CHECKS THE FOLLOWING CONDITION
!
!                              A(K) = C(1)
!                              C(K) = C(1)
!                              B(K) = B(1)
!
!                        FOR K=1,2,...,N.
!
!     LDIMF       THE ROW (OR FIRST) DIMENSION OF THE THREE-DIMENSIONAL
!                 ARRAY F AS IT APPEARS IN THE PROGRAM CALLING POIS3D.
!                 THIS PARAMETER IS USED TO SPECIFY THE VARIABLE DIMENSION
!                 OF F.    LDIMF MUST BE AT LEAST L.
!
!     MDIMF       THE COLUMN (OR SECOND) DIMENSION OF THE THREE-DIMENSIONAL
!                 ARRAY F AS IT APPEARS IN THE PROGRAM CALLING POIS3D.
!                 THIS PARAMETER IS USED TO SPECIFY THE VARIABLE DIMENSION
!                 OF F.    MDIMF MUST BE AT LEAST M.
!
!     F           A THREE-DIMENSIONAL ARRAY THAT SPECIFIES THE VALUES OF
!                 THE RIGHT SIDE OF THE LINEAR SYSTEM OF EQUATIONS GIVEN
!                 ABOVE.    F MUST BE DIMENSIONED AT LEAST L X M X N.
!
!     W            A ONE-DIMENSIONAL ARRAY THAT MUST BE PROVIDED BY THE
!                 USER FOR WORK SPACE.    THE LENGTH OF W MUST BE AT LEAST
!                 30 + L + M + 2*N + MAX(L,M,N) +
!                 7*(INT((L+1)/2) + INT((M+1)/2)).
!
!
```

```
!               * * * * * *     ON OUTPUT     * * * * * *
!
!       F               CONTAINS THE SOLUTION X.
!
!       IERROR      AN ERROR FLAG THAT INDICATES INVALID INPUT PARAMETERS.
!                   EXCEPT FOR NUMBER ZERO, A SOLUTION IS NOT ATTEMPTED.
!                   = 0    NO ERROR
!                   = 1    IF LPEROD .LT. 0 OR .GT. 4
!                   = 2    IF L .LT. 3
!                   = 3    IF MPEROD .LT. 0 OR .GT. 4
!                   = 4    IF M .LT. 3
!                   = 5    IF NPEROD .LT. 0 OR .GT. 1
!                   = 6    IF N .LT. 3
!                   = 7    IF LDIMF .LT. L
!                   = 8    IF MDIMF .LT. M
!                   = 9    IF A(K) .NE. C(1) OR C(K) .NE. C(1) OR B(I) .NE.B(1)
!                              FOR SOME K=1,2,...,N.
!                   = 10 IF NPEROD = 1 AND A(1) .NE. 0 OR C(N) .NE. 0
!
!                   SINCE THIS IS THE ONLY MEANS OF INDICATING A POSSIBLY
!                   INCORRECT CALL TO POIS3D, THE USER SHOULD TEST IERROR
!                   AFTER THE CALL.
!
!
!       * * * * * * * *     PROGRAM SPECIFICATIONS     * * * * * * * * * * * *
!
!       DIMENSION OF     A(N),B(N),C(N),F(LDIMF,MDIMF,N),
!       ARGUMENTS        W(SEE ARGUMENT LIST)
!
!       LATEST           DECEMBER 1, 1978
!       REVISION
!
!       SUBPROGRAMS      POIS3D,POS3D1,TRID,RFFTI,RFFTF,RFFTF1,RFFTB,
!       REQUIRED         RFFTB1,COSTI,COST,SINTI,SINT,COSQI,COSQF,COSQF1
!                        COSQB,COSQB1,SINQI,SINQF,SINQB,CFFTI,CFFTI1,
!                        CFFTB,CFFTB1,PASSB2,PASSB3,PASSB4,PASSB,CFFTF,
```

```
!                       CFFTF1,PASSF1,PASSF2,PASSF3,PASSF4,PASSF,PIMACH,
!
!     SPECIAL           NONE
!     CONDITIONS
!
!     COMMON              VALUE
!     BLOCKS
!
!     I/O               NONE
!
!     PRECISION         SINGLE
!
!     SPECIALIST        ROLAND SWEET
!
!     LANGUAGE            FORTRAN
!
!     HISTORY           WRITTEN BY ROLAND SWEET AT NCAR IN JULY,1977
!
!     ALGORITHM           THIS SUBROUTINE SOLVES THREE-DIMENSIONAL BLOCK
!                       TRIDIAGONAL LINEAR SYSTEMS ARISING FROM FINITE
!                       DIFFERENCE APPROXIMATIONS TO THREE-DIMENSIONAL
!                       POISSON EQUATIONS USING THE FOURIER TRANSFORM
!                       PACKAGE SCLRFFTPAK WRITTEN BY PAUL SWARZTRAUBER.
!
!     SPACE             6561(DECIMAL) = 14641(OCTAL) LOCATIONS ON THE
!     REQUIRED           NCAR CONTROL DATA 7600
!
!     TIMING AND              THE EXECUTION TIME T ON THE NCAR CONTROL DATA
!     ACCURACY           7600 FOR SUBROUTINE POIS3D IS ROUGHLY PROPORTIONAL
!                       TO L*M*N*(LOG2(L)+LOG2(M)+5), BUT ALSO DEPENDS ON
!                       INPUT PARAMETERS LPEROD AND MPEROD.    SOME TYPICAL
!                       VALUES ARE LISTED IN THE TABLE BELOW WHEN NPEROD=0.
!                            TO MEASURE THE ACCURACY OF THE ALGORITHM A
!                       UNIFORM RANDOM NUMBER GENERATOR WAS USED TO CREATE
!                       A SOLUTION ARRAY X FOR THE SYSTEM GIVEN IN THE
!                       'PURPOSE' WITH
```

```
!
!                    A(K) = C(K) = -0.5*B(K) = 1,          K=1,2,...,N
!
!                  AND, WHEN NPEROD = 1
!
!                    A(1) = C(N) = 0
!                    A(N) = C(1) = 2.
!
!                  THE SOLUTION X WAS SUBSTITUTED INTO THE GIVEN SYS-
!                  TEM AND, USING DOUBLE PRECISION, A RIGHT SIDE Y WAS
!                  COMPUTED.  USING THIS ARRAY Y SUBROUTINE POIS WAS
!                  CALLED TO PRODUCE AN APPROXIMATE SOLUTION Z.  THEN
!                  THE RELATIVE ERROR, DEFINED AS
!
!                  E = MAX(ABS(Z(I,J,K)-X(I,J,K)))/MAX(ABS(X(I,J,K)))
!
!                  WHERE THE TWO MAXIMA ARE TAKEN OVER I=1,2,...,L,
!                  J=1,2,...,M AND K=1,2,...,N, WAS COMPUTED.  THE
!                  VALUE OF E IS GIVEN IN THE TABLE BELOW FOR SOME
!                  TYPICAL VALUES OF L,M AND N.
!
!
!                    L(=M=N)   LPEROD    MPEROD    T(MSECS)    E
!                    ------    ------    ------    --------  ------
!
!                      16         0         0         272     1.E-13
!                      15         1         1         287     4.E-13
!                      17         3         3         338     2.E-13
!                      32         0         0        1755     2.E-13
!                      31         1         1        1894     2.E-12
!                      33         3         3        2042     7.E-13
!
!
!     PORTABILITY    AMERICAN NATIONAL STANDARDS INSTITUTE FORTRAN.
!                    THE MACHINE DEPENDENT CONSTANT PI IS DEFINED IN
!                    FUNCTION PIMACH.
```

```
!
!       REQUIRED            COS,SIN,ATAN
!       RESIDENT
!       ROUTINES
!
!       REFERENCE           NONE
!
!     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!
      DIMENSION             A(1)            ,B(1)           ,C(1)           , &
     &                      F(LDIMF,MDIMF,1)          ,W(1)           ,SAVE(6)
       LP = LPEROD+1
       MP = MPEROD+1
       NP = NPEROD+1
!
!       CHECK FOR INVALID INPUT.
!
       IERROR = 0
       IF (LP.LT.1 .OR. LP.GT.5) IERROR = 1
       IF (L .LT. 3) IERROR = 2
       IF (MP.LT.1 .OR. MP.GT.5) IERROR = 3
       IF (M .LT. 3) IERROR = 4
       IF (NP.LT.1 .OR. NP.GT.2) IERROR = 5
       IF (N .LT. 3) IERROR = 6
       IF (LDIMF .LT. L) IERROR = 7
       IF (MDIMF .LT. M) IERROR = 8
       IF (NP .NE. 1) GO TO 103
       DO 101 K=1,N
          IF (A(K) .NE. C(1)) GO TO 102
          IF (C(K) .NE. C(1)) GO TO 102
          IF (B(K) .NE. B(1)) GO TO 102
  101 CONTINUE
       GO TO 104
  102 IERROR = 9
  103 IF (NPEROD.EQ.1 .AND. (A(1).NE.0. .OR. C(N).NE.0.)) IERROR = 10
  104 IF (IERROR .NE. 0) GO TO 122
```

```
      IWYRT = L+1
      IWT = IWYRT+M
      IWD = IWT+MAX0(L,M,N)+1
      IWBB = IWD+N
      IWX = IWBB+N
      IWY = IWX+7*((L+1)/2)+15
      GO TO (105,114),NP
!
!     REORDER UNKNOWNS WHEN NPEROD = 0.
!
  105 NH = (N+1)/2
      NHM1 = NH-1
      NODD = 1
      IF (2*NH .EQ. N) NODD = 2
      DO 111 I=1,L
         DO 110 J=1,M
            DO 106 K=1,NHM1
               NHPK = NH+K
               NHMK = NH-K
               W(K) = F(I,J,NHMK)-F(I,J,NHPK)
               W(NHPK) = F(I,J,NHMK)+F(I,J,NHPK)
  106       CONTINUE
            W(NH) = 2.*F(I,J,NH)
            GO TO (108,107),NODD
  107       W(N) = 2.*F(I,J,N)
  108       DO 109 K=1,N
               F(I,J,K) = W(K)
  109       CONTINUE
  110    CONTINUE
  111 CONTINUE
      SAVE(1) = C(NHM1)
      SAVE(2) = A(NH)
      SAVE(3) = C(NH)
      SAVE(4) = B(NHM1)
      SAVE(5) = B(N)
      SAVE(6) = A(N)
```

```
      C(NHM1) = 0.
      A(NH) = 0.
      C(NH) = 2.*C(NH)
      GO TO (112,113),NODD
112 B(NHM1) = B(NHM1)-A(NH-1)
      B(N) = B(N)+A(N)
      GO TO 114
113 A(N) = C(NH)
114 CONTINUE
      CALL POS3D1 (LP,L,MP,M,N,A,B,C,LDIMF,MDIMF,F,W,W(IWYRT),W(IWT), &
     &                    W(IWD),W(IWX),W(IWY),C1,C2,W(IWBB))
      GO TO (115,122),NP
115 DO 121 I=1,L
         DO 120 J=1,M
            DO 116 K=1,NHM1
               NHMK = NH-K
               NHPK = NH+K
               W(NHMK) = .5*(F(I,J,NHPK)+F(I,J,K))
               W(NHPK) = .5*(F(I,J,NHPK)-F(I,J,K))
116         CONTINUE
            W(NH) = .5*F(I,J,NH)
            GO TO (118,117),NODD
117         W(N) = .5*F(I,J,N)
118         DO 119 K=1,N
               F(I,J,K) = W(K)
119         CONTINUE
120      CONTINUE
121 CONTINUE
      C(NHM1) = SAVE(1)
      A(NH) = SAVE(2)
      C(NH) = SAVE(3)
      B(NHM1) = SAVE(4)
      B(N) = SAVE(5)
      A(N) = SAVE(6)
122 CONTINUE
      RETURN
```

```
      END


      SUBROUTINE POS3D1 (LP,L,MP,M,N,A,B,C,LDIMF,MDIMF,F,XRT,YRT,T,D, &
     &                   WX,WY,C1,C2,BB)
      DIMENSION          A(1)          ,B(1)          ,C(1)          , &
     &                   F(LDIMF,MDIMF,1)          ,XRT(1)        ,YRT(1)        , &
     &                   T(1)          ,D(1)          ,WX(1)         ,WY(1)         , &
     &                   BB(1)
      PI = PIMACH(DUM)
      LR = L
      MR = M
      NR = N
!
!     GENERATE TRANSFORM ROOTS
!
      LRDEL = ((LP-1)*(LP-3)*(LP-5))/3
      SCALX = LR+LRDEL
      DX = PI/(2.*SCALX)
      GO TO (108,103,101,102,101),LP
  101 DI = 0.5
      SCALX = 2.*SCALX
      GO TO 104
  102 DI = 1.0
      GO TO 104
  103 DI = 0.0
  104 DO 105 I=1,LR
          XRT(I) = -4.*C1*(SIN((FLOAT(I)-DI)*DX))**2
  105 CONTINUE
      SCALX = 2.*SCALX
      GO TO (112,106,110,107,111),LP
  106 CALL SINTI (LR,WX)
      GO TO 112
  107 CALL COSTI (LR,WX)
      GO TO 112
  108 XRT(1) = 0.
      XRT(LR) = -4.*C1
```

```
      DO 109 I=3,LR,2
         XRT(I-1) = -4.*C1*(SIN(FLOAT((I-1))*DX))**2
         XRT(I) = XRT(I-1)
109 CONTINUE
      CALL RFFTI (LR,WX)
      GO TO 112
110 CALL SINQI (LR,WX)
      GO TO 112
111 CALL COSQI (LR,WX)
112 CONTINUE
      MRDEL = ((MP-1)*(MP-3)*(MP-5))/3
      SCALY = MR+MRDEL
      DY = PI/(2.*SCALY)
      GO TO (120,115,113,114,113),MP
113 DJ = 0.5
      SCALY = 2.*SCALY
      GO TO 116
114 DJ = 1.0
      GO TO 116
115 DJ = 0.0
116 DO 117 J=1,MR
         YRT(J) = -4.*C2*(SIN((FLOAT(J)-DJ)*DY))**2
117 CONTINUE
      SCALY = 2.*SCALY
      GO TO (124,118,122,119,123),MP
118 CALL SINTI (MR,WY)
      GO TO 124
119 CALL COSTI (MR,WY)
      GO TO 124
120 YRT(1) = 0.
      YRT(MR) = -4.*C2
      DO 121 J=3,MR,2
         YRT(J-1) = -4.*C2*(SIN(FLOAT((J-1))*DY))**2
         YRT(J) = YRT(J-1)
121 CONTINUE
      CALL RFFTI (MR,WY)
```

```
      GO TO 124
  122 CALL SINQI (MR,WY)
      GO TO 124
  123 CALL COSQI (MR,WY)
  124 CONTINUE
      IFWRD = 1
      IS = 1
  125 CONTINUE
!
!       TRANSFORM X
!
      DO 141 J=1,MR
         DO 140 K=1,NR
            DO 126 I=1,LR
               T(I) = F(I,J,K)
  126       CONTINUE
            GO TO (127,130,131,134,135),LP
  127       GO TO (128,129),IFWRD
  128       CALL RFFTF (LR,T,WX)
            GO TO 138
  129       CALL RFFTB (LR,T,WX)
            GO TO 138
  130       CALL SINT (LR,T,WX)
            GO TO 138
  131       GO TO (132,133),IFWRD
  132       CALL SINQF (LR,T,WX)
            GO TO 138
  133       CALL SINQB (LR,T,WX)
            GO TO 138
  134       CALL COST (LR,T,WX)
            GO TO 138
  135       GO TO (136,137),IFWRD
  136       CALL COSQF (LR,T,WX)
            GO TO 138
  137       CALL COSQB (LR,T,WX)
  138       CONTINUE
```

```
                DO 139 I=1,LR
                    F(I,J,K) = T(I)
  139           CONTINUE
  140       CONTINUE
  141 CONTINUE
        GO TO (142,164),IFWRD
!
!       TRANSFORM Y
!
  142 CONTINUE
        DO 158 I=1,LR
            DO 157 K=1,NR
                DO 143 J=1,MR
                    T(J) = F(I,J,K)
  143           CONTINUE
                GO TO (144,147,148,151,152),MP
  144           GO TO (145,146),IFWRD
  145           CALL RFFTF (MR,T,WY)
                GO TO 155
  146           CALL RFFTB (MR,T,WY)
                GO TO 155
  147           CALL SINT (MR,T,WY)
                GO TO 155
  148           GO TO (149,150),IFWRD
  149           CALL SINQF (MR,T,WY)
                GO TO 155
  150           CALL SINQB (MR,T,WY)
                GO TO 155
  151           CALL COST (MR,T,WY)
                GO TO 155
  152           GO TO (153,154),IFWRD
  153           CALL COSQF (MR,T,WY)
                GO TO 155
  154           CALL COSQB (MR,T,WY)
  155           CONTINUE
                DO 156 J=1,MR
```

```
                F(I,J,K) = T(J)
  156           CONTINUE
  157       CONTINUE
  158 CONTINUE
      GO TO (159,125),IFWRD
  159 CONTINUE
!
!       SOLVE TRIDIAGONAL SYSTEMS IN Z
!
      DO 163 I=1,LR
         DO 162 J=1,MR
            DO 160 K=1,NR
               BB(K) = B(K)+XRT(I)+YRT(J)
               T(K) = F(I,J,K)
  160          CONTINUE
               CALL TRID (NR,A,BB,C,T,D)
               DO 161 K=1,NR
                  F(I,J,K) = T(K)
  161          CONTINUE
  162       CONTINUE
  163 CONTINUE
      IFWRD = 2
      IS = -1
      GO TO 142
  164 CONTINUE
      DO 167 I=1,LR
         DO 166 J=1,MR
            DO 165 K=1,NR
               F(I,J,K) = F(I,J,K)/(SCALX*SCALY)
  165          CONTINUE
  166       CONTINUE
  167 CONTINUE
      RETURN
      END

      SUBROUTINE RADB2 (IDO,L1,CC,CH,WA1)
```

```
      DIMENSION         CC(IDO,2,L1)           ,CH(IDO,L1,2)           ,&
     &                  WA1(1)
      DO 101 K=1,L1
         CH(1,K,1) = CC(1,1,K)+CC(IDO,2,K)
         CH(1,K,2) = CC(1,1,K)-CC(IDO,2,K)
101 CONTINUE
      IF (IDO-2) 107,105,102
102 IDP2 = IDO+2
      DO 104 K=1,L1
         DO 103 I=3,IDO,2
            IC = IDP2-I
            CH(I-1,K,1) = CC(I-1,1,K)+CC(IC-1,2,K)
            TR2 = CC(I-1,1,K)-CC(IC-1,2,K)
            CH(I,K,1) = CC(I,1,K)-CC(IC,2,K)
            TI2 = CC(I,1,K)+CC(IC,2,K)
            CH(I-1,K,2) = WA1(I-2)*TR2-WA1(I-1)*TI2
            CH(I,K,2) = WA1(I-2)*TI2+WA1(I-1)*TR2
103      CONTINUE
104 CONTINUE
      IF (MOD(IDO,2) .EQ. 1) RETURN
105 DO 106 K=1,L1
         CH(IDO,K,1) = CC(IDO,1,K)+CC(IDO,1,K)
         CH(IDO,K,2) = -(CC(1,2,K)+CC(1,2,K))
106 CONTINUE
107 RETURN
      END

      SUBROUTINE RADB3 (IDO,L1,CC,CH,WA1,WA2)
      DIMENSION         CC(IDO,3,L1)           ,CH(IDO,L1,3)           , &
     &                  WA1(1)        ,WA2(1)
      DATA TAUR,TAUI /-.5,.866025403784439/
      DO 101 K=1,L1
         TR2 = CC(IDO,2,K)+CC(IDO,2,K)
         CR2 = CC(1,1,K)+TAUR*TR2
         CH(1,K,1) = CC(1,1,K)+TR2
         CI3 = TAUI*(CC(1,3,K)+CC(1,3,K))
```

```
          CH(1,K,2) = CR2-CI3
          CH(1,K,3) = CR2+CI3
101 CONTINUE
      IF (IDO .EQ. 1) RETURN
      IDP2 = IDO+2
      DO 103 K=1,L1
          DO 102 I=3,IDO,2
              IC = IDP2-I
              TR2 = CC(I-1,3,K)+CC(IC-1,2,K)
              CR2 = CC(I-1,1,K)+TAUR*TR2
              CH(I-1,K,1) = CC(I-1,1,K)+TR2
              TI2 = CC(I,3,K)-CC(IC,2,K)
              CI2 = CC(I,1,K)+TAUR*TI2
              CH(I,K,1) = CC(I,1,K)+TI2
              CR3 = TAUI*(CC(I-1,3,K)-CC(IC-1,2,K))
              CI3 = TAUI*(CC(I,3,K)+CC(IC,2,K))
              DR2 = CR2-CI3
              DR3 = CR2+CI3
              DI2 = CI2+CR3
              DI3 = CI2-CR3
              CH(I-1,K,2) = WA1(I-2)*DR2-WA1(I-1)*DI2
              CH(I,K,2) = WA1(I-2)*DI2+WA1(I-1)*DR2
              CH(I-1,K,3) = WA2(I-2)*DR3-WA2(I-1)*DI3
              CH(I,K,3) = WA2(I-2)*DI3+WA2(I-1)*DR3
102       CONTINUE
103 CONTINUE
      RETURN
      END

      SUBROUTINE RADB4 (IDO,L1,CC,CH,WA1,WA2,WA3)
      DIMENSION       CC(IDO,4,L1)           ,CH(IDO,L1,4)           , &
     &                WA1(1)      ,WA2(1)      ,WA3(1)
      DATA SQRT2 /1.414213562373095/
      DO 101 K=1,L1
          TR1 = CC(1,1,K)-CC(IDO,4,K)
          TR2 = CC(1,1,K)+CC(IDO,4,K)
```

```
            TR3 = CC(IDO,2,K)+CC(IDO,2,K)
            TR4 = CC(1,3,K)+CC(1,3,K)
            CH(1,K,1) = TR2+TR3
            CH(1,K,2) = TR1-TR4
            CH(1,K,3) = TR2-TR3
            CH(1,K,4) = TR1+TR4
101 CONTINUE
      IF (IDO-2) 107,105,102
102 IDP2 = IDO+2
      DO 104 K=1,L1
         DO 103 I=3,IDO,2
            IC = IDP2-I
            TI1 = CC(I,1,K)+CC(IC,4,K)
            TI2 = CC(I,1,K)-CC(IC,4,K)
            TI3 = CC(I,3,K)-CC(IC,2,K)
            TR4 = CC(I,3,K)+CC(IC,2,K)
            TR1 = CC(I-1,1,K)-CC(IC-1,4,K)
            TR2 = CC(I-1,1,K)+CC(IC-1,4,K)
            TI4 = CC(I-1,3,K)-CC(IC-1,2,K)
            TR3 = CC(I-1,3,K)+CC(IC-1,2,K)
            CH(I-1,K,1) = TR2+TR3
            CR3 = TR2-TR3
            CH(I,K,1) = TI2+TI3
            CI3 = TI2-TI3
            CR2 = TR1-TR4
            CR4 = TR1+TR4
            CI2 = TI1+TI4
            CI4 = TI1-TI4
            CH(I-1,K,2) = WA1(I-2)*CR2-WA1(I-1)*CI2
            CH(I,K,2) = WA1(I-2)*CI2+WA1(I-1)*CR2
            CH(I-1,K,3) = WA2(I-2)*CR3-WA2(I-1)*CI3
            CH(I,K,3) = WA2(I-2)*CI3+WA2(I-1)*CR3
            CH(I-1,K,4) = WA3(I-2)*CR4-WA3(I-1)*CI4
            CH(I,K,4) = WA3(I-2)*CI4+WA3(I-1)*CR4
103      CONTINUE
104 CONTINUE
```

```
      IF (MOD(IDO,2) .EQ. 1) RETURN
  105 CONTINUE
      DO 106 K=1,L1
         TI1 = CC(1,2,K)+CC(1,4,K)
         TI2 = CC(1,4,K)-CC(1,2,K)
         TR1 = CC(IDO,1,K)-CC(IDO,3,K)
         TR2 = CC(IDO,1,K)+CC(IDO,3,K)
         CH(IDO,K,1) = TR2+TR2
         CH(IDO,K,2) = SQRT2*(TR1-TI1)
         CH(IDO,K,3) = TI2+TI2
         CH(IDO,K,4) = -SQRT2*(TR1+TI1)
  106 CONTINUE
  107 RETURN
      END


      SUBROUTINE RADB5 (IDO,L1,CC,CH,WA1,WA2,WA3,WA4)
      DIMENSION       CC(IDO,5,L1)           ,CH(IDO,L1,5)          , &
     &                WA1(1)      ,WA2(1)      ,WA3(1)      ,WA4(1)
      DATA TR11,TI11,TR12,TI12 /.309016994374947,.951056516295154, &
     &-.809016994374947,.587785252292473/
      DO 101 K=1,L1
         TI5 = CC(1,3,K)+CC(1,3,K)
         TI4 = CC(1,5,K)+CC(1,5,K)
         TR2 = CC(IDO,2,K)+CC(IDO,2,K)
         TR3 = CC(IDO,4,K)+CC(IDO,4,K)
         CH(1,K,1) = CC(1,1,K)+TR2+TR3
         CR2 = CC(1,1,K)+TR11*TR2+TR12*TR3
         CR3 = CC(1,1,K)+TR12*TR2+TR11*TR3
         CI5 = TI11*TI5+TI12*TI4
         CI4 = TI12*TI5-TI11*TI4
         CH(1,K,2) = CR2-CI5
         CH(1,K,3) = CR3-CI4
         CH(1,K,4) = CR3+CI4
         CH(1,K,5) = CR2+CI5
  101 CONTINUE
```

```
IF (IDO .EQ. 1) RETURN
IDP2 = IDO+2
DO 103 K=1,L1
    DO 102 I=3,IDO,2
        IC = IDP2-I
        TI5 = CC(I,3,K)+CC(IC,2,K)
        TI2 = CC(I,3,K)-CC(IC,2,K)
        TI4 = CC(I,5,K)+CC(IC,4,K)
        TI3 = CC(I,5,K)-CC(IC,4,K)
        TR5 = CC(I-1,3,K)-CC(IC-1,2,K)
        TR2 = CC(I-1,3,K)+CC(IC-1,2,K)
        TR4 = CC(I-1,5,K)-CC(IC-1,4,K)
        TR3 = CC(I-1,5,K)+CC(IC-1,4,K)
        CH(I-1,K,1) = CC(I-1,1,K)+TR2+TR3
        CH(I,K,1) = CC(I,1,K)+TI2+TI3
        CR2 = CC(I-1,1,K)+TR11*TR2+TR12*TR3
        CI2 = CC(I,1,K)+TR11*TI2+TR12*TI3
        CR3 = CC(I-1,1,K)+TR12*TR2+TR11*TR3
        CI3 = CC(I,1,K)+TR12*TI2+TR11*TI3
        CR5 = TI11*TR5+TI12*TR4
        CI5 = TI11*TI5+TI12*TI4
        CR4 = TI12*TR5-TI11*TR4
        CI4 = TI12*TI5-TI11*TI4
        DR3 = CR3-CI4
        DR4 = CR3+CI4
        DI3 = CI3+CR4
        DI4 = CI3-CR4
        DR5 = CR2+CI5
        DR2 = CR2-CI5
        DI5 = CI2-CR5
        DI2 = CI2+CR5
        CH(I-1,K,2) = WA1(I-2)*DR2-WA1(I-1)*DI2
        CH(I,K,2) = WA1(I-2)*DI2+WA1(I-1)*DR2
        CH(I-1,K,3) = WA2(I-2)*DR3-WA2(I-1)*DI3
        CH(I,K,3) = WA2(I-2)*DI3+WA2(I-1)*DR3
        CH(I-1,K,4) = WA3(I-2)*DR4-WA3(I-1)*DI4
```

```
            CH(I,K,4) = WA3(I-2)*DI4+WA3(I-1)*DR4
            CH(I-1,K,5) = WA4(I-2)*DR5-WA4(I-1)*DI5
            CH(I,K,5) = WA4(I-2)*DI5+WA4(I-1)*DR5
102      CONTINUE
103 CONTINUE
      RETURN
      END

      SUBROUTINE RADBG (IDO,IP,L1,IDL1,CC,C1,C2,CH,CH2,WA)
      DIMENSION         CH(IDO,L1,IP)          ,CC(IDO,IP,L1)          , &
     &                  C1(IDO,L1,IP)          ,C2(IDL1,IP), &
     &                  CH2(IDL1,IP)           ,WA(1)
      DATA TPI/6.28318530717959/
      ARG = TPI/FLOAT(IP)
      DCP = COS(ARG)
      DSP = SIN(ARG)
      IDP2 = IDO+2
      NBD = (IDO-1)/2
      IPP2 = IP+2
      IPPH = (IP+1)/2
      IF (IDO .LT. L1) GO TO 103
      DO 102 K=1,L1
         DO 101 I=1,IDO
            CH(I,K,1) = CC(I,1,K)
101      CONTINUE
102 CONTINUE
      GO TO 106
103 DO 105 I=1,IDO
         DO 104 K=1,L1
            CH(I,K,1) = CC(I,1,K)
104      CONTINUE
105 CONTINUE
106 DO 108 J=2,IPPH
         JC = IPP2-J
         J2 = J+J
         DO 107 K=1,L1
```

```
            CH(1,K,J) = CC(IDO,J2-2,K)+CC(IDO,J2-2,K)
            CH(1,K,JC) = CC(1,J2-1,K)+CC(1,J2-1,K)
107      CONTINUE
108 CONTINUE
      IF (IDO .EQ. 1) GO TO 116
      IF (NBD .LT. L1) GO TO 112
      DO 111 J=2,IPPH
         JC = IPP2-J
         DO 110 K=1,L1
            DO 109 I=3,IDO,2
               IC = IDP2-I
               CH(I-1,K,J) = CC(I-1,2*J-1,K)+CC(IC-1,2*J-2,K)
               CH(I-1,K,JC) = CC(I-1,2*J-1,K)-CC(IC-1,2*J-2,K)
               CH(I,K,J) = CC(I,2*J-1,K)-CC(IC,2*J-2,K)
               CH(I,K,JC) = CC(I,2*J-1,K)+CC(IC,2*J-2,K)
109         CONTINUE
110      CONTINUE
111 CONTINUE
      GO TO 116
112 DO 115 J=2,IPPH
         JC = IPP2-J
         DO 114 I=3,IDO,2
            IC = IDP2-I
            DO 113 K=1,L1
               CH(I-1,K,J) = CC(I-1,2*J-1,K)+CC(IC-1,2*J-2,K)
               CH(I-1,K,JC) = CC(I-1,2*J-1,K)-CC(IC-1,2*J-2,K)
               CH(I,K,J) = CC(I,2*J-1,K)-CC(IC,2*J-2,K)
               CH(I,K,JC) = CC(I,2*J-1,K)+CC(IC,2*J-2,K)
113         CONTINUE
114      CONTINUE
115 CONTINUE
116 AR1 = 1.
      AI1 = 0.
      DO 120 L=2,IPPH
         LC = IPP2-L
         AR1H = DCP*AR1-DSP*AI1
```

```
      AI1 = DCP*AI1+DSP*AR1
      AR1 = AR1H
      DO 117 IK=1,IDL1
         C2(IK,L) = CH2(IK,1)+AR1*CH2(IK,2)
         C2(IK,LC) = AI1*CH2(IK,IP)
117   CONTINUE
      DC2 = AR1
      DS2 = AI1
      AR2 = AR1
      AI2 = AI1
      DO 119 J=3,IPPH
         JC = IPP2-J
         AR2H = DC2*AR2-DS2*AI2
         AI2 = DC2*AI2+DS2*AR2
         AR2 = AR2H
         DO 118 IK=1,IDL1
            C2(IK,L) = C2(IK,L)+AR2*CH2(IK,J)
            C2(IK,LC) = C2(IK,LC)+AI2*CH2(IK,JC)
118      CONTINUE
119   CONTINUE
120 CONTINUE
    DO 122 J=2,IPPH
      DO 121 IK=1,IDL1
         CH2(IK,1) = CH2(IK,1)+CH2(IK,J)
121   CONTINUE
122 CONTINUE
    DO 124 J=2,IPPH
      JC = IPP2-J
      DO 123 K=1,L1
         CH(1,K,J) = C1(1,K,J)-C1(1,K,JC)
         CH(1,K,JC) = C1(1,K,J)+C1(1,K,JC)
123   CONTINUE
124 CONTINUE
    IF (IDO .EQ. 1) GO TO 132
    IF (NBD .LT. L1) GO TO 128
    DO 127 J=2,IPPH
```

```
      JC = IPP2-J
      DO 126 K=1,L1
         DO 125 I=3,IDO,2
            CH(I-1,K,J) = C1(I-1,K,J)-C1(I,K,JC)
            CH(I-1,K,JC) = C1(I-1,K,J)+C1(I,K,JC)
            CH(I,K,J) = C1(I,K,J)+C1(I-1,K,JC)
            CH(I,K,JC) = C1(I,K,J)-C1(I-1,K,JC)
125         CONTINUE
126      CONTINUE
127 CONTINUE
    GO TO 132
128 DO 131 J=2,IPPH
      JC = IPP2-J
      DO 130 I=3,IDO,2
         DO 129 K=1,L1
            CH(I-1,K,J) = C1(I-1,K,J)-C1(I,K,JC)
            CH(I-1,K,JC) = C1(I-1,K,J)+C1(I,K,JC)
            CH(I,K,J) = C1(I,K,J)+C1(I-1,K,JC)
            CH(I,K,JC) = C1(I,K,J)-C1(I-1,K,JC)
129         CONTINUE
130      CONTINUE
131 CONTINUE
132 CONTINUE
    IF (IDO .EQ. 1) RETURN
    DO 133 IK=1,IDL1
       C2(IK,1) = CH2(IK,1)
133 CONTINUE
    DO 135 J=2,IP
       DO 134 K=1,L1
          C1(1,K,J) = CH(1,K,J)
134    CONTINUE
135 CONTINUE
    IF (NBD .GT. L1) GO TO 139
    IS = -IDO
    DO 138 J=2,IP
       IS = IS+IDO
```

```
          IDIJ = IS
          DO 137 I=3,IDO,2
              IDIJ = IDIJ+2
              DO 136 K=1,L1
                  C1(I-1,K,J) = WA(IDIJ-1)*CH(I-1,K,J)-WA(IDIJ)*CH(I,K,J)
                  C1(I,K,J) = WA(IDIJ-1)*CH(I,K,J)+WA(IDIJ)*CH(I-1,K,J)
136               CONTINUE
137       CONTINUE
138 CONTINUE
      GO TO 143
139 IS = -IDO
      DO 142 J=2,IP
          IS = IS+IDO
          DO 141 K=1,L1
              IDIJ = IS
              DO 140 I=3,IDO,2
                  IDIJ = IDIJ+2
                  C1(I-1,K,J) = WA(IDIJ-1)*CH(I-1,K,J)-WA(IDIJ)*CH(I,K,J)
                  C1(I,K,J) = WA(IDIJ-1)*CH(I,K,J)+WA(IDIJ)*CH(I-1,K,J)
140               CONTINUE
141       CONTINUE
142 CONTINUE
143 RETURN
      END


    SUBROUTINE RADF2 (IDO,L1,CC,CH,WA1)
      DIMENSION         CH(IDO,2,L1)              ,CC(IDO,L1,2)             , &
    &                   WA1(1)
      DO 101 K=1,L1
          CH(1,1,K) = CC(1,K,1)+CC(1,K,2)
          CH(IDO,2,K) = CC(1,K,1)-CC(1,K,2)
101 CONTINUE
      IF (IDO-2) 107,105,102
102 IDP2 = IDO+2
      DO 104 K=1,L1
          DO 103 I=3,IDO,2
```

```
            IC = IDP2-I
            TR2 = WA1(I-2)*CC(I-1,K,2)+WA1(I-1)*CC(I,K,2)
            TI2 = WA1(I-2)*CC(I,K,2)-WA1(I-1)*CC(I-1,K,2)
            CH(I,1,K) = CC(I,K,1)+TI2
            CH(IC,2,K) = TI2-CC(I,K,1)
            CH(I-1,1,K) = CC(I-1,K,1)+TR2
            CH(IC-1,2,K) = CC(I-1,K,1)-TR2
103       CONTINUE
104 CONTINUE
      IF (MOD(IDO,2) .EQ. 1) RETURN
105 DO 106 K=1,L1
          CH(1,2,K) = -CC(IDO,K,2)
          CH(IDO,1,K) = CC(IDO,K,1)
106 CONTINUE
107 RETURN
      END

      SUBROUTINE RADF3 (IDO,L1,CC,CH,WA1,WA2)
      DIMENSION         CH(IDO,3,L1)        ,CC(IDO,L1,3)       , &
     &                  WA1(1)      ,WA2(1)
      DATA TAUR,TAUI /-.5,.866025403784439/
      DO 101 K=1,L1
          CR2 = CC(1,K,2)+CC(1,K,3)
          CH(1,1,K) = CC(1,K,1)+CR2
          CH(1,3,K) = TAUI*(CC(1,K,3)-CC(1,K,2))
          CH(IDO,2,K) = CC(1,K,1)+TAUR*CR2
101 CONTINUE
      IF (IDO .EQ. 1) RETURN
      IDP2 = IDO+2
      DO 103 K=1,L1
          DO 102 I=3,IDO,2
              IC = IDP2-I
              DR2 = WA1(I-2)*CC(I-1,K,2)+WA1(I-1)*CC(I,K,2)
              DI2 = WA1(I-2)*CC(I,K,2)-WA1(I-1)*CC(I-1,K,2)
              DR3 = WA2(I-2)*CC(I-1,K,3)+WA2(I-1)*CC(I,K,3)
              DI3 = WA2(I-2)*CC(I,K,3)-WA2(I-1)*CC(I-1,K,3)
```

```
            CR2 = DR2+DR3
            CI2 = DI2+DI3
            CH(I-1,1,K) = CC(I-1,K,1)+CR2
            CH(I,1,K) = CC(I,K,1)+CI2
            TR2 = CC(I-1,K,1)+TAUR*CR2
            TI2 = CC(I,K,1)+TAUR*CI2
            TR3 = TAUI*(DI2-DI3)
            TI3 = TAUI*(DR3-DR2)
            CH(I-1,3,K) = TR2+TR3
            CH(IC-1,2,K) = TR2-TR3
            CH(I,3,K) = TI2+TI3
            CH(IC,2,K) = TI3-TI2
102      CONTINUE
103 CONTINUE
      RETURN
      END

      SUBROUTINE RADF4 (IDO,L1,CC,CH,WA1,WA2,WA3)
      DIMENSION         CC(IDO,L1,4)           ,CH(IDO,4,L1)           , &
     &                  WA1(1)      ,WA2(1)      ,WA3(1)
      DATA HSQT2 /.7071067811865475/
      DO 101 K=1,L1
         TR1 = CC(1,K,2)+CC(1,K,4)
         TR2 = CC(1,K,1)+CC(1,K,3)
         CH(1,1,K) = TR1+TR2
         CH(IDO,4,K) = TR2-TR1
         CH(IDO,2,K) = CC(1,K,1)-CC(1,K,3)
         CH(1,3,K) = CC(1,K,4)-CC(1,K,2)
101 CONTINUE
      IF (IDO-2) 107,105,102
102 IDP2 = IDO+2
      DO 104 K=1,L1
         DO 103 I=3,IDO,2
            IC = IDP2-I
            CR2 = WA1(I-2)*CC(I-1,K,2)+WA1(I-1)*CC(I,K,2)
            CI2 = WA1(I-2)*CC(I,K,2)-WA1(I-1)*CC(I-1,K,2)
```

```
            CR3 = WA2(I-2)*CC(I-1,K,3)+WA2(I-1)*CC(I,K,3)
            CI3 = WA2(I-2)*CC(I,K,3)-WA2(I-1)*CC(I-1,K,3)
            CR4 = WA3(I-2)*CC(I-1,K,4)+WA3(I-1)*CC(I,K,4)
            CI4 = WA3(I-2)*CC(I,K,4)-WA3(I-1)*CC(I-1,K,4)
            TR1 = CR2+CR4
            TR4 = CR4-CR2
            TI1 = CI2+CI4
            TI4 = CI2-CI4
            TI2 = CC(I,K,1)+CI3
            TI3 = CC(I,K,1)-CI3
            TR2 = CC(I-1,K,1)+CR3
            TR3 = CC(I-1,K,1)-CR3
            CH(I-1,1,K) = TR1+TR2
            CH(IC-1,4,K) = TR2-TR1
            CH(I,1,K) = TI1+TI2
            CH(IC,4,K) = TI1-TI2
            CH(I-1,3,K) = TI4+TR3
            CH(IC-1,2,K) = TR3-TI4
            CH(I,3,K) = TR4+TI3
            CH(IC,2,K) = TR4-TI3
103         CONTINUE
104 CONTINUE
      IF (MOD(IDO,2) .EQ. 1) RETURN
105 CONTINUE
      DO 106 K=1,L1
         TI1 = -HSQT2*(CC(IDO,K,2)+CC(IDO,K,4))
         TR1 = HSQT2*(CC(IDO,K,2)-CC(IDO,K,4))
         CH(IDO,1,K) = TR1+CC(IDO,K,1)
         CH(IDO,3,K) = CC(IDO,K,1)-TR1
         CH(1,2,K) = TI1-CC(IDO,K,3)
         CH(1,4,K) = TI1+CC(IDO,K,3)
106 CONTINUE
107 RETURN
      END

      SUBROUTINE RADF5 (IDO,L1,CC,CH,WA1,WA2,WA3,WA4)
```

```
      DIMENSION       CC(IDO,L1,5)          ,CH(IDO,5,L1)          , &
     &                WA1(1)      ,WA2(1)      ,WA3(1)      ,WA4(1)
      DATA TR11,TI11,TR12,TI12 /.309016994374947,.951056516295154, &
     &-.809016994374947,.587785252292473/
      DO 101 K=1,L1
         CR2 = CC(1,K,5)+CC(1,K,2)
         CI5 = CC(1,K,5)-CC(1,K,2)
         CR3 = CC(1,K,4)+CC(1,K,3)
         CI4 = CC(1,K,4)-CC(1,K,3)
         CH(1,1,K) = CC(1,K,1)+CR2+CR3
         CH(IDO,2,K) = CC(1,K,1)+TR11*CR2+TR12*CR3
         CH(1,3,K) = TI11*CI5+TI12*CI4
         CH(IDO,4,K) = CC(1,K,1)+TR12*CR2+TR11*CR3
         CH(1,5,K) = TI12*CI5-TI11*CI4
  101 CONTINUE
      IF (IDO .EQ. 1) RETURN
      IDP2 = IDO+2
      DO 103 K=1,L1
         DO 102 I=3,IDO,2
            IC = IDP2-I
            DR2 = WA1(I-2)*CC(I-1,K,2)+WA1(I-1)*CC(I,K,2)
            DI2 = WA1(I-2)*CC(I,K,2)-WA1(I-1)*CC(I-1,K,2)
            DR3 = WA2(I-2)*CC(I-1,K,3)+WA2(I-1)*CC(I,K,3)
            DI3 = WA2(I-2)*CC(I,K,3)-WA2(I-1)*CC(I-1,K,3)
            DR4 = WA3(I-2)*CC(I-1,K,4)+WA3(I-1)*CC(I,K,4)
            DI4 = WA3(I-2)*CC(I,K,4)-WA3(I-1)*CC(I-1,K,4)
            DR5 = WA4(I-2)*CC(I-1,K,5)+WA4(I-1)*CC(I,K,5)
            DI5 = WA4(I-2)*CC(I,K,5)-WA4(I-1)*CC(I-1,K,5)
            CR2 = DR2+DR5
            CI5 = DR5-DR2
            CR5 = DI2-DI5
            CI2 = DI2+DI5
            CR3 = DR3+DR4
            CI4 = DR4-DR3
            CR4 = DI3-DI4
            CI3 = DI3+DI4
```

```
                CH(I-1,1,K) = CC(I-1,K,1)+CR2+CR3
                CH(I,1,K) = CC(I,K,1)+CI2+CI3
                TR2 = CC(I-1,K,1)+TR11*CR2+TR12*CR3
                TI2 = CC(I,K,1)+TR11*CI2+TR12*CI3
                TR3 = CC(I-1,K,1)+TR12*CR2+TR11*CR3
                TI3 = CC(I,K,1)+TR12*CI2+TR11*CI3
                TR5 = TI11*CR5+TI12*CR4
                TI5 = TI11*CI5+TI12*CI4
                TR4 = TI12*CR5-TI11*CR4
                TI4 = TI12*CI5-TI11*CI4
                CH(I-1,3,K) = TR2+TR5
                CH(IC-1,2,K) = TR2-TR5
                CH(I,3,K) = TI2+TI5
                CH(IC,2,K) = TI5-TI2
                CH(I-1,5,K) = TR3+TR4
                CH(IC-1,4,K) = TR3-TR4
                CH(I,5,K) = TI3+TI4
                CH(IC,4,K) = TI4-TI3
102         CONTINUE
103 CONTINUE
        RETURN
        END

        SUBROUTINE RADFG (IDO,IP,L1,IDL1,CC,C1,C2,CH,CH2,WA)
        DIMENSION       CH(IDO,L1,IP)       ,CC(IDO,IP,L1)       , &
       &                C1(IDO,L1,IP)       ,C2(IDL1,IP), &
       &                CH2(IDL1,IP)        ,WA(1)
        DATA TPI/6.28318530717959/
        ARG = TPI/FLOAT(IP)
        DCP = COS(ARG)
        DSP = SIN(ARG)
        IPPH = (IP+1)/2
        IPP2 = IP+2
        IDP2 = IDO+2
        NBD = (IDO-1)/2
        IF (IDO .EQ. 1) GO TO 119
```

```
      DO 101 IK=1,IDL1
         CH2(IK,1) = C2(IK,1)
101 CONTINUE
      DO 103 J=2,IP
         DO 102 K=1,L1
            CH(1,K,J) = C1(1,K,J)
102      CONTINUE
103 CONTINUE
      IF (NBD .GT. L1) GO TO 107
      IS = -IDO
      DO 106 J=2,IP
         IS = IS+IDO
         IDIJ = IS
         DO 105 I=3,IDO,2
            IDIJ = IDIJ+2
            DO 104 K=1,L1
               CH(I-1,K,J) = WA(IDIJ-1)*C1(I-1,K,J)+WA(IDIJ)*C1(I,K,J)
               CH(I,K,J) = WA(IDIJ-1)*C1(I,K,J)-WA(IDIJ)*C1(I-1,K,J)
104         CONTINUE
105      CONTINUE
106 CONTINUE
      GO TO 111
107 IS = -IDO
      DO 110 J=2,IP
         IS = IS+IDO
         DO 109 K=1,L1
            IDIJ = IS
            DO 108 I=3,IDO,2
               IDIJ = IDIJ+2
               CH(I-1,K,J) = WA(IDIJ-1)*C1(I-1,K,J)+WA(IDIJ)*C1(I,K,J)
               CH(I,K,J) = WA(IDIJ-1)*C1(I,K,J)-WA(IDIJ)*C1(I-1,K,J)
108         CONTINUE
109      CONTINUE
110 CONTINUE
111 IF (NBD .LT. L1) GO TO 115
      DO 114 J=2,IPPH
```

```
          JC = IPP2-J
          DO 113 K=1,L1
              DO 112 I=3,IDO,2
                  C1(I-1,K,J) = CH(I-1,K,J)+CH(I-1,K,JC)
                  C1(I-1,K,JC) = CH(I,K,J)-CH(I,K,JC)
                  C1(I,K,J) = CH(I,K,J)+CH(I,K,JC)
                  C1(I,K,JC) = CH(I-1,K,JC)-CH(I-1,K,J)
  112         CONTINUE
  113     CONTINUE
  114 CONTINUE
      GO TO 121
  115 DO 118 J=2,IPPH
          JC = IPP2-J
          DO 117 I=3,IDO,2
              DO 116 K=1,L1
                  C1(I-1,K,J) = CH(I-1,K,J)+CH(I-1,K,JC)
                  C1(I-1,K,JC) = CH(I,K,J)-CH(I,K,JC)
                  C1(I,K,J) = CH(I,K,J)+CH(I,K,JC)
                  C1(I,K,JC) = CH(I-1,K,JC)-CH(I-1,K,J)
  116         CONTINUE
  117     CONTINUE
  118 CONTINUE
      GO TO 121
  119 DO 120 IK=1,IDL1
          C2(IK,1) = CH2(IK,1)
  120 CONTINUE
  121 DO 123 J=2,IPPH
          JC = IPP2-J
          DO 122 K=1,L1
              C1(1,K,J) = CH(1,K,J)+CH(1,K,JC)
              C1(1,K,JC) = CH(1,K,JC)-CH(1,K,J)
  122     CONTINUE
  123 CONTINUE
!
      AR1 = 1.
      AI1 = 0.
```

```
      DO 127 L=2,IPPH
         LC = IPP2-L
         AR1H = DCP*AR1-DSP*AI1
         AI1 = DCP*AI1+DSP*AR1
         AR1 = AR1H
         DO 124 IK=1,IDL1
            CH2(IK,L) = C2(IK,1)+AR1*C2(IK,2)
            CH2(IK,LC) = AI1*C2(IK,IP)
  124    CONTINUE
         DC2 = AR1
         DS2 = AI1
         AR2 = AR1
         AI2 = AI1
         DO 126 J=3,IPPH
            JC = IPP2-J
            AR2H = DC2*AR2-DS2*AI2
            AI2 = DC2*AI2+DS2*AR2
            AR2 = AR2H
            DO 125 IK=1,IDL1
               CH2(IK,L) = CH2(IK,L)+AR2*C2(IK,J)
               CH2(IK,LC) = CH2(IK,LC)+AI2*C2(IK,JC)
  125       CONTINUE
  126    CONTINUE
  127 CONTINUE
      DO 129 J=2,IPPH
         DO 128 IK=1,IDL1
            CH2(IK,1) = CH2(IK,1)+C2(IK,J)
  128    CONTINUE
  129 CONTINUE
!
      IF (IDO .LT. L1) GO TO 132
      DO 131 K=1,L1
         DO 130 I=1,IDO
            CC(I,1,K) = CH(I,K,1)
  130    CONTINUE
  131 CONTINUE
```

```
      GO TO 135
  132 DO 134 I=1,IDO
          DO 133 K=1,L1
              CC(I,1,K) = CH(I,K,1)
  133     CONTINUE
  134 CONTINUE
  135 DO 137 J=2,IPPH
          JC = IPP2-J
          J2 = J+J
          DO 136 K=1,L1
              CC(IDO,J2-2,K) = CH(1,K,J)
              CC(1,J2-1,K) = CH(1,K,JC)
  136     CONTINUE
  137 CONTINUE
      IF (IDO .EQ. 1) RETURN
      IF (NBD .LT. L1) GO TO 141
      DO 140 J=2,IPPH
          JC = IPP2-J
          J2 = J+J
          DO 139 K=1,L1
              DO 138 I=3,IDO,2
                  IC = IDP2-I
                  CC(I-1,J2-1,K) = CH(I-1,K,J)+CH(I-1,K,JC)
                  CC(IC-1,J2-2,K) = CH(I-1,K,J)-CH(I-1,K,JC)
                  CC(I,J2-1,K) = CH(I,K,J)+CH(I,K,JC)
                  CC(IC,J2-2,K) = CH(I,K,JC)-CH(I,K,J)
  138         CONTINUE
  139     CONTINUE
  140 CONTINUE
      RETURN
  141 DO 144 J=2,IPPH
          JC = IPP2-J
          J2 = J+J
          DO 143 I=3,IDO,2
              IC = IDP2-I
              DO 142 K=1,L1
```

```
                CC(I-1,J2-1,K) = CH(I-1,K,J)+CH(I-1,K,JC)
                CC(IC-1,J2-2,K) = CH(I-1,K,J)-CH(I-1,K,JC)
                CC(I,J2-1,K) = CH(I,K,J)+CH(I,K,JC)
                CC(IC,J2-2,K) = CH(I,K,JC)-CH(I,K,J)
142           CONTINUE
143       CONTINUE
144 CONTINUE
      RETURN
      END

      SUBROUTINE RFFTB1 (N,C,CH,WA,IFAC)
      DIMENSION       CH(1)      ,C(1)       ,WA(1)      ,IFAC(1)
      NF = IFAC(2)
      NA = 0
      L1 = 1
      IW = 1
      DO 116 K1=1,NF
         IP = IFAC(K1+2)
         L2 = IP*L1
         IDO = N/L2
         IDL1 = IDO*L1
         IF (IP .NE. 4) GO TO 103
         IX2 = IW+IDO
         IX3 = IX2+IDO
         IF (NA .NE. 0) GO TO 101
         CALL RADB4 (IDO,L1,C,CH,WA(IW),WA(IX2),WA(IX3))
         GO TO 102
101      CALL RADB4 (IDO,L1,CH,C,WA(IW),WA(IX2),WA(IX3))
102      NA = 1-NA
         GO TO 115
103      IF (IP .NE. 2) GO TO 106
         IF (NA .NE. 0) GO TO 104
         CALL RADB2 (IDO,L1,C,CH,WA(IW))
         GO TO 105
104      CALL RADB2 (IDO,L1,CH,C,WA(IW))
105      NA = 1-NA
```

```
      GO TO 115
106   IF (IP .NE. 3) GO TO 109
      IX2 = IW+IDO
      IF (NA .NE. 0) GO TO 107
      CALL RADB3 (IDO,L1,C,CH,WA(IW),WA(IX2))
      GO TO 108
107   CALL RADB3 (IDO,L1,CH,C,WA(IW),WA(IX2))
108   NA = 1-NA
      GO TO 115
109   IF (IP .NE. 5) GO TO 112
      IX2 = IW+IDO
      IX3 = IX2+IDO
      IX4 = IX3+IDO
      IF (NA .NE. 0) GO TO 110
      CALL RADB5 (IDO,L1,C,CH,WA(IW),WA(IX2),WA(IX3),WA(IX4))
      GO TO 111
110   CALL RADB5 (IDO,L1,CH,C,WA(IW),WA(IX2),WA(IX3),WA(IX4))
111   NA = 1-NA
      GO TO 115
112   IF (NA .NE. 0) GO TO 113
      CALL RADBG (IDO,IP,L1,IDL1,C,C,C,CH,CH,WA(IW))
      GO TO 114
113   CALL RADBG (IDO,IP,L1,IDL1,CH,CH,CH,C,C,WA(IW))
114   IF (IDO .EQ. 1) NA = 1-NA
115   L1 = L2
      IW = IW+(IP-1)*IDO
116 CONTINUE
    IF (NA .EQ. 0) RETURN
    DO 117 I=1,N
      C(I) = CH(I)
117 CONTINUE
    RETURN
    END

 SUBROUTINE RFFTB (N,R,WSAVE)
    DIMENSION       R(1)      ,WSAVE(1)
```

```
      IF (N .EQ. 1) RETURN
      CALL RFFTB1 (N,R,WSAVE,WSAVE(N+1),WSAVE(2*N+1))
      RETURN
      END

      SUBROUTINE RFFTF1 (N,C,CH,WA,IFAC)
      DIMENSION       CH(1)      ,C(1)      ,WA(1)      ,IFAC(1)
      NF = IFAC(2)
      NA = 1
      L2 = N
      IW = N
      DO 111 K1=1,NF
         KH = NF-K1
         IP = IFAC(KH+3)
         L1 = L2/IP
         IDO = N/L2
         IDL1 = IDO*L1
         IW = IW-(IP-1)*IDO
         NA = 1-NA
         IF (IP .NE. 4) GO TO 102
         IX2 = IW+IDO
         IX3 = IX2+IDO
         IF (NA .NE. 0) GO TO 101
         CALL RADF4 (IDO,L1,C,CH,WA(IW),WA(IX2),WA(IX3))
         GO TO 110
  101    CALL RADF4 (IDO,L1,CH,C,WA(IW),WA(IX2),WA(IX3))
         GO TO 110
  102    IF (IP .NE. 2) GO TO 104
         IF (NA .NE. 0) GO TO 103
         CALL RADF2 (IDO,L1,C,CH,WA(IW))
         GO TO 110
  103    CALL RADF2 (IDO,L1,CH,C,WA(IW))
         GO TO 110
  104    IF (IP .NE. 3) GO TO 106
         IX2 = IW+IDO
         IF (NA .NE. 0) GO TO 105
```

```
      CALL RADF3 (IDO,L1,C,CH,WA(IW),WA(IX2))
      GO TO 110
105   CALL RADF3 (IDO,L1,CH,C,WA(IW),WA(IX2))
      GO TO 110
106   IF (IP .NE. 5) GO TO 108
      IX2 = IW+IDO
      IX3 = IX2+IDO
      IX4 = IX3+IDO
      IF (NA .NE. 0) GO TO 107
      CALL RADF5 (IDO,L1,C,CH,WA(IW),WA(IX2),WA(IX3),WA(IX4))
      GO TO 110
107   CALL RADF5 (IDO,L1,CH,C,WA(IW),WA(IX2),WA(IX3),WA(IX4))
      GO TO 110
108   IF (IDO .EQ. 1) NA = 1-NA
      IF (NA .NE. 0) GO TO 109
      CALL RADFG (IDO,IP,L1,IDL1,C,C,C,CH,CH,WA(IW))
      NA = 1
      GO TO 110
109   CALL RADFG (IDO,IP,L1,IDL1,CH,CH,CH,C,C,WA(IW))
      NA = 0
110   L2 = L1
111 CONTINUE
    IF (NA .EQ. 1) RETURN
    DO 112 I=1,N
      C(I) = CH(I)
112 CONTINUE
    RETURN
    END

    SUBROUTINE RFFTF (N,R,WSAVE)
    DIMENSION       R(1)       ,WSAVE(1)
    IF (N .EQ. 1) RETURN
    CALL RFFTF1 (N,R,WSAVE,WSAVE(N+1),WSAVE(2*N+1))
    RETURN
    END
```

```
      SUBROUTINE RFFTI1 (N,WA,IFAC)
      DIMENSION      WA(1)      ,IFAC(1)     ,NTRYH(4)
      DATA NTRYH(1),NTRYH(2),NTRYH(3),NTRYH(4)/4,2,3,5/
      NL = N
      NF = 0
      J = 0
  101 J = J+1
      IF (J-4) 102,102,103
  102 NTRY = NTRYH(J)
      GO TO 104
  103 NTRY = NTRY+2
  104 NQ = NL/NTRY
      NR = NL-NTRY*NQ
      IF (NR) 101,105,101
  105 NF = NF+1
      IFAC(NF+2) = NTRY
      NL = NQ
      IF (NTRY .NE. 2) GO TO 107
      IF (NF .EQ. 1) GO TO 107
      DO 106 I=2,NF
         IB = NF-I+2
         IFAC(IB+2) = IFAC(IB+1)
  106 CONTINUE
      IFAC(3) = 2
  107 IF (NL .NE. 1) GO TO 104
      IFAC(1) = N
      IFAC(2) = NF
      TPI = 6.28318530717959
      ARGH = TPI/FLOAT(N)
      IS = 0
      NFM1 = NF-1
      L1 = 1
      IF (NFM1 .EQ. 0) RETURN
      DO 110 K1=1,NFM1
         IP = IFAC(K1+2)
         LD = 0
```

```
      L2 = L1*IP
      IDO = N/L2
      IPM = IP-1
      DO 109 J=1,IPM
        LD = LD+L1
        I = IS
        ARGLD = FLOAT(LD)*ARGH
        FI = 0.
        DO 108 II=3,IDO,2
          I = I+2
          FI = FI+1.
          ARG = FI*ARGLD
          WA(I-1) = COS(ARG)
          WA(I) = SIN(ARG)
  108   CONTINUE
        IS = IS+IDO
  109 CONTINUE
      L1 = L2
  110 CONTINUE
      RETURN
      END

      SUBROUTINE RFFTI (N,WSAVE)
      DIMENSION       WSAVE(1)
      IF (N .EQ. 1) RETURN
      CALL RFFTI1 (N,WSAVE(N+1),WSAVE(2*N+1))
      RETURN
      END

      SUBROUTINE SINQB (N,X,WSAVE)
      DIMENSION       X(1)        ,WSAVE(1)
      IF (N .GT. 1) GO TO 101
      X(1) = 4.*X(1)
      RETURN
  101 NS2 = N/2
      DO 102 K=2,N,2
```

```
      X(K) = -X(K)
102 CONTINUE
      CALL COSQB (N,X,WSAVE)
      DO 103 K=1,NS2
         KC = N-K
         XHOLD = X(K)
         X(K) = X(KC+1)
         X(KC+1) = XHOLD
103 CONTINUE
      RETURN
      END

      SUBROUTINE SINQF (N,X,WSAVE)
      DIMENSION       X(1)       ,WSAVE(1)
      IF (N .EQ. 1) RETURN
      NS2 = N/2
      DO 101 K=1,NS2
         KC = N-K
         XHOLD = X(K)
         X(K) = X(KC+1)
         X(KC+1) = XHOLD
101 CONTINUE
      CALL COSQF (N,X,WSAVE)
      DO 102 K=2,N,2
         X(K) = -X(K)
102 CONTINUE
      RETURN
      END

      SUBROUTINE SINQI (N,WSAVE)
      DIMENSION       WSAVE(1)
      CALL COSQI (N,WSAVE)
      RETURN
      END

      SUBROUTINE SINT1(N,WAR,WAS,XH,X,IFAC)
```

```
      DIMENSION WAR(1),WAS(1),X(1),XH(1),IFAC(1)
      DATA SQRT3 /1.73205080756888/
      DO 100 I=1,N
      XH(I) = WAR(I)
      WAR(I) = X(I)
100 CONTINUE
      IF (N-2) 101,102,103
101 XH(1) = XH(1)+XH(1)
      GO TO 106
102 XHOLD = SQRT3*(XH(1)+XH(2))
      XH(2) = SQRT3*(XH(1)-XH(2))
      XH(1) = XHOLD
      GO TO 106
103 NP1 = N+1
      NS2 = N/2
      X(1) = 0.
      DO 104 K=1,NS2
         KC = NP1-K
         T1 = XH(K)-XH(KC)
         T2 = WAS(K)*(XH(K)+XH(KC))
         X(K+1) = T1+T2
         X(KC+1) = T2-T1
104 CONTINUE
      MODN = MOD(N,2)
      IF (MODN .NE. 0) X(NS2+2) = 4.*XH(NS2+1)
      CALL RFFTF1 (NP1,X,XH,WAR,IFAC)
      XH(1) = .5*X(1)
      DO 105 I=3,N,2
         XH(I-1) = -X(I)
         XH(I) = XH(I-2)+X(I-1)
105 CONTINUE
      IF (MODN .NE. 0) GO TO 106
      XH(N) = -X(N+1)
106 DO 107 I=1,N
      X(I) = WAR(I)
      WAR(I) = XH(I)
```

```
107 CONTINUE
      RETURN
      END

      SUBROUTINE SINT (N,X,WSAVE)
      DIMENSION       X(1)        ,WSAVE(1)
      NP1 = N+1
      IW1 = N/2+1
      IW2 = IW1+NP1
      IW3 = IW2+NP1
      CALL SINT1(N,X,WSAVE,WSAVE(IW1),WSAVE(IW2),WSAVE(IW3))
      RETURN
      END

      SUBROUTINE SINTI (N,WSAVE)
      DIMENSION       WSAVE(1)
      DATA PI /3.14159265358979/
      IF (N .LE. 1) RETURN
      NS2 = N/2
      NP1 = N+1
      DT = PI/FLOAT(NP1)
      DO 101 K=1,NS2
         WSAVE(K) = 2.*SIN(K*DT)
101 CONTINUE
      CALL RFFTI (NP1,WSAVE(NS2+1))
      RETURN
      END

      SUBROUTINE TRID (MR,A,B,C,Y,D)
      DIMENSION       A(1)        ,B(1)        ,C(1)        ,Y(1)        , &
                     &       D(1)
      M = MR
      MM1 = M-1
      Z = 1./B(1)
      D(1) = C(1)*Z
      Y(1) = Y(1)*Z
```

```
      DO 101 I=2,MM1
         Z = 1./(B(I)-A(I)*D(I-1))
         D(I) = C(I)*Z
         Y(I) = (Y(I)-A(I)*Y(I-1))*Z
101 CONTINUE
      Z = B(M)-A(M)*D(MM1)
      IF (Z .NE. 0.) GO TO 102
      Y(M) = 0.
      GO TO 103
102 Y(M) = (Y(M)-A(M)*Y(MM1))/Z
103 CONTINUE
      DO 104 IP=1,MM1
         I = M-IP
         Y(I) = Y(I)-D(I)*Y(I+1)
104 CONTINUE
      RETURN
      END
```

References

[1] F. Mandl, et,al. Quantum Mechanics. University of California press, 1992.

[2] Linys Pauling. THE CHEMICAL BOND, A Brief Introduction to Modern Structural Chemistry, 3. GEORGE BANTA, INC, 1967.

[3] G.Aruldhas. Quantum Mechanics. Prentice'Hall of India Private Limited New Delhi, 2002.

[4] Stephen Gasiorowicz. Quantum Physics. John Wiley & Sons, Inc, 1996

[5] Michael P.Marder. Condensed Matter Physics. A Wiley-Interscience Publication JOHN WILEY & SONS, INC, 2000.

[6] Carl Trindle Donald Shillady. ELECTRONIC STRUCTURE MODELING. CRC Press Taylor & Francis Group Boca Raton London New York, 2008.

[7] Robert A. Evarestov. Quantum Chemistry of Solids. The LCAO First Principles Treatment of Crystals.Springer Berlin Heidelberg, 2007.

[8] Errol Lewars. Computational Chemistry. Introduction to the Theory and Appliccations of Molecular and Quantum Mechanics. Kluwer Academic Publishers, 2003.