國立交通大學

應用數學系

碩 士 論 文

原子基態能量的計算

Computation of the ground state energy of atoms

研 究 生：陳廷彰

指導教授：葉立明　教授

中 華 民 國 九 十 八 年 七 月

原子基態能量的計算
Computation of the ground state energy of atoms

研 究 生：陳廷彰　　Student：Ting-Jang 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 Computer and 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

原子基態能的量計算

學生：陳廷彰　　　　　　　　　　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

基態能量在量子力學裡是代表最低能量。我們利用從 Kohn-Sham 方程式以及 local density approximation 中化簡出來之電子密度 ρ 的基態能量泛函 $E_G[\rho]$ 來計算原子的基態能量。如果要計算 $E_G[\rho]$，我們必須先算出原子基態的波函數以及電子密度，而電子密度是跟波函數有關的函數。所以第一步，要利用解 Kohn-Sham 方程式 $H\Psi = \varepsilon\Psi$ 找出波函數。Kohn-Sham 方程式是一個二階偏微分方程，在 Kohn-Sham 方程式中，ε 的最小值所對應的函數 Ψ，就是原子基態能量的波函數。為了在計算上的方便，我們將 H 離散化，讓問題變成解特徵值的計算之後，利用自洽來解出波函數。此篇計算所得到的數據和實際值的數據誤差不超過 15%。

中 華 民 國 九 十 八 年 七 月

Computation of the ground state energy of atoms

Student: Ting-Jang Chen **Advisor:** Dr. Li-Ming Yeh

Department of Applied Mathematics

National Chiao Tung University

Hsinchu, Taiwan, R.O.C.

Abstract

The ground state is the lowest-energy state of the quantum mechanical system. We compute the ground state energy by using the ground state energy functional $E_G[\rho]$ of electronic density ρ, which is deduced from the Kohn-Sham total-energy functional and local density approximation. To compute the ground state energy of atoms, we have to compute each electronic density of the ground state of atoms. The electronic density of atoms is a function about the wave function of atoms, so the first step is to determine the wave functions of ground state of atoms by solving the Kohn-Sham equation $H\Psi = \varepsilon\Psi$. The Kohn-Sham equation is a problem of the second order partial differential equation. The wave function of the ground state of atoms is the function Ψ corresponding to the minimal ε of the Kohn-Sham equation. For the convenience of solving the Kohn-Sham equation, we discretize the Hamiltonian H of the Kohn-Sham equation and the problem become an eigenvalue problem. In this computation, we determine the wave function of the ground state of atoms by self-consistency. The errors between the computation and the realistic values are less than 15%.

誌　　謝

能完成這篇論文，首先要感謝我的指導教授葉立明教授。有耐心的引導我學習，糾正我錯誤的學習態度以及習慣，並常常為了我的論文和我討論到深夜，真的是由衷的感謝您。

再來要感謝和我一起畢業的同學以及學弟妹，無條件的幫我檢查論文，討論論文的內容，讓我的論文更加完整。

還要感謝我的女友劉翊婷，在我最困難最煩惱的時候總是陪著我幫我解決問題，半夜陪著我準備口試，不停的幫我加油打氣。

也感謝我的口試委員葉立明教授、王夏聲教授、黃聰明教授、李華輪教授。

最後要感謝我的父母，時時掛念我的健康，感謝以上各位的幫助以及支持，使我度過充實的研究生活。

Contents

1 Introduction

1.1 Density functional theory

1.1.1 Electronic density

The starting point of this computation is the observation of Hohenbergy and Kohn (1964) [1] that electronic density ρ contains in principle all the information contained in a many-electron wave function. The electronic density ρ of a many-electron system at point $\vec{r}$ is defined to be (In this paper, we use the atomic unit with $\hbar = m_e = e = 1$)

$$\rho(r) = \sum_{i \in occupied} f_i |\Psi_i|^2,$$

where f_i are the occupation numbers of the orbitals denoted by i, and Ψ_i is the wave function. It is convenient to employ the spherical coordinates r, θ, ϕ, the wave function can be written as :

$$\Psi_i = R_{nl(i)}(r) Y_{lm(i)}(\theta, \phi). \tag{1.1}$$

The electronic orbitals are represented by the index $i = \{n, l, m\}$. n, l, m are the main quantum number, the angular momentum quantum number and the magnetic quantum number respectively.

1.1.2 Density functional theory and the ground state energy functional

Density functional theory is a theory of quantum mechanics. With this theory, the properties of a many-electron system can be determined by using the functionals which is dependent on the electronic density. Hence the name density functional theory comes from the use of functionals of the electronic density. Hohenberg and Kohn observed [1] that the ground-state energy is a functional of the electronic density ρ and can written as

$$E_G = E_G[\rho] = T_0[\rho] + E_{ee}[\rho] + E_{ex}[\rho] + E_{xc}[\rho],$$

where $T_0[\rho]$ is the kinetic energy functional, $E_{ee}[\rho]$ is the functional the classical Coulomb repulsion energy, $E_{ex}[\rho]$ is the energy of the electrons in the external field of the nuclei, and $E_{xc}[\rho]$ is the exchange-correlation energy.

1.1.3 Local density approximation

The principle of local density approximation is to compute the exchange-correlation energies per particle. Hohenberg and Kohn provided some motivation for using approximate methods to describe the exchange-correlation energies as a function of a electronic density. The simplest method of describing exchange-correlation energy is local density approximation. In local density approximation, the exchange-correlation energy of an electronic system is written as [7, 10]

$$E_{xc} = \int_{R^3}\int \varepsilon_{xc} \cdot \rho(r) dr d\omega,$$

with$\varepsilon_{xc} = \varepsilon_x + \varepsilon_c$.

Notation:

Here we define the integral

$$\int_{R^3}\int g(r) dr d\omega$$

for a function $g(r)$ to be

$$\int_{R^3}\int g(r) dr d\omega = \int_0^{\pi}\int_0^{2\pi}\int_0^{\infty} g(r) r^2 sin\phi dr d\theta d\phi = 4\pi \int_0^{\infty} g(r) r^2 dr. \quad (1.2)$$

1.2 The ground state energy functional

We use the ground state energy functional to compute the ground state energy of atoms. In local density approximation [10] the ground state energy of a many-electron system are expressed as a functional $E_G[\rho]$ of the electronic density $\rho(r)$. The ground state energy functional can be written as

$$E_G[\rho] = F[\rho(r)] + E_{ex}[\rho].$$

And

$$E_{ex}[\rho] = \int_{R^3}\int v_{ex}\cdot\rho(r)drd\omega = \int_{R^3}\int -\frac{z}{r}\rho(r)drd\omega = -4\pi\int_0^\infty z\cdot r\cdot\rho(r)dr, \tag{1.3}$$

where z is the atomic number and $v_{ex}(r)$ is the external potential

$$v_{ex}(r) = -\frac{z}{r}.$$

The functional $F[\rho(r)]$ is given by [10]

$$\begin{aligned} F[\rho(r)] &= E_{ee}[\rho] + T_0[\rho] + E_{xc}[\rho] \\ &= \frac{1}{2}\int_{R^3}\int V_{ee}(r)\rho(r)drd\omega + T_0[\rho] + E_{xc}[\rho] \\ &= \frac{1}{2}\int_{R^3}\int\int_{R^3}\int \frac{\rho(r)\rho(r')}{|r-r'|}dr'd\omega drd\omega + T_0[\rho] + E_{xc}[\rho]. \end{aligned}$$

And we obtain that

$$E_G[\rho] = \int_{R^3}\int V_{ee}(r)\rho(r)drd\omega + T_0[\rho] + E_{xc}[\rho] + \int_{R^3}\int v_{ex}(r)\rho(r)drd\omega. \tag{1.4}$$

The first term of the right hand side of (1.4) is the Coulomb energy of the electron charge densities, with $V_{ee}(r)$ being the electron-electron Coulomb potential [1]:

$$V_{ee}(r) = \int_{R^3}\int \rho(r')\frac{1}{r>}dr'd\omega = 4\pi\int_0^\infty \rho(r')\frac{r'^2}{r>}dr'$$

(here we define the variable "$r>$" be $max(r, r')$). So the first term of the right hand side of (1.4) is

$$\frac{1}{2}\int_{R^3}\int\left(\int_{R^3}\int\frac{\rho(r)\rho(r')}{|r-r'|}dr'd\omega\right)drd\omega = \frac{1}{2}\cdot(4\pi)^2\int_0^\infty\int_0^\infty\frac{r^2r'^2}{r>}\rho(r)\rho(r')dr'dr. \tag{1.5}$$

$E_{xc}[\rho]$ is the exchange-correlation energy. In the local density approximation, $E_{xc}[\rho]$ is written as

$$E_{xc}[\rho] = \int_{\Omega}\int \varepsilon_{xc}[\rho(r)]\rho(r)drd\omega = 4\pi\int_0^{\infty} \varepsilon_{xc}[\rho(r)]\cdot\rho(r)\cdot r^2dr. \qquad (1.6)$$

Now, we discuss the exchange-correlation energy ε_{xc} in (1.6). The exchange-correlation energy may be decomposed into a sum of the exchange energy and the correlation energy [10].

$$\varepsilon_{xc} = \varepsilon_x + \varepsilon_c.$$

ε_{xc} is expressed in terms of the radius r_s of a unit charge defined by

$$r_s = (\frac{3}{4\pi\rho})^{\frac{1}{3}}.$$

The exchange term can be calculated exactly for a uniform electron gas, and is given by

$$\varepsilon_x[\rho] = -\frac{3}{4\pi r_s}(\frac{9}{4})^{\frac{1}{3}} = -\frac{0.458}{r_s}.$$

The correlation energy is given by

$$\varepsilon_c = \begin{cases} -\dfrac{\gamma}{1+\beta_1\sqrt{r_s}+\beta_2 r_s} & ;for\ r_s \geq 1, low\ density \\ A\ln r_s + B + Cr_s\ln r_s + Dr_s & ;for\ r_s < 1,\ high\ density \end{cases},$$

where the values of the constants are given by

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

and

$$A = 0.0311;\ B = -0.0480;\ C = 0.0020;\ D = -0.0116.$$

The term $T_0[\rho]$ is the kinetic energy functional. The kinetic energy functional is expressed in terms of a system of noninteracting electrons. (Here the wave function $\Psi(r,\theta,\phi) = R_{nl}(r)\cdot Y_{lm}(\theta,\phi)$.)

$$T_0[\rho] = \sum_{i\in occupied} f_i \int_{R^3}\int \Psi_i^*(\frac{-\nabla^2}{2})\Psi_i drd\omega. \qquad (1.7)$$

The natation that ∇^2 in spherical coordinate can be written as [4]

$$\nabla^2 = \frac{1}{r^2}\frac{\partial}{\partial r}(r^2\frac{\partial}{\partial r}) + \frac{1}{r^2 sin^2\theta}\frac{\partial^2}{\partial\phi^2} + \frac{1}{r^2 sin\theta}\frac{\partial}{\partial\theta}(sin\theta\frac{\partial}{\partial\theta}).$$

And we obtain that

$$\begin{aligned}
-\frac{1}{2}\nabla^2\Psi_i &= -\frac{1}{2}\left\{\frac{1}{r^2}\frac{\partial}{\partial r}(r^2\frac{\partial}{\partial r}\Psi_i) + \frac{1}{r^2}\left[\frac{1}{sin\theta}\frac{\partial}{\partial\theta}(sin\theta\frac{\partial\Psi_i}{\partial\theta}) + \frac{1}{sin^2\theta}\frac{\partial^2\Psi_i}{\partial\phi^2}\right]\right\} \\
&= -\frac{1}{2}\left\{\frac{1}{r^2}\frac{\partial}{\partial r}(r^2\frac{\partial}{\partial r}R_{nl(i)})Y_{lm(i)} + \frac{1}{r^2}R_{nl(i)}\left[\frac{1}{sin\theta}\frac{\partial}{\partial\theta}(sin\theta\frac{\partial Y_{lm(i)}}{\partial\theta})\right.\right. \\
&\quad \left.\left. + \frac{1}{sin^2\theta}\frac{\partial^2 Y_{lm(i)}}{\partial\phi^2}\right]\right\}.
\end{aligned} \tag{1.8}$$

Because of the identity [4]

$$\frac{1}{sin\theta}\frac{\partial}{\partial\theta}sin\theta\frac{\partial Y_{lm(i)}}{\partial\theta} + \frac{1}{sin^2\theta}\frac{\partial^2 Y_{lm(i)}}{\partial\phi^2} = -l(l+1)Y_{lm(i)},$$

we obtain the equation

$$\begin{aligned}
-\frac{1}{2}\nabla^2\Psi_i &= -\frac{1}{2}\left\{\frac{1}{r^2}\frac{\partial}{\partial r}(r^2\frac{\partial}{\partial r}R_{nl(i)})Y_{lm(i)} - \frac{1}{r^2}R_{nl(i)}\left[\frac{1}{r^2}\cdot l(l+1)\cdot Y_{lm(i)}\right]\right\} \\
&= \left\{-\frac{1}{2r^2}\frac{d}{dr}(r^2\frac{dR_{nl(i)}}{dr}) + \frac{l(l+1)}{2r^2}R_{nl(i)}\right\}Y_{lm(i)}.
\end{aligned} \tag{1.9}$$

By (1.9) we have

$$\begin{aligned}
\int_{R^3}\int\Psi_i^*(-\frac{1}{2}\nabla^2)\Psi_i dr d\omega &= \int_{R^3}\int Y_{lm(i)}R_{nl(i)}\left[-\frac{1}{2r^2}\frac{d}{dr}(r^2\frac{dR_{nl(i)}}{dr}) + \right. \\
&\quad \left.\frac{l(l+1)}{2r^2}R_{nl(i)}\right]Y_{lm(i)}dr d\omega \\
&= \int_{R^3}\int |Y_{lm(i)}|^2 R_{nl(i)}\left[-\frac{1}{2r^2}\frac{d}{dr}(r^2\frac{dR_{nl(i)}}{dr}) + \right. \\
&\quad \left.\frac{l(l+1)}{2r^2}R_{nl(i)}\right]dr d\omega.
\end{aligned} \tag{1.10}$$

By (1.7), (1.10), and the normalization of Y_{lm} (in section 2.6 we will discuss the normalization) [7]

$$\int_0^{\pi}\int_0^{2\pi}|Y_{lm(i)}|^2 sin\phi d\theta d\phi = 1,$$

we get the equation

$$\begin{aligned} T_0 &= \sum_{i\in occupied} f_i \left\{ \int_0^\infty r^2 R_{nl(i)} \left[-\frac{1}{2r^2}\frac{d}{dr}(r^2 \frac{dR_{nl(i)}}{dr}) + \frac{l(l+1)}{2r^2} R_{nl(i)} \right] dr \right\} \\ &= \sum_{i\in occupied} f_i \left\{ \int_0^\infty \frac{-1}{2} R_{nl(i)} \cdot r \cdot (2 \cdot \frac{dR_{nl(i)}}{dr} + \frac{d^2 R_{nl(i)}}{r^2}) dr \right. \\ &\quad \left. + \int_0^\infty R_{nl(i)} \frac{l(l+1)}{2r^2} R_{nl(i)} r^2 dr \right\}. \end{aligned} \tag{1.11}$$

By (1.2), (1.4), (1.5), (1.6), (1.11), we get the formula of ground state energy of atoms.

$$\begin{aligned} E_G[\rho] = & \frac{1}{2} \int_0^\infty \int_0^\infty \frac{r^2 r'^2}{r>} \rho(r)\rho(r') \cdot (4\pi)^2 dr dr' \\ & + \sum_{i\in occupied} f_i \left\{ \int_0^\infty \frac{-1}{2} R_{nl(i)} \cdot r \cdot (2 \cdot \frac{dR_{nl(i)}}{dr} + \frac{d^2 R_{nl(i)}}{r^2}) dr \right. \\ & \left. + \int_0^\infty R_{nl(i)} \frac{l(l+1)}{2r^2} R_{nl(i)} r^2 dr \right\} + 4\pi \int_0^\infty \varepsilon_{xc}[\rho(r)] \cdot \rho(r) \cdot r^2 dr \\ & -4\pi \int_0^\infty z \cdot r \cdot \rho(r) dr. \end{aligned} \tag{1.12}$$

1.3 Atomic System

In order to calculate the ground state energy functional. we must find each wave function of the atoms by solving the Kohn-Sham equation. The Kohn-Sham equation can be written as [1]:

$$\left(-\frac{1}{2}\nabla^2 + v_{eff}[\rho] \right) \Psi_i(r) = \varepsilon_i \Psi_i(r), \tag{1.13}$$

where ε_i is a Lagrange multiplier and can be interpreted as the orbital energy of the state represented by Ψ_i.

By (1.9) and (1.13), we have the equation

$$\left(-\frac{1}{2}\frac{1}{r^2}\frac{d}{dr}(r^2 \frac{dR_{nl}}{dr}) + \frac{l(l+1)}{2r^2} R_{nl} \right) Y_{lm} + v_{eff} R_{nl} Y_{lm} = e_{nl} R_{nl} Y_{lm}. \tag{1.14}$$

Multiplying both side of (1.14) by $\frac{1}{Y_{lm}}$, we obtain the equation

$$-\frac{1}{2}\frac{1}{r^2}\frac{d}{dr}(r^2 \frac{dR_{nl}}{dr}) + \frac{l(l+1)}{2r^2} R_{nl} + v_{eff} R_{nl} = e_{nl} R_{nl}. \tag{1.15}$$

The effective potential can be written as

$$v_{eff} = v_{ex}[\rho(r)] + V_{ee}[\rho(r)] + v_{xc}[\rho(r)]. \tag{1.16}$$

In section 1.2 we have known that

$$v_{ex} = -\frac{z}{r},\ V_{ee} = \int_{R^3}\int \rho(r')\frac{1}{r>}dr'.$$

The exchange-correlation potential v_{xc} is given by [1]

$$v_{xc}[\rho(r)] = \frac{\delta\left(\varepsilon_{xc}[\rho(r)]\cdot\rho(r)\right)}{\delta\rho(r)} = \varepsilon_{xc}[\rho(r)] + \rho(r)\frac{d\varepsilon_{xc}}{d\rho},$$

where $\varepsilon_{xc} = \varepsilon_x + \varepsilon_x$ is written in section 1.2. For the convenience of computing (1.15), we presume a function P_{nl} such that

$$P_{nl}(r) = rR_{nl}(r).$$

Multiplying both sides of (1.15) by r, this equation may also be written as

$$-\frac{1}{2}\frac{d^2P_{nl}(r)}{dr^2} + \frac{l(l+1)}{2r^2}P_{nl}(r) + v_{eff}[\rho]P_{nl}(r) = e_{nl}P_{nl}(r). \tag{1.17}$$

So

$$\left(-\frac{1}{2}\frac{d^2}{dr^2} + \frac{l(l+1)}{2r^2} + v_{eff}[\rho]\right)P_{nl}(r) = e_{nl}P_{nl}(r). \tag{1.18}$$

The electronic density $\rho(r)$ is written as

$$\begin{aligned}\rho(r) &= \sum_{i\in occupied} f_i|\Psi_i(r,\theta,\phi)|^2\\ &= \sum_{i\in occupied} f_i\left(\sum_{m=-l}^{l}|R_{nl(i)}(r)\cdot Y_{lm(i)}(\theta,\phi)|^2\right)\\ &= \sum_{i\in occupied} f_i\left(\sum_{m=-l}^{l}|R_{nl(i)}(r)|^2\cdot|Y_{lm(i)}(\theta,\phi)|^2\right)\\ &= \sum_{i\in occupied} f_i\left(|R_{nl(i)}(r)|^2\cdot\sum_{m=-l}^{l}|Y_{lm(i)}(\theta,\phi)|^2\right).\end{aligned}$$

For a closed shell system we have the identity that [2]

$$\sum_{m=-l}^{l}|Y_{lm}(\theta,\phi)|^2 = 1,$$

we obtain the equation

$$\rho(r) = \sum_{i \in occupied} f_i |R_{nl(i)}(r)|^2.$$

In section 2, we will describe the discretization of the equation (1.18).

2 Calculate the ground state energy of atoms

2.1 The process of the computation

In the beginning of this computation, we give an initial guess of the electronic density to determine the correct wave function and electronic density by self-consistency (In section 2.7 we will introduce self-consistency). After finding the correct wave function and electronic density, we can calculate the ground state energy by using the ground state energy functional E_G. We choose the electronic density builded by the wave function of hydrogen [2] to be the initial guess of the computation of helium. After the computation of helium being finished, we can use the electronic density of helium to be the initial guess of the computation of lithium. Such like the process of the computation of helium, we can compute the ground state energy of the atoms one by one by using the preceding atom to be the initial guess.

2.2 The integration and the differentiation

By the functional (1.12), we know that if we want to the compute ground state energy, we must find the wave functions and electronic density. To find the wave functions, we will solve the equation (1.18). Now, we discuss the computational method and the discretization of this computation. In this computation, we try to compute in the numerical method. We take the computational domain $r = [0\mathring{A}, 10\mathring{A}]$ and divide it into 400 equal parts. The domain is represented as

$$\vec{r} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{399} \end{bmatrix}, \text{where } r_i = h \cdot i,$$

and h is $10\mathring{A}$ over 400, $h = \dfrac{10\mathring{A}}{400}$.

Now, we define the integration and the differentiation in this computation. First, we use the "trapezoidal rule" to compute the integration in this computation. The domain has been discretized as the vector $\vec{r}$.

So we defined the integration as

$$\begin{aligned} \int_{r_i}^{r_k} f(r)dr &= \frac{f(r_i) + f(r_{i+1})}{2} \cdot h + \frac{f(r_{i+1}) + f(r_{i+2})}{2} \cdot h \\ &+ ... + \frac{f(r_{k-1}) + f(r_k)}{2} \cdot h \\ &= h \cdot (\frac{f(r_i)}{2} + \sum_{j=i+1}^{k-1} f(r_j) + \frac{f(r_k)}{2}) \text{ , } for \ 0 \leq i < k \leq 399 \end{aligned} \tag{2.19}$$

$$\int_0^\infty f(r)dr \approx \int_{0\mathring{A}}^{10\mathring{A}} f(r)dr = h \cdot (\frac{f(r_1)}{2} + \sum_{j=2}^{398} f(r_j) + \frac{f(r_{399})}{2}). \tag{2.20}$$

And the differentiation in this computation is defined as

$$f'(r_i) = \left.\frac{df(x)}{dx}\right|_{x=r_i} = \frac{f(r_{i+1}) - f(r_{i-1})}{r_{i+1} - r_{i-1}}, for \ 0 \leq i < k \leq 399. \tag{2.21}$$

The differentiation of $f(r_1)$,$f(r_{399})$ is defined as

$$f'(r_1) = \frac{f(r_2) - f(r_1)}{r_2 - r_1}, \ f'(r_{399}) = \frac{f(r_{399}) - f(r_{398})}{r_{399} - r_{398}}. \tag{2.22}$$

2.3 Discretization of $-\dfrac{1}{2}\dfrac{d^2P_{nl}(r)}{dr^2}$

The discretization of $-\frac{1}{2}\frac{d^2P_{nl}(r)}{dr^2}$ is an approximation.

By the Taylor series

$$P(x+h) = P(x) + P'(x)h + \frac{P''}{2!}h^2 + \frac{P'''}{3!}h^3 + \cdots,$$

and

$$P(x-h) = P(x) + P(x)(-h) + \frac{P''}{2!}(-h)^2 + \frac{P'''}{3!}(-h)^3 + \cdots.$$

So that

$$P(x+h) + P(x-h) = 2P(x) + P''h^2 + O(h^4).$$

We have

$$\frac{-P(x+h) + 2P(x) - P(x-h)}{h^2} \approx -P'',$$

and we obtain

$$-\frac{1}{2}P'' \approx M_p = \begin{bmatrix} \frac{1}{h^2} & \frac{-1}{2h^2} & \cdots & 0 \\ \frac{-1}{2h^2} & \frac{1}{h^2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \frac{-1}{2h^2} \\ 0 & \cdots & \frac{-1}{2h^2} & \frac{1}{h^2} \end{bmatrix}_{399\times399}. \tag{2.23}$$

2.4 Discretization of $\frac{l(l+1)}{2r_i^2}$

$\frac{l(l+1)}{2r_i^2}$ is discretized as

$$M_l = \begin{bmatrix} \frac{l(l+1)}{2r_1^2} & & & 0 \\ & \frac{l(l+1)}{2r_2^2} & & \\ & & \ddots & \\ 0 & & & \frac{l(l+1)}{2r_{399}^2} \end{bmatrix}_{399\times399}. \tag{2.24}$$

2.5 Discretization of effective potential

By (1.16) we have known that effective potential contains Coulomb potential, exchange-correlation potential and external potential. Now we discuss their discretization.

2.5.1 Discretization of Coulomb potential

The Coulomb potential

$$\begin{aligned} v_{ee} &= \int_{R^3}\int \frac{\rho(r')}{|r-r'|}dr'd\omega \\ &= 4\pi\int_0^\infty r'^2\rho(r')\frac{1}{r>}dr' \\ &= 4\pi\int_0^r r'^2\rho(r')\frac{1}{r}dr' + 4\pi\int_r^\infty r'^2\rho(r')\frac{1}{r'}dr' \\ &= IntV_{ee}(r). \end{aligned}$$

We obtain the matrix of Coulomb potential

$$M_{ee} = \begin{bmatrix} IntV_{ee}(r_1) & & 0 \\ & IntV_{ee}(r_2) & \\ & & \ddots & \\ 0 & & & IntV_{ee}(r_{399}) \end{bmatrix}_{399\times399}. \tag{2.25}$$

By (2.19) and (2.20), the integral of $IntV_{ee}(r_i)$ can be discretized.

2.5.2 Discretization of exchange-correlation potential

The exchange-correlation potential is written as

$$\begin{aligned} v_{xc} &= \frac{\varepsilon_{xc}[\rho(r)]\cdot\rho(r)}{\delta\rho(r)} \\ &= \varepsilon_{xc}[\rho(r)] + \rho\cdot\frac{d\varepsilon_{xc}[\rho(r)]}{d\rho(r)} \\ &= \varepsilon_{xc}[\rho(r)] + \rho\cdot\frac{d\varepsilon_{xc}[\rho(r)]}{dr}\cdot\frac{dr}{d\rho(r)} \\ &= Dif\varepsilon_{xc}(r), \end{aligned}$$

where $\varepsilon_{xc} = \varepsilon_x + \varepsilon_c$ is written in section 1.2.

$$M_{xc} = \begin{bmatrix} Dif\varepsilon_{xc}(r_1) & & & 0 \\ & Dif\varepsilon_{xc}(r_2) & & \\ & & \ddots & \\ 0 & & & Dif\varepsilon_{xc}(r_{399}) \end{bmatrix}_{399\times 399} . \tag{2.26}$$

By (2.21) and (2.22), the differentiation of $Dif\varepsilon_{xc}(r_i)$ can be discretized.

2.5.3 Discretization of external potential

The external potential is written as $v_{ex} = -\frac{z}{r}$ so after the dicretization the matrix is

$$M_r = \begin{bmatrix} -\frac{z}{r_1} & & & 0 \\ & -\frac{z}{r_2} & & \\ & & \ddots & \\ 0 & & & -\frac{z}{r_{399}} \end{bmatrix}_{399\times 399} , \tag{2.27}$$

where z is the atomic number.

2.6 Normalization of wave function

The square of a wave function is called the probability density. In quantum mechanics, the probability density can describe the distribution of the electrons. In this computation, we have to normalize the wave function to find the wave function that can provide a reasonable probability. The definition of normalization is

$$1 = \int_{R^3}\int P_p \cdot dr d\omega,$$

which the definition of the probability density P_p is

$$P_p = \Psi^*\Psi = R_{nl}^* R_{nl} Y_{lm}^* Y_{lm}.$$

So that we have

$$1 = \int_{R^3}\int P_p dr d\omega = \int_0^\infty r^2 R_{nl}^* R_{nl} dr \int_o^\pi \int_0^{2\pi} Y_{lm}^* Y_{lm} sin\phi d\theta d\phi. \tag{2.28}$$

The radial probability density $P_{pr}(r)$ is defined by

$$P_{pr}(r) = r^2 R_{nl}^* R_{nl}.$$

Because of the integral [7]

$$\int_o^{\pi} \int_0^{2\pi} Y_{lm}^* Y_{lm} sin\phi d\theta d\phi = 1$$

and (2.28), we have the result

$$\int_0^{\infty} P_{pr}(r)dr = \int_0^{\infty} r^2 R_{nl}^* R_{nl} dr = 1. \tag{2.29}$$

Therefore, we have the conclusion that if we want to normalize a radial wave function R_{nl}, we just compute the integral

$$\int_0^{\infty} r^2 R_{nl}^* R_{nl} dr = k, \ k \text{ is a constant},$$

and divided R_{nl} by $\sqrt{k}$. Then the new function $R'_{nl} = \frac{R_{nl}}{\sqrt{k}}$ is the radial wave function which has been normalized.

2.7 Self-consistency

Define M_{eff} be the matrix which signifies the effective potential v_{eff}, and

$$M_{eff} = M_{ee} + M_{xc} + M_r. \tag{2.30}$$

By (2.23), (2.24) and (2.30), the equation (1.18) can be represented as the form

$$(M_p + M_l + M_{eff})\vec{P_{nl}} = e_{nl}\vec{P_{nl}},$$

where the vector $\vec{P_{nl}}$ is

$$\vec{P_{nl}} = \begin{bmatrix} P_{nl}(\vec{r_1}) \\ P_{nl}(\vec{r_2}) \\ \vdots \\ P_{nl}(\vec{r_{399}}) \end{bmatrix}.$$

It is an eigenvalue problem.

We can determine the correct electronic density and $\vec{P_{nl}}$ by executing self-consistency. The table (2.7) can describe the process of self-consistency.

In the beginning, build a vector $\vec{\rho_{in}}$ to be the initial guess, and we will get a Hamiltonian H_{kh} (here $H_{kh} = M_p + M_l + M_{eff}$). Then, computing the minimal eigenvalue e_{min} of H_{kh}, and find an eigenvector $\vec{P}$ corresponding e_{min}. Normalizing $\vec{P}$ and we get the eigenvector $\vec{P_{out}}$. Then we can compute the electronic density $\vec{\rho_{out}}$. Now, if $\vec{\rho_{out}} = \vec{\rho_{in}}$ it means that we get the right P_{nl}, otherwise, generate a new $\vec{\rho_{in}}$ as this $\vec{\rho_{out}}$ and construct a new H_{kh}. After finding the correct electronic density and wave function, we can calculate the ground state energy by using the formula (1.12).

We compute the ground state energy of He ($z = 2$) to No ($z = 102$) in the order, and use the preceding atom's $\vec{\rho_{nl}}$ to be the initial guess. For example, if we want to compute the ground state energy of Mg ($z = 12$), we will use the $\vec{\rho_{nl}}$ of Na ($z = 11$) to be its initial guess.

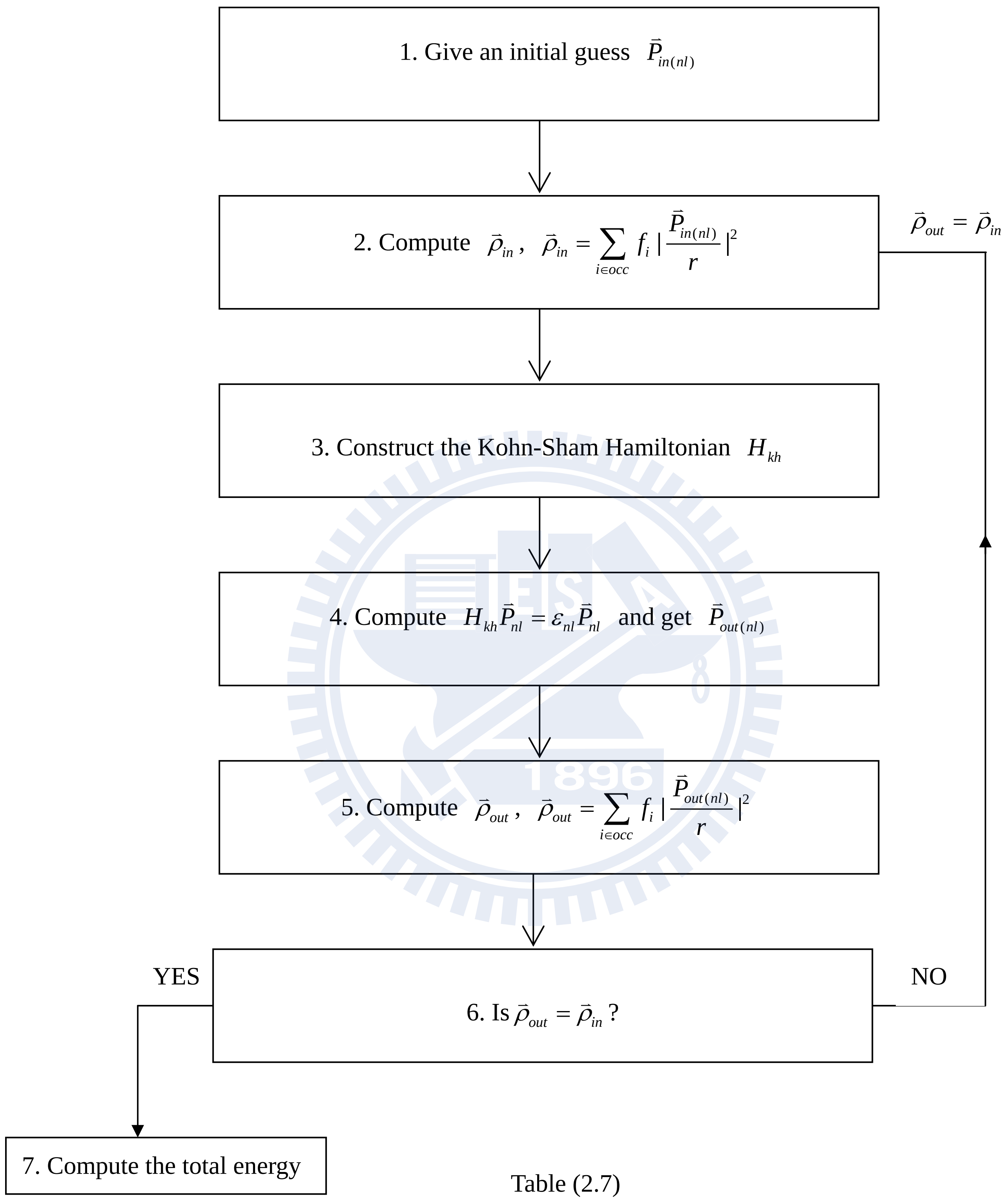
1. Give an initial guess $\vec{P}_{in(nl)}$
2. Compute $\vec{\rho}_{in}$, $\vec{\rho}_{in} = \sum_{i \in occ} f_i \left| \frac{\vec{P}_{in(nl)}}{r} \right|^2$
$\vec{\rho}_{out} = \vec{\rho}_{in}$
3. Construct the Kohn-Sham Hamiltonian H_{kh}
4. Compute $H_{kh}\vec{P}_{nl} = \varepsilon_{nl}\vec{P}_{nl}$ and get $\vec{P}_{out(nl)}$
5. Compute $\vec{\rho}_{out}$, $\vec{\rho}_{out} = \sum_{i \in occ} f_i \left| \frac{\vec{P}_{out(nl)}}{r} \right|^2$
YES
NO
6. Is $\vec{\rho}_{out} = \vec{\rho}_{in}$?
7. Compute the total energy

Table (2.7)

3 Results

The errors between the computation and the realistic values are less than 15%. The list (3.1) is the comparison of the ideal answer [3] and the answer of this computation with atomic unit, and figure (3.2) is the scale error between the realistic values (Idea data) and the answer of this computation .

The formula of the errors is

$$\text{errors} = \frac{|a_i - b_i|}{a_i},$$

where a_i is the data from this computation and b_i is the idea data.

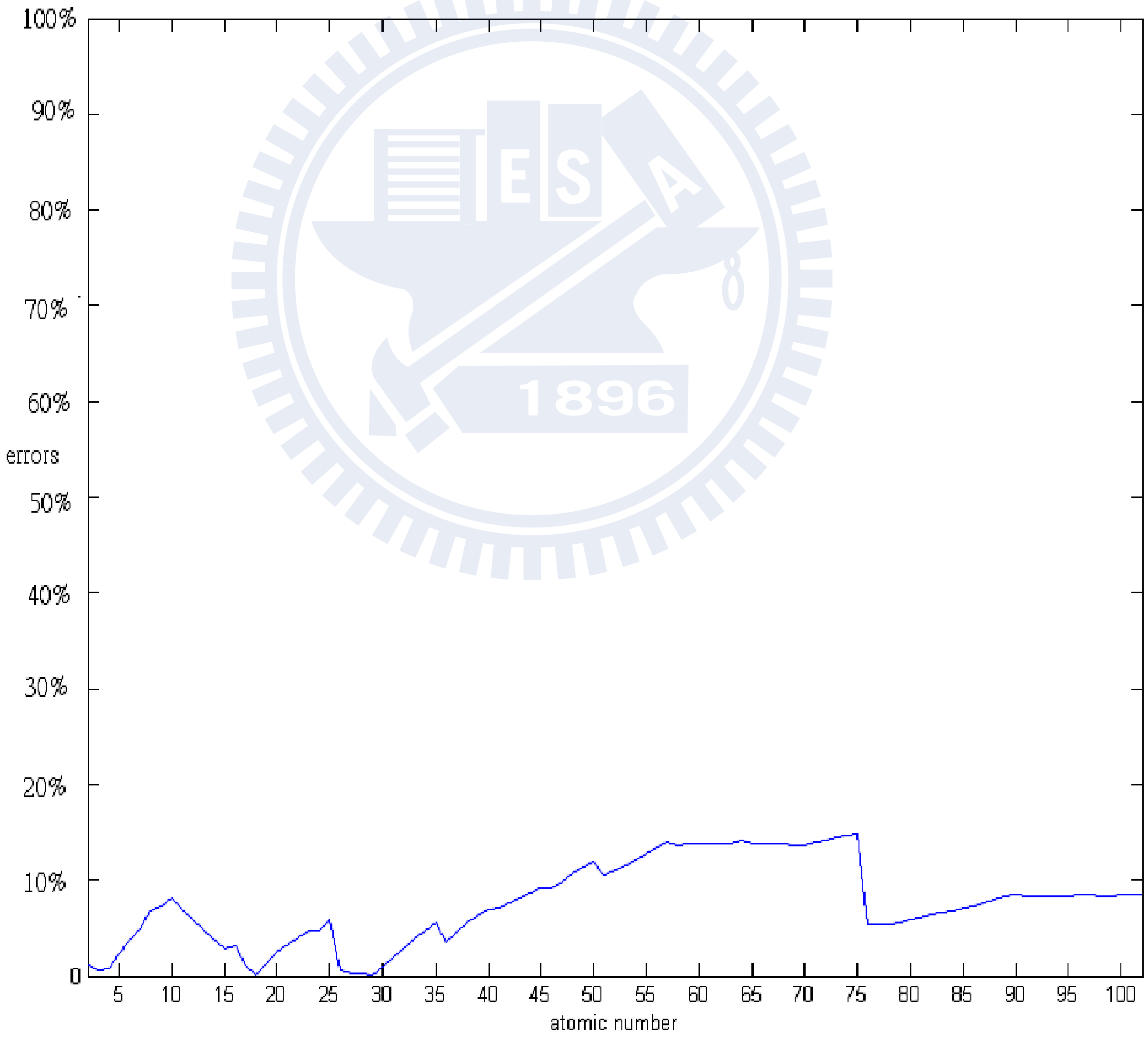

Figure (3.2)

Atomic number	Idea data –E(a.u.)	Data in this computation	Errors(%)
1	0.500	0.500000	0.0000
2	2.862	2.893138	1.0800
3	7.433	7.388389	0.6000
4	14.57	14.67078	0.6900
5	24.53	25.11337	2.3200
6	37.69	39.11898	3.6500
7	54.40	57.24088	4.9600
8	74.41	79.84583	6.8100
9	99.41	107.2576	7.3200
10	128.5	139.8187	8.1000
11	161.9	173.8986	6.9000
12	199.6	212.1369	5.9100
13	241.9	254.2069	4.8400
14	289	300.3957	3.7900
15	340.7	350.7181	2.8600
16	397.5	410.3264	3.1300
17	459.5	464.3733	1.0500
18	526,8	527.6907	0.1700
19	599.2	592.0264	1.2100
20	676.8	659.9070	2.5600
21	759.7	735.5189	3.2900
22	848.4	815.8062	4.0000
23	942.9	900.7615	4.6800
24	1043	995.9873	4.7200
25	1150	1085.262	5.9700
26	1262	1270.632	0.6800
27	1381	1384.305	0.2400
28	1507	1503.796	0.2100
29	1639	1640.438	0.0900
30	1778	1761.187	0.9500
31	1923	1886.647	1.9300
32	2075	2016.884	2.8800
33	2234	2151.999	3.8100
34	2400	2292.058	4.7100
35	2572	2437.159	5.5300

Atomic number	Idea data -E(a.u.)	Data in this computation	Errors(%)
36	2752	2657.920	3.5400
37	2938	2809.000	4.5900
38	3132	2964.275	5.6600
39	3332	3134.383	6.3000
40	3539	3310.443	6.9000
41	3754	3504.019	7.1300
42	3975	3692.796	7.6400
43	4205	3887.294	8.1700
44	4441	4087.941	8.6400
45	4686	4294.852	9.1100
46	4938	4523.195	9.1700
47	5198	4728.309	9.9300
48	5465	4937.590	10.6800
49	5740	5155.548	11.3400
50	6023	5379.200	11.9700
51	6314	5713.260	10.5100
52	6612	5953.587	11.0600
53	6918	6200.008	11.5800
54	7232	6452.556	12.0800
55	7554	6697.584	12.7900
56	7884	6947.461	13.4800
57	8221	7215.920	13.9300
58	8567	7535.974	13.6800
59	8921	7844.062	13.7300
60	9284	8159.394	13.7800
61	9655	8482.952	13.8200
62	10035	8814.835	13.8400
63	10423	9155.146	13.8500
64	10820	9476.594	14.1800
65	11226	9860.293	13.8500
66	11641	10227.65	13.8200
67	12065	10602.66	13.7900
68	12498	10986.59	13.7600
69	12940	11379.53	13.7100
70	13391	11781.56	13.6600

Atomic number	Idea data -E(a.u.)	Data in this computation	Errors(%)
71	13852	12159.12	13.9200
72	14323	12542.46	14.2000
73	14800	12933.05	14.4400
74	15287	13330.92	14.6700
75	15784	13736.11	14.9100
76	16293	15464.24	5.3600
77	16806	15932.53	5.4800
78	17333	16442.05	5.4200
79	17866	16929.71	5.5300
80	18409	17391.06	5.8500
81	18962	17868.97	6.1200
82	19524	18354.69	6.3700
83	20096	18848.17	6.6200
84	20676	19349.36	6.8600
85	21267	19858.31	7.0900
86	21867	20375.08	7.3200
87	22476	20871.18	7.6900
88	23094	21373.11	8.0500
89	23722	21907.81	8.2800
90	24360	22450.80	8.5000
91	25007	23074.43	8.3800
92	25664	23674.40	8.4000
93	26331	24320.71	8.2700
94	27008	24940.68	8.2900
95	27696	25568.91	8.3200
96	28392	26166.23	8.5100
97	29100	26811.44	8.5400
98	29817	27510.10	8.3900
99	30545	28178.74	8.4000
100	31283	28855.63	8.4100
101	32031	29542.39	8.4200
102	32790	30239.09	8.4400

List (3.1)

References

[1] P.Marder, Micheal. *Condensed Matter Physics.* A Wiley-Tnterscience Publication, 203-253. 1960.

[2] Gasiorowicz, Stephen. *Quantum Physics.* John Wiley & Sons, Inc, 197-200. 1995.

[3] Fisher, Forse. *Atomic Data. erratum Atom. Data. Nucl. Tables 12 87.* 301. 1972.

[4] R. L. Flyrry Jr. *Quantum chemistry- An introduction.* Englewood Cliffs, NJ: Prentice-Hall, 32-85. 1983.

[5] Parr, Robert G. *Density-Function Theory of Atoms and Molecules.* New York: Oxford University Press, 1989.

[6] Fiolhais, Carlos, Nogueira. *Primer in Density Functional Theory.* Berlin; New York: Springer,. 2003.

[7] Menguc, Mustafa, Pinar. *Modeling of Radiate Heat transfer in Multi-didimensional Enclosures using Spherical Harmonics Approximation.* A Bell & Howel information Company, 17. 1985.

[8] Gilbarg David. *Elliptic partial differential equations of second order.* New York: Springer, 2001.

[9] Richard, Courant and David, Hilbert. *Methods of mathematical Physics.* 1962.

[10] Dahl, Jens Peder ed. *Local Density Approximation in Quantum Chemistry and Solid State Physics.* New York: Plenum, 1984.

Appendix

The source file of this computation

```
module global

  implicit none
  integer ,parameter ::d=400
  integer ,parameter ::sz=399
  integer i,j,Bool,k,times
  real ,parameter ::pi=3.1416
  real ,parameter ::bdd =10.0
  real Yini(sz),Nini(sz),Pin(sz),Rin(sz),Yin(sz),ABNd(sz)
  real Nin(sz),Pout(sz),Rout(sz),Yout(sz),Nout(sz),R(sz),N(sz),Nd(sz)
  real Sch(sz,sz),L1(sz,sz),L2(sz,sz)
  real Veff(sz,sz),Exc(sz,sz),Eee(sz,sz),Rv(sz,sz)
  real s,Ncin,Ncout
  real A_2nd(sz),B_2nd(sz),A_3rd(sz),B_3rd(sz)
  real A_4th(sz),B_4th(sz),A_5th(sz),B_5th(sz)
  real A_6th(sz),B_6th(sz),A_7th(sz),B_7th(sz)
  integer k_2nd,k_3rd,k_4th,k_5th,k_6th,k_7th
  real Pin_2s(sz),Pout_2s(sz),H_2s(sz),Rout_2s(sz)
  real Pout_3s(sz),Pin_3s(sz),Rout_3s(sz),H_3s(sz)
  real Pin_2p(sz),Pout_2p(sz),H_2p(sz),Rout_2p(sz)
  real H_3p(sz),Pout_3p(sz),Pin_3p(sz),Rout_3p(sz)
  real Pin_4s(sz),Pout_4s(sz),H_4s(sz),Rout_4s(sz)
  real Pin_3d(sz),Pout_3d(sz),H_3d(sz),Rout_3d(sz)
  real Pin_4p(sz),Pout_4p(sz),H_4p(sz),Rout_4p(sz)
  real Pin_5s(sz),Pout_5s(sz),H_5s(sz),Rout_5s(sz)
  real Pin_4d(sz),Pout_4d(sz),H_4d(sz),Rout_4d(sz)
  real Pin_5p(sz),Pout_5p(sz),H_5p(sz),Rout_5p(sz)
  real Pin_6s(sz),Pout_6s(sz),H_6s(sz),Rout_6s(sz)
  real Pin_5d(sz),Pout_5d(sz),H_5d(sz),Rout_5d(sz)
  real Pin_4f(sz),Pout_4f(sz),H_4f(sz)
  real Rout_4f(sz)
  real Pin_6p(sz),Pout_6p(sz),H_6p(sz),Rout_6p(sz)
  real Pin_7s(sz),Pout_7s(sz),H_7s(sz),Rout_7s(sz)
  real Pin_6d(sz),Pout_6d(sz),H_6d(sz),Rout_6d(sz)
```

```
 real Pin_5f(sz),Pout_5f(sz),H_5f(sz),Rout_5f(sz)
 real Pin_7p(sz),Pout_7p(sz),H_7p(sz),Rout_7p(sz)
!sch 變數
 real H(sz),eigenvalue(sz),eigenvector(sz,sz)
!L1 變數
 real hg
!L2 變數
!Veff 變數
!Ec 變數
 real Ec(sz,sz)
 real ,parameter ::Ge=0.1423
 real ,parameter ::Be1=1.0529
 real ,parameter ::Be2=0.3334
 real Ech(sz,sz)
 real ,parameter ::Aec=0.0311
 real ,parameter ::Bec=-0.048
 real ,parameter ::Cec=0.002
 real ,parameter ::Dec=-0.0116
!Ex 變數
 real Rs(sz),Ex(sz,sz)
!Ediff 變數
 real Addxc(sz,sz),Edr(sz,sz),Ndr(sz,sz),Ediff(sz,sz)
!Exc 變數
!intg1 變數
!real ,external ::Intg1

!intg1 變數
!real ,external ::Intg2
!Eee 變數
!Exc 變數
!Energy 的變數
 real total
 real
lPout_1sl(sz),lPout_2sl(sz),lPout_3sl(sz),lPout_4sl(sz),lPout_5sl(sz)
,lPout_6sl(sz),lPout_7sl(sz)
 real
lPout_2pl(sz),lPout_3pl(sz),lPout_4pl(sz),lPout_5pl(sz),lPout_6pl(sz)
 real lPout_3dl(sz),lPout_4dl(sz),lPout_5dl(sz),lPout_6dl(sz)
```

```
  real lPout_4fl(sz),lPout_5fl(sz)
  real
Hn(sz),Hn_2s(sz),Hn_3s(sz),Hn_4s(sz),Hn_5s(sz),Hn_6s(sz),Hn_7s(sz)
  real Hn_2p(sz),Hn_3p(sz),Hn_4p(sz),Hn_5p(sz),Hn_6p(sz)
  real Hn_3d(sz),Hn_4d(sz),Hn_5d(sz),Hn_6d(sz)
  real Hn_4f(sz),Hn_5f(sz)
end module


!-----------------------------------------
Program main

  use IMSL
  use global
  implicit none

 call self_he()

 open(unit=11,file='Reslet1.txt')
 open(unit=12,file='eigenvalueofHe.txt')
 call total_energy(2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
  write(*,*)"He(2)"
  write(*,*)times
  write(*,*)total
 write(11,*)"He(2)"
  write(11,*)times
  write(11,*)total
  !do i=1,sz
  !write(12,*)eigenvalue(i)
  !end do
 call Li3()
 call Be4()
 call B5()
 call C6()
 call N7()
 call O8()
 call F9()
 call Ne10()
 call Na11()
 call Mg12()
 call Al13()
 call Si14()
 call P15()
 call S16()
 call Cl17()
 call Ar18()
 call K19()
 call Ca20()
 call Sc21()
 call Ti22()
 call V23()
 call Cr24()
 call Mn25()
 call Fe26()
 call Co27()
 call Ni28()
 call Cu29()
 call Zn30()
 call Ga31()
 call Ge32()
 call As33()
 call Se34()
 call Br35()
 call Kr36()
 call Rb37()
 call Sr38()
 call Y39()
 call Zr40()
 call Nb41()
 call Mo42()
 call Tc43()
 call Ru44()
 call Rh45()
 call Pb46()
 call Ag47()
 call Cd48()
 call In49()
 call Sn50()
 call Sb51()
 call Te52()
 call I53()
 call Xe54()
 call Cs55()
 call Ba56()
 call La57()
 call Ce58()
 call Pr59()
 call Nd60()
 call Pm61()
 call Sm62()
 call Eu63()
 call Gd64()
 call Tb65()
 call Dy66()
 call Ho67()
 call Er68()
 call Tm69()
 call Yb70()
 call Lu71()
 call Hf72()
 call Ta73()
 call W74()
 call Re75()
 call Os76()
 call Ir77()
 call Pt78()
 call Au79()
 call Hg80()
 call Tl81()
 call Pb82()
 call Bi83()
 call Po84()
 call At85()
 call Rn86()
```

```
 call Fr87()
 call Ra88()
 call Ac89()
 call Th90()
 call Pa91()
 call U92()
 call Np93()
 call Pu94()
 call Am95()
 call Cm96()
 call Bk97()
 call Cf98()
 call Es99()
 call Fm100()
 call Md101()
 call No102()
 call Lr103()
 call Rf104()
 call Db105()
 call Sg106()
 call Bh107()

end

!---------------------------------------------------
subroutine subSch_7s() !矩陣合
  use global
  use IMSL
  implicit none
  real ,external :: norl
 do i=1,sz
 do j=1,sz
  Sch(i,j)=L1(i,j)+L2(i,j)+Veff(i,j)
 end do
 end do
  eigenvalue=eig(Sch,V=eigenvector)
!爲了存取特徵値的最小値位置
  open(unit=10,file="7s_1st.txt")
  write(10,*)minloc(eigenvalue)
  rewind(10)
  read(10,*)k
!--------------
 do i=1,sz
  H(i)=eigenvector(i,k)
 end do
 do i=1,sz
  Hn(i)=H(i)/nor1(R,H)
 end do
 do i=1,sz
 A_2nd(i)=0
 end do
 A_2nd(k)=100000
 do i=1,sz
 B_2nd(i)=A_2nd(i)+eigenvalue(i)
 end do
  open(unit=10,file="7s_2nd.txt")
  write(10,*)minloc(B_2nd)
  rewind(10)
  read(10,*)k_2nd
 do i=1,sz
 H_2s(i)=eigenvector(i,k_2nd)
 end do
 do i=1,sz
  Hn_2s(i)=H_2s(i)/nor1(R,H_2s)
 end do
 !------
 do i=1,sz
 A_3rd(i)=0
 end do
 A_3rd(k_2nd)=100000
 do i=1,sz
 B_3rd(i)=B_2nd(i)+A_3rd(i)
 end do
  open(unit=10,file="7s_3rd.txt")
  write(10,*)minloc(B_3rd)
  rewind(10)
  read(10,*)k_3rd
 do i=1,sz
 H_3s(i)=eigenvector(i,k_3rd)
 end do
 do i=1,sz
  Hn_3s(i)=H_3s(i)/norl(R,H_3s)
 end do
 do i=1,sz
 A_4th(i)=0
 end do
 A_4th(k_3rd)=100000
 do i=1,sz
 B_4th(i)=B_3rd(i)+A_4th(i)
 end do
  open(unit=10,file="7s_4th.txt")
  write(10,*)minloc(B_4th)
  rewind(10)
  read(10,*)k_4th
 do i=1,sz
 H_4s(i)=eigenvector(i,k_4th)
 end do
 do i=1,sz
  Hn_4s(i)=H_4s(i)/norl(R,H_4s)
 end do
 do i=1,sz
 A_5th(i)=0
 end do
 A_5th(k_4th)=100000
 do i=1,sz
 B_5th(i)=B_4th(i)+A_5th(i)
 end do
  open(unit=10,file="7s_5th.txt")
  write(10,*)minloc(B_5th)
  rewind(10)
  read(10,*)k_5th
 do i=1,sz
 H_5s(i)=eigenvector(i,k_5th)
 end do
```

```
do i=1,sz
 Hn_5s(i)=H_5s(i)/norl(R,H_5s)
end do
do i=1,sz
A_6th(i)=0
end do
A_6th(k_5th)=100000
do i=1,sz
B_6th(i)=B_5th(i)+A_6th(i)
end do
 open(unit=10,file="7s_6th.txt")
 write(10,*)minloc(B_6th)
 rewind(10)
 read(10,*)k_6th
do i=1,sz
H_6s(i)=eigenvector(i,k_6th)
end do
do i=1,sz
 Hn_6s(i)=H_6s(i)/norl(R,H_6s)
end do
do i=1,sz
A_7th(i)=0
end do
A_7th(k_6th)=100000
do i=1,sz
B_7th(i)=B_6th(i)+A_7th(i)
end do
 open(unit=10,file="7s_7th.txt")
 write(10,*)minloc(B_7th)
 rewind(10)
 read(10,*)k_7th
do i=1,sz
H_7s(i)=eigenvector(i,k_7th)
end do
do i=1,sz
 Hn_7s(i)=H_7s(i)/norl(R,H_7s)
end do
```

```
end
subroutine subSch_6p() !矩陣合
  use global
  use IMSL
  implicit none
 real ,external :: norl
 do i=1,sz
 do j=1,sz
  Sch(i,j)=L1(i,j)+L2(i,j)+Veff(i,j)
 end do
 end do
  eigenvalue=eig(Sch,V=eigenvector)
!爲了存取特徵値的最小値位置
  open(unit=10,file="6p_1st.txt")
  write(10,*)minloc(eigenvalue)
  rewind(10)
  read(10,*)k
!--------------
 do i=1,sz
  H_2p(i)=eigenvector(i,k)
 end do
 do i=1,sz
  Hn_2p(i)=H_2p(i)/norl(R,H_2p)
 end do
 do i=1,sz
 A_2nd(i)=0
 end do
 A_2nd(k)=100000
 do i=1,sz
 B_2nd(i)=A_2nd(i)+eigenvalue(i)
 end do
  open(unit=10,file="6P_2nd.txt")
  write(10,*)minloc(B_2nd)
  rewind(10)
  read(10,*)k_2nd
 do i=1,sz
 H_3p(i)=eigenvector(i,k_2nd)
 end do
```

```
do i=1,sz
 Hn_3p(i)=H_3p(i)/norl(R,H_3p)
end do
do i=1,sz
A_3rd(i)=0
end do
A_3rd(k_2nd)=100000
do i=1,sz
B_3rd(i)=A_3rd(i)+B_2nd(i)
end do
 open(unit=10,file="6P_3rd.txt")
 write(10,*)minloc(B_3rd)
 rewind(10)
 read(10,*)k_3rd
do i=1,sz
H_4p(i)=eigenvector(i,k_3rd)
end do
do i=1,sz
 Hn_4p(i)=H_4p(i)/norl(R,H_4p)
end do
!---------------------------
do i=1,sz
A_4th(i)=0
end do
A_4th(k_3rd)=100000
do i=1,sz
B_4th(i)=A_4th(i)+B_3rd(i)
end do
 open(unit=10,file="6P_4th.txt")
 write(10,*)minloc(B_4th)
 rewind(10)
 read(10,*)k_4th
do i=1,sz
H_5p(i)=eigenvector(i,k_4th)
end do
do i=1,sz
 Hn_5p(i)=H_5p(i)/norl(R,H_5p)
end do
```

```
 do i=1,sz
 A_5th(i)=0
 end do
 A_5th(k_4th)=100000
 do i=1,sz
 B_5th(i)=A_5th(i)+B_4th(i)
 end do
  open(unit=10,file="6P_5th.txt")
  write(10,*)minloc(B_5th)
  rewind(10)
  read(10,*)k_5th
 do i=1,sz
 H_6p(i)=eigenvector(i,k_5th)
 end do
 do i=1,sz
  Hn_6p(i)=H_6p(i)/norl(R,H_6p)
 end do
end
subroutine subSch_6d() !矩陣合
  use global
  use IMSL
  implicit none
  real ,external ::norl
 do i=1,sz
 do j=1,sz
  Sch(i,j)=L1(i,j)+L2(i,j)+Veff(i,j)
 end do
 end do
  eigenvalue=eig(Sch,V=eigenvector)
!爲了存取特徵値的最小値位置
  open(unit=10,file="6d_1st.txt")
  write(10,*)minloc(eigenvalue)
  rewind(10)
  read(10,*)k
!--------------
 do i=1,sz
  H_3d(i)=eigenvector(i,k)
 end do
 do i=1,sz
  Hn_3d(i)=H_3d(i)/norl(R,H_3d)
 end do
 do i=1,sz
 A_2nd(i)=0
 end do
 A_2nd(k)=100000
 do i=1,sz
 B_2nd(i)=A_2nd(i)+eigenvalue(i)
 end do
  open(unit=10,file="6d_2nd.txt")
  write(10,*)minloc(B_2nd)
  rewind(10)
  read(10,*)k_2nd
 do i=1,sz
 H_4d(i)=eigenvector(i,k_2nd)
 end do
 do i=1,sz
  Hn_4d(i)=H_4d(i)/norl(R,H_4d)
 end do
do i=1,sz
 A_3rd(i)=0
 end do
 A_3rd(k_2nd)=100000
 do i=1,sz
 B_3rd(i)=A_3rd(i)+B_2nd(i)
 end do
  open(unit=10,file="6d_3rd.txt")
  write(10,*)minloc(B_3rd)
  rewind(10)
  read(10,*)k_3rd
 do i=1,sz
 H_5d(i)=eigenvector(i,k_3rd)
 end do
 do i=1,sz
  Hn_5d(i)=H_5d(i)/norl(R,H_5d)
 end do
do i=1,sz
```

```
 A_4th(i)=0
 end do
 A_4th(k_3rd)=100000
 do i=1,sz
 B_4th(i)=A_4th(i)+B_3rd(i)
 end do
  open(unit=10,file="6d_4th.txt")
  write(10,*)minloc(B_4th)
  rewind(10)
  read(10,*)k_4th
 do i=1,sz
 H_6d(i)=eigenvector(i,k_4th)
 end do
 do i=1,sz
  Hn_6d(i)=H_6d(i)/norl(R,H_6d)
 end do
end

subroutine subSch_5f() !矩陣合
  use global
  use IMSL
  implicit none
  real ,external :: norl
 do i=1,sz
 do j=1,sz
  Sch(i,j)=L1(i,j)+L2(i,j)+Veff(i,j)
 end do
 end do
  eigenvalue=eig(Sch,V=eigenvector)
!為了存取特徵值的最小值位置
  open(unit=10,file="5f_1st.txt")
  write(10,*)minloc(eigenvalue)
  rewind(10)
  read(10,*)k
!-------------
 do i=1,sz
  H_4f(i)=eigenvector(i,k)
 end do
 do i=1,sz
  Hn_4f(i)=H_4f(i)/norl(R,H_4f)
 end do
 do i=1,sz
 A_2nd(i)=0
 end do
 A_2nd(k)=100000
 do i=1,sz
 B_2nd(i)=A_2nd(i)+eigenvalue(i)
 end do
  open(unit=10,file="5f_2nd.txt")
  write(10,*)minloc(B_2nd)
  rewind(10)
  read(10,*)k_2nd
 do i=1,sz
 H_5f(i)=eigenvector(i,k_2nd)
 end do
 do i=1,sz
  Hn_5f(i)=H_5f(i)/norl(R,H_5f)
 end do
end
subroutine subL1() !兩次倒數用泰勒展開之後逼近的結果
  use IMSL
  use global
  implicit none
  L1=eye(sz)
  hg=bdd/d
 do i=1,sz
  L1(i,i)=1.0/(hg**2.0)
 end do
 do i=2,sz
  L1(i,i-1)=-1.0/(2.0*(hg**2.0))
 end do
 do i=1,sz-1
  L1(i,i+1)=-1.0/(2.0*(hg**2.0))
 end do
end
```

```
subroutine subL2(l) !l(l+1)/2r^2 那項
  use IMSL
  use global
  implicit none
  integer l
  L2=eye(sz)
  hg=bdd/d
 do i=1,sz
  L2(i,i)=(l*(l+1))/(2.0*((hg*i)**2.0))
 end do
end
subroutine subVeff(z) !Veff=Vee+Vxc+V 那項(總合)
  use IMSL
  use global
  implicit none
  integer z
  hg=bdd/d
!-z/r(Vr 那項)
  Rv=eye(sz)
 do i=1,sz
  Rv(i,i)=-z/(hg*i)
 end do
 do i=1,sz
 do j=1,sz
  Veff(i,j)=Eee(i,j)+Exc(i,j)+Rv(i,j)
 end do
 end do
end
subroutine subExc() !Vex=Exc+n*dExc/dn(總合)
  use global
  use IMSL
  implicit none
do j=1,sz
 do i=1,sz
 If (Rs(i)>=1) then
  Exc(i,j)=Ex(i,j)+Ec(i,j)+Ediff(i,j)
  else
  Exc(i,j)=Ex(i,j)+Ech(i,j)+Ediff(i,j)
```

```
end if
 end do
 end do
end
subroutine subEx()
  use IMSL
  use global
  implicit none
  Ex=eye(sz)
 do i=1,sz
  rs(i)=(3.0/(4.0*pi*N(i)))**(1.0/3.0)
 end do
 do i=1,sz
  Ex(i,i)=(-0.458)/(rs(i))
 end do
end
subroutine subEc()
  use IMSL
  use global
  implicit none
  Ec=eye(sz)
 do i=1,sz
  Ec(i,i)=-Ge/(1.0+Be1*(Rs(i)**(0.5))+Be2*Rs(i))
 end do
end
subroutine subEch()
  use IMSL
  use global
  implicit none
  Ech=eye(sz)
 do i=1,sz
 Ech(i,i)=Aec*LOG(Rs(i))+Bec+Cec*Rs(i)*LOG(Rs(i))+Dec*Rs(i)
 end do
 end
subroutine subEdiff() !用數值微分
  use global
  use IMSL
  implicit none
  Ediff=eye(sz)
 do i=1,sz
  R(i)=(bdd/d)*i
 end do
  Edr=eye(sz)
  Ndr=eye(sz)
 do j=1,sz
 do i=1,sz
if (Rs(i)>=1) then
  Addxc(i,j)=Ex(i,j)+Ec(i,j) !其實就是 Exc
  else
 Addxc(i,j)=Ex(i,j)+Ech(i,j)
end if
 end do
 end do
  Edr(1,1)=(Addxc(1,1)-Addxc(2,2))/(R(1)-R(2))
  Edr(sz,sz)=(Addxc(sz-1,sz-1)-Addxc(sz,sz))/(R(sz-1)-R(sz))
 do i=2,sz-1
  Edr(i,i)=(Addxc(i+1,i+1)-Addxc(i-1,i-1))/(R(i+1)-R(i-1))
 end do
  Ndr(1,1)=(N(1)-N(2))/(R(1)-R(2))
  Ndr(sz,sz)=(N(sz-1)-N(sz))/(R(sz-1)-R(sz))
 do i=2,sz-1
  Ndr(i,i)=(N(i+1)-N(i-1))/(R(i+1)-R(i-1))
 end do
 do i=1,sz
  Ediff(i,i)=Edr(i,i)*(1/Ndr(i,i))*N(i)
 end do
end
subroutine subEee()
  use Global
  use IMSL
  implicit none
  real ,external::Intg1
  real ,external::Intg2
  Eee=eye(sz)
do i=1,sz
  R(i)=(bdd/d)*i
 end do
 do i=1,sz
  Eee(i,i)=Intg1(i,n)+Intg2(i,n)
 end do
end
real function Intg1(a,n) !Eee 的數值積分
  use IMSL
  implicit none
  integer ,parameter::d=400
  integer ,parameter::sz=399
  real ,parameter::bdd=10.0
  real N(sz)
  real hg,Int
 real ,parameter ::pi=3.1416
  integer i,a
  Int=0.0
  hg=bdd/d
 do i=1,a-1
Int=Int+((((hg*i)**2.0)*N(i)/(a*hg))+(((hg*(i+1))**2.0)*N(i+1)/(a*hg)
)) *hg*0.5*4*pi
 end do
  Intg1=Int
end function
real function Intg2(b,n)
  use IMSL
  implicit none
  integer ,parameter ::d=400
  integer ,parameter ::sz=399
  real ,parameter::bdd=10.0
 real ,parameter ::pi=3.1416
  real N(sz)
  real hg,Int
  integer b,i
  Int=0.0
  hg=bdd/d
 do i=b,sz-1
Int=Int+((((hg*i)**2.0)*N(i))/(hg*i)+(((hg*(i+1))**2.0)*N(i+1))/(hg*(
i+1))) *hg*0.5*4*pi
```

```
 end do
  Intg2=Int !+( (((hg*sz)**2.0)*N(sz))/(hg*sz) )*0.5*hg
end function
subroutine subProcess_6p(l,z)!所有的 subroutine 都呼叫一次
  use IMSL
  use global
  implicit none
  integer l,z
  call subL1()
  call subL2(l)
  call subEee()
  call subEx()
  call subEc()
  call subEdiff()
  call subExc()
  call subVeff(z)
  call subSch_6p()
end
subroutine subProcess_7s(l,z)!所有的 subroutine 都呼叫一次
  use IMSL
  use global
  implicit none
  integer l,z
  call subL1()
  call subL2(l)
  call subEee()
  call subEx()
  call subEc()
  call subEdiff()
  call subExc()
  call subVeff(z)
  call subSch_7s()
end
subroutine subProcess_6d(l,z)!所有的 subroutine 都呼叫一次
  use IMSL
  use global
  implicit none
  integer l,z
  call subL1()
  call subL2(l)
  call subEee()
  call subEx()
  call subEc()
  call subEdiff()
  call subExc()
  call subVeff(z)
  call subSch_6d()
end
subroutine subProcess_5f(l,z)!所有的 subroutine 都呼叫一次
  use IMSL
  use global
  implicit none
  integer l,z
  call subL1()
  call subL2(l)
  call subEee()
  call subEx()
  call subEc()
  call subEdiff()
  call subExc()
  call subVeff(z)
  call subSch_5f()
end
!-Total Energy---
subroutine
Total_Energy(z,N1s,N2s,N2p,N3s,N3p,N4s,N3d,N4p,N5s,N4d,N5p,N6s,N5d,N4
f,N6p,N7s,N6d,N5f)!4d 的 energy
 use IMSL
  use global
  implicit none
  integer
z,N1s,N2s,N3s,N4s,N5s,N6s,N7s,N2p,N3p,N4p,N5p,N6p,N3d,N4d,N5d,N6d,N4f
,N5f
  real ,external :: Env
  real ,external :: Enee
  real ,external :: Enxc
```

```
   real ,external :: Intk1
   real ,external :: Intk2
   real ,external :: norl
   real Enkk
   do i=1,sz
   Rout(i)=Pout(i)/R(i)
   Rout_2s(i)=Pout_2s(i)/R(i)
   Rout_3s(i)=Pout_3s(i)/R(i)
   Rout_3p(i)=Pout_3p(i)/R(i)
   Rout_4s(i)=Pout_4s(i)/R(i)
   Rout_3d(i)=Pout_3d(i)/R(i)
   Rout_4p(i)=Pout_4p(i)/R(i)
   Rout_5s(i)=Pout_5s(i)/R(i)
   Rout_4d(i)=Pout_4d(i)/R(i)
   Rout_5p(i)=Pout_5p(i)/R(i)
   Rout_6s(i)=Pout_6s(i)/R(i)
   Rout_4f(i)=Pout_4f(i)/R(i)
   Rout_5d(i)=Pout_5d(i)/R(i)
   Rout_6p(i)=Pout_6p(i)/R(i)
   Rout_7s(i)=Pout_7s(i)/R(i)
   Rout_5f(i)=Pout_5f(i)/R(i)
   Rout_6d(i)=Pout_6d(i)/R(i)
   end do
  Enkk =
N1s*( Intk1(Rout,R)+Intk2(Rout,R,0) )+N2s*( Intk1(Rout_2s,R)+Intk2(Ro
ut_2s,R,0) )+N2p*( Intk1(Rout_2p,R)+Intk2(Rout_2p,R,1) )+N3s*( Intk1(
Rout_3s,R)+Intk2(Rout_3s,R,0) )+N3p*( Intk1(Rout_3p,R)+Intk2(Rout_3p,
R,1) )+N4s*( Intk1(Rout_4s,R)+Intk2(Rout_4s,R,0) )+N3d*( Intk1(Rout_3
d,R)+Intk2(Rout_3d,R,2) )+N4p*( Intk1(Rout_4p,R)+Intk2(Rout_4p,R,1) )
+N5s*( Intk1(Rout_5s,R)+Intk2(Rout_5s,R,0) )+N4d*( Intk1(Rout_4d,R)+I
ntk2(Rout_4d,R,2) )+N5p*( Intk1(Rout_5p,R)+Intk2(Rout_5p,R,1) )+N6s*(
Intk1(Rout_6s,R)+Intk2(Rout_6s,R,0))+N5d*(Intk1(Rout_5d,R)+Intk2(Rout
_5d,R,2))+N4f*(Intk1(Rout_4f,R)+Intk2(Rout_4f,R,3))+N6p*(Intk1(Rout_6
p,R)+Intk2(Rout_6p,R,1))+N7s*( Intk1(Rout_7s,R)+Intk2(Rout_7s,R,0) )+
N6d*( Intk1(Rout_6d,R)+Intk2(Rout_6d,R,2) )+N5f*( Intk1(Rout_5f,R)+In
tk2(Rout_5f,R,3) )
  total = Enkk+Env(Nout,R,z)+Enxc(Nout,R,Addxc)+Enee(Nout,R,z)
end
```

```
real function Env(n,r,z)!(外位能)
  implicit none
  integer i,z
  real ,parameter ::pi=3.1416
  integer ,parameter ::d=400
  integer ,parameter ::sz=399
  real bdd
  real n(sz),r(sz)
  Env=0.0
  bdd=10.0
 do i=1,sz-1
    Env = Env-( z*r(i)*n(i)+z*r(i+1)*n(i+1) )*4*pi*0.5*(bdd/d)
 end do
end function

real function Enxc(n,r,x)!Exc
  implicit none
  integer i
  real ,parameter ::pi=3.1416
  integer ,parameter ::d=400
  integer ,parameter ::sz=399
  real hg,bdd
  real n(sz),r(sz),x(sz,sz)
  bdd=10.0
  hg=bdd/d
  Enxc=0.0
 do i=1,sz-1

Enxc=Enxc+4*pi*(  x(i,i)*n(i)*(r(i)**2)+x(i+1,i+1)*n(i+1)*(r(i+1)**2)
  )*0.5*(bdd/d)
 end do
end function
real function Enee(n,r,z)!Eee
  implicit none
  integer i,j,z
  real ,parameter ::pi=3.1416
  integer ,parameter ::d=400
  integer ,parameter ::sz=399
  real n(sz),r(sz),bdd
  bdd=10.0
  Enee=0.0
 do i=1,sz
 do j=1,sz
Enee=Enee+0.5*( (( r(i)*r(j) )**2)/( max(i,j)*(bdd/d) ) )*n(i)*n(j)*(
bdd/d)*(bdd/d)*4*pi*4*pi
 end do
 end do
 end

real function Intk1(Rc,r)
  implicit none
  integer i,j
  real ,parameter ::pi=3.1416
  integer ,parameter ::d=400
  integer ,parameter ::sz=399
  real hg
  real n(sz),r(sz),bdd,Rc(sz),inti1
  bdd=10.0
  inti1=0.0
  hg=bdd/d
 do i=2,sz-2
Inti1=Inti1+(( -0.5*Rc(i)*r(i)*( (Rc(i+1)-Rc(i-1))/(r(i+1)-r(i-1))+(R
c(i+1)-2*Rc(i)+Rc(i-1))/(hg**2) )  )+( -0.5*Rc(i+1)*r(i+1)*( (Rc(i+2)
-Rc(i))/(r(i+2)-r(i))+(Rc(i+2)-2*Rc(i+1)+Rc(i))/(hg**2) )  ))*hg*0.5
  !(  Rc(i)*( (-1.0/r(i))*( (Rc(i-1)-Rc(i+1))/(R(i-1)-R(i+1)) ) )-0.5
*( (Rc(i+1)-2*Rc(i)+Rc(i-1))/((bdd/d)**2) )*hg*0.5 ) +
(  Rc(i+1)*( (-1.0/r(i+1))*( (Rc(i)-Rc(i+2))/(R(i)-R(i+2)) ) )-0.5*(
(Rc(i+2)-2*Rc(i+1)+Rc(i))/((bdd/d)**2))*hg*0.5  )
  end do
Intk1=inti1
end function
real function Intk2(Rc,r,l)
  implicit none
  integer i,j,l
  real ,parameter ::pi=3.1416
  integer ,parameter ::d=400
```

```
   integer ,parameter ::sz=399
   real hg,Intk
   real n(sz),r(sz),bdd,Rc(sz)
   bdd=10.0
   intk=0.0
   hg=bdd/d
  do i=1,sz-1
Intk=Intk+(  ( (Rc(i)**2)*l*(l+1)*0.5 )+( (Rc(i+1)**2)*l*(l+1)*0.5 )
 )*0.5*hg
 end do
Intk2=Intk
end function

subroutine self_he()
 use IMSL
 use global
 implicit none
 real ,external :: norl
 ! bool=1  !自洽迴圈用的值
  hg=bdd/d !空隙
 do i=1,sz
  R(i)=hg*i !R 向量 domain 的值
 end do
!以下爲初始值
 do i=1,sz
  Yini(i)=exp(-R(i)/0.53)!*2.0*((2.0/0.53)**(1.5))
 end do
 do i=1,sz
  Nini(i)=2.0*((Yini(i))**2.0)
 end do
!---初始值帶入矩陣
 do i=1,sz
  N(i)=Nini(i)
 end do
call subProcess_7s(0,2)
 do i=1,sz
  Pout(i)=Hn(i)
 end do
```

```
do i=1,sz
  Nout(i)=2.0*((Pout(i)/R(i))**2.0)
 end do
!(以上)----初始直的結果--
!-給 s 初值--
  s=10
  times=0
!-----自洽---He
 do while (s>=10**(-3.0)) !誤差值在 10**-6 如果寫 s=0 跑不完
 times=times+1
 do i=1,sz
  N(i)=Nout(i)
 end do
  call subProcess_7s(0,2)
   do i=1,sz
  Pin(i)=Hn(i)
 end do
 do i=1,sz
  Nin(i)=2.0*((Pin(i)/R(i))**2.0)
 end do
 do i=1,sz
  Nd(i)=Nin(i)-Nout(i)
 end do
 do i=1,sz
  ABNd(i)=ABS(Nd(i))
 end do
  s=maxval(ABNd)
!write(*,*)times
!write(*,*)s
!--------------
do i=1,sz
Nout(i)=Nin(i)
 end do
end do
end

subroutine Li3()
 use IMSL
 use global
 implicit none
!-----自洽---Li
 call sub_self(3,2,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(3,2,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
  write(*,*)"Li(3)"
  write(*,*)times
  write(*,*)total
  write(11,*)"Li(3)"
  write(11,*)times
  write(11,*)total
end

subroutine Be4()
 use IMSL
 use global
 implicit none

!-----自洽---Be
 call sub_self(4,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(4,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
  write(*,*)"Be(4)"
  write(*,*)times
  write(*,*)total
 write(11,*)"Be(4)"
  write(11,*)times
  write(11,*)total
end

subroutine B5()
 use IMSL
 use global
 implicit none
!----B 自洽 -------
call sub_self(5,2,2,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(5,2,2,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
  write(*,*)"B(5)"
  write(*,*)times
  write(*,*)total
 write(11,*)"B(5)"
  write(11,*)times
  write(11,*)total
end

subroutine C6()
 use IMSL
 use global
 implicit none
!---------------C
call sub_self(6,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(6,2,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
  write(*,*)"C(6)"
  write(*,*)times
  write(*,*)total
 write(11,*)"C(6)"
  write(11,*)times
  write(11,*)total
  end
!-------N----


subroutine N7()
 use IMSL
 use global
 implicit none
call sub_self(7,2,2,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(7,2,2,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
  write(*,*)"N(7)"
  write(*,*)times
  write(*,*)total
 write(11,*)"N(7)"
  write(11,*)times
  write(11,*)total
end

subroutine O8()
```

```
 use IMSL
 use global
 implicit none
!-----O----
call sub_self(8,2,2,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(8,2,2,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
   write(*,*)"O(8)"
   write(*,*)times
   write(*,*)total
 write(11,*)"O(8)"
  write(11,*)times
  write(11,*)total
end

subroutine F9()
 use IMSL
 use global
 implicit none
!----F------
call sub_self(9,2,2,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(9,2,2,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
   write(*,*)"F(9)"
   write(*,*)times
   write(*,*)total
 write(11,*)"F(9)"
  write(11,*)times
  write(11,*)total
end

subroutine Ne10()
 use IMSL
 use global
 implicit none
!------Ne-----

 call sub_self(10,2,2,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(10,2,2,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
   write(*,*)"Ne(10)"
   write(*,*)times
   write(*,*)total
 write(11,*)"Ne(10)"
  write(11,*)times
  write(11,*)total
end

subroutine Na11()
 use IMSL
 use global
 implicit none
!---Na---
call sub_self(11,2,2,6,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(11,2,2,6,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Na(11)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Na(11)"
  write(11,*)times
  write(11,*)total
end

subroutine Mg12()
 use IMSL
 use global
 implicit none
!---Mg---
call sub_self(12,2,2,6,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(12,2,2,6,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Mg(12)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Mg(12)"
  write(11,*)times
  write(11,*)total
end

subroutine AL13()
```

```
 use IMSL
 use global
 implicit none
!----AL----
call sub_self(13,2,2,6,2,1,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(13,2,2,6,2,1,0,0,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"AL(13)"
 write(*,*)times
 write(*,*)total
 write(11,*)"AL(13)"
  write(11,*)times
  write(11,*)total
end

subroutine SI14()
 use IMSL
 use global
 implicit none
!----SI----
call sub_self(14,2,2,6,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(14,2,2,6,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"SI(14)"
 write(*,*)times
 write(*,*)total
 write(11,*)"SI(14)"
  write(11,*)times
  write(11,*)total
end

subroutine P15()
 use IMSL
 use global
 implicit none
!----P----
call sub_self(15,2,2,6,2,3,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(15,2,2,6,2,3,0,0,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"P(15)"
 write(*,*)times
 write(*,*)total
 write(11,*)"P(15)"
  write(11,*)times
  write(11,*)total
end

subroutine S16()
 use IMSL
 use global
 implicit none
!----S----
call sub_self(16,2,2,6,2,4,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(16,2,6,2,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"S(16)"
 write(*,*)times
 write(*,*)total
 write(11,*)"S(16)"
  write(11,*)times
  write(11,*)total
end

subroutine CL17()
 use IMSL
 use global
 implicit none
!----CL----
call sub_self(17,2,2,6,2,5,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(17,2,2,6,2,5,0,0,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"CL(17)"
 write(*,*)times
 write(*,*)total
 write(11,*)"CL(17)"
  write(11,*)times
  write(11,*)total
end

subroutine Ar18()
 use IMSL
 use global
 implicit none
!----Ar----
call sub_self(18,2,2,6,2,6,0,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(18,2,2,6,2,6,0,0,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Ar(18)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Ar(18)"
  write(11,*)times
  write(11,*)total
end

subroutine K19()
 use IMSL
 use global
 implicit none
!-----K----
call sub_self(19,2,2,6,2,6,1,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(19,2,2,6,2,6,1,0,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"K(19)"
 write(*,*)times
 write(*,*)total
 write(11,*)"K(19)"
  write(11,*)times
  write(11,*)total
end

subroutine Ca20()
 use IMSL
 use global
 implicit none
!-----Ca-----
call sub_self(20,2,2,6,2,6,2,0,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(20,2,2,6,2,6,2,0,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Ca(20)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Ca(20)"
  write(11,*)times
  write(11,*)total
end

subroutine Sc21()
 use IMSL
 use global
 implicit none
!-----Sc-----
call sub_self(21,2,2,6,2,6,2,1,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(21,2,2,6,2,6,2,1,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Sc(21)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Sc(21)"
  write(11,*)times
  write(11,*)total
end

subroutine Ti22()
 use IMSL
 use global
 implicit none
!-----Ti-----
call sub_self(22,2,2,6,2,6,2,2,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(22,2,2,6,2,6,2,2,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Ti(22)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Ti(22)"
  write(11,*)times
  write(11,*)total
end

subroutine V23()
 use IMSL
 use global
 implicit none
!-----V-----
call sub_self(23,2,2,6,2,6,2,3,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(23,2,2,6,2,6,2,3,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"V(23)"
 write(*,*)times
 write(*,*)total
 write(11,*)"V(23)"
  write(11,*)times
  write(11,*)total
end

subroutine Cr24()
 use IMSL
 use global
 implicit none
!-----Cr----
call sub_self(24,2,2,6,2,6,1,5,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(24,2,2,6,2,6,1,5,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Cr(24)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Cr(24)"
  write(11,*)times
  write(11,*)total
end

subroutine Mn25()
 use IMSL
 use global
 implicit none
!-----Mn-----
call sub_self(25,2,2,6,2,6,2,5,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(25,2,2,6,2,6,2,5,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Mn(25)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Mn(25)"
```

```
  write(11,*)times
  write(11,*)total
end

subroutine Fe26()
 use IMSL
 use global
 implicit none
!-----Fe-----
call sub_self(26,2,2,6,2,6,2,6,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(26,2,2,6,2,6,2,6,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Fe(26)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Fe(26)"
  write(11,*)times
  write(11,*)total
end

subroutine Co27()
 use IMSL
 use global
 implicit none
!-----Co-----
call sub_self(27,2,2,6,2,6,2,7,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(27,2,2,6,2,6,2,7,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Co(27)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Co(27)"
  write(11,*)times
  write(11,*)total
end

subroutine Ni28()
 use IMSL
 use global
 implicit none
```

```
!-----Ni-----
call sub_self(28,2,2,6,2,6,2,8,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(28,2,2,6,2,6,2,8,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Ni(28)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Ni(28)"
  write(11,*)times
  write(11,*)total
end

subroutine Cu29()
 use IMSL
 use global
 implicit none
!-----Cu-----
call sub_self(29,2,2,6,2,6,1,10,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(29,2,2,6,2,6,1,10,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Cu(29)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Cu(29)"
  write(11,*)times
  write(11,*)total
end

subroutine Zn30()
 use IMSL
 use global
 implicit none
!-----Zn-----
call sub_self(30,2,2,6,2,6,2,10,0,0,0,0,0,0,0,0,0,0,0)
call total_energy(30,2,2,6,2,6,2,10,0,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Zn(30)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Zn(30)"
  write(11,*)times
  write(11,*)total
end

subroutine Ga31()
 use IMSL
 use global
 implicit none
!-----Ga-----
call sub_self(31,2,2,6,2,6,2,10,1,0,0,0,0,0,0,0,0,0,0)
call total_energy(31,2,2,6,2,6,2,10,1,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Ga(31)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Ga(31)"
  write(11,*)times
  write(11,*)total
end

subroutine Ge32()
 use IMSL
 use global
 implicit none
!-----Ge-----
call sub_self(32,2,2,6,2,6,2,10,2,0,0,0,0,0,0,0,0,0,0)
call total_energy(32,2,2,6,2,6,2,10,2,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Ge(32)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Ge(32)"
  write(11,*)times
  write(11,*)total
end

subroutine As33()
 use IMSL
 use global
 implicit none
!-----As-----
```

```
call sub_self(33,2,2,6,2,6,2,10,3,0,0,0,0,0,0,0,0,0,0)
call total_energy(33,2,2,6,2,6,2,10,3,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"As(33)"
 write(*,*)times
 write(*,*)total
 write(11,*)"As(33)"
  write(11,*)times
  write(11,*)total
end

subroutine Se34()
 use IMSL
 use global
 implicit none
!-----Se-----
call sub_self(34,2,2,6,2,6,2,10,4,0,0,0,0,0,0,0,0,0,0)
call total_energy(34,2,2,6,2,6,2,10,4,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Se(34)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Se(34)"
  write(11,*)times
  write(11,*)total
end

subroutine Br35()
 use IMSL
 use global
 implicit none
!-----Br-----
call sub_self(35,2,2,6,2,6,2,10,5,0,0,0,0,0,0,0,0,0,0)
call total_energy(35,2,2,6,2,6,2,10,5,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Br(35)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Br(35)"
  write(11,*)times
  write(11,*)total
```

```
end

subroutine Kr36()
 use IMSL
 use global
 implicit none

!-----Kr-----
call sub_self(36,2,2,6,2,6,2,10,6,0,0,0,0,0,0,0,0,0,0)
call total_energy(36,2,2,6,2,6,2,10,6,0,0,0,0,0,0,0,0,0,0)
 write(*,*)"Kr(36)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Kr(36)"
  write(11,*)times
  write(11,*)total
end

subroutine Rb37()
 use IMSL
 use global
 implicit none
!-----Rb-----
call sub_self(37,2,2,6,2,6,2,10,6,1,0,0,0,0,0,0,0,0,0)
call total_energy(37,2,2,6,2,6,2,10,6,1,0,0,0,0,0,0,0,0,0)
 write(*,*)"Rb(37)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Rb(37)"
  write(11,*)times
  write(11,*)total
end

subroutine Sr38()
 use IMSL
 use global
 implicit none
!-----Sr-----
```

```
call sub_self(38,2,2,6,2,6,2,10,6,2,0,0,0,0,0,0,0,0,0)
call total_energy(38,2,2,6,2,6,2,10,6,2,0,0,0,0,0,0,0,0,0)
 write(*,*)"Sr(38)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Sr(38)"
  write(11,*)times
  write(11,*)total
end

subroutine Y39()
 use IMSL
 use global
 implicit none

!-----Y-----
call sub_self(39,2,2,6,2,6,2,10,6,2,1,0,0,0,0,0,0,0,0)
call total_energy(39,2,2,6,2,6,2,10,6,2,1,0,0,0,0,0,0,0,0)
 write(*,*)"Y(39)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Y(39)"
  write(11,*)times
  write(11,*)total
end

subroutine Zr40()
 use IMSL
 use global
 implicit none
!-----Zr-----
call sub_self(40,2,2,6,2,6,2,10,6,2,2,0,0,0,0,0,0,0,0)
call total_energy(40,2,2,6,2,6,2,10,6,2,2,0,0,0,0,0,0,0,0)
 write(*,*)"Zr(40)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Zr(40)"
  write(11,*)times
  write(11,*)total
end

subroutine Nb41()
 use IMSL
 use global
 implicit none
!-----Nb-----
call sub_self(41,2,2,6,2,6,2,10,6,1,4,0,0,0,0,0,0,0,0)
call total_energy(41,2,2,6,2,6,2,10,6,1,4,0,0,0,0,0,0,0,0)
 write(*,*)"Nb(41)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Nb(41)"
  write(11,*)times
  write(11,*)total
end

subroutine Mo42()
 use IMSL
 use global
 implicit none
!-----Mo-----
call sub_self(42,2,2,6,2,6,2,10,6,1,5,0,0,0,0,0,0,0,0)
call total_energy(42,2,2,6,2,6,2,10,6,1,5,0,0,0,0,0,0,0,0)
 write(*,*)"Mo(42)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Mo(42)"
  write(11,*)times
  write(11,*)total
end

subroutine Tc43()
 use IMSL
 use global
 implicit none
!-----Tc-----
```

```
call sub_self(43,2,2,6,2,6,2,10,6,1,6,0,0,0,0,0,0,0,0)
call total_energy(43,2,2,6,2,6,2,10,6,1,6,0,0,0,0,0,0,0,0)
 write(*,*)"Tc(43)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Tc(43)"
  write(11,*)times
  write(11,*)total
end

subroutine Ru44()
 use IMSL
 use global
 implicit none
!-----Ru-----
call sub_self(44,2,2,6,2,6,2,10,6,1,7,0,0,0,0,0,0,0,0)
call total_energy(44,2,2,6,2,6,2,10,6,1,7,0,0,0,0,0,0,0,0)
 write(*,*)"Ru(44)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Ru(44)"
  write(11,*)times
  write(11,*)total
end

subroutine Rh45()
 use IMSL
 use global
 implicit none
!-----Rh-----
call sub_self(45,2,2,6,2,6,2,10,6,1,8,0,0,0,0,0,0,0,0)
call total_energy(45,2,2,6,2,6,2,10,6,1,8,0,0,0,0,0,0,0,0)
 write(*,*)"Rh(45)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Rh(45)"
  write(11,*)times
  write(11,*)total
```

```
end

subroutine Pb46()
 use IMSL
 use global
 implicit none
!-----Pb-----
call sub_self(46,2,2,6,2,6,2,10,6,0,10,0,0,0,0,0,0,0,0)
call total_energy(46,2,2,6,2,6,2,10,6,0,10,0,0,0,0,0,0,0,0)
 write(*,*)"Pd(46)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Pd(46)"
  write(11,*)times
  write(11,*)total
end

subroutine Ag47()
 use IMSL
 use global
 implicit none
!-----Ag-----
call sub_self(47,2,2,6,2,6,2,10,6,1,10,0,0,0,0,0,0,0,0)
call total_energy(47,2,2,6,2,6,2,10,6,1,10,0,0,0,0,0,0,0,0)
 write(*,*)"Ag(47)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Ag(47)"
  write(11,*)times
  write(11,*)total
end

subroutine Cd48()
 use IMSL
 use global
 implicit none
!-----Cd-----
call sub_self(48,2,2,6,2,6,2,10,6,2,10,0,0,0,0,0,0,0,0)
call total_energy(48,2,2,6,2,6,2,10,6,2,10,0,0,0,0,0,0,0,0)
 write(*,*)"Cd(48)"
 write(*,*)times
 write(*,*)total
 write(11,*)"Cd(48)"
  write(11,*)times
  write(11,*)total
end

subroutine In49()
 use IMSL
 use global
 implicit none
!----In---
call sub_self(49,2,2,6,2,6,2,10,6,2,10,1,0,0,0,0,0,0,0)
call total_energy(49,2,2,6,2,6,2,10,6,2,10,1,0,0,0,0,0,0,0)
write(*,*)"In(49)"
write(*,*)times
write(*,*)total
 write(11,*)"In(49)"
  write(11,*)times
  write(11,*)total
end

subroutine Sn50()
 use IMSL
 use global
 implicit none
!----Sn---
call sub_self(50,2,2,6,2,6,2,10,6,2,10,2,0,0,0,0,0,0,0)
call total_energy(50,2,2,6,2,6,2,10,6,2,10,2,0,0,0,0,0,0,0)
write(*,*)"Sn(50)"
write(*,*)times
write(*,*)total
 write(11,*)"Sn(50)"
  write(11,*)times
  write(11,*)total
end
```

```
subroutine Sb51()
 use IMSL
 use global
 implicit none
!----Sb---
call sub_self(51,2,2,6,2,6,2,10,6,2,10,3,0,0,0,0,0,0,0)
call total_energy(51,2,2,6,2,6,2,10,6,2,10,3,0,0,0,0,0,0,0)
write(*,*)"Sb(51)"
write(*,*)times
write(*,*)total
 write(11,*)"Sb(51)"
  write(11,*)times
  write(11,*)total
end

subroutine Te52()
 use IMSL
 use global
 implicit none
!----Te---
call sub_self(52,2,2,6,2,6,2,10,6,2,10,4,0,0,0,0,0,0,0)
call total_energy(52,2,2,6,2,6,2,10,6,2,10,4,0,0,0,0,0,0,0)
write(*,*)"Te(52)"
write(*,*)times
write(*,*)total
 write(11,*)"Te(52)"
  write(11,*)times
  write(11,*)total
end

subroutine I53()
 use IMSL
 use global
 implicit none
!----I---
call sub_self(53,2,2,6,2,6,2,10,6,2,10,5,0,0,0,0,0,0,0)
call total_energy(53,2,2,6,2,6,2,10,6,2,10,5,0,0,0,0,0,0,0)
```

```
write(*,*)"I(53)"
write(*,*)times
write(*,*)total
 write(11,*)"I(53)"
  write(11,*)times
  write(11,*)total
 end

subroutine Xe54()
 use IMSL
 use global
 implicit none
!----Xe---
call sub_self(54,2,2,6,2,6,2,10,6,2,10,6,0,0,0,0,0,0,0)
call total_energy(54,2,2,6,2,6,2,10,6,2,10,6,0,0,0,0,0,0,0)
write(*,*)"Xe(54)"
write(*,*)times
write(*,*)total
 write(11,*)"Xe(54)"
  write(11,*)times
  write(11,*)total
end

subroutine Cs55()
 use IMSL
 use global
 implicit none
!----Cs---
call sub_self(55,2,2,6,2,6,2,10,6,2,10,6,1,0,0,0,0,0,0)
call total_energy(55,2,2,6,2,6,2,10,6,2,10,6,1,0,0,0,0,0,0)
write(*,*)"Cs(55)"
write(*,*)times
write(*,*)total
 write(11,*)"Cs(55)"
  write(11,*)times
  write(11,*)total
end
```

```
subroutine Ba56()
 use IMSL
 use global
 implicit none
!----Ba---
call sub_self(56,2,2,6,2,6,2,10,6,2,10,6,2,0,0,0,0,0,0)
call total_energy(56,2,2,6,2,6,2,10,6,2,10,6,2,0,0,0,0,0,0)
write(*,*)"Ba(56)"
write(*,*)times
write(*,*)total
 write(11,*)"Ba(56)"
  write(11,*)times
  write(11,*)total
end

subroutine La57()
 use IMSL
 use global
 implicit none
!----La----
call sub_self(57,2,2,6,2,6,2,10,6,2,10,6,2,1,0,0,0,0,0)
call total_energy(57,2,2,6,2,6,2,10,6,2,10,6,2,1,0,0,0,0,0)
write(*,*)"La(57)"
write(*,*)times
write(*,*)total
 write(11,*)"La(57)"
  write(11,*)times
  write(11,*)total
end

subroutine Ce58()
 use IMSL
 use global
 implicit none
!----Ce----
call sub_self(58,2,2,6,2,6,2,10,6,2,10,6,2,0,2,0,0,0,0)
call total_energy(58,2,2,6,2,6,2,10,6,2,10,6,2,0,2,0,0,0,0)
write(*,*)"Ce(58)"
write(*,*)times
write(*,*)total
 write(11,*)"Ce(58)"
  write(11,*)times
  write(11,*)total
end

subroutine Pr59()
 use IMSL
 use global
 implicit none
!----Pr----
call sub_self(59,2,2,6,2,6,2,10,6,2,10,6,2,0,3,0,0,0,0)
call total_energy(59,2,2,6,2,6,2,10,6,2,10,6,2,0,3,0,0,0,0)
write(*,*)"Pr(59)"
write(*,*)times
write(*,*)total
 write(11,*)"Pr(59)"
  write(11,*)times
  write(11,*)total
end

subroutine Nd60()
 use IMSL
 use global
 implicit none
!----Nd----
call sub_self(60,2,2,6,2,6,2,10,6,2,10,6,2,0,4,0,0,0,0)
call total_energy(60,2,2,6,2,6,2,10,6,2,10,6,2,0,4,0,0,0,0)
write(*,*)"Nd(60)"
write(*,*)times
write(*,*)total
 write(11,*)"Nd(60)"
  write(11,*)times
  write(11,*)total
end

subroutine Pm61()
```

```
 use IMSL
 use global
 implicit none
!----Pm----
call sub_self(61,2,2,6,2,6,2,10,6,2,10,6,2,0,5,0,0,0,0)
call total_energy(61,2,2,6,2,6,2,10,6,2,10,6,2,0,5,0,0,0,0)
write(*,*)"Pm(61)"
write(*,*)times
write(*,*)total
 write(11,*)"Pm(61)"
  write(11,*)times
  write(11,*)total
end

subroutine Sm62()
 use IMSL
 use global
 implicit none
!----Sm----
call sub_self(62,2,2,6,2,6,2,10,6,2,10,6,2,0,6,0,0,0,0)
call total_energy(62,2,2,6,2,6,2,10,6,2,10,6,2,0,6,0,0,0,0)
write(*,*)"Sm(62)"
write(*,*)times
write(*,*)total
 write(11,*)"Sm(62)"
  write(11,*)times
  write(11,*)total
end

subroutine Eu63()
 use IMSL
 use global
 implicit none
!----En----
call sub_self(63,2,2,6,2,6,2,10,6,2,10,6,2,0,7,0,0,0,0)
call total_energy(63,2,2,6,2,6,2,10,6,2,10,6,2,0,7,0,0,0,0)
write(*,*)"En(63)"
write(*,*)times
write(*,*)total
 write(11,*)"En(63)"
  write(11,*)times
  write(11,*)total
end

subroutine Gd64()
 use IMSL
 use global
 implicit none
!----Gd----
call sub_self(64,2,2,6,2,6,2,10,6,2,10,6,2,1,7,0,0,0,0)
call total_energy(64,2,2,6,2,6,2,10,6,2,10,6,2,1,7,0,0,0,0)
write(*,*)"Gd(64)"
write(*,*)times
write(*,*)total
 write(11,*)"Gd(64)"
  write(11,*)times
  write(11,*)total
end

subroutine Tb65()
 use IMSL
 use global
 implicit none
!----Tb----
call sub_self(65,2,2,6,2,6,2,10,6,2,10,6,2,0,9,0,0,0,0)
call total_energy(65,2,2,6,2,6,2,10,6,2,10,6,2,0,9,0,0,0,0)
write(*,*)"Tb(65)"
write(*,*)times
write(*,*)total
 write(11,*)"Tb(65)"
  write(11,*)times
  write(11,*)total
end

subroutine Dy66()
 use IMSL
 use global
 implicit none
!----Dy----
call sub_self(66,2,2,6,2,6,2,10,6,2,10,6,2,0,10,0,0,0,0)
call total_energy(66,2,2,6,2,6,2,10,6,2,10,6,2,0,10,0,0,0,0)
write(*,*)"Dy(66)"
write(*,*)times
write(*,*)total
 write(11,*)"Dy(66)"
  write(11,*)times
  write(11,*)total
end

subroutine Ho67()
 use IMSL
 use global
 implicit none
!----Ho----
call sub_self(67,2,2,6,2,6,2,10,6,2,10,6,2,0,11,0,0,0,0)
call total_energy(67,2,2,6,2,6,2,10,6,2,10,6,2,0,11,0,0,0,0)
write(*,*)"Ho(67)"
write(*,*)times
write(*,*)total
 write(11,*)"Ho(67)"
  write(11,*)times
  write(11,*)total
end

subroutine Er68()
 use IMSL
 use global
 implicit none
!----Er----
call sub_self(68,2,2,6,2,6,2,10,6,2,10,6,2,0,12,0,0,0,0)
call total_energy(68,2,2,6,2,6,2,10,6,2,10,6,2,0,12,0,0,0,0)
write(*,*)"Er(68)"
write(*,*)times
write(*,*)total
```

```
 write(11,*)"Er(68)"
  write(11,*)times
  write(11,*)total
end

subroutine Tm69()
 use IMSL
 use global
 implicit none
!----Tm----
call sub_self(69,2,2,6,2,6,2,10,6,2,10,6,2,0,13,0,0,0,0)
call total_energy(69,2,2,6,2,6,2,10,6,2,10,6,2,0,13,0,0,0,0)
write(*,*)"Tm(69)"
write(*,*)times
write(*,*)total
 write(11,*)"Tm(69)"
  write(11,*)times
  write(11,*)total
end

subroutine Yb70()
 use IMSL
 use global
 implicit none
!----Yb----
call sub_self(70,2,2,6,2,6,2,10,6,2,10,6,2,0,14,0,0,0,0)
call total_energy(70,2,2,6,2,6,2,10,6,2,10,6,2,0,14,0,0,0,0)
write(*,*)"Yb(70)"
write(*,*)times
write(*,*)total
 write(11,*)"Yb(70)"
  write(11,*)times
  write(11,*)total
end

subroutine Lu71()
 use IMSL
 use global
 implicit none
!----Lu----
call sub_self(71,2,2,6,2,6,2,10,6,2,10,6,2,1,14,0,0,0,0)
call total_energy(71,2,2,6,2,6,2,10,6,2,10,6,2,1,14,0,0,0,0)
write(*,*)"Lu(71)"
write(*,*)times
write(*,*)total
 write(11,*)"Lu(71)"
  write(11,*)times
  write(11,*)total
end

subroutine Hf72()
 use IMSL
 use global
 implicit none
!----Hf----
call sub_self(72,2,2,6,2,6,2,10,6,2,10,6,2,2,14,0,0,0,0)
call total_energy(72,2,2,6,2,6,2,10,6,2,10,6,2,2,14,0,0,0,0)
write(*,*)"Hf(72)"
write(*,*)times
write(*,*)total
 write(11,*)"Hf(72)"
  write(11,*)times
  write(11,*)total
end

subroutine Ta73()
 use IMSL
 use global
 implicit none
!----Ta----
call sub_self(73,2,2,6,2,6,2,10,6,2,10,6,2,3,14,0,0,0,0)
call total_energy(73,2,2,6,2,6,2,10,6,2,10,6,2,3,14,0,0,0,0)
write(*,*)"Ta(73)"
write(*,*)times
write(*,*)total
 write(11,*)"Ta(73)"
```

```
  write(11,*)times
  write(11,*)total
end

subroutine W74()
 use IMSL
 use global
 implicit none
!----W----
call sub_self(74,2,2,6,2,6,2,10,6,2,10,6,2,4,14,0,0,0,0)
call total_energy(74,2,2,6,2,6,2,10,6,2,10,6,2,4,14,0,0,0,0)
write(*,*)"W(74)"
write(*,*)times
write(*,*)total
 write(11,*)"W(74)"
  write(11,*)times
  write(11,*)total
end

subroutine Re75()
 use IMSL
 use global
 implicit none
!----Re----
call sub_self(75,2,2,6,2,6,2,10,6,2,10,6,2,5,14,0,0,0,0)
call total_energy(75,2,2,6,2,6,2,10,6,2,10,6,2,5,14,0,0,0,0)
write(*,*)"Re(75)"
write(*,*)times
write(*,*)total
 write(11,*)"Re(75)"
  write(11,*)times
  write(11,*)total
end

subroutine Os76()
 use IMSL
 use global
 implicit none
```

```
!----Os----
call sub_self(76,2,2,6,2,6,2,10,6,2,10,6,2,6,14,0,0,0,0)
call total_energy(76,2,2,6,2,6,2,10,6,2,10,6,2,6,14,0,0,0,0)
write(*,*)"Os(76)"
write(*,*)times
write(*,*)total
 write(11,*)"Os(76)"
  write(11,*)times
  write(11,*)total
end

subroutine Ir77()
 use IMSL
 use global
 implicit none
!----Ir----
call sub_self(77,2,2,6,2,6,2,10,6,2,10,6,2,7,14,0,0,0,0)
call total_energy(77,2,2,6,2,6,2,10,6,2,10,6,2,7,14,0,0,0,0)
write(*,*)"Ir(77)"
write(*,*)times
write(*,*)total
 write(11,*)"Ir(77)"
  write(11,*)times
  write(11,*)total
end

subroutine Pt78()
 use IMSL
 use global
 implicit none
!----pt----
call sub_self(78,2,2,6,2,6,2,10,6,2,10,6,1,9,14,0,0,0,0)
call total_energy(78,2,2,6,2,6,2,10,6,2,10,6,1,9,14,0,0,0,0)
write(*,*)"Pt(78)"
write(*,*)times
write(*,*)total
 write(11,*)"Pt(78)"
  write(11,*)times
  write(11,*)total
end

subroutine Au79()
 use IMSL
 use global
 implicit none
!----Au----
call sub_self(79,2,2,6,2,6,2,10,6,2,10,6,1,10,14,0,0,0,0)
call total_energy(79,2,2,6,2,6,2,10,6,2,10,6,1,10,14,0,0,0,0)
write(*,*)"Au(79)"
write(*,*)times
write(*,*)total
 write(11,*)"Au(79)"
  write(11,*)times
  write(11,*)total
end

subroutine Hg80()
 use IMSL
 use global
 implicit none
!----Hg----
call sub_self(80,2,2,6,2,6,2,10,6,2,10,6,2,10,14,0,0,0,0)
call total_energy(80,2,2,6,2,6,2,10,6,2,10,6,2,10,14,0,0,0,0)
write(*,*)"Hg(80)"
write(*,*)times
write(*,*)total
 write(11,*)"Hg(80)"
  write(11,*)times
  write(11,*)total
end

subroutine Tl81()
 use IMSL
 use global
 implicit none
!----Tl----
```

```
call sub_self(81,2,2,6,2,6,2,10,6,2,10,6,2,10,14,1,0,0,0)
call total_energy(81,2,2,6,2,6,2,10,6,2,10,6,2,10,14,1,0,0,0)
write(*,*)"Tl(81)"
write(*,*)times
write(*,*)total
 write(11,*)"Tl(81)"
  write(11,*)times
  write(11,*)total
end

subroutine Pb82()
 use IMSL
 use global
 implicit none
!----Pb----
call sub_self(82,2,2,6,2,6,2,10,6,2,10,6,2,10,14,2,0,0,0)
call total_energy(82,2,2,6,2,6,2,10,6,2,10,6,2,10,14,2,0,0,0)
write(*,*)"Pb(82)"
write(*,*)times
write(*,*)total
 write(11,*)"Pb(82)"
  write(11,*)times
  write(11,*)total
end

subroutine Bi83()
 use IMSL
 use global
 implicit none
!----Bi----
call sub_self(83,2,2,6,2,6,2,10,6,2,10,6,2,10,14,3,0,0,0)
call total_energy(83,2,2,6,2,6,2,10,6,2,10,6,2,10,14,3,0,0,0)
write(*,*)"Bi(83)"
write(*,*)times
write(*,*)total
 write(11,*)"Bi(83)"
  write(11,*)times
  write(11,*)total
```

```
end

subroutine Po84()
 use IMSL
 use global
 implicit none
!----Po----
call sub_self(84,2,2,6,2,6,2,10,6,2,10,6,2,10,14,4,0,0,0)
call total_energy(84,2,2,6,2,6,2,10,6,2,10,6,2,10,14,4,0,0,0)
write(*,*)"Po(84)"
write(*,*)times
write(*,*)total
 write(11,*)"Po(84)"
  write(11,*)times
  write(11,*)total
end

subroutine At85()
 use IMSL
 use global
 implicit none
!----At----
call sub_self(85,2,2,6,2,6,2,10,6,2,10,6,2,10,14,5,0,0,0)
call total_energy(85,2,2,6,2,6,2,10,6,2,10,6,2,10,14,5,0,0,0)
write(*,*)"At(85)"
write(*,*)times
write(*,*)total
 write(11,*)"At(85)"
  write(11,*)times
  write(11,*)total
end

subroutine Rn86()
 use IMSL
 use global
 implicit none
!----Rn----
call sub_self(86,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,0,0,0)
call total_energy(86,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,0,0,0)
write(*,*)"Rn(86)"
write(*,*)times
write(*,*)total
 write(11,*)"Rn(86)"
  write(11,*)times
  write(11,*)total
end

subroutine Fr87()
 use IMSL
 use global
 implicit none
!-----Fr---
call sub_self(87,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,1,0,0)
call total_energy(87,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,1,0,0)
write(*,*)"Fr(87)"
write(*,*)times
write(*,*)total
 write(11,*)"Fr(87)"
  write(11,*)times
  write(11,*)total
end

subroutine Ra88()
 use IMSL
 use global
 implicit none
!-----Ra---
call sub_self(88,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,0)
call total_energy(88,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,0)
write(*,*)"Ra(88)"
write(*,*)times
write(*,*)total
 write(11,*)"Ra(88)"
  write(11,*)times
  write(11,*)total
end
```

```
subroutine Ac89()
 use IMSL
 use global
 implicit none
!-----Ac---
call sub_self(89,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,1,0)
call total_energy(89,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,1,0)
write(*,*)"Ac(89)"
write(*,*)times
write(*,*)total
 write(11,*)"Ac(89)"
  write(11,*)times
  write(11,*)total
end

subroutine Th90()
 use IMSL
 use global
 implicit none
!-----Th---
call sub_self(90,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,2,0)
call total_energy(90,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,2,0)
write(*,*)"Th(90)"
write(*,*)times
write(*,*)total
 write(11,*)"Th(90)"
  write(11,*)times
  write(11,*)total
end

subroutine Pa91()
 use IMSL
 use global
 implicit none
!-----Pa---
call sub_self(91,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,1,2)
call total_energy(91,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,1,2)
```

```
write(*,*)"Pa(91)"
write(*,*)times
write(*,*)total
 write(11,*)"Pa(91)"
  write(11,*)times
  write(11,*)total
end

subroutine U92()
 use IMSL
 use global
 implicit none
!-----U---
call sub_self(92,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,1,3)
call total_energy(92,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,1,3)
write(*,*)"U(92)"
write(*,*)times
write(*,*)total
 write(11,*)"U(92)"
  write(11,*)times
  write(11,*)total
end

subroutine Np93()
 use IMSL
 use global
 implicit none
!-----Np---
call sub_self(93,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,5)
call total_energy(93,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,5)
write(*,*)"Np(93)"
write(*,*)times
write(*,*)total
 write(11,*)"Np(93)"
  write(11,*)times
  write(11,*)total
end
```

```
subroutine Pu94()
 use IMSL
 use global
 implicit none
!----Pu----
call sub_self(94,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,6)
call total_energy(94,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,6)
write(*,*)"Pu(94)"
write(*,*)times
write(*,*)total
 write(11,*)"Pu(94)"
  write(11,*)times
  write(11,*)total
end

subroutine Am95()
 use IMSL
 use global
 implicit none
!----Am----
call sub_self(95,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,7)
call total_energy(95,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,7)
write(*,*)"Am(95)"
write(*,*)times
write(*,*)total
 write(11,*)"Am(95)"
  write(11,*)times
  write(11,*)total
end

subroutine Cm96()
 use IMSL
 use global
 implicit none
!-----Cm---
call sub_self(96,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,1,7)
call total_energy(96,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,1,7)
write(*,*)"Cm(96)"
write(*,*)times
write(*,*)total
 write(11,*)"Cm(96)"
  write(11,*)times
  write(11,*)total
end

subroutine Bk97()
 use IMSL
 use global
 implicit none
!-----Bk---
call sub_self(97,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,1,8)
call total_energy(97,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,1,8)
write(*,*)"Bk(97)"
write(*,*)times
write(*,*)total
 write(11,*)"Bk(97)"
  write(11,*)times
  write(11,*)total
end

subroutine Cf98()
 use IMSL
 use global
 implicit none
!----Cf----
call sub_self(98,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,10)
call total_energy(98,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,10)
write(*,*)"Cf(98)"
write(*,*)times
write(*,*)total
 write(11,*)"Cf(98)"
  write(11,*)times
  write(11,*)total
end

subroutine Es99()
```

```
 use IMSL
 use global
 implicit none
!----Es----
call sub_self(99,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,11)
call total_energy(99,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,11)
write(*,*)"Es(99)"
write(*,*)times
write(*,*)total
 write(11,*)"Es(99)"
  write(11,*)times
  write(11,*)total
end

subroutine Fm100()
 use IMSL
 use global
 implicit none
!----Fm----
call sub_self(100,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,12)
call total_energy(100,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,12)
write(*,*)"Fm(100)"
write(*,*)times
write(*,*)total
 write(11,*)"Fm(100)"
  write(11,*)times
  write(11,*)total
end

subroutine Md101()
 use IMSL
 use global
 implicit none
!----Md----
call sub_self(101,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,13)
call total_energy(101,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,13)
write(*,*)"Md(101)"
write(*,*)times
write(*,*)total
 write(11,*)"Md(101)"
  write(11,*)times
  write(11,*)total
end

subroutine No102()
 use IMSL
 use global
 implicit none
!----No----
call sub_self(102,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,14)
call total_energy(102,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,0,14)
write(*,*)"No(102)"
write(*,*)times
write(*,*)total
 write(11,*)"No(102)"
  write(11,*)times
  write(11,*)total
end

subroutine Lr103()
 use IMSL
 use global
 implicit none
!----Lr----
call sub_self(103,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,1,14)
call total_energy(103,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,1,14)
write(*,*)"Lr(103)"
write(*,*)times
write(*,*)total
 write(11,*)"Lr(103)"
  write(11,*)times
  write(11,*)total
end

subroutine Rf104()
 use IMSL
 use global
 implicit none
!----Rf----
call sub_self(104,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,2,14)
call total_energy(104,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,2,14)
write(*,*)"Rf(104)"
write(*,*)times
write(*,*)total
 write(11,*)"Rf(104)"
  write(11,*)times
  write(11,*)total
end

subroutine Db105()
 use IMSL
 use global
 implicit none
!----Db----
call sub_self(105,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,3,14)
call total_energy(105,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,3,14)
write(*,*)"Db(105)"
write(*,*)times
write(*,*)total
 write(11,*)"Db(105)"
  write(11,*)times
  write(11,*)total
end

subroutine Sg106()
 use IMSL
 use global
 implicit none
!----Sg----
call sub_self(106,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,4,14)
call total_energy(106,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,4,14)
write(*,*)"Sg(106)"
write(*,*)times
write(*,*)total
```

```
 write(11,*)"Sg(106)"
  write(11,*)times
  write(11,*)total
end

subroutine Bh107()
 use IMSL
 use global
 implicit none
!----Bh----
call sub_self(107,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,5,14)
call total_energy(107,2,2,6,2,6,2,10,6,2,10,6,2,10,14,6,2,5,14)
write(*,*)"Bh(107)"
write(*,*)times
write(*,*)total
 write(11,*)"Bh(107)"
  write(11,*)times
  write(11,*)total
end
subroutine
sub_self(Zn,Nt1s,Nt2s,Nt2p,Nt3s,Nt3p,Nt4s,Nt3d,Nt4p,Nt5s,Nt4d,Nt5p,Nt
6s,Nt5d,Nt4f,Nt6p,Nt7s,Nt6d,Nt5f)
 use IMSL
 use global
 implicit none
 real ,external :: norl
 integer
Zn,Nt1s,Nt2s,Nt3s,Nt4s,Nt5s,Nt6s,Nt7s,Nt2p,Nt3p,Nt4p,Nt5p,Nt6p,Nt3d,N
t4d,Nt5d,Nt6d,Nt4f,Nt5f

 do i=1,sz
N(i)=Nin(i)
 end do

 call subprocess_7s(0,Zn)
  do i=1,sz
 Pout(i)=Hn(i)!/norl(R,H)
 Pout_2s(i)=Hn_2s(i)!/norl(R,H_2s)
 Pout_3s(i)=Hn_3s(i)!/norl(R,H_3s)
 Pout_4s(i)=Hn_4s(i)!/norl(R,H_4s)
 Pout_5s(i)=Hn_5s(i)!/norl(R,H_5s)
 Pout_6s(i)=Hn_6s(i)!/norl(R,H_6s)
 Pout_7s(i)=Hn_7s(i)!/norl(R,H_7s)
  end do

 call subprocess_6p(1,Zn)
  do i=1,sz
 Pout_2p(i)=Hn_2p(i)!/norl(R,H_2p)
 Pout_3p(i)=Hn_3p(i)!/norl(R,H_3p)
 Pout_4p(i)=Hn_4p(i)!/norl(R,H_4p)
 Pout_5p(i)=Hn_5p(i)!/norl(R,H_5p)
 Pout_6p(i)=Hn_6p(i)!/norl(R,H_6p)
  end do
 call subprocess_6d(2,Zn)
  do i=1,sz
 Pout_3d(i)=Hn_3d(i)!/norl(R,H_3d)
 Pout_4d(i)=Hn_4d(i)!/norl(R,H_4d)
 Pout_5d(i)=Hn_5d(i)!/norl(R,H_5d)
 Pout_6d(i)=Hn_6d(i)!/norl(R,H_6d)
  end do
 call subprocess_5f(3,Zn)
  do i=1,sz
 Pout_4f(i)=Hn_4f(i)!/norl(R,H_4f)
 Pout_5f(i)=Hn_5f(i)!/norl(R,H_5f)
  end do

 do i=1,sz

Nout(i)=Nt1s*((Pout(i)/R(i))**2.0)+Nt2s*((Pout_2s(i)/R(i))**2.0)+Nt2p
*((Pout_2p(i)/R(i))**2.0)+Nt3s*((Pout_3s(i)/R(i))**2.0)+Nt3p*((Pout_3
p(i)/R(i))**2.0)+Nt4s*((Pout_4s(i)/R(i))**2.0)+Nt3d*((Pout_3d(i)/R(i)
)**2.0)+Nt4p*((Pout_4p(i)/R(i))**2.0)+Nt5s*((Pout_5s(i)/R(i))**2.0)+N
t4d*((Pout_4d(i)/R(i))**2)+Nt5p*((Pout_5p(i)/R(i))**2)+Nt6s*((Pout_6s
(i)/R(i))**2)+Nt5d*((Pout_5d(i)/R(i))**2)+Nt4f*((Pout_4f(i)/R(i))**2)
+Nt6p*((Pout_6p(i)/R(i))**2)+Nt7s*((Pout_7s(i)/R(i))**2)+Nt6d*((Pout_
6d(i)/R(i))**2)+Nt5f*((Pout_5f(i)/R(i))**2)
```

```
 end do
 do i=1,sz
  Nd(i)=Nin(i)-Nout(i)
 end do

 do i=1,sz
  ABNd(i)=ABS(Nd(i))
 end do
  s=sum(ABNd)
  times=0
   do while (s>=10**(-2.0))

  times=times+1
do i=1,sz
  N(i)=Nout(i)
 end do
  call subprocess_7s(0,Zn)
  do i=1,sz
 Pin(i)=Hn(i)!/norl(R,H)
 Pin_2s(i)=Hn_2s(i)!/norl(R,H_2s)
 Pin_3s(i)=Hn_3s(i)!/norl(R,H_3s)
 Pin_4s(i)=Hn_4s(i)!/norl(R,H_4s)
 Pin_5s(i)=Hn_5s(i)!/norl(R,H_5s)
 Pin_6s(i)=Hn_6s(i)!/norl(R,H_6s)
 Pin_7s(i)=Hn_7s(i)!/norl(R,H_7s)
 end do
 call subprocess_6p(1,Zn)
  do i=1,sz
 Pin_2p(i)=Hn_2p(i)!/norl(R,H_2p)
 Pin_3p(i)=Hn_3p(i)!/norl(R,H_3p)
 Pin_4p(i)=Hn_4p(i)!/norl(R,H_4p)
 Pin_5p(i)=Hn_5p(i)!/norl(R,H_5p)
 Pin_6p(i)=Hn_6p(i)!/norl(R,H_6p)
  end do

 call subprocess_6d(2,Zn)
  do i=1,sz
 Pin_3d(i)=Hn_3d(i)!/norl(R,H_3d)
 Pin_4d(i)=Hn_4d(i)!/norl(R,H_4d)
 Pin_5d(i)=Hn_5d(i)!/norl(R,H_5d)
 Pin_6d(i)=Hn_6d(i)!/norl(R,H_6d)
  end do
 call subprocess_5f(3,Zn)
  do i=1,sz
 Pin_4f(i)=Hn_4f(i)!/norl(R,H_4f)
 Pin_5f(i)=Hn_5f(i)!/norl(R,H_5f)
  end do
 do i=1,sz

Nin(i)=Nt1s*((Pin(i)/R(i))**2.0)+Nt2s*((Pin_2s(i)/R(i))**2.0)+Nt2p*((
Pin_2p(i)/R(i))**2.0)+Nt3s*((Pin_3s(i)/R(i))**2.0)+Nt3p*((Pin_3p(i)/R
(i))**2.0)+Nt4s*((Pin_4s(i)/R(i))**2.0)+Nt3d*((Pin_3d(i)/R(i))**2.0)+
Nt4p*((Pin_4p(i)/R(i))**2.0)+Nt5s*((Pin_5s(i)/R(i))**2.0)+Nt4d*((Pin_
4d(i)/R(i))**2)+Nt5p*((Pin_5p(i)/R(i))**2)+Nt6s*((Pin_6s(i)/R(i))**2.
0)+Nt5d*((Pin_5d(i)/R(i))**2)+Nt4f*((Pin_4f(i)/R(i))**2)+Nt6p*((Pin_6
p(i)/R(i))**2)+Nt7s*((Pin_7s(i)/R(i))**2)+Nt6d*((Pin_6d(i)/R(i))**2)+
Nt5f*((Pin_5f(i)/R(i))**2)

 end do



 do i=1,sz
  Nd(i)=Nin(i)-Nout(i)
 end do

 do i=1,sz
  ABNd(i)=ABS(Nd(i))
 end do
  s=maxval(ABNd)
!--------------
do i=1,sz
 Nout(i)=Nin(i)
 end do
end do
end
```

```
real function norl(R,Pe)
use IMSL
implicit none
integer ,parameter ::sz=399
integer ,parameter ::d=400
real ,parameter :: bdd=10.0
real ,parameter :: pi=3.1416
real R(sz),Pe(sz),k
integer ::i
k=0.0
do i=1,sz-1
!k=
k+( (R(i)**2)*((Pe(i)/R(i))**2)+(R(i+1)**2)*((Pe(i+1)/R(i+1))**2)  )*
(bdd/d)*0.5
 k=k+(Pe(i)**2+Pe(i+1)**2)*(bdd/d)*0.5
end do

norl=k**(0.5)

end function
```