1998 年度 卒業論文

リカージョン法による分子軌道計算のプログラムの開発

9420002

木村・齋藤研　安藤 泰夫

電気通信大学 電子工学科 電子デバイス講座

指導教官 齋藤 理一郎 助教授

提出日 平成 11 年 2 月 10 日

【概要】
本研究では、カーボンナノチューブのような原子の数が非常に多い分子の分子軌道を計算するプログラムを開発することを目的としている。分子軌道の計算には、第一原理計算とタイトバインディングハミルトニアンによる近似計算がある。タイトバインディングによる計算は、計算時間を短縮するのに非常に有用である。ここでは、タイトバインディングによる近似計算を利用して分子軌道を計算している。タイトバインディングによる分子軌道計算は、分子軌道法を利用した計算であるが、分子軌道法では原子の数が多い巨大な分子では、非常に時間がかかる計算になる。分子軌道法では、分子を構成する原子の価電子帯の原子軌道数を次元とする行列が作られるが、計算時間はこの行列の次元の 3 乗に比例する。それでは、カーボンナノチューブのような巨大な分子では計算に時間がかかりすぎる。そこで、計算時間を原子軌道の数の 1 乗に比例する程度まで短縮することが可能なリカージョン法を用いた分子軌道計算プログラムを完成させることを目的として、リカージョン法に用いるハミルトニアン行列から状態密度を計算するプログラムを作成した。また、リカージョン法による計算結果を評価するために、分子軌道法による分子軌道計算プログラムを作成した。計算はフラーレンの C_{60} について行った。タイトバインディングパラメーターは、フラーレンのために開発された原子間の距離の関数になっているものを採用した。分子軌道法の計算の結果は、同じパラメーターを使用した計算結果と一致する結果を得た。また、リカージョン法のチェーンモデルの構成情報から分子の状態密度を計算するプログラムを開発したが、チェーンモデルを作るプログラムについては開発することができなかった。分子軌道法による C_{60} の分子軌道の計算速度から考察すると、カーボンナノチューブのような巨大な分子では、リカージョン法による分子軌道計算が非常に有用であると思われる。

もくじ

第 1 章

序論

1.1 背景

炭素の構造をあらわす理論の研究として、第一原理を基礎とした計算方法と、タイトバインディングハミルトニアンを利用した近似計算がある。第一原理を利用した計算はタイトバインディングによる計算と比べて非常に多くの計算努力を必要とするので、タイトバインディングによる計算は有用であると考えられる。炭素には、グラファイト、ダイヤモンド、フラーレン、カーボンナノチューブという同素体がある。岡田氏らによる博士論文 [3] で、フラーレンに関して良い結果を得たタイトバインディングパラメーターが開発された。図 1.1(a) はその論文の結果で、C_{60} 分子の縮重度である。

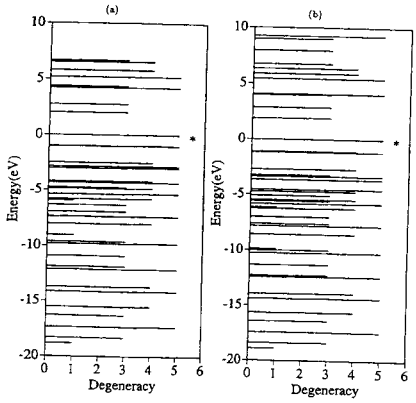


図 1.1 (a) タイトバインディング計算 (b) 第一原理の C_{60} 分子の縮重度 [1]

$4(\boldsymbol{eV})$ 未満において、第一原理計算の結果と良く一致している。このパラメーターによって、フラーレンやカーボンナノチューブの分子軌道計算に良い結果が得られることが期待できる。

[1] ファイル名：./eps/haikei.eps

しかし、カーボンナノチューブのように原子の数が 1000 以上あるような分子の分子軌道を計算しようとしたとき、タイトバインディングによる分子軌道法を用いると計算に時間がかかる。以下に計算の対象となる炭素の同素体を説明する。

1.1.1 フラーレン

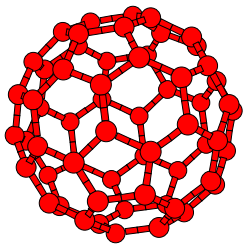


図 1.2 C_{60} 分子 [2]

1985 年に Kroto、 Smalley の実験により発見された C_{60} 分子がフラーレンに関するはじめての発見である。フラーレンは C_{60} 以外にも、 C_{70} や C_{76}、 C_{84} といった大きな粒子が発見されている。すべてのフラーレンは 5 員環と 6 員環で構成される閉殻構造の多面体であり、その数はオイラーの定理により知ることがでる。オイラーの定理によると、五員環は常に 12 個ある。

1.1.2 カーボンナノチューブ

カーボンナノチューブは、グラファイトを丸めてチューブ状にしたような形をしている。一般に、その先端は開いているわけでなく閉じている。フラーレンの構造と関連してくるが、先端が閉じているということは、オイラーの定理からチューブの先端部には五員環があることがわかる。カーボンナノチューブには、１つのチューブのみの単層ナノチューブ (図 1.2) と幾つかのチューブが入れ子になっている多層ナノチューブ (図 1.3) がある。多層ナノチューブの物性はバルクのグラファイトのそれと大差ないのに比べ、単層ナノチューブは六員環ネットワークのトポロジーによって支配される特異な性質を持つ。

カーボンナノチューブはその構造によって、金属にも半導体にもなる。この特性を利用すれば、幾つかの構造のカーボンナノチューブをつなげ、ナノメートルサイズのデバイスを作ることが可能になるであろう。

[2] ファイル名：./eps/C60-1.eps

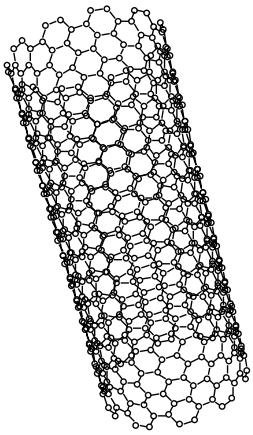


図 1.3　単層ナノチューブ [3]

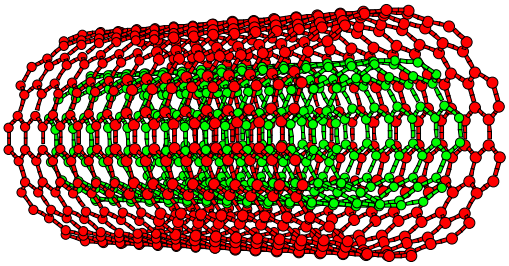


図 1.4　多層ナノチューブ [4]

ナノチューブの特徴には

- 直径がナノメートルのサイズである。
- 円筒で多くはらせん構造である。
- サイズが分子とバルク (分子の数量が極めて多い) の中間である。
- 構成する原子の位置が既知である。
- 柔軟構造である。
- 熱的に安定構造である。

[3]ファイル名：./eps/10-10-1.ps

[4]ファイル名：./eps/ireko.ps

- 自己支持構造 (六員環を形成する炭素原子の結合はダイヤモンドより強い) である。

があります。以上のようなユニークさを持つため、広範囲な研究者に興味をもたれ、研究対象になっている。もし、ナノチューブを自由につくることができれば、半導体の主力が炭素に移る日がくるであろう。

1.1.3 分子軌道計算

分子の電子状態を計算する方法として、分子軌道法がある。この分子軌道法を使って、炭素で構成される分子の電子状態の計算をした場合、原子の数が 1000 程度までの計算ならば、汎用コンピューターを用いても可能であるが、 10000 といった原子の数が多くなった場合には、分子軌道法によるトランスファー行列とオーバーラップ行列をつくって計算する方法では、非常に時間のかかる計算となる。特に炭素の場合、価電子帯の $2s, 2p_x, 2p_y, 2p_z$ の 4 つの軌道だけを考えて、原子の数の 4 倍の大きさのハミルトニアン行列をつくる。求めるエネルギー固有値は、この行列の大きさ N の 3 乗に比例する。この計算を N の 1 乗に比例する程度まで速くすることが可能な計算方法をオーダー N 法と呼び、その一つとしてリカージョン法がある。この方法は原子の数が多くなっても非常に速く計算ができるようになるため、この方法を用いれば、汎用コンピューターでも大きな分子の電子状態を計算することが可能になる。原子の数が 10000 あるとすると、計算時間をオーダー N^3 からオーダー N することは計算時間の短縮に非常に有用である。

1.2 目的

分子軌道法で、距離の関数であるタイトバインディングパラメーターを用いたプログラムを開発する。これによって、任意の座標での分子でも分子軌道の計算を可能とする。しかしながら、原子数が多くなった場合、分子軌道法を用いて計算すると、非常に時間がかかってしまう。

そこで、炭素で構成される分子における状態密度を、原子の数が多い場合でも汎用コンピューターで計算ができるように、プログラムをオーダー N に改良する。方法はリカージョン法を用いる。計算は、フラーレンの C_{60} について行う。。

第 2 章

計算方法

2.1 タイトバインディングパラメーターを用いた分子軌道法による計算

2.1.1 原理

分子軌道は、原子のそばでは原子のポテンシャルエネルギーを感じるはずであるので、近似的に分子軌道 Ψ_i を原子軌道 φ_i の線形結合で表す。

$$\Psi_i = \sum_{j=1}^{N} C_{ij}\varphi_j(r), (i = 1, \cdots, N) \tag{2.1}$$

この時、分子軌道のエネルギーは

$$E_i = \frac{\int \Psi_i^* \mathcal{H} \Psi_i d\boldsymbol{r}}{\int \Psi_i^* \Psi_i d\boldsymbol{r}} \tag{2.2}$$

で表すことができる。ここで、

$$\begin{aligned} \int \Psi_i^* \mathcal{H} \Psi_i d\boldsymbol{r} &= \langle \Psi_i^* | \mathcal{H} | \Psi_i \rangle \\ \int \Psi_i^* \Psi_i d\boldsymbol{r} &= \langle \Psi_i^* | \Psi_i \rangle \end{aligned} \tag{2.3}$$

であらわすと、

$$E_i = \frac{\langle \Psi_i^* | \mathcal{H} | \Psi_i \rangle}{\langle \Psi_i^* | \Psi_i \rangle} \tag{2.4}$$

この式に 2.1 式を代入すると、

$$E_i = \frac{\sum_{j,j'=1}^{N} C_{ij}^* C_{ij'} \langle \varphi_j | \mathcal{H} | \varphi_{j'} \rangle}{\sum_{j,j'=1}^{N} C_{ij}^* C_{ij'} \langle \varphi_j | \varphi_{j'} \rangle} \tag{2.5}$$

ここで、

$$H_{jj'} = \langle \varphi_j \,|\mathcal{H}|\, \varphi_{j'} \rangle \tag{2.6}$$

$$S_{jj'} = \langle \varphi_j | \varphi_{j'} \rangle \tag{2.7}$$

とする。E_i を C_{ij}^* で偏微分すると極小の条件から 0 になる。

式 (2.5) を変形して、

$$\sum_{j'=1}^{N} H_{jj'} C_{ij'} = E_i \sum_{j'=1}^{N} S_{jj'} C_{ij'} \tag{2.8}$$

ここで列ベクトル $\boldsymbol{C}_i = {}^t(C_{i1}, \cdots, C_{iN})$ を定義すれば、

$$H\boldsymbol{C}_i = E_i S \boldsymbol{C}_i \tag{2.9}$$

移項して

$$(H - E_i S)\boldsymbol{C}_i = 0 \tag{2.10}$$

としたとき、もし行列 $(H - E_i S)$ に逆行列が存在したとすると、両辺に逆行列をかけて $\boldsymbol{C}_i = 0$ となり、Ψ_i が分子軌道にならない。したがって、行列 $(H - E_i S)$ には逆行列が存在しない。したがって、以下の式が成り立つ。

$$\boldsymbol{det}\,(H - E_i S) = 0 \tag{2.11}$$

これを永年方程式といい、この行列式を解いてエネルギー固有値 E_i を求める。この作業は、実際にはハウスホルダー法等の行列対角化のプログラムを用いる。

2.1.2 炭素における、タイトバインディング計算

次に 2.1.1 の計算を炭素の場合に応用することを考える。炭素の価電子帯の電子の原子軌道は $2s, 2p_x, 2p_y, 2p_z$ の 4 つである。よって、炭素でのトランスファー行列 H とオーバーラップ行列 S は、この 4 つの原子軌道の原子積分の値で作られる。原子軌道 ϕ_i と ϕ_j の原子積分は、以下の式で計算されるものである。[2]

$$H_{ij} = \int \phi_i^* \mathcal{H} \phi_j d\boldsymbol{r} \tag{2.12}$$

$$S_{ij} = \int \phi_i^* \phi_j d\boldsymbol{r} \tag{2.13}$$

ここでは、行列の要素の値となる原子積分の値は、タイトバインディングパラメーターを用いて計算した。ここで用いたパラメーターは、原子間の距離よる関数になっ

ているもの [3] を使って計算している。また、トランスファー行列、オーバーラップ行列の対角項は、

$$H_{ii} = \int \phi_i^* \mathcal{H} \phi_i d\boldsymbol{r} \tag{2.14}$$

$$S_{ii} = \int \phi_i^* \phi_i d\boldsymbol{r} \tag{2.15}$$

であるが、$2s$ 軌道の時は H_{ii}, S_{ii} はそれぞれ $\varepsilon_{2s}, 1$、$2p$ 軌道の時はそれぞれ $\varepsilon_{2p}, 1$ となる。S_{ii} が 1 となるのは、原子軌道関数が規格化されているからである。

2.1.3 原子積分の計算

炭素原子の原子積分の値は以下の 8 通りある。

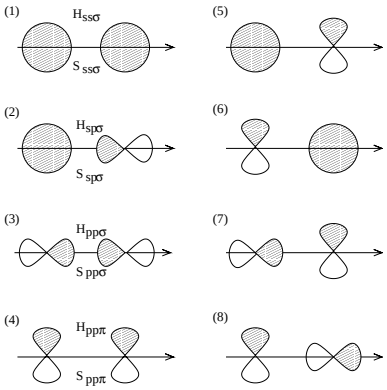


図 2.1 $2s$ 軌道と $2p$ 軌道の原子積分 [1]

このうち 4 つは値が 0 になる。したがって、残りの 4 通りの成分の値をパラメーターにして計算すればよい。実際の値は、この 8 通りの成分をいくつか含む形になったいるので、この 8 通りの成分に分解して計算する。

二次元の場合 (図 2.2)

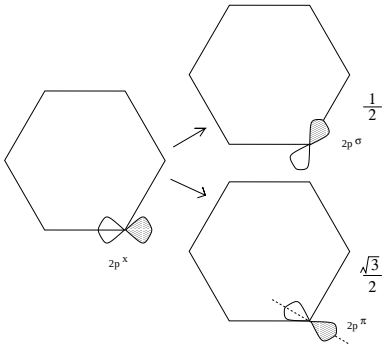


図 2.2 $2p$ 軌道の成分の分解 [2]

$2s$ 軌道は方向を持たないため、$2s$ 軌道同士の原子積分はパラメーターをそのまま適

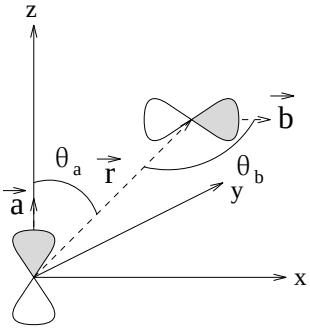
[1] ファイル名：./eps/model8.eps

[2] ファイル名：./eps/model9.eps

用できるが、$2p$ 軌道は方向を持つために、成分を分解して計算をする。図のようになっていた場合以下のように計算して値を求めている。

a と b の 2 つの $2p$ 軌道を $\boldsymbol{r}$ に垂直な成分 (π 成分) と $\boldsymbol{r}$ 方向の成分 (σ 成分) に分解する。それぞれの成分の大きさをかけたものに π 成分ならば、 π 成分のパラメーター、σ 成分ならば σ 成分のパラメーターをかけて原子積分の値とする。

実際の分子では三次元であるので、三次元の場合について説明する。



であらわされる。ここで、$2p_x$ どうし、$2p_y$ どうし、$2p_z$ どうしの場合、$\cos\theta_a = -\cos\theta_b$ であるので σ 成分は

$$H_{p_i p_j \sigma} = H_{pp\sigma}(-\cos^2\theta_a) \tag{2.19}$$

となる。π 成分は、図 2.4 のようにねじれはなく同じ方向を向いてるので、

$$H_{p_i p_j \pi} = H_{pp\pi}(1-\cos^2\theta_a) \tag{2.20}$$

となる。

2.1.4 タイトバインディングパラメーター

タイトバインディングパラメーターは、原子積分の値をパラメーターにしたものである。ここで、使用した炭素原子の 2 原子間のパラメーターは、原子間の距離の関数になっている以下の式で求められるものである。
$Q(r)$ はカットオフ関数、R は機能関数、v, u はパラメーター (表 2.1) で、a と b は２つの原子それぞれをあらわす。

トランスファー行列 H のパラメーターの関数 [3]

$$t_{ab}^{ss\sigma}(r) = -7\sqrt{v_a^s v_b^s}R^{ss\sigma}(4r/(r_s^a + r_s^b))Q(r) \tag{2.21}$$

$$t_{ab}^{sp\sigma}(r) = -7\sqrt{v_a^s v_b^{s\sigma}}R^{sp\sigma}(4r/(r_s^a + r_{p\sigma}^b))Q(r) \tag{2.22}$$

$$t_{ab}^{pp\sigma}(r) = -7\sqrt{v_a^{p\sigma} v_b^{p\sigma}}R^{pp\sigma}(4r/(r_{p\sigma}^a + r_{p\sigma}^b))Q(r) \tag{2.23}$$

$$t_{ab}^{pp\pi}(r) = -7\sqrt{v_a^{p\pi} v_b^{p\pi}}R^{pp\pi}(4r/(r_{p\pi}^a + r_{p\pi}^b))Q(r) \tag{2.24}$$

オーバーラップ行列 S のパラメーターの関数 [3]

$$s_{ab}^{ss\sigma}(r) = 7\sqrt{u_a^s u_b^s}R^{ss\sigma}(4r/(r_s^a + r_s^b))Q(r) \tag{2.25}$$

$$s_{ab}^{sp\sigma}(r) = 7\sqrt{u_a^s u_b^{s\sigma}}R^{sp\sigma}(4r/(r_s^a + r_{p\sigma}^b))Q(r) \tag{2.26}$$

$$s_{ab}^{pp\sigma}(r) = 7\sqrt{u_a^{p\sigma} u_b^{p\sigma}}R^{pp\sigma}(4r/(r_{p\sigma}^a + r_{p\sigma}^b))Q(r) \tag{2.27}$$

$$s_{ab}^{pp\pi}(r) = 7\sqrt{u_a^{p\pi} u_b^{p\pi}}R^{pp\pi}(4r/(r_{p\pi}^a + r_{p\pi}^b))Q(r) \tag{2.28}$$

$r^s, r^{p\sigma}, r^{p\pi}$ は原子軌道の種類による距離のパラメーターで、それぞれ $r^s = 0.620(\text{Å})$, $r^{p\sigma} = 0.810(\text{Å})$, $r^{p\pi} = 0.550(\text{Å})$ である。

表 2.1: パラメーター [3]

v	$(\boldsymbol{eV})$	u	
v^s	6.6	u_s	$\frac{1}{7}$
$v^{p\sigma}$	4.3	$u_{p\sigma}$	$\frac{1}{7}$
$v^{p\pi}$	4.5	$u_{p\pi}$	$\frac{1}{7}$
ε_{2s}	-7.0		

ε_{2s} は、 $\varepsilon_{2p} = 0$ としたときの相対的な値

カットオフ関数 [3]

$$Q(r) = \left\{ \begin{array}{ll} 1 & (for\ r < r_1) \\ \frac{1}{2}(1 + \cos\frac{\pi(r-r_1)}{r_2-r_1}) & (for\ r_1 < r < r_2) \\ 0 & (for\ r_2 < r) \end{array} \right\} \tag{2.29}$$

カットオフパラメーターは $r_1 = 3.6(\text{Å})$、 $r_2 = 4.0(\text{Å})$ である。

機能関数 [3]

$$R^{ss\sigma}(x) = e^{-x}(1 + x + \frac{x^2}{3}) \tag{2.30}$$

$$R^{sp\sigma}(x) = e^{-x}(x + \frac{x^2}{3}) \tag{2.31}$$

$$R^{pp\sigma}(x) = e^{-x}(-1 + x + \frac{x^2}{3}) \tag{2.32}$$

$$R^{pp\pi}(x) = e^{-x}(1 + x + \frac{x^2}{3}) \tag{2.33}$$

この式に原子間の距離 r を代入し求められた、 $(\ t_{ab}^{ss\sigma}, t_{ab}^{sp\sigma}, t_{ab}^{pp\sigma}, t_{ab}^{pp\pi})$ $(s_{ab}^{ss\sigma}, s_{ab}^{sp\sigma}, s_{ab}^{pp\sigma}, s_{ab}^{pp\pi})$ がタイトバインディングパラメータとなる。

2.1.5 ハミルトニアン行列の作成

作成したプログラムでやっていることは、波動関数に重なりがある 2 つの原子の座標を取り出して、それぞれの原子の $2s, 2p_x, 2p_y, 2p_z$ でつくられる 4×4 のハミルトニアン小行列の要素を計算し、その 4×4 の小行列を分子全体のハミルトニアン行列の対応する場所に入れてハミルトニアン行列をつくっている。そのできた行列を、DEIGAB というトランスファー行列とオーバーラップ行列を入力すると固有値と固有ベクトルの計算をするプログラムに入力して、値を得る。 DEIGAB については 2.1.7 で説明する。

2 つの原子 i, j のハミルトニアン小行列

$$H_{ij} = \begin{pmatrix} H_{s_i s_j} & H_{s_i p x_j} & H_{s_i p y_j} & H_{s_i p z_j} \\ H_{p x_i s_j} & H_{p x_i p x_j} & H_{p x_i p y_j} & H_{p x_i p z_j} \\ H_{p y_i s_j} & H_{p y_i p x_j} & H_{p y_i p y_j} & H_{p y_i p z_j} \\ H_{p z_i s_j} & H_{p z_i p x_j} & H_{p z_i p y_j} & H_{p z_i p z_j} \end{pmatrix}$$

分子全体のハミルトニアン行列は小行列 H_{ij} を用いて

$$H = \begin{pmatrix} H_{1\ 1} & H_{1\ 2} & H_{1\ 3} & H_{1\ 4} & H_{1\ 5} & H_{1\ 6} & \cdots & H_{1\ 60} \\ H_{2\ 1} & H_{2\ 2} & H_{2\ 3} & H_{2\ 4} & \cdots & \cdots & \cdots & H_{2\ 60} \\ H_{3\ 1} & H_{3\ 2} & H_{3\ 3} & H_{3\ 4} & \cdots & \cdots & \cdots & H_{3\ 60} \\ \vdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ H_{60\ 1} & H_{60\ 2} & H_{60\ 3} & H_{60\ 4} & \cdots & \cdots & \cdots & H_{60\ 60} \end{pmatrix}$$

であらわされる。

2.1.6 行列の要素の計算

- 2 つの原子の座標 $(x_i, y_i, z_i), (x_j, y_j, z_j)$ から原子間の距離 r_{ij} を求める。
- 式 (2.29) で r_{ij} がカットオフディスタンス r_2 より大きいものは、値を 0 とするので計算しない。
- 距離 r_{ij} からタイトバインディングパラメーターを計算する。
- 以下の式から原子積分の値を計算する。

$$H_{s_i s_j} = t^{ss\sigma}(r) \tag{2.34}$$

$$H_{s_i p a_j} = t^{sp\sigma}(r)(\cos\theta) \tag{2.35}$$

$$\cos\theta_a = \frac{a_i - a_j}{r}, \cos\theta_b = \frac{b_i - b_j}{r} \tag{2.36}$$

$$H_{p a_i p a_j} = t^{pp\sigma}(r)(-\cos^2\theta) + t^{pp\pi}(r)(1 - \cos^2\theta) \tag{2.37}$$

$$H_{p a_1 p b_2} = t^{pp\sigma}(r)\cos\theta_a \cos\theta_b + t^{pp\pi}(r)\left(\vec{a} - (\vec{a}\cdot\frac{\vec{r}}{r})\frac{\vec{r}}{r}\right)\cdot\left(\vec{b} - (\vec{b}\cdot\frac{\vec{r}}{r})\frac{\vec{r}}{r}\right) \tag{2.38}$$

r は原子間の距離、 a, b は x, y, z のいずれかで、 $\vec{a}, \vec{b}$ はその x, y, z いずれかの単位ベクトルである。

2.1.7 DEIGAB(付録 p.35) の説明 (固有値、固有ベクトルの計算)

固有値を得るとき、行列の方程式

$$Ax = \lambda Mx \tag{2.39}$$

を解く。A は対称行列、M は正値対称行列である。正値対称行列 M は $M = U^T U$ とコレスキー分解できる。[5] ただし、U は上三角行列 U^T は下三角行列である。$z = Ux$ とおくと、M をコレスキー分解すると式 (2.34) は

$$Ax = \lambda U^T Ux \tag{2.40}$$

となる。コレスキー分解とは、行列 M を $U^T U$ という、2 つの行列に分解するもので、

$$\left\{ \begin{array}{ll} u_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} u_{ik}^2} & (i > 1) \quad u_{11} = \sqrt{m_{11}} \\ u_{ji} = \dfrac{(m_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk})}{l_{ii}} & (i > j) \quad l_{1i} = \frac{a_{1i}}{l_{11}} \end{array} \right\} \tag{2.41}$$

で得られる。$z = Ux$ として、式 (2.35) の両辺に左から $(U^T)^{-1}$ をかけると、

$$(U^T)^{-1} A U^{-1} z = \lambda z \tag{2.42}$$

となる。$B = (U^T)^{-1} A U^{-1}$ とすれば、

$$Bz = \lambda z \tag{2.43}$$

となる。つまり、行列 B の固有値と固有ベクトルを求めればよい [5](p.215)。また、この式の固有ベクトル z はもとの式の固有ベクトル x とは違うが、x は $z = Ux$ に U^{-1} をかけて、$x = U^{-1} z$ で求められる。この、固有値、固有ベクトルの計算は、行列 B の形により計算速度がかわってくる。ここでは、実対称行列であるので、ヤコビ法、ハウスホルダ法・二分法・逆反復法、二分法・同時逆反復法、QL 法、divide and conquer 法がある [5]。計算は行列 B が密な行列であるほど、時間がかかる。M が単位行列であり、A が帯行列の場合には、B が A と同じ帯幅の帯行列となるために、帯行列に対する対角化が可能である。しかし、M が A と同じような帯行列である場合は、B の帯幅は大きくなる。実際に

$$M = \begin{pmatrix} 1 & a & 0 & 0 \\ a & 1 & b & 0 \\ 0 & b & 1 & c \\ 0 & 0 & c & 1 \end{pmatrix} \tag{2.44}$$

をコレスキー分解してみると、

$$U = \begin{pmatrix} 1 & a & 0 & 0 \\ 0 & \sqrt{1-a^2} & b & 0 \\ 0 & 0 & \sqrt{1-b^2} & c \\ 0 & 0 & 0 & \sqrt{1-c^2} \end{pmatrix} \tag{2.45}$$

ここではまだ帯幅は大きくなっていないが、この逆行列は、

$$U^{-1} = \begin{pmatrix} 1 & \frac{-a}{\sqrt{1-a^2}} & \frac{ab}{\sqrt{1-a^2}\sqrt{1-b^2}} & \frac{-abc}{\sqrt{1-a^2}\sqrt{1-b^2}\sqrt{1-c^2}} \\ 0 & \frac{1}{\sqrt{1-a^2}} & \frac{-b}{\sqrt{1-a^2}\sqrt{1-b^2}} & \frac{bc}{\sqrt{1-a^2}\sqrt{1-b^2}\sqrt{1-c^2}} \\ 0 & 0 & \frac{1}{\sqrt{1-b^2}} & \frac{-c}{\sqrt{1-b^2}\sqrt{1-c^2}} \\ 0 & 0 & 0 & \frac{1}{\sqrt{1-c^2}} \end{pmatrix} \tag{2.46}$$

というように、上三角行列で密なものになっている。同様に、$(U^T)^{-}1$ も下三角行列で密になり、$(U^T)^{-1}AU^{-1}$ は密な行列になってしまう。このように行列 B が、コレスキー分解の際に元の行列 A, M よりもかなり密な行列になってしまうために、計算時間を要してしまう。これらの方法では計算時間をオーダー N にすることはできない。このことから、分子軌道法を使った計算方法では計算時間の短縮に限界がある。

2.2 リカージョン法

2.2.1 オーバーラップ行列 $S = 1$ の場合のリカージョン法

- 波動関数を束ねて、リニアチェーンのようにハミルトニアン行列を最初から三重対角化行列にし、計算を速くする。また、つくられるハミルトニアン行列の次元は、対称性を考慮すれば元になる分子の波動関数の数より少なくなる。対称性がなければ、次元は同じとなる。

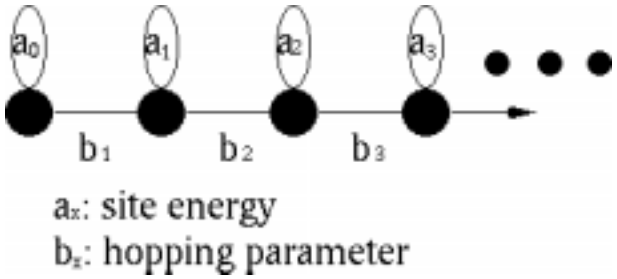


図 2.5 リニアチェーン [5]

新たに構成された波動関数は、連続して直線的に存在する軌道 $\{\boldsymbol{u}_0,\boldsymbol{u}_1,\cdots\}$ と、2 つの実数のパラメーターである $\{a_0,a_1,\cdots\}$ と $\{b_1,b_2,\cdots\}$ で、

$$\begin{cases} \boldsymbol{H}\boldsymbol{u}_n = a_n\boldsymbol{u}_n + b_{n+1}\boldsymbol{u}_{n+1} + b_n\boldsymbol{u}_{n-1} \\ a_n = \boldsymbol{u}_n^\dagger \boldsymbol{S}\boldsymbol{H}\boldsymbol{u}_n \\ b_{n+1}^2 = [(\boldsymbol{H}-a_n)\boldsymbol{u}_n - b_n\boldsymbol{u}_{n-1}]^\dagger \boldsymbol{S}[(\boldsymbol{H}-a_n)\boldsymbol{u}_n - b_n\boldsymbol{u}_{n-1}] \end{cases} \tag{2.47}$$

のようにあらわされる。

図のような、隣接するもの以外は直交化されているモデルをつくることで、三重対角化された以下のようなハミルトニアン行列がつくることが可能になる。

$$\begin{pmatrix} a_0 & b_1 & 0 & 0 & \cdots \\ b_1 & a_1 & b_2 & 0 & \cdots \\ 0 & b_2 & a_2 & b_3 & \cdots \\ 0 & 0 & b_3 & a_3 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

この三重対角化行列で表されるモデルの状態密度 $n_0(E)$ はグリーン関数の虚部により、

$$n_0(E) = -Im(1/E - a_0 - b_1^2/[E - a_1 - b_2^2/(E - a_2\cdots)])/\pi \tag{2.48}$$

[5] ファイル名：./eps/linear-chain.eps

のように簡単な計算式 [1] で表される。高速に計算できることが予測される。ここでの E は微少で一定である虚数部を含んでいる。この虚数部の大きさを変えることで状態密度のピークの幅を変えることができる。

2.2.2 オーバーラップ行列 S がある場合のリカージョン法

最初の波動関数を決める (特定の原子基底である必要はない) その波動関数と重なりがある波動関数の重ね合わせで次の波動関数をつくる。さらにその波動関数から直交化されていない波動関数の重ね合わせで次の波動関数をつくる。それを分子全体に適用する。つくられた新たな波動関数で新たなハミルトニアン行列をつくる。このモデルで新たにつくられた波動関数では隣接する波動関数以外は直交化されているため、三重対角化行列を実現することができる。三重対角化行列の固有値、固有ベクトルの計算時間は N^2 に比例する [5]。

オーバーラップ行列 $\boldsymbol{S}$ の要素は、

$$[S]_{mn} = \int \phi_m^*(r)\phi_n(r)d\tau \tag{2.49}$$

で定義される。
また、ハミルトニアン行列 $\boldsymbol{H}$ は次のように定義される。

$$\mathcal{H}\phi_n(r) = \sum_{m=0}^{\infty} H_{mn}\phi_m(r) \tag{2.50}$$

$$\boldsymbol{H} = [H_{mn}] \tag{2.51}$$

$$\mathcal{H}_{mn} = \int \phi_m^*(r)\mathcal{H}\phi_n(r)\tau \tag{2.52}$$

$$\mathcal{H}_{mn} = [SH]_{mn} \tag{2.53}$$

$$\boldsymbol{H} = \boldsymbol{S}^{-1}[\mathcal{H}_{mn}] \tag{2.54}$$

最初の状態 $\boldsymbol{u}_0$ は任意に決めることができる。 $\boldsymbol{u}_0$ を一般化するために

$$\boldsymbol{u}_0\boldsymbol{S}\boldsymbol{u}_0 = 1 \tag{2.55}$$

とする。その最初の状態 $\boldsymbol{u}_0$ のサイトエネルギーである a_0 は

$$a_0 = \boldsymbol{u}_0^\dagger\boldsymbol{S}\boldsymbol{H}\boldsymbol{u}_0 \tag{2.56}$$

であらわされる。
また、 $b_1\boldsymbol{u}_1$ は

$$b_1\boldsymbol{u}_1 = (\boldsymbol{H} - a_0)\boldsymbol{u}_0 \tag{2.57}$$

であり、 $\boldsymbol{u}_1^\dagger \boldsymbol{S} \boldsymbol{u}_1 = 1$ であるので、

$$b_1^2 = b_1 \boldsymbol{u}_1^\dagger \boldsymbol{S} b_1 \boldsymbol{u}_1 = [(\boldsymbol{H} - a_0)\boldsymbol{u}_0]^\dagger \boldsymbol{S} [(\boldsymbol{H} - a_0)\boldsymbol{u}_0] \tag{2.58}$$

となる。これで b_1 が求まる。 (2.57) 式を b_1 でわると、

$$\boldsymbol{u}_1 = (\boldsymbol{H} - a_0)\boldsymbol{u}_0 / b_1 \tag{2.59}$$

となり、 $\boldsymbol{u}_1$ が求まる。
$n > 1$ での一般的な式は、

$$\boldsymbol{H}\boldsymbol{u}_n = a_n \boldsymbol{u}_n + b_{n+1}\boldsymbol{u}_{n+1} + b_n \boldsymbol{u}_{n-1} \tag{2.60}$$

である。 a_n は

$$a_n = \boldsymbol{u}_n^\dagger \boldsymbol{S} \boldsymbol{H} \boldsymbol{u}_n \tag{2.61}$$

さらに、

$$b_{n+1}\boldsymbol{u}_{n+1} = (\boldsymbol{H} - a_n) - b_n \boldsymbol{u}_{n-1} \tag{2.62}$$

よって、 b_{n+1}^2 は $\boldsymbol{u}_{n+1}^\dagger \boldsymbol{S} \boldsymbol{u}_{n+1} = 1$ であるので、

$$b_{n+1}^2 = [(\boldsymbol{H} - a_n)\boldsymbol{u}_n - b_n \boldsymbol{u}_{n-1}]^\dagger \boldsymbol{S} [(\boldsymbol{H} - a_n)\boldsymbol{u}_n - b_n \boldsymbol{u}_{n-1}] \tag{2.63}$$

となる。したがって、 $\boldsymbol{u}_{n+1}$ は

$$\boldsymbol{u}_{n+1} = [(\boldsymbol{H} - a_n)\boldsymbol{u}_n - b_n \boldsymbol{u}_{n-1}] / b_{n+1} \tag{2.64}$$

である。

2.2.3 正方格子の場合

正方格子の場合で考えてみる。一つの原子に一つの波動関数のみと考え、隣接する原子以外では波動関数が直交化されているものとする。

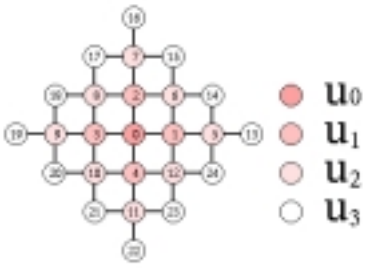


図 2.6 n が 3 までの正方格子 [6]

[6] ファイル名：./eps/rec.eps

まず、最初の波動関数を 0 番の原子にとってみる。この波動関数を u_0 とすると、次の u_1 は u_0 の隣にある四つの波動関数 $\phi_1, \phi_2, \phi_3, \phi_4$ で構成される。 u_2 はさらにその外側にある隣の波動関数 $\phi_5 \sim \phi_{12}$ で構成される。次はさらにその外側の波動関数 $\phi_{13} \sim \phi_{24}$ で構成されることになる。これを、分子全体に適用するまで続ける。このモデルで、 n が 3 までのときの a_n、 b_n を計算してみる。

t はホッピングパラメーターで、この正方格子のモデルではすべての t が等価であると考えて計算式をつくった。

$$t_{ij} =< \phi_i \mid \mathcal{H} \mid \phi_j >= t \tag{2.65}$$

$$\boldsymbol{u}_0 = (\phi_0) \tag{2.66}$$

$$a_0 = \boldsymbol{u}_0 \boldsymbol{S} \boldsymbol{H} \boldsymbol{u}_0 \tag{2.67}$$

n が 1 のとき

$\boldsymbol{u}_1$ を構成する波動関数は $\phi_1, \phi_2, \phi_3, \phi_4$ である。したがって、 $b_1 \boldsymbol{u}_1$ は、

$$b_1 \boldsymbol{u}_1 = \begin{pmatrix} \phi_1 t \\ \phi_2 t \\ \phi_3 t \\ \phi_4 t \end{pmatrix} \tag{2.68}$$

${b_1}^2$ は、

$${b_1}^2 = [(\boldsymbol{H} - a_0)\boldsymbol{u}_0]^\dagger \boldsymbol{S} [(\boldsymbol{H} - a_0)\boldsymbol{u}_0] \tag{2.69}$$

この式に、 $b_1 \boldsymbol{u}_1 = (\boldsymbol{H} - a_0)\boldsymbol{u}_0$ を代入すると、

$${b_1}^2 = (b_1 \boldsymbol{u}_1)^\dagger \boldsymbol{S} (b_1 \boldsymbol{u}_1) \tag{2.70}$$

になる。この正方格子のモデルでは $\boldsymbol{u}_n$ を構成するそれぞれの波動関数が直交しているので、 $\boldsymbol{S}$ は

$$\boldsymbol{S} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{2.71}$$

のようになり、 ${b_1}^2$ の値は、そのそれぞれの要素の 2 乗の合計となる。よって、

$${b_1}^2 = (\phi_1 t, \phi_2 t, \phi_3 t, \phi_4 t) \boldsymbol{S} \begin{pmatrix} \phi_1 t \\ \phi_2 t \\ \phi_3 t \\ \phi_4 t \end{pmatrix} = 4t^2 \tag{2.72}$$

${b_1}^2$ は正値をとるので、

$$b_1 = 2t \tag{2.73}$$

$b_1 \boldsymbol{u}_1$ を上で求めた b_1 で割ると $\boldsymbol{u}_1$ が求められる。

$$\boldsymbol{u}_1 = \begin{pmatrix} \frac{\phi_1}{2} \\ \frac{\phi_2}{2} \\ \frac{\phi_3}{2} \\ \frac{\phi_4}{2} \end{pmatrix} \tag{2.74}$$

$$a_1 = \boldsymbol{u}_1^\dagger \boldsymbol{S}\boldsymbol{H}\boldsymbol{u}_1 = (\frac{\phi_1}{2}, \frac{\phi_2}{2}, \frac{\phi_3}{2}, \frac{\phi_4}{2})\boldsymbol{S}\boldsymbol{H} \begin{pmatrix} \frac{\phi_1}{2} \\ \frac{\phi_2}{2} \\ \frac{\phi_3}{2} \\ \frac{\phi_4}{2} \end{pmatrix} = a_0 \tag{2.75}$$

n が 2 のとき

$\boldsymbol{u}_2$ を構成する波動関数のうち $\phi_6, \phi_8, \phi_{10}, \phi_{12}$ は、 $\boldsymbol{u}_1$ を構成する波動関数の 2 つと重なりをもつ。したがって、ホッピングパラメーターは、その合計であるから次のようになる。

$$b_2\boldsymbol{u}_2 = \begin{pmatrix} \phi_5 \frac{t}{2} \\ \phi_6(\frac{t}{2} + \frac{t}{2}) \\ \phi_7 \frac{t}{2} \\ \phi_8(\frac{t}{2} + \frac{t}{2}) \\ \phi_9 \frac{t}{2} \\ \phi_{10}(\frac{t}{2} + \frac{t}{2}) \\ \phi_{11} \frac{t}{2} \\ \phi_{12}(\frac{t}{2} + \frac{t}{2}) \end{pmatrix} \tag{2.76}$$

${b_2}^2$ は、

$${b_2}^2 = (\phi_5 \frac{t}{2}, \phi_6 t, \phi_7 \frac{t}{2}, \phi_8 t, \phi_9 \frac{t}{2}, \phi_{10} t, \phi_{11} \frac{t}{2}, \phi_{12} t)\boldsymbol{S} \begin{pmatrix} \phi_5 \frac{t}{2} \\ \phi_6 t \\ \phi_7 \frac{t}{2} \\ \phi_8 t \\ \phi_9 \frac{t}{2} \\ \phi_{10} t \\ \phi_{11} \frac{t}{2} \\ \phi_{12} t \end{pmatrix} = 5t^2 \tag{2.77}$$

したがって $\boldsymbol{u}_2$ は、

$$\boldsymbol{u}_2 = \begin{pmatrix} \frac{\phi_5}{2\sqrt{5}} \\ \frac{\phi_6}{\sqrt{5}} \\ \frac{\phi_7}{2\sqrt{5}} \\ \frac{\phi_8}{\sqrt{5}} \\ \frac{\phi_9}{2\sqrt{5}} \\ \frac{\phi_{10}}{\sqrt{5}} \\ \frac{\phi_{11}}{2\sqrt{5}} \\ \frac{\phi_{12}}{\sqrt{5}} \end{pmatrix} \tag{2.78}$$

a_2 は、

$$a_2 = \boldsymbol{u}_2^{\dagger} \boldsymbol{S} \boldsymbol{H} \boldsymbol{u}_2 = a_0 \tag{2.79}$$

n が 3 のとき

$$b_3 \boldsymbol{u}_3 = \begin{pmatrix} \phi_{13}\frac{t}{2\sqrt{5}} \\ \phi_{14}\left(\frac{t}{2\sqrt{5}} + \frac{t}{\sqrt{5}}t\right) \\ \phi_{15}\left(\frac{t}{2\sqrt{5}} + \frac{t}{\sqrt{5}}t\right) \\ \phi_{16}\frac{t}{2\sqrt{5}} \\ \phi_{17}\left(\frac{t}{2\sqrt{5}} + \frac{t}{\sqrt{5}}t\right) \\ \phi_{18}\left(\frac{t}{2\sqrt{5}} + \frac{t}{\sqrt{5}}t\right) \\ \vdots \\ \phi_{24}\left(\frac{t}{2\sqrt{5}} + \frac{t}{\sqrt{5}}t\right) \end{pmatrix} \tag{2.80}$$

$$b_3{}^2 = (\phi_{13}\frac{t}{2\sqrt{5}}, \phi_{14}\frac{3t}{2\sqrt{5}}, \phi_{15}\frac{3t}{2\sqrt{5}}, \cdots, \phi_{24}\frac{3t}{2\sqrt{5}})\boldsymbol{S} \begin{pmatrix} \phi_{13}\frac{t}{2\sqrt{5}} \\ \phi_{14}\frac{3t}{2\sqrt{5}} \\ \phi_{15}\frac{3t}{2\sqrt{5}} \\ \vdots \\ \phi_{24}\frac{3t}{2\sqrt{5}} \end{pmatrix} = \frac{19}{5}t^2 \tag{2.81}$$

$$\boldsymbol{u}_3 = \begin{pmatrix} \phi_{13}\frac{1}{2\sqrt{19}} \\ \phi_{14}\frac{3}{2\sqrt{19}} \\ \phi_{15}\frac{3}{2\sqrt{19}} \\ \vdots \\ \phi_{24}\frac{3}{2\sqrt{19}} \end{pmatrix} \tag{2.82}$$

$$a_3 = \boldsymbol{u}_3^{\dagger} \boldsymbol{S} \boldsymbol{H} \boldsymbol{u}_3 = a_0 \tag{2.83}$$

このように計算していき、チェーンモデルのトランスファー行列をつくる。求めた a_n, b_n を 2.40 式に入れて状態密度が求められる。$t = 1, a_0 = 0$ として、原子が 144 個ある n が 8 までの正方格子で計算した結果を以下に示す。

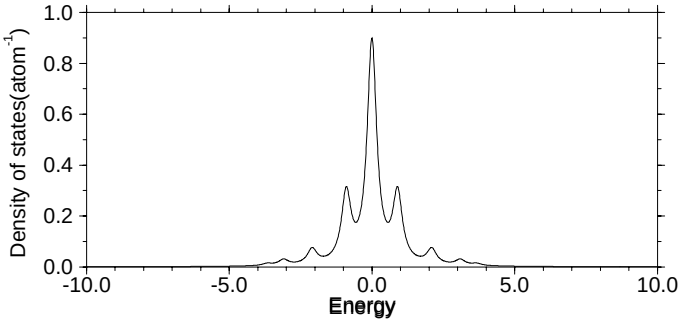


図 2.7　n が 8 までの正方格子の状態密度 [7]

表 2.2: a_n, b_n の値

n	a_n	b_n
0	0.0	
1	0.0	1.00000
2	0.0	2.23606
3	0.0	1.94935
4	0.0	1.90567
5	0.0	1.90727
6	0.0	1.91762
7	0.0	1.92801
8	0.0	1.93669

[7]ファイル名：./eps/seihou.eps

第 3 章

結果及び考察

計算は C_{60} のエネルギー固有値と状態密度について、タイトバインディング計算で行った。また、リカージョン法については計算プログラムが未完成のため、計算の考察をした。

3.1 任意の座標に対するタイトバインディング計算

C_{60} についてエネルギー固有値と固有ベクトルを計算をした。原子の座標から計算をしているが、入力した座標は Tersoff ポテンシャルで最適化されているものを使用した。

C_{60} の分子軌道は 60 ある炭素原子の 240 の価電子帯の原子軌道から構成されるので 240 のエネルギー固有値を持つ。計算結果を見るといくつかエネルギー固有値が縮重している。縮重とは、いくつかのエネルギー固有値が同じ値になっている状態で、重なっているエネルギー固有値の数を縮重度という。 C_{60} では、最大で 5 重の縮重が見られる。エネルギー固有値と縮重度の関係を表したものを図 3.1,3.2 に示す。(a) は 2 章の式 (2.21) から式 (2.33) 及び表 2.1 にかかれた原子間の距離の関数をタイトバインディングパラメーターとして計算したもので、同じタイトバインディングパラメーターを使用して計算された他の論文 [3] の C_{60} のエネルギー固有値の結果と一致した結果を得られた。また、(b) は、 [2] で用いられているパラメーターが定数であるものを使用してカットオフディスタンス 1.6(Å) で計算したものである。

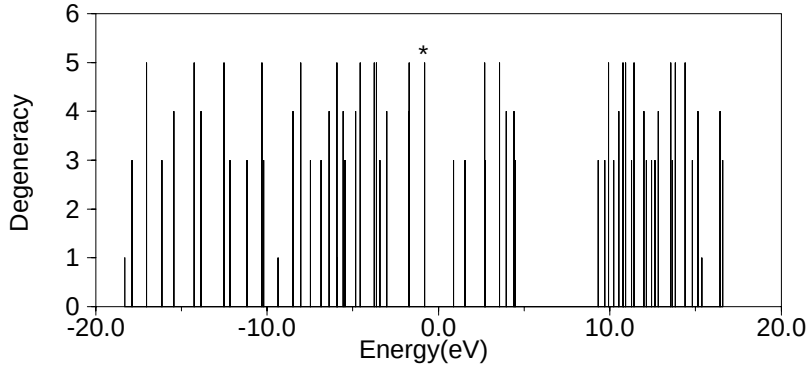


図 3.1 (a) 距離の関数であるタイトバインディングパラメーター [1]

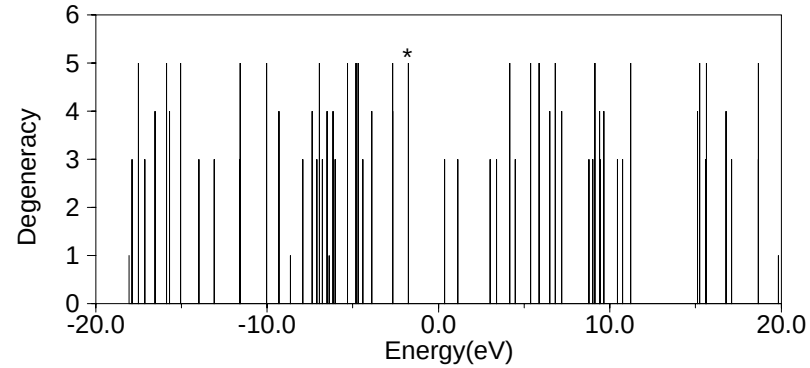


図 3.2 (b) 定数であるタイトバインディングパラメーター [2]

第一原理計算で計算された C_{60} のエネルギー固有値と比較すると、(a) の方がよく一致している。フラーレンのタイトバインディングパラメーターには、式 (2.21) から式 (2.33) 及び表 2.1 のパラメータ [3] で計算する方が、より正確な値になるといえる。

縮重度であるが、分子のように有限の大きさをもった物体は、何かある一点のまわりの対称操作に関してハミルトニアンは不変である。この不変になる操作をしたときにお互いに等価な波動関数のエネルギー固有値は同じになり、縮重するという。この対称操作ができることが縮重度としてあらわれ、分子の場合は点群の既約表現の縮重度としてあらわれる。C_{60} の 5 重の縮重は、C_{60} が正二十面体群 ($\mathbf{I}_h$) をもつためである。正二十面体群は 6 本の 5 回転軸、10 本の 3 回転軸、15 本の 2 回転軸をもつ。また、空間反転ももつ。空間反転とは、座標 $\boldsymbol{r}$ を $-\boldsymbol{r}$ に変換する操作である。この、正二十面体群と空間反転をあわせた群を $\mathbf{I}_h$ と呼ぶ。

[1] ファイル名：./eps/shuku.eps

[2] ファイル名：./eps/shu2.eps

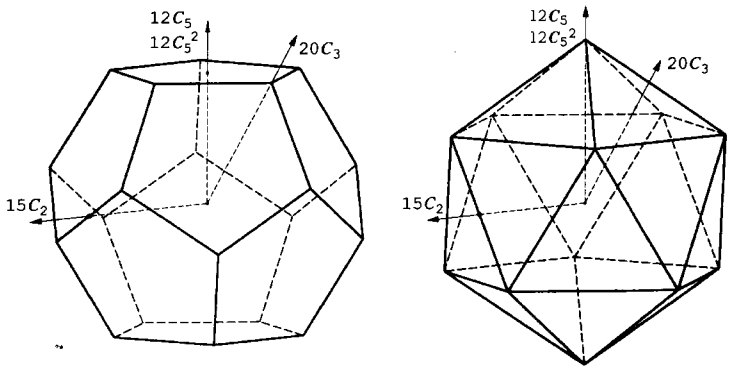


図 3.3　正二十面体群 [3]

C_{60} には $\mathbf{I}_h$ 対称性があるため 3 重、 4 重、 5 重の縮重がみられる。特に、 2 重の縮重度はない。

炭素原子あたり価電子が 4 個で、 60 個の原子があるので、 240 の電子を持つ。この電子はエネルギー準位の低い分子軌道から占有していき、 1 つの分子軌道には上向きのスピンと下向のスピンの 2 つの電子が占有するので、実際に電子が占有しているのはエネルギーが最も低い分子軌道から 120 番目のエネルギー固有値のところまでである。この、電子が占有している最もエネルギーが高いところ (Highest occupied molecular orbital) は図では、 5 重に縮重している $*$ である。縮重している５つすべてに電子は占有している。また、電子がない最もエネルギーが低い分子軌道 (Lowest unoccupied molecular orbital) は 3 重に縮重しているところで、 (a) での HOMO と LUMO のエネルギーギャップは $2.0(\boldsymbol{eV})$ であった。同じタイトバインディングパラメーターを使った計算結果と一致し、第一原理計算の結果の $1.9(\boldsymbol{eV})$ と非常に近い値になった。 (b) のエネルギーギャップは、 $2.1(\boldsymbol{eV})$ で、こちらの結果でも、 (a) の方が良い結果を得ている。

また、状態密度をエネルギー固有値と固有ベクトルから求めた。固有ベクトルは、あるエネルギー固有値に対して、分子を構成する原子のそれぞれの原子軌道が持つ成分を表していて、エネルギー固有値 E_i に対して C_i であらわす。オーバーラップ行列 S がある場合は、 $C^{\dagger}SC = 1$ にならなければいけない。 C_{60} の場合、 E_i に対応する固有ベクトルは次のように表される、

$$C_i = (C_{is_1}, C_{ip_{x1}}, C_{ip_{y1}}, C_{ip_{z1}}, C_{is_2}, C_{ip_{x2}}, C_{ip_{y2}}, \cdots, C_{is_{60}}, C_{ip_{x60}}, C_{ip_{y60}}, C_{ip_{z60}}) \quad (3.1)$$

状態密度は、出てきた固有ベクトルの値を、エネルギー固有値に対して幅を持つようにするため、カウス関数を用い、その曲線の重ね合わせで状態密度を表した。

[3]ファイル名：./eps/taishou.eps

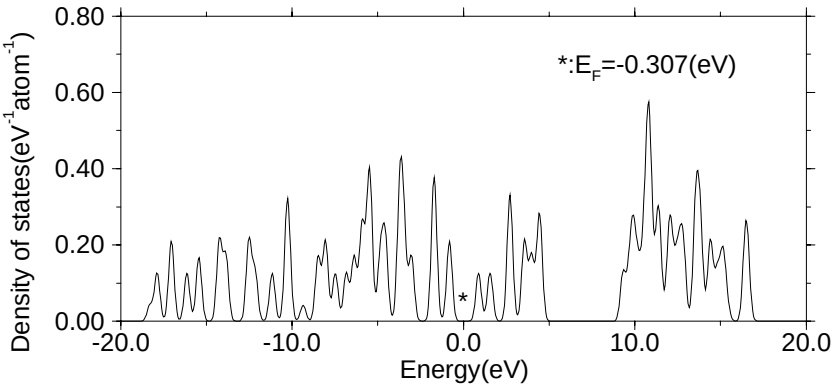


図 3.4 C_{60} の状態密度 [4]

図 3.4 では、ガウス関数 $e^{-(E-E_i)^2/a^2}$ の a の値を $0.22(\boldsymbol{eV})$ とした。この値は平均的な分子軌道の level 間隔におけばよい。

状態密度は、炭素原子 1 個あたりの状態密度である。 1 つの原子には $2s, 2p_x, 2p_y, 2p_z$ の 4 つの原子軌道があるので、状態密度の曲線全体を積分をすると 4 になるはずである。実際に計算して得られた曲線を積分した結果も 4 となった。フェルミエネルギー E_F はエネルギーの低い方から積分して 2 になった図 3.4 の $*$ である。

表 3.1: (b) のタイトバインディングパラメーター [2]

t	値 $(\boldsymbol{eV})$	s	値
t^{σ}_{ss}	-6.769	s^{σ}_{ss}	0.212
t^{σ}_{sp}	-5.580	s^{σ}_{sp}	0.102
t^{σ}_{pp}	-5.037	s^{σ}_{pp}	0.146
t^{π}_{pp}	-3.033	s^{π}_{pp}	0.129
ε_{2s}	-8.868		

ε_{2s} は， $\varepsilon_{2p} = 0$ とした時の相対的な値

分子軌道法では C_{60} の場合 240×240 の行列ができる。計算時間は、行列の次元を N とすると、 N^3 に比例する。実際の C_{60} での計算時間は cone を使用した場合 6.1 秒であった。このことから、原子数 $1000(N = 4000)$ で 28240 秒、原子数 $10000(N = 40000)$ で 28240740 秒の計算時間がかかることが予測される。原子数が多くなった場合固有値を求めるのに非常に時間がかかる。

[4] ファイル名：./eps/jmlen.eps

3.2 リカージョン法

3.2.1 ハミルトニアン行列の作成

C_{60} に重なり行列がある場合のリカージョン法を適用する。ここでは、ハミルトニアン行列を作るプログラムが未完成であるので、ハミルトニアン行列は途中までしか作られていない。

最初の波動関数は、任意にとった原子の番号を 1 として、その原子にある 4 つの原子軌道の波動関数 $(\phi_{2s_1}, \phi_{2px_1}, \phi_{2py_1}, \phi_{2pz_1})$ の重ね合わせで構成した。

$$\boldsymbol{u}_0 = \begin{pmatrix} \phi_{2s_1} \\ \phi_{2px_1} \\ \phi_{2py_1} \\ \phi_{2pz_1} \end{pmatrix} \tag{3.2}$$

この場合、次の波動関数 $\boldsymbol{u}_1$ は、 $\boldsymbol{u}_0$ を構成するすべての波動関数と重なりを持つ波動関数で構成される。最近接の原子以外は直交化していると考えると、 $\boldsymbol{u}_1$ は原子 1 に隣接する 3 つの原子を 2,3,4 とするとそれらの原子の 12 個の原子軌道の波動関数 $\phi_{2s_2}, \phi_{2px_2}, \phi_{2py_2}, \phi_{2pz_2}$ $\phi_{2s_3}, \phi_{2px_3}, \phi_{2py_3}, \phi_{2pz_3}$ $\phi_{2s_4}, \phi_{2px_4}, \phi_{2py_4}, \phi_{2pz_4}$ で構成される。同様に $\boldsymbol{u}_n$ はつくることができる。この論文で採用したタイトバインディングパラメーターのカットオフディスタンス 4.0Å を使った場合、 1 つの原子にある原子軌道は同じ位置にあるので、原子をひとかたまりでみる。作られたプログラムでは、 C_{60} で図 3.5 のように 4 つの基底しか作ることができなかった。 C_{60} では分子を構成する原子の価電子帯に合計 240 の原子軌道があるので 240 の基底が必要である。

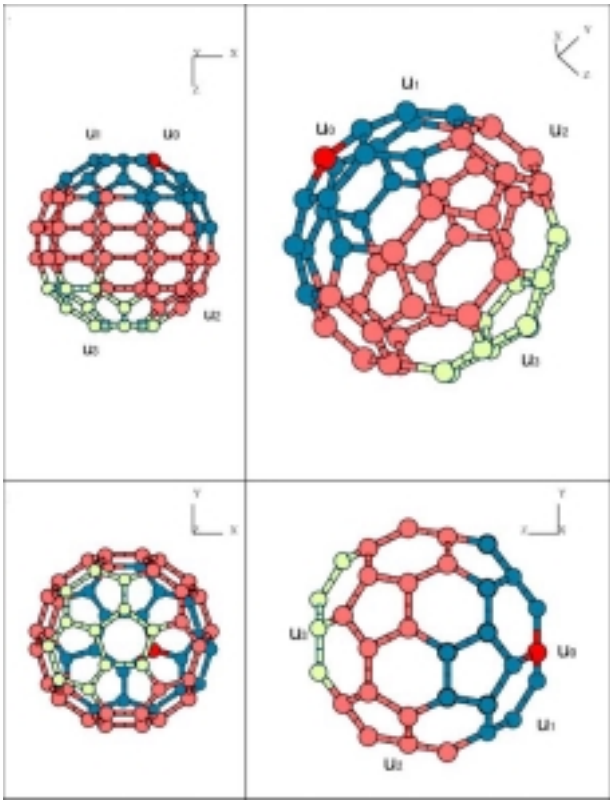


図 3.5　C_{60} のモデル [5]

カットオフディスタンス 4.0(Å) では、一つの基底の中にある波動関数が多くなる。

$\boldsymbol{u}_0$ から $\boldsymbol{u}_3$ の 4 つの新たな波動関数で構成されていると考えると、このモデルのハミルトニアン行列は 4×4 の行列になる。

$$\mathrm{H} = \begin{pmatrix} a_0 & b_1 & 0 & 0 \\ b_1 & a_1 & b_2 & 0 \\ 0 & b_2 & a_2 & b_3 \\ 0 & 0 & b_3 & a_3 \end{pmatrix} \tag{3.3}$$

また、カットオフディスタンスを 1.6(Å) としたときで、作られたプログラムでのリカージョン法のモデルを図 3.6 に示す。

[5] ファイル名：./eps/chain1.eps

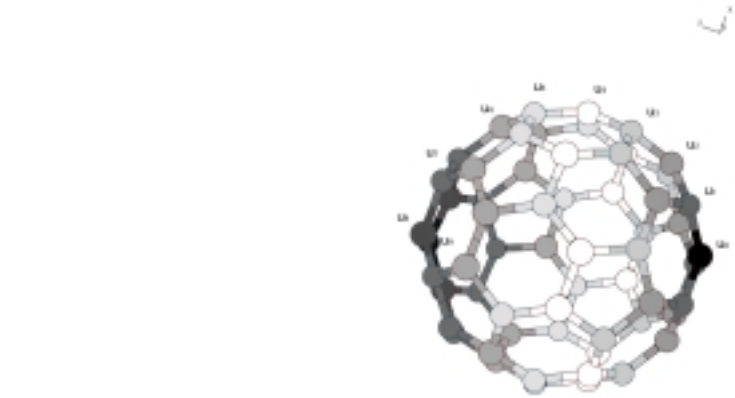


図 3.6　カットオフディスタンス 1.6(Å) のモデル [6]

この時、このモデルの波動関数は $\boldsymbol{u}_0$ から $\boldsymbol{u}_9$ までで構成される。したがって基底の数は 10 となる。この $\boldsymbol{u}_9$ で基底を切ってしまっては正しい結果を得ることはできない。$\boldsymbol{u}_n = 0$ になるまで繰り返す必要がある。基底の数は最初の基底の取り方によってかわる。この最初の基底を対称性が最も低いようにとると、240 個の基底がとれるはずである。

本研究で作成したプログラムでは、C_{60} 分子の場合、カットオフディスタンス 1.6(Å) で基底を 10 個までしか作ることができなかった。作られたモデルから状態密度を計算するプログラムを作成したが、モデルを作るプログラムが未完成なためにこのプログラムが正しい動作をしているか検証することはできなかった。

チェーンモデルを作るプログラムは、基底を最後まで作ることができるプログラムに改良する必要がある。また、チェーンモデルではハミルトニアン行列の次元を小さくしすぎないために最初の波動関数 u_0 をうまくとらなければならない。

[6] ファイル名：./eps/c60.eps

第 4 章

結論

タイトバインディングによる分子軌道法では、行列の固有方程式の計算時間を短縮できないために、原子の数が非常に多いと計算時間がかかりすぎるという結果を得た。このことから、カーボンナノチューブのような原子の数が多い分子では、オーダー N 法であるリカージョン法による計算は非常に有用であるといえる。しかし、リカージョン法では対称性の高い分子では、最初の基底を最も対称性の低いものにしなければ、正しい状態密度を計算できない。

この実験では、リカージョン法による結果を出すことができなかったため、分子軌道法による結果を再現することができなかった。

これからの課題として、リカージョン法によるプログラムを完成させ、実際に計算をし、チェーンモデルの最初の $\boldsymbol{u}_0$ の良い取り方を考える必要がある。この最初の $\boldsymbol{u}_0$ は状態密度を正しく計算する上で非常に重要である。

【謝辞】本研究を進めるにあたって多大な御指導、御助言を頂きました電気通信大学電子工学科齋藤理一郎助教授に心よりお礼申し上げます。また、本研究に数々の有益な御助言を頂いた木村忠正教授、湯郷成美助教授、一色秀夫助手に深く感謝申し上げます。また、研究活動をともにし、多くの援助をいただいた八木将志氏、松尾竜馬氏、山岡寛明氏、田代哲正氏に深く感謝致します。そして、数々の御援助を頂いた木村研究室、湯郷研究室の大学院生、卒研生の方々に感謝致します。

参考文献

[1] R. Haydock,Solid state Physics vol 35, ed F. seits, D. Turnbull and H. Ehrenreich (New York academic) 216 (1980). *The recursion method and its applications* Editors D.G. Petitfor and D.L. Weaire (Springer Verlag), Spronger Series in Solid State Science **58** (1984).

[2] 齋藤 理一郎 著 「量子物理学」 培風館

[3] Okada 東京工業大学博士論文

[4] 小野寺 嘉孝 著 「群論入門」 裳華房

[5] 小国 力 編著 「行列計算ソフトウェア」 丸善株式会社

[6] 齋藤 弥八 坂東 俊治 著 「カーボンナノチューブの基礎」 コロナ社

【付録】

プログラムソース

1. ハミルトニアン行列をつくるプログラム

入力ファイル : c60.xyz 原子の座標データ
: SIZES プログラムのパラメータ (コンパイルに必要)
AN: 原子数,NSIZE: 原子軌道数
出力ファイル : Hmn60.dat H 行列
: Smn60.dat S 行列

```
c     C60 のエネルギーバンドの計算
c
c     Rxyz: 座標の入った配列
c     Hmn: ハミルトニアン行列
c     E: エネルギー固有値
c
c
c     型
c
      implicit real*8(a-z)
      integer num,snum,cou1,cou2,k,l,o,sho1,sho2,n,nsize,ne,nv,
     #     A_N,AN,CHECK
      CHARACTER*20 DUM_CHA
      INCLUDE 'SIZES'
C      parameter(
C     #     AN=60,
C     #     NSIZE=240)
      dimension Hehe(4,4),Sese(4,4),Rxyz(AN,3),
     #     Hmn(NSIZE,NSIZE),Smn(NSIZE,NSIZE)
c
c
c     パラメーター
c
      parameter(
c     隣合う原子間の距離の基準
     #     LENa=4.0,
c     e2p と e2s の値
     #     e2p=0.0,
     #     e2s=(-7.0),
     #     EPS= 1.0D-24)
      n=240
      ne=240
      nv=240
      LENb_2=3.6*3.6
c
c
c     基準の原子間の距離の 2 乗
      LENa_2=LENa*LENa
c
c     行列の初期化
      DO 1 cou1=1,NSIZE
         DO 2 cou2=1,NSIZE
            Hmn(cou1,cou2)=0.
            Smn(cou1,cou2)=0.
 2       CONTINUE
 1    CONTINUE
c
c
c     原子の座標の読み込み
c
      open(15,FILE='c60.xyz',
     #     status='old')
      rewind 15
      read(15,*) A_N
      write(*,*) A_N
      DO 10 cou1=1,AN
         read(15,*) DUM_CHA,Rxyz(cou1,1),Rxyz(cou1,2),Rxyz(cou1,3)
c         write(*,*) Rxyz(cou1,1),Rxyz(cou1,2),Rxyz(cou1,3)
 10   continue
      close(15,status='keep')
c
c
c
c     ここから Hmn を求めるプログラム
c
```

```
c      R 原子の決定
       DO 1000 num=1,AN
          R1x=Rxyz(num,1)
          R1y=Rxyz(num,2)
          R1z=Rxyz(num,3)
c
c
c      隣の原子の決定
          DO 900 snum=num,AN
c
c      同じ原子の場合は 550 へ
             IF(snum.EQ.num) GO TO 550
c
c      X 成分が基準より大きい場合，飛ばす
             IF(ABS(Rxyz(snum,1)-Rxyz(num,1)).GE.LENa) GO TO 900
c
c      Y 成分が基準より大きい場合，飛ばす
             IF(ABS(Rxyz(snum,2)-Rxyz(num,2)).GE.LENa) GO TO 900
c
c      Z 成分が基準より大きい場合，飛ばす
             IF(ABS(Rxyz(snum,3)-Rxyz(num,3)).GE.LENa) GO TO 900
c
c      原子間の距離の 2 乗を求める
             LENxyz_2 =
     #          (Rxyz(snum,1)-Rxyz(num,1))*(Rxyz(snum,1)-Rxyz(num,1))
     #        +(Rxyz(snum,2)-Rxyz(num,2))*(Rxyz(snum,2)-Rxyz(num,2))
     #        +(Rxyz(snum,3)-Rxyz(num,3))*(Rxyz(snum,3)-Rxyz(num,3))
c             write(*,*) LENxyz_2
c
c      原子間の距離が基準より大きい場合，飛ばす
             IF(LENxyz_2.GE.LENa_2) GO TO 900
c
c      Qr
             IF(LENxyz_2.GE.LENb_2)then
                CHECK=1
             else
                CHECK=0
             end if
c
c      隣の原子と決定された原子の座標を入れる
             Nx=Rxyz(snum,1)
             Ny=Rxyz(snum,2)
             Nz=Rxyz(snum,3)
c
c      ---------------- 隣である --------------

c             write(*,*) num,snum
c
c      Hij に座標を入れる

          call Hij(R1x,R1y,R1z,Nx,Ny,Nz,LENxyz_2,Hehe,SeSe,CHECK)
c
c
c      結果を行列に入れる
          DO 200 k=1,4
             DO 100 l=1,4
c                write(*,*) (num-1)*4+k,(snum-1)*4+l
                Hmn((num-1)*4+k,(snum-1)*4+l)=Hehe(k,l)
                Smn((num-1)*4+k,(snum-1)*4+l)=Sese(k,l)
                Hmn((snum-1)*4+k,(num-1)*4+l)=Hehe(l,k)
                Smn((snum-1)*4+k,(num-1)*4+l)=Sese(l,k)
 100         CONTINUE
 200      CONTINUE
          GO TO 900
c
c
c      ------------- 同じ原子の時の Hmn の値を入れる -----------
c
c 550
 550      Hmn(num*4-3,snum*4-3)=e2s
          Smn(num*4-3,snum*4-3)=1.
          DO 560 O=1,3
             Hmn(num*4-3+o,snum*4-3+o)=e2p
             Smn(num*4-3+o,snum*4-3+o)=1.
 560      CONTINUE
c
c      ------------ 離れている時（既に 0 を入れている） ----------
```

```
c
c
c
 900     CONTINUE
 1000 CONTINUE
c
c     Hmn をファイルに出力
      open(20,FILE='Hmn60.dat',access='sequential',
     #     form='formatted')
      DO 51 sho1=1,NSIZE
         DO 52 sho2=1,NSIZE
            write(20,*) Hmn(sho1,sho2)
 52      continue
 51   continue
      close(20,status='keep')
c
c     Smn をファイルに出力
      open(21,FILE='Smn60.dat',access='sequential',
     #     form='formatted')
      DO 61 sho1=1,NSIZE
         DO 62 sho2=1,NSIZE
            write(21,*) Smn(sho1,sho2)
 62      continue
 61   continue
      close(21,status='keep')
c
      Stop
      END
c
c
c     ------------------------------------------------
c     C60 の Hij を計算するプログラム
c
      subroutine Hij(R_x,R_y,R_z,N_x,N_y,N_z,LEN_2,Hehe,Sese,CHECK)
c
      implicit real*8(a-z)
      integer CHECK
      parameter(
     #     pi=3.14159265,
     #     vs=6.6,
     #     vsg=4.3,
     #     vpi=4.5,
C     #     u=1.0/7.0,
     #     rs=0.62,
     #     rsg=0.81,
     #     rpi=0.55 )
c
c     型宣言
      dimension Hehe(4,4),Sese(4,4)
c
c     --------- 値の定義 ---------
c
C      u=1.0/7.0
      rx=R_x
      ry=R_y
      rz=R_z
      n1x=N_x
      n1y=N_y
      n1z=N_z
      ZERO=0.0
      ONE=1.0
c
c
c     -------- 計算式 -------
c
c     |Rn-R|
c
      R1L_1=sqrt(LEN_2)
c      write(*,*) R1L_1
c
c     タイトバインディングパラメーター
      Hss=(-7.0*vs*Rssfun(4.0*R1L_1/(rs+rs)))
      Hsp=(-7.0*(sqrt(vs*vsg))*Rspfun(4.0*R1L_1/(rs+rsg)))
      Hsg=(-7.0*vsg*Rsigfun(4.0*R1L_1/(rsg+rsg)))
      Hpi=(-7.0*vpi*Rpifun(4.0*R1L_1/(rpi+rpi)))
      Sss = Rssfun(4.0*R1L_1/(rs+rs))
      Ssp = Rspfun(4.0*R1L_1/(rs+rsg))
```

```
      Ssg = Rsigfun(4.0*R1L_1/(rsg+rsg))
      Spi = Rpifun(4.0*R1L_1/(rpi+rpi))

      IF(CHECK.eq.1)then
         Q = Qr(R1L_1,(pi*(R1L1-3.6)/0.4))
         Hss=Hss*Q
         Hsp=Hsp*Q
         Hsg=Hsg*Q
         Hpi=Hpi*Q
         Sss=Sss*Q
         Ssp=Ssp*Q
         Ssg=Ssg*Q
         Spi=Spi*Q
      end if

C      write(*,*) Hss,Sss
C      write(*,*) Hsp,Ssp
C      write(*,*) Hsg,Ssg
C      write(*,*) Hpi,Spi
c      COSx （N 側の成分からみて R 方向の COS）
      cosx = CSTH(r_x,n_x,R1L_1)
      cosxx = cosx**2
c      COSy
      cosy = CSTH(r_y,n_y,R1L_1)
      cosyy = cosy**2
c      COSz
      cosz = CSTH(r_z,n_z,R1L_1)
      coszz = cosz**2
c
c
c     H の計算式
c
c     Hsp,Ssp
c
c     値の代入
      Hspx = Hsp * cosx
      Hspy = Hsp * cosy
      Hspz = Hsp * cosz
c
      Sspx = Ssp * cosx
      Sspy = Ssp * cosy
      Sspz = Ssp * cosz
c
c     Hpapa,Spapa
c
c     値の代入
c     Hpxpx
      Hpxpx1 = Hsg * (-(cosxx))
      Hpxpx2 = Hpi * (1 - cosxx)
c
c     Hpypy
      Hpypy1 = Hsg * (-(cosyy))
      Hpypy2 = Hpi * (1 - cosyy)
c
c     Hpzpz
      Hpzpz1 = Hsg * (-(coszz))
      Hpzpz2 = Hpi * (1 - coszz)
c
      Hpxpx = (Hpxpx1 + Hpxpx2)
      Hpypy = (Hpypy1 + Hpypy2)
      Hpzpz = (Hpzpz1 + Hpzpz2)
c
c     Spxpx
      Spxpx = (Ssg * (-(cosxx)) + Spi * (1 - cosxx))
c     Spypy
      Spypy = (Ssg * (-(cosyy)) + Spi * (1 - cosyy))
c     Spzpz
      Spzpz = (Ssg * (-(coszz)) + Spi * (1 - coszz))
c
c     Hpapb
c
c     値の代入
c
c     Hpxpy
      Hpxpy1 = Hsg * (-(cosx * cosy))
```

```
      Hpxpy2 = Hpi*nejire(rx,ry,rz,n1x,n1y,n1z,R1L_1,
     #     ONE,ZERO,ZERO,ZERO,ONE,ZERO)
c
c     Hpxpz
      Hpxpz1 = Hsg * (-(cosx * cosz))
      Hpxpz2 = Hpi*nejire(rx,ry,rz,n1x,n1y,n1z,R1L_1,
     #     ONE,ZERO,ZERO,ZERO,ZERO,ONE)
c
c     Hpypz
      Hpypz1 = Hsg * (-(cosy * cosz))
      Hpypz2 = Hpi*nejire(rx,ry,rz,n1x,n1y,n1z,R1L_1,
     #     ZERO,ONE,ZERO,ZERO,ZERO,ONE)
c
c
      Hpxpy = (Hpxpy1 + Hpxpy2)
      Hpxpz = (Hpxpz1 + Hpxpz2)
      Hpypz = (Hpypz1 + Hpypz2)
      Hpypx = Hpxpy
      Hpzpx = Hpxpz
      Hpzpy = Hpypz
c
      Spxpy = (Hpxpy1*Ssg/Hsg + Hpxpy2*Spi/Hpi)
      Spxpz = (Hpxpz1*Ssg/Hsg + Hpxpz2*Spi/Hpi)
      Spypz = (Hpypz1*Ssg/Hsg + Hpypz2*Spi/Hpi)
      Spypx = Spxpy
      Spzpx = Spxpz
      Spzpy = Spypz
c
c
c
c          Hsasa
      Hehe(1,1) = Hss
c          Hspx
      Hehe(1,2) = Hsp * (cosx)
c          Hspy
      Hehe(1,3) = Hsp * (cosy)
c          Hspz
      Hehe(1,4) = Hsp * (cosz)
c          Hpxs
      Hehe(2,1) = (-Hehe(1,2))
c          Hpxpx
      Hehe(2,2) = Hpxpx
c          Hpxpy
      Hehe(2,3) = Hpxpy
c          Hpxpz
      Hehe(2,4) = Hpxpz
c          Hpys
      Hehe(3,1) = (-Hehe(1,3))
c          Hpypx
      Hehe(3,2) = Hpypx
c          Hpypy
      Hehe(3,3) = Hpypy
c          Hpypz
      Hehe(3,4) = Hpypz
c          Hpzs
      Hehe(4,1) = (-Hehe(1,4))
c          Hpzpx
      Hehe(4,2) = Hpzpx
c          Hpzpy
      Hehe(4,3) = Hpzpy
c          Hpzpz
      Hehe(4,4) = Hpzpz
c
      Sese(1,1) = (Sss)
      Sese(1,2) = (Sspx)
      Sese(1,3) = (Sspy)
      Sese(1,4) = (Sspz)
      Sese(2,1) = (-Sese(1,2))
      Sese(2,2) = (Spxpx)
      Sese(2,3) = (Spxpy)
      Sese(2,4) = (Spxpz)
      Sese(3,1) = (-Sese(1,3))
      Sese(3,2) = (Spypx)
```

```
      Sese(3,3) = (Spypy)
      Sese(3,4) = (Spypz)
      Sese(4,1) = (-Sese(1,4))
      Sese(4,2) = (Spzpx)
      Sese(4,3) = (Spzpy)
      Sese(4,4) = (Spzpz)
c
c
      return
      END
c
c     よく出てくる cos,sin の関数の定義
c------------ 副関数 --------------
      real*8 function CSTH(a,b,c)
      real*8 a,b,c
      if(abs(c).lt.1.0D-14) THEN
         CSTH = 0
         STOP
      ELSE
         CSTH = (a-b)/c
      END IF
      return
      end

c
c     ねじれ (π成分)
      real*8 function Nejire(R1x,R1y,R1z,Nx,Ny,Nz,LEN,
     #     RPx,RPy,RPz,NPx,NPy,NPz)
      implicit real*8 (a-z)
      dimension R1(3),NJ(3),RJ(3),NK(3),RK(3)
C      parameter(
C     #     Hpi=(-3.033))
c
c     R->N ベクトル
      R1(1)=Nx-R1x
      R1(2)=Ny-R1y
      R1(3)=Nz-R1z
c
c     NP の NR 方向の成分の大きさ + 正負
      NNAI=(R1(1)*NPx + R1(2)*NPy + R1(3)*NPz)/LEN
c      write(*,*) R1(1)*NPx
c
c     NP の NR 方向の成分
      NJ(1)=(R1(1)/LEN)*NNAI
      NJ(2)=(R1(2)/LEN)*NNAI
      NJ(3)=(R1(3)/LEN)*NNAI
c
c     NP の残りの成分
      NK(1)=NPx-NJ(1)
      NK(2)=NPy-NJ(2)
      NK(3)=NPz-NJ(3)
c
c     RP の NR 方向の成分の大きさ + 正負
      RNAI=(R1(1)*RPx + R1(2)*RPy + R1(3)*RPz)/LEN
c
c     RP の NR 方向の成分
      RJ(1)=(R1(1)/LEN)*RNAI
      RJ(2)=(R1(2)/LEN)*RNAI
      RJ(3)=(R1(3)/LEN)*RNAI
c
c     RP の残りの成分
      RK(1)=RPx-RJ(1)
      RK(2)=RPy-RJ(2)
      RK(3)=RPz-RJ(3)
c
      Nejire = (NK(1)*RK(1) + NK(2)*RK(2) + NK(3)*RK(3))
      return
      end
c
c
c
      real*8 function Rssfun(x)
      real*8 x
      implicit real*8 (a-z)
      Rssfun=exp(-x)*(1.0+x+x*x/3.0)
      return
```

```
      end

      real*8 function Rspfun(x)
      real*8 x
      implicit real*8 (a-z)
      Rspfun=(exp(-x)*(x+x*x/3.0))
      return
      end

      real*8 function Rsigfun(x)
      real*8 x
      implicit real*8 (a-z)
      Rsigfun=(exp(-x)*(-1.0+x+x*x/3.0))
      return
      end

      real*8 function Rpifun(x)
      real*8 x
      Rpifun=exp(-x)*(1.0+x+x*x/3.0)
      return
      end

      real*8 function Qr(x,theta)
      real*8 x,theta
      Qr = 0.5*(1.0+cos(theta))
      return
      end
```

2. 固有値、固有ベクトルの計算プログラム（固有値、固有ベクトルの計算はプログラム DIGAB を引用）

入力ファイル : Hmn60.dat(H60am.f,lenham.f により出力)
: Smn60.dat(同上)
: SIZES
出力ファイル : energyV60.dat 固有ベクトルのデータ
: energyV60.edt エネルギー固有値のデータ
: energy60.dat エネルギー固有値の縮重度 (xvgr 用)

```
c     Hmn を daigab に代入するプログラム
c
c
c
      integer cou1,cou2,scou,count,counter,n,nsize,ne,nv,AN
      implicit real*8(a-z)
      INCLUDE 'SIZES'
      parameter(
     #     EPS= 1.0D-20,
     #     kijun=(-0.00006))
      dimension Hmn(NSIZE,NSIZE),Smn(NSIZE,NSIZE),E(NSIZE),
     #     W(NSIZE,7),V(NSIZE,NSIZE)
c
      N  = NSIZE
      NE = NSIZE
      NV = NSIZE
      open(15,FILE='Hmn60.dat',
     #     status='old',access='sequential',form='formatted')
      rewind 15
      DO 10 cou1=1,NSIZE
         DO 11 cou2=1,NSIZE
            read(15,*) Hmn(cou1,cou2)
 11      continue
 10   continue
      close(15,status='keep')

      open(16,FILE='Smn60.dat',
     #     status='old',access='sequential',form='formatted')
      rewind 16
      DO 20 cou1=1,NSIZE
         DO 21 cou2=1,NSIZE
            read(16,*) Smn(cou1,cou2)
 21      continue
 20   continue
      close(16,status='keep')
c
c     DAIGAB に値を代入
      call DEIGAB(Hmn,Smn,N,NSIZE,NE,NV,EPS,W,E,V)
```

```
c
c     E の出力
c
c     固有ベクトルの出力
c     V(count,SCOU):E(SCOU) に対応するベクトル
c
      open(22,FILE='energyV60.dat',access='sequential',
     #     form='formatted')
      write(22,*) AN
      DO 101 SCOU=1,NSIZE
C         TMP=0.0
C         DO 103 count=1,NSIZE
C            TMP=TMP+V(count,SCOU)**2
C 103     continue
C         TMP2=0.0
         DO 102 count=1,NSIZE
            write(22,*) V(SCOU,NSIZE+1-count)
C/sqrt(TMP)
C            TMP2=TMP2+(V(count,SCOU)/sqrt(TMP))**2
 102     continue
C         write(*,*) TMP2
 101  continue

c
c     固有値の出力
      open(21,FILE='energyV60.edt',access='sequential',
     #     form='formatted')
      DO 100 count=1,NSIZE
         write(21,*) E(NSIZE+1-count)
 100  CONTINUE
      close(21,status='keep')
      write(*,*) '120',E(120)
      open(20,FILE='energy60.dat',access='sequential',
     #     form='formatted')
      counter=1
      do 110 cou2=1,NSIZE-1
         if(counter.eq.1) then
            energy=e(COU2)
         endif

         if((e(COU2+1)-e(COU2)).GE.kijun) then
            counter=counter+1
            energy=energy+e(COU2+1)
            if(cou2.eq.NSIZE-1) then
               do 14 SCOU=0,counter
                  write(20,*) energy/counter,SCOU
 14            continue
            endif
         else
            do 12 SCOU=0,counter
               write(20,*) energy/counter,SCOU
 12         continue
            write(20,*) energy/counter,0
            if(cou2.eq.NSIZE-1) then
               do 13 SCOU=0,1
                  write(20,*) e(NSIZE),SCOU
 13            continue
            endif
            counter=1
         endif

 110  continue
      close(20,status='keep')
      Stop
      END
c
      SUBROUTINE DEIGAB( A, B, N, NSIZE, NE, NV, EPS, W, E, V )
      IMPLICIT REAL*8 (A-H,O-Q,S-Z)
C         SUBPROGRAM FOR GENERALIZED EIGENVALUE PROBLEM
C                    (A) X = LAMBDA (B) X
C         FOR REAL SYMMETRIC MATRICES (A) & (B), THE LATTER BEING
C         POSITIVE DEFINITE.
C
C       * USAGE - -
C     CALL DEIGAB( A, B, N, NSIZE, NE, NV, EPS, W, E, V )
C
C         INPUT - -
```

```
C         A     R *8   - 2-DIM. ARRAY CONTAINING REAL SYMMETRIC MATRIX.
C         B     R *8   - 2-DIM. ARRAY CONTAINING REAL SYMMETRIC
C                        POSITIVE DEFINITE MATRIX.
C         N     I *4   - ORDER OF MATRIX.
C       NSIZE   I *4   - SIZE OF THE 2-DIM. ARRAYS  A, B, W  & V
C                        DEFINED IN 'DIMENSION' STATEMENT (FIRST INDEX).
C        NE     I *4   - NUMBER OF EIGENVALUES TO BE OBTAINED.
C                        IN DESCENDING ORDER  WHEN  NE > 0,
C                        IN ASCENDING  ORDER  WHEN  NE < 0.
C        NV     I *4   - NUMBER OF EIGENVECTORS TO BE OBTAINED.
C        EPS    R *8   - ACCURACY  ( STANDARD VALUE  1.0D-16 ).
C
C         OUTPUT - -
C         E     R *8   - 1-DIM. ARRAY CONTAINING THE OUTPUT EIGENVALUES
C         V     R *8   - 2-DIM. ARRAY CONTAINING THE OUTPUT EIGENVECTORS
C                        THE VECTOR ( V(1,K), V(2,K),..., V(N,K) )
C                        BELONGS TO THE EIGENVALUE  E(K).
C         WORKING SPACE - -
C         W     R *8   - 2-DIM. ARRAY  (N,7)  USED AS THE WORKING AREA.
C
C       * NOTE - -
C         THE MATRICES (A) AND (B) ARE DESTROYED.
C       * METHOD - -
C         CHOLESKY REDUCTION FOLLOWED BY HOUSEHOLDER DIAGONALIZATION.
C       * SUBROUTINE USED - -  DEIGRS
      IMPLICIT INTEGER  (R)
C
C
      DIMENSION  A(NSIZE,NSIZE), B(NSIZE,NSIZE), W(NSIZE,7)
      DIMENSION  V(NSIZE,NSIZE), E(NSIZE)
C
C                    CHECK THE INPUT DATA.
      NEV = NE
      NEVA = IABS(NEV)
      NVEC = NV
      IF( N )            8, 8, 1
    1 IF( NSIZE - N )    8, 2, 2
    2 IF( NEVA )         3, 8, 3
    3 IF( N - NEVA )     8, 4, 4
    4 IF( NVEC )         8, 5, 5
    5 IF( NEVA - NVEC ) 8, 10, 10
    8 WRITE(6,9)  N, NSIZE, NEV, NVEC
    9  FORMAT(' (SUBR.DEIGAB)  INVALID ARGUMENT.   N, NSIZE, NE, NV = '4
     #I5)
        RETURN
C
   10 CONTINUE
      IF( N .NE. 1 )  GO TO  20
      T = B(1,1)
      IF ( T ) 24,24,15
   15 B(1,1) = DSQRT(1.0D0/T)
   18 E(1) = A(1,1)/T
      V(1,1) = 1.0D0
        RETURN
C
   20 CONTINUE
C                    CHOLESKY DECOMPOSITION OF THE POSITIVE DEFINITE
C                    MATRIX (B)  INTO A PRODUCT OF A LOWER TRIANGULAR
C                    MATRIX (L)  WITH ITS TRANSPOSED MATRIX.
      T = B(1,1)
      IF( T )   24, 24, 21
   21 T = DSQRT(1.0D0/T)
      B(1,1) = T
      DO  22  I=2, N
   22 B(1,I) = B(I,1) * T
      DO  29  R=2, N
      RSUB1 = R - 1
      SUM = 0.0
        DO  23  K=1, RSUB1
   23   SUM = B(K,R) ** 2  + SUM
      T = B(R,R) - SUM
      IF( T )   24, 24, 26
   26 T = DSQRT(1.0D0/T)
      B(R,R) = T
      IF( R .GE. N )  GO TO  30
      RADD1 = R + 1
        DO  28  I=RADD1, N
        SUM = 0.0
          DO  27  K=1, RSUB1
```

```
   27     SUM = B(K,I) * B(K,R) + SUM
          B(R,I) = ( B(I,R) - SUM ) * T
   28   CONTINUE
   29 CONTINUE
      ENTRY  DABSUB( A, EPS, E, V )
      IF ( N.NE.1 ) GO TO 30
      T = 1.0D0/B(1,1)**2
      GO TO 18
   30 CONTINUE
C                  CHOLESKY DECOMPOSITION IS COMPLETED.
C                  B = L L(T),  THE UPPER TRIANGULAR MATRIX  L(T)  IS
C                  STORED IN THE ARRAY  B.
C                       NOW, PROCEED TO EVALUATE  L**(-1) A L(T)**(-1)
C                       FIRST, PRE-MULTIPLY  L**(-1).
      DO  31  J=1, N
   31 A(1,J) = A(J,1) * B(1,1)
      DO  35  J=1, N
      DO  34  R=2, N
      RSUB1 = R - 1
      SUM = 0.0
        DO  32  K=1, RSUB1
   32   SUM = B(K,R) * A(K,J)  +  SUM
C
C
C
      A(R,J) = ( A(R,J) - SUM ) * B(R,R)
   34 CONTINUE
   35 CONTINUE
C                  NEXT, POST-MULTIPLY  L(T)**(-1).
      DO  36  J=1, N
   36 A(J,1) = A(J,1) * B(1,1)
      DO  40  R=2, N
      RSUB1 = R - 1
      T1 = B(R,R)
        DO  38  K=1, RSUB1
        T = - B(K,R)
          DO  37  J=R, N
   37     A(J,R) = A(J,K) * T  +  A(J,R)
   38   CONTINUE
      DO  39  J=R, N
   39 A(J,R) = A(J,R) * T1
   40 CONTINUE
C                  FIND EIGENVALUES AND EIGENVECTORS
C                  OF THE TRANSFORMED MATRIX.
      CALL DEIGRS( A, N, NSIZE, NEV, NVEC, EPS, W, W(1,7), E, V )
C
C                  BACK TRANSFORMATION OF EIGENVECTORS.
      IF( NVEC )  41, 50, 41
   41 CONTINUE
      DO  45  J=1, NVEC
      K = N
   42 T = V(K,J) * B(K,K)
      V(K,J) = T
      K1 = K - 1
      IF( K1 )   43, 45, 43
   43 DO  44  R=1, K1
   44 V(R,J) = V(R,J) - B(R,K) * T
      K = K1
        GO TO  42
   45 CONTINUE
   50 RETURN
   24   WRITE(6,25)
   25   FORMAT (' (SUBR.DEIGAB)  MATRIX  B  IS NOT POSITIVE DEFINITE.')
        RETURN
      END
c
      SUBROUTINE DEIGRS( A, N, N1, NE, NV, EPS, W, LW, E, V )
c
c Solve Eigenvaule Problem for Real Symmetrical Matrix
c which is taken from MSL, Tokyo-Univ. Comp. Center.
c
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL SW, LW
      DIMENSION A(N1,N1), W(N1,7), LW(N1), E(N1), V(N1,N1)
      NEA=IABS(NE)
      IF(NEA.NE.0) GO TO 1
      WRITE(6,1000) NE
      RETURN
    1 NVA=IABS(NV)
```

```
      IF( NVA.LE.NEA .AND. NEA.LE.N .AND. N.LE.N1) GO TO 2
      WRITE(6,2000) NV, NE, N, N1
      E(1)=0.
      RETURN
    2 NM1=N-1
      NM2=N-2
      IF( EPS.LT.0.0 ) EPS=1.0D-16
      IF(NM2) 10, 20, 50
C  WHEN N=1
   10 E(1)=A(1,1)
      IF ( NV.NE.0 ) V(1,1) = 1.0D0
      RETURN
C  WHEN N=2
C  COMPUTE EIGENVALUES OF 2*2 MATRIX
   20 CALL ERRSET(207,256,-1,1)
      W(1,1)=A(2,1)
      T = 0.5D0*(A(1,1)+A(2,2))
      R=A(1,1)*A(2,2)-A(2,1)*A(2,1)
      D=T*T-R
      Q=DABS(T)+DSQRT(D)
      IF(T.LT.0.) Q=-Q
      T = T*DFLOAT(NE)
      IF(T) 40, 30, 30
   30 E(1)=Q
      IF(NEA.EQ.2) E(2)=R/Q
      GO TO 250
   40 E(1)=R/Q
      IF(NEA.EQ.2) E(2)=Q
      GO TO 250
C  WHEN N=3,4,...
C  REDUCE TO TRIDIAGONAL FORM BY HOUSEHOLDER'S METHOD
   50 DO 130 K=1,NM2
      K1=K+1
      S=0.
      DO  60 I=K1,N
   60 S=S+A(I,K)*A(I,K)
      W(K,1)=0.
      IF(S.EQ.0.) GO TO 130
      SR=DSQRT(S)
      A1=A(K1,K)
      IF(A1.LT.0.) SR=-SR
      W(K,1)=-SR
      R = 1.0D0/(S+A1*SR)
      A(K1,K)=A1+SR
      DO  90 I=K1,N
      S=0.
      DO  70 J=K1,I
   70 S=S+A(I,J)*A(J,K)
      IF(I.EQ.N) GO TO 90
      I1=I+1
      DO  80 J=I1,N
   80 S=S+A(J,I)*A(J,K)
   90 W(I,1)=S*R
      S=0.
      DO 100 I=K1,N
  100 S=S+A(I,K)*W(I,1)
      T = 0.5D0*S*R
      DO 110 I=K1,N
  110 W(I,1)=W(I,1)-T*A(I,K)
      DO 120 J=K1,N
      WJ1=W(J,1)
      AJK=A(J,K)
      DO 120 I=J,N
  120 A(I,J)=A(I,J)-A(I,K)*WJ1-W(I,1)*AJK
  130 CONTINUE
      W(NM1,1)=A(N,NM1)
C  COMPUTE EIGENVALUES BY BISECTION METHOD
      CALL ERRSET(207,256,-1,1)
      DO 135 I=1,N
  135 W(I,6)=A(I,I)
      R=DMAX1((DABS(W(1,6))+DABS(W(1,1))),(DABS(W(NM1,1))+DABS(W(N,6))))
      DO 140 I=2,NM1
      T=DABS(W(I-1,1))+DABS(W(I,6))+DABS(W(I,1))
      IF(T.GT.R) R=T
  140 CONTINUE
      EPS1=R*1.0D-16
      EPS2=R*EPS
      DO 150 I=1,NM1
  150 W(I,2)=W(I,1)*W(I,1)
```

```
      IF(NE.LT.0) R=-R
      F=R
      DO 160 I=1,NEA
  160 E(I)=-R
      DO 240 K=1,NEA
      D=E(K)
  170 T = 0.5D0*(D+F)
C
      IF( DABS(D-F).LE.EPS2 .OR. T.EQ.D .OR. T.EQ.F ) GO TO 240
C
      J=0
      I=1
  180 Q=W(I,6)-T
  190 IF(Q.GE.0.) J=J+1
      IF(Q.EQ.0.) GO TO 200
      I=I+1
      IF(I.GT.N) GO TO 210
      CALL OVERFL(L)
      Q=W(I,6)-T-W(I-1,2)/Q
      CALL OVERFL(L)
      IF(L.NE.1) GO TO 190
      J=J+1
      I=I-1
  200 I=I+2
      IF(I.LE.N) GO TO 180
  210 IF(NE.LT.0) J=N-J
      IF(J.GE.K) GO TO 220
      F=T
      GO TO 170
  220 D=T
      M=MIN0(J,NEA)
      DO 230 I=K,M
  230 E(I)=T
      GO TO 170
  240 E(K)=T
C  COMPUTE EIGENVECTORS BY INVERSE ITERATION
  250 CALL ERRSET(207, 10, 5,2)
      IF(NV.EQ.0) RETURN
      IF( N.NE.2 ) GO TO 255
      W(1,6)=A(1,1)
      W(2,6)=A(2,2)
  255 CALL ERRSET(207,256,-1,1)
C
      W(N,1)=0.
      MM=584287
      DO 410 I=1,NVA
      DO 260 J=1,N
      W(J,2)=W(J,6)-E(I)
      W(J,3)=W(J,1)
  260 V(J,I) = 1.0D0
      SW=.FALSE.
C  REDUCE TO TRIANGULAR FORM
      DO 280 J=1,NM1
      IF(DABS(W(J,2)).LT.DABS(W(J,1))) GO TO 270
      IF(W(J,2).EQ.0) W(J,2)=1.0D-30
      W(J,5)=W(J,1)/W(J,2)
      LW(J)=.FALSE.
      W(J+1,2)=W(J+1,2)-W(J,5)*W(J,3)
      W(J,4)=0.
      GO TO 280
  270 W(J,5)=W(J,2)/W(J,1)
      LW(J)=.TRUE.
      W(J,2)=W(J,1)
      T=W(J,3)
      W(J,3)=W(J+1,2)
      W(J,4)=W(J+1,3)
      W(J+1,2)=T-W(J,5)*W(J,3)
      W(J+1,3)=-W(J,5)*W(J,4)
  280 CONTINUE
      IF(W(N,2).EQ.0.) W(N,2)=1.0D-30
C  BEGIN BACK SUBSTITUTION
      IF(I.EQ.1) GO TO 300
      IF(DABS(E(I)-E(I-1)).GE.EPS1) GO TO 300
C  GENERATE RANDOM NUMBERS
      DO 290 J=1,N
      MM=MM*48828125
  290 V(J,I)=FLOAT(MM)*0.4656613E-9
  300 CALL OVERFL(L)
      T=V(N,I)
```

```
      R=V(N-1,I)
  310 V(N,I)=T/W(N,2)
      V(NM1,I)=(R-W(NM1,3)*V(N,I))/W(NM1,2)
      CALL OVERFL(L)
      IF(L.NE.1) GO TO 330
      DO 320 J=1,NM2
  320 V(J,I)=V(J,I)*1.D-5
      T=T*1.D-5
      R=R*1.D-5
      GO TO 310
  330 IF(N.EQ.2) GO TO 380
      K=NM2
  340 T=V(K,I)
  350 V(K,I)=(T-W(K,3)*V(K+1,I)-W(K,4)*V(K+2,I))/W(K,2)
      CALL OVERFL(L)
      IF(L.NE.1) GO TO 370
      DO 360 J=1,N
  360 V(J,I)=V(J,I)*1.D-5
      T=T*1.D-5
      GO TO 350
  370 K=K-1
      IF(K) 380, 380, 340
  380 IF(SW) GO TO 410
      SW=.TRUE.
      DO 400 J=1,NM1
      IF(LW(J)) GO TO 390
      V(J+1,I)=V(J+1,I)-W(J,5)*V(J,I)
      GO TO 400
  390 T=V(J,I)
      V(J,I)=V(J+1,I)
      V(J+1,I)=T-W(J,5)*V(J+1,I)
  400 CONTINUE
      GO TO 300
  410 CONTINUE
C  BEGIN BACK TRANSFORMATION
      CALL ERRSET(207, 10, 5,2)
      IF(N.EQ.2) GO TO 470
      DO 415 I=1,NM2
  415 W(I,1)=-W(I,1)*A(I+1,I)
      DO 460 I=1,NVA
      K=NM2
  420 R=W(K,1)
      IF(R.EQ.0.) GO TO 450
      R = 1.0D0/R
      S=0.
      K1=K+1
      DO 430 J=K1,N
  430 S=S+A(J,K)*V(J,I)
      R=R*S
      DO 440 J=K1,N
  440 V(J,I)=V(J,I)-R*A(J,K)
  450 K=K-1
      IF(K.GE.1) GO TO 420
  460 CONTINUE
C  NORMALIZE EIGENVECTORS
C  NORMALIZE AS MAXIMUM ELEMENT = 1
  470 DO 490 I=1,NVA
      T=DABS(V(1,I))
      K=1
      DO 480 J=2,N
      R=DABS(V(J,I))
      IF(T.GE.R) GO TO 480
      T=R
      K=J
  480 CONTINUE
      T = 1.0D0/V(K,I)
      DO 490 J=1,N
  490 V(J,I)=V(J,I)*T
      IF(NV.LT.0) RETURN
C  ORTHONORMALIZE AS NORM = 1
      DO 550 I=1,NVA
      IF(I.EQ.1) GO TO 520
      IF(DABS(E(I)-E(I-1)).GE.EPS1) GO TO 520
C  ORTHONORMALIZE EIGENVECTORS FOR DEGENERATED EIGENVALUES
      I1=I-1
      DO 510 J=M,I1
      S=0.
      DO 500 K=1,N
  500 S=S+V(K,J)*V(K,I)
```

```
      DO 510 K=1,N
  510 V(K,I)=V(K,I)-S*V(K,J)
      GO TO 530
  520 M=I
C  NORMALIZE AS NORM = 1
  530 S=0.
      DO 540 J=1,N
  540 S=S+V(J,I)*V(J,I)
      T=0.0
      IF ( S.NE.0.0 ) T = DSQRT(1.0D0/S)
      DO 550 J=1,N
  550 V(J,I)=V(J,I)*T
      RETURN
 1000 FORMAT(1H0,'(SUBR. DEIGRS) NE=',I5,', NE SHOULD BE NON-ZERO. RETUR
     *N WITH NO CALCULATION.')
 2000 FORMAT(1H0,'(SUBR. DEIGRS) NV=',I5,', NE=',I5,', N=',I5,', N1=',I5
     *,1H,/1H ,'NV, NE, N, N1 SHOULD SATISFY THE FOLLOWING INEQUALITIES,
     * !NV! <= !NE! <= N <= N1.'/1H ,'RETURN WITH NO CALCULATION.' )
      END
      SUBROUTINE ERRSET(I,J,K,L)
      RETURN
      END
      SUBROUTINE OVERFL(L)
      RETURN
      END
```

3. 状態密度を計算するプログラム

入力ファイル : Smn60.dat (H60am.f,lenham.f により出力)
: energyV60.dat (c60.f により出力)
: energyV60.edt (c60.f により出力)
: SIZES
出力ファイル : jm-all.dat2 1 原子あたり状態密度 (xvgr 用)
: jm-s.dat2 2s 軌道の状態密度
: jm-px.dat2 2px 軌道の状態密度
: jm-py.dat2 2py 軌道の状態密度
: jm-pz.dat2 2pz 軌道の状態密度

```
c
c     状態密度の計算プログラム
c
      implicit real*8 (a-z)
      integer NSIZE,AN,M,N,I,J,A_N
C d95.f 用
      integer n2,nm,nc,NUMNUM
C------
      INCLUDE 'SIZES'
C d95.f 用
      parameter (n2=800,nm=240)
C------
      dimension C(NSIZE,NSIZE),S(NSIZE,NSIZE),E(NSIZE),
     #     CN(NSIZE,NSIZE),amoall(NSIZE),amos(NSIZE),amopx(NSIZE),
     #     amopy(NSIZE),amopz(NSIZE)
C d95.f 用
      dimension yall(n2),ys(n2),ypx(n2),ypy(n2),ypz(n2),xc(n2)
C------
c
c
c     行列の読み込み
c     エネルギー固有値
c     energyV.edt の形式は E(k)

      open(18,FILE='energyV60.edt')
      do 9 I=1,NSIZE
         read(18,*) E(I)
 9    continue
      close(18,status='keep')

c
c     固有ベクトル
c     energyV.dat の形式は C(i,k) i: 原子軌道 k: エネルギー固有値 E(k)
c     に対応
c     読み込む時は C(k,i) に変更
c
```

```
      open(19,FILE='energyV60.dat')
      read(19,*) A_N
      do 10 M=1,NSIZE
         do 11 N=1,NSIZE
c             write(*,*) M,N
            read(19,*) C(N,M)
 11      continue
 10   continue
      close(19,status='keep')

c
c     Smn60.dat の形式は S(m,n) m,n: 原子軌道に対応
c
      open(21,FILE='Smn60.dat')
      do 12 M=1,NSIZE
         do 13 N=1,NSIZE
            read(21,*) S(M,N)
 13      continue
 12   continue
      close(21,status='keep')
c
c
c     行列の計算 + 規格化
      DO 20 M=1,NSIZE
         DO 21 N=1,NSIZE
            CN(M,N)=C(M,N)*S(M,N)
 21      continue
         TMP=0.0
         DO 22 N=1,NSIZE
            TMP=CN(M,N)*CN(M,N)+TMP
 22      continue
         DO 23 N=1,NSIZE
            CN(M,N)=CN(M,N)/sqrt(TMP)
 23      continue
         TMP2=0.0
         do 24 N=1,NSIZE
            TMP2=TMP2+CN(M,N)*CN(M,N)
 24      continue
c         write(*,*) TMP2
 20   continue
      do 25 i=1,NSIZE
         amos(i) = 0.0
         amopx(i)= 0.0
         amopy(i)= 0.0
         amopz(i)= 0.0
 25   continue

C      DO 125 M=1,NSIZE
C         TMP3=0.0
C         DO 126 N=1,NSIZE
C            TMP3=TMP3+CN(M,N)*CN(M,N)
C 126     continue
C         write(*,*) TMP3
C 125  continue
C
      nc=0
C
      do 30 M=1,NSIZE
         do 31 N=1,AN
C     E(M) に対する s,px,py,pz の成分
C     s
            amos(M)=amos(M)+CN(M,(N-1)*4+1)*CN(M,(N-1)*4+1)
C     px
            amopx(M)=amopx(M)+CN(M,(N-1)*4+2)*CN(M,(N-1)*4+2)
C     py
            amopy(M)=amopy(M)+CN(M,(N-1)*4+3)*CN(M,(N-1)*4+3)
C     pz
            amopz(M)=amopz(M)+CN(M,(N-1)*4+4)*CN(M,(N-1)*4+4)
C
            nc = nc+1
 31      continue
C
         amoall(M)=amos(M)+amopx(M)+amopy(M)+amopz(M)
C         write(*,*) amoall(M)
 30   continue
      nc = int(nc/nm)
```

```
c      do 39 i=1,NSIZE
c         SUMS=amoall(i)
c 39   continue
c      write(*,*) SUMS

c     d95.fより
c-----------------------------------------
C      nm=240
C      n2=400

      dx=(abs(E(1)-E(nm))+10.0)/float(n2-1)
      de=dx
      write(*,*) 'de = ',de
      de2=de*de
      dd=16.0*de

      do i=1,n2
c
         xc(i)=float(int(e(1))-5.0)+dx*float(i-1)
c
         ys(i)= 0.0
         ypx(i)= 0.0
         ypy(i)= 0.0
         ypz(i)= 0.0
C         yspxpy(i)=0.0
         yall(i)=0.0
c
      end do


      do i= 1,nm
         do j=1,n2
            if(abs(E(i)-xc(j)).lt.dd) then
               ys(j)=ys(j)+exp(-(E(i)-xc(j))**2/(de2*16.0))*amos(i)
               ypx(j)=ypx(j)+exp(-(E(i)-xc(j))**2/(de2*16.0))*amopx(i)
               ypy(j)=ypy(j)+exp(-(E(i)-xc(j))**2/(de2*16.0))*amopy(i)
               ypz(j)=ypz(j)+exp(-(E(i)-xc(j))**2/(de2*16.0))*amopz(i)
C     yspxpy(j)=yspxpy(j)+exp(-(E(i)-xc(j))**2/de2)*amospxpy(i)
               yall(j)=yall(j)+exp(-(E(i)-xc(j))**2/(de2*16.0))
     #              *amoall(i)
            endif
         end do
      end do


c--------------------------- c sc: Scaling Factor
      sc=1.0/(float(nc)*de*4.0*sqrt(3.1415926535))
      write(*,*) 'scaling factor = ',sc
      NUMNUM=1
c
      do i=1,n2
         ys(i)=ys(i)*sc
         ypx(i)=ypx(i)*sc
         ypy(i)=ypy(i)*sc
         ypz(i)=ypz(i)*sc
C         yspxpy(i)=yspxpy(i)*sc
         yall(i)=yall(i)*sc
c
         sumys=sumys+ys(i)
         sumypx=sumypx+ypx(i)
         sumypy=sumypy+ypy(i)
         sumypz=sumypz+ypz(i)
C         sumyspxpy=sumyspxpy+yspxpy(i)
         sumyall=sumyall+yall(i)
         if(NUMNUM.eq.1) then
            if((sumyall*dx).GE.(3.9999/2.0))then
               write(*,*) xc(i)
               NUMNUM=0
            end if
         end if
      end do

         sumys=sumys*dx
         sumypx=sumypx*dx
```

```
         sumypy=sumypy*dx
         sumypz=sumypz*dx
C         sumyspxpy=sumyspxpy*dx
         sumyall=sumyall*dx
c
      write(*,*) 'Sum yall=8 ',sumyall
      write(*,*) 'Sum ys=2',sumys
      write(*,*) 'Sum ypx=2',sumypx
      write(*,*) 'Sum ypy=2',sumypy
      write(*,*) 'Sum ypz=2',sumypz
C      write(*,*) 'Sum yspxpy=6 ',sumyspxpy
c
C      j=index(jobname,' ')-1
c
c

      open(62,file='jm-all.dat2')
      do i=1,n2
         write(62,1000) xc(i), yall(i)
      end do
      write(62,*) '#'
      close(62)
      open(64,file='jm-s.dat2')
      do i=1,n2
         write(64,1000) xc(i), ys(i)
      end do
      write(64,*) '#'
      close(64)

      open(65,file='jm-px.dat2')
      do i=1,n2
         write(65,1000) xc(i), ypx(i)
      end do
      write(65,*) '#'
      close(65)

      open(66,file='jm-py.dat2')
      do i=1,n2
         write(66,1000) xc(i),ypy(i)
      end do
      write(66,*) '#'
      close(66)

      open(67,file='jm-pz.dat2')
      do i=1,n2
         write(67,1000) xc(i),ypz(i)
      end do
      write(67,*) '#'
      close(67)

 1000 format(f10.5,f13.8)
 1010 format(f10.5,a)
 1005 format(I4)
      stop
      end
```

4. チェーンモデルをつくるプログラム

入力ファイル : c60.xyz 原子の座標データ
: c60.sizes プログラムのパラメーター
出力ファイル : n.dat 基底の中の原子の番号のデータ
: Ares.dat an の計算のためのデータ
(n の中での隣接する原子の情報)
: Bres.dat bn の計算のためのデータ
(n と n-1 での隣接原子の情報)

: N.par ハミルトニアン行列計算プログラム
(Sham.f,Lhamr.f,Lhamlen.f) パラメータ定義ファイル

: ZAHYOU.dat
Xmol で見るための途中までのチェーンモデルの座標データ

```
---------------------------------
      implicit real*8(a-y)
      implicit complex*16(z)
      CHARACTER*20 DUM_CHA
      INTEGER I,J,K,L,N,COUNT,AN,MVOL,NVOL,MBANG,NBANG,NCOUNT,
```

```
     #     MOTO,NOKORI,CHECK,N_MAX,DAT_MAX,NSIZE,MAXSIZE,MAXD
C     MOTO   :N にある原子の原子番号
C     NOKORI :N での残りの原子番号
C     N_MAX  :N の最大数
C     AN     : 原子数
C     DAT_MAX:MOTO の最大数

      INCLUDE 'c60.sizes'
      parameter(KIJUN=1.6)
      dimension R(AN,3)
      dimension MOTO(N_MAX,DAT_MAX)
      dimension NOKORI(N_MAX,AN-1)
      dimension MVOL(N_MAX)
      dimension NVOL(N_MAX)
      MAXSIZE=0
      open(20,FILE='c60.xyz')
      read(20,*) DUM

      DO 1001 COUNT=1,AN
         read(20,*) DUM_CHA,R(COUNT,1),R(COUNT,2),R(COUNT,3)
C         DO 1002 L=1,3
C            read(20,*) R(COUNT,L)
C 1002    continue
 1001 continue
      close(20,STATUS='keep')

      MVOL(1)=1
      NVOL(1)=AN-1
      MOTO(1,1)=1

      DO 12 N=1,N_MAX
         DO 11 I=1,AN-1
            NOKORI(N,I)=0
 11      continue
 12   continue
      DO 10 I=1,AN-1
         NOKORI(1,I)=I+1
 10   continue

      open(22,FILE='Bres.dat')
      DO 101 N=1,N_MAX
         MBANG=0
         NBANG=0
         DO 102 J=1,NVOL(N)
            CHECK=0
            DO 103 K=1,MVOL(N)
C              if(NOKORI(N,J).eq.MOTO(N,K)) STOP
               DO 104 L=1,3
                  if(ABS(R(NOKORI(N,J),L)-R(MOTO(N,K),L)).GE.KIJUN)
     #                then
                     go to 103
                  endif
 104           continue
               RLEN_2=0.0
               DO 401 L=1,3
                  RLEN_2 = RLEN_2+(R(NOKORI(N,J),L)-R(MOTO(N,K),L))
     #                 *(R(NOKORI(N,J),L)-R(MOTO(N,K),L))
 401           continue
               if(RLEN_2.GE.(KIJUN*KIJUN)) go to 103
C     隣であった場合ここを通る
               write(22,*) K,MOTO(N,K),NOKORI(N,J),RLEN_2
               if(CHECK.eq.0) then
               MBANG=MBANG+1
               if(MBANG.GE.DAT_MAX+1) then
                  write(*,*) 'DAT_MAX が足りません'
                  close(22,STATUS='keep')
                  STOP
               end if
               MOTO(N+1,MBANG) = NOKORI(N,J)
               CHECK=1
               end if
 103        continue
            if(CHECK.eq.0)then
C     隣ではなかった時
               NBANG=NBANG+1
               NOKORI(N+1,NBANG)=NOKORI(N,J)
            end if
```

```
 102     continue
         MVOL(N+1)=MBANG
C         MBANG=0
         NVOL(N+1)=NBANG
         IF(NBANG.GE.MAXSIZE)then
            MAXSIZE=NBANG
         END IF
C         NBANG=0
         write(22,*) '  0  0',MVOL(N+1),'  0'
         if(NBANG.eq.0) then
            NSIZE=N+1
            go to 100
         end if
 101   continue
 100   if(NBANG.GE.0.1) then
         write(*,*) 'N_MAXが足りません'
         close(22,STATUS='keep')
         STOP
       end if
       close(22,STATUS='keep')
       open(21,FILE='res.dat')
C       write(*,*) N
       DO 200 N=1,10
         write(21,*) MVOL(N)
         IF(MVOL(N).GE.MAXD)then
            MAXD=MVOL(N)
         END IF
 200   continue
       close(21,STATUS='keep')
       open(22,FILE='n.dat')
       DO 201 N=1,NSIZE
         DO 202 I=1,MVOL(N)
            write(22,*) MOTO(N,I)
 202     continue
         write(22,*) '0'
 201   continue
       close(22,STATUS='keep')
       open(23,FILE='Ares.dat')
       DO 301 N=1,NSIZE
C          write(23,*) N
         NCOUNT=0
         DO 302 I=1,MVOL(N)
            DO 303 J=I,MVOL(N)
               DO 304 L=1,3
                  if(ABS(R(MOTO(N,I),L)-R(MOTO(N,J),L)).GE.KIJUN)
     #                 then
                     go to 303
                  endif
 304           continue
               RLEN_2=0.0
               DO 305 L=1,3
                  RLEN_2 = RLEN_2+(R(MOTO(N,I),L)-R(MOTO(N,J),L))
     #                 *(R(MOTO(N,I),L)-R(MOTO(N,J),L))
 305           continue
               if(RLEN_2.GE.(KIJUN*KIJUN)) go to 303
               write(23,*) MOTO(N,I),MOTO(N,J),J,RLEN_2
               NCOUNT=NCOUNT+1
 303        continue
 302     continue
         IF(NCOUNT.GE.MAXSIZE)then
            MAXSIZE=NCOUNT
         END IF
         write(23,*) '  0   0   0   0'
 301   continue
       close(23,STATUS='keep')
       write(*,*) 'Nは',NSIZE,'まであります'

       open(24,FILE='N.par')
       write(24,*) '      parameter('
       write(24,*) '     #     AN=',AN,','
       write(24,*) '     #     MAX=',MAXSIZE+1,','
       write(24,*) '     #     MAXD=',MAXD,','
       write(24,*) '     #     NSIZE=',NSIZE-1,')'
       close(24,STATUS='keep')
       open(25,FILE='ZAHYOU.dat')
       write(25,*) '60'
```

```
      write(25,*) ' '
      DO 501 N=1,NSIZE
         DO 502 I=1,MVOL(N)
            write(25,*) 'C',R(MOTO(N,I),1),R(MOTO(N,I),2),R(MOTO(N,I),3)
 502     continue
 501  continue
      close(25,STATUS='keep')
      STOP
      END
```

5. リカージョン法による状態密度計算プログラム

入力ファイル : c60.xyz
: n.dat(Lsoot1.f により出力)
: Ares.dat(同上)
: Bres.dat(同上)
: N.par パラメータ定義ファイル (同上)
出力ファイル : ANres.dat an の計算結果
(Lhamr.f) : BNres.dat bn の計算結果
: C60mitsu.plt 状態密度

```
---------------------------------------
      implicit real*8 (a-h,o-y)
      implicit integer*4 (i-n)
      implicit complex*16 (z)
      real*8 N_xyz,hehe,KEISUU
      integer AN,cou1,WIDTH_L,WIDTH
      INCLUDE 'N.par'
      parameter(
     #     e2s=(-8.868),
     #     WIDTH_L=4000,
     #     pi=3.14159265358979323846)
      CHARACTER*20 DUM_CHA
      dimension R(AN,3),AVOL(0:NSIZE),BVOL(NSIZE),NVOL(0:NSIZE),
     #     ARESU(0:NSIZE),B(0:NSIZE),RESUB(0:NSIZE,MAXD*4),
     #     MOTOA_1(0:NSIZE,MAX),MOTOA_2(0:NSIZE,MAX),
     #     MOTOA_3(0:NSIZE,MAX),
     #     MOTOB_1(NSIZE,MAX),MOTOB_2(NSIZE,MAX),MOTOB_T(NSIZE,MAX),
     #     ALEN_2(0:NSIZE,MAX),BLEN_2(NSIZE,MAX),R_xyz(3),N_xyz(3),
     #     RE(WIDTH_L*2+1),NDAT(0:NSIZE,MAX),
     #     Hmn(MAXD*4,MAXD*4),Smn(MAXD*4,MAXD*4),
     #     hehe(4,4),sese(4,4),haha(4,4),
     #     KEISDAT(NSIZE,MAXD*4),KEISMAX(MAXD*4)


      Er=0.01
      ZEi=(0.0,0.3)



c     原子の座標の読み込み
c
      open(20,FILE='c60.xyz',STATUS='old')
      read(20,*) II
      DO 10 cou1=1,AN
         read(20,*) DUM_CHA,R(cou1,1),R(cou1,2),R(cou1,3)
c         write(*,*) R(cou1,1),R(cou1,2),R(cou1,3)
 10   continue
      close(20,status='keep')
      open(19,FILE='n.dat')
      DO 21 N=0,NSIZE
         DO 22 I=1,MAX
            read(19,*) NDAT(N,I)
            if(NDAT(N,I).eq.0) then
               NVOL(N)=I-1
               go to 21
            end if
 22      continue
 21   continue

      open(21,FILE='Ares.dat')
      DO 101 N=0,NSIZE
         DO 102 I=1,MAX
        read(21,*) MOTOA_1(N,I),MOTOA_2(N,I),MOTOA_3(N,I),ALEN_2(N,I)
            if(MOTOA_1(N,I).eq.0) then
```

```
               AVOL(N)=I-1
               go to 101
            end if
 102     continue
 101   continue
       close(21,STATUS='keep')
       open(22,FILE='Bres.dat')
       DO 103 N=1,NSIZE
          DO 104 I=1,MAX
             read(22,*) MOTOB_T(N,I),MOTOB_1(N,I),MOTOB_2(N,I),
     #            BLEN_2(N,I)
             if(MOTOB_1(N,I).eq.0) then
                BVOL(N)=I-1
                go to 103
             end if
 104     continue
 103   continue
       close(22,STATUS='keep')
C      データの読み込み終了
C
C      A0 の計算
       N=0
       J=0
       ARESU(N)=0.0
       DO 151 I=1,AVOL(N)
          if(ALEN_2(N,I).lt.0.1) then
C      KEISMAX(J):J の原子の隣の原子の数
             J=J+1
             KEISMAX(J)=0
C      H 行列
             DO 161 K=1,4
                DO 162 L=1,4
                   Sese(K,L)=0.0
 162            continue
                haha(K,K)=e2p
                Sese(K,K)=1.0
 161         continue
             haha(1,1)=e2s
          else
C      hehe,sese の計算
             R_xyz(1)=R(MOTOA_1(N,I),1)
             R_xyz(2)=R(MOTOA_1(N,I),2)
             R_xyz(3)=R(MOTOA_1(N,I),3)
             N_xyz(1)=R(MOTOA_2(N,I),1)
             N_xyz(2)=R(MOTOA_2(N,I),2)
             N_xyz(3)=R(MOTOA_2(N,I),3)
             RLEN_2=ALEN_2(N,I)
             call Hij(R_xyz,N_xyz,RLEN_2,haha,sese)
          end if
C      行列の値の代入
          DO 152 K=1,4
             DO 153 L=1,4
                SUM0=SUM0+sese(K,L)
                ARESU(N)=ARESU(N)+haha(K,L)
 153         continue
 152      continue
 151   continue
       DO 154 K=1,4
          RESUB(0,K)=1.0/sqrt(SUM0)
 154   continue
       ARESU(0)=ARESU(0)/SUM0
       B(0)=1
C
C      N について
       DO 201 N=1,NSIZE
C          write(*,*) N
          DO 501 K=1,NVOL(N)*4
             DO 502 L=1,NVOL(N)*4
                Smn(K,L)=0.0
 502         continue
 501      continue
C      B(n) について
          DO 302 I=1,BVOL(N)
C      I が最初
             if(I.eq.1)then
                J=1
```

```
C      最初ではない時
            else
C      前回と同じ原子番号の時 CHECK=1
               if(MOTOB_2(N,I-1).eq.MOTOB_2(N,I))then
                  CHECK=1
C      違う原子番号の時 CHECK=0
               else
                  CHECK=0
                  J=J+1
               end if
            end if
C
            R_xyz(1)=R(MOTOB_1(N,I),1)
            R_xyz(2)=R(MOTOB_1(N,I),2)
            R_xyz(3)=R(MOTOB_1(N,I),3)
            N_xyz(1)=R(MOTOB_2(N,I),1)
            N_xyz(2)=R(MOTOB_2(N,I),2)
            N_xyz(3)=R(MOTOB_2(N,I),3)
            RLEN_2=BLEN_2(N,I)
C            write(*,*) J
            call Hb(R_xyz,N_xyz,RLEN_2,haha)
C      もとの原子番号が同じ時 前回の結果 RESUB にたす
            if(CHECK.eq.1) then
               RESUB(N,J*4-3)=RESUB(N,J*4-3) +
     #              Haha(1,1)*RESUB(N-1,MOTOB_T(N,I)*4-3) +
     #              Haha(2,1)*RESUB(N-1,MOTOB_T(N,I)*4-2) +
     #              Haha(3,1)*RESUB(N-1,MOTOB_T(N,I)*4-1) +
     #              Haha(4,1)*RESUB(N-1,MOTOB_T(N,I)*4)
               RESUB(N,J*4-2)=RESUB(N,J*4-2) +
     #              Haha(1,2)*RESUB(N-1,MOTOB_T(N,I)*4-3) +
     #              Haha(2,2)*RESUB(N-1,MOTOB_T(N,I)*4-2) +
     #              Haha(3,2)*RESUB(N-1,MOTOB_T(N,I)*4-1) +
     #              Haha(4,2)*RESUB(N-1,MOTOB_T(N,I)*4)
               RESUB(N,J*4-1)=RESUB(N,J*4-1) +
     #              Haha(1,3)*RESUB(N-1,MOTOB_T(N,I)*4-3) +
     #              Haha(2,3)*RESUB(N-1,MOTOB_T(N,I)*4-2) +
     #              Haha(3,3)*RESUB(N-1,MOTOB_T(N,I)*4-1) +
     #              Haha(4,3)*RESUB(N-1,MOTOB_T(N,I)*4)
               RESUB(N,J*4)  =RESUB(N,J*4)   +
     #              Haha(1,3)*RESUB(N-1,MOTOB_T(N,I)*4-3) +
     #              Haha(2,3)*RESUB(N-1,MOTOB_T(N,I)*4-2) +
     #              Haha(3,3)*RESUB(N-1,MOTOB_T(N,I)*4-1) +
     #              Haha(4,3)*RESUB(N-1,MOTOB_T(N,I)*4)
C      違う時 新しい結果 RESUB をつくる
            else
               RESUB(N,J*4-3)=Haha(1,1)*RESUB(N-1,MOTOB_T(N,I)*4-3) +
     #              Haha(2,1)*RESUB(N-1,MOTOB_T(N,I)*4-2) +
     #              Haha(3,1)*RESUB(N-1,MOTOB_T(N,I)*4-1) +
     #              Haha(4,1)*RESUB(N-1,MOTOB_T(N,I)*4)
               RESUB(N,J*4-2)=Haha(1,2)*RESUB(N-1,MOTOB_T(N,I)*4-3) +
     #              Haha(2,2)*RESUB(N-1,MOTOB_T(N,I)*4-2) +
     #              Haha(3,2)*RESUB(N-1,MOTOB_T(N,I)*4-1) +
     #              Haha(4,2)*RESUB(N-1,MOTOB_T(N,I)*4)
               RESUB(N,J*4-1)=Haha(1,3)*RESUB(N-1,MOTOB_T(N,I)*4-3) +
     #              Haha(2,3)*RESUB(N-1,MOTOB_T(N,I)*4-2) +
     #              Haha(3,3)*RESUB(N-1,MOTOB_T(N,I)*4-1) +
     #              Haha(4,3)*RESUB(N-1,MOTOB_T(N,I)*4)
               RESUB(N,J*4)  =Haha(1,4)*RESUB(N-1,MOTOB_T(N,I)*4-3) +
     #              Haha(2,4)*RESUB(N-1,MOTOB_T(N,I)*4-2) +
     #              Haha(3,4)*RESUB(N-1,MOTOB_T(N,I)*4-1) +
     #              Haha(4,4)*RESUB(N-1,MOTOB_T(N,I)*4)
            end if
 302     continue
C         RESUB() を sqrt(B(N-1)) でわる.
         DO 601 M=1,NVOL(N)*4
            RESUB(N,M)=RESUB(N,M)/sqrt(B(N-1))
 601     continue
C      An と Bn の前の計算
         J=0
         ARESU(N)=0.0
         DO 202 I=1,AVOL(N)
            if(ALEN_2(N,I).lt.0.1) then
C      KEISMAX(J):J の原子の隣の原子の数
               J=J+1
               KEISMAX(J)=0
C      K: 行列を入れる場所
C                K=J
```

```
C     H行列S行列
                DO 220 K=1,4
                   DO 221 L=1,4
                      Hehe(K,L)=0.0
                      Sese(K,L)=0.0
 221               continue
                   Hehe(K,K)=e2p
                   Sese(K,K)=1.0
 220            continue
                Hehe(1,1)=e2s
             else
C     KEISDAT:JについてJ以外で計算する場所の記憶番号
                KEISMAX(J)=KEISMAX(J)+1
                KEISDAT(J,KEISMAX(J))=MOTOA_3(N,I)
C     hehe,seseの計算
                R_xyz(1)=R(MOTOA_1(N,I),1)
                R_xyz(2)=R(MOTOA_1(N,I),2)
                R_xyz(3)=R(MOTOA_1(N,I),3)
                N_xyz(1)=R(MOTOA_2(N,I),1)
                N_xyz(2)=R(MOTOA_2(N,I),2)
                N_xyz(3)=R(MOTOA_2(N,I),3)
                RLEN_2=ALEN_2(N,I)
                call Hij(R_xyz,N_xyz,RLEN_2,hehe,sese)
             end if
C     Jが自分の原子の通し番号 MOTOA_3(N,I)が相手の原子の通し番号
             DO 200 k=1,4
                DO 100 l=1,4
                   Hmn((J-1)*4+k,(MOTOA_3(N,I)-1)*4+l)=Hehe(k,l)
                   Smn((J-1)*4+k,(MOTOA_3(N,I)-1)*4+l)=Sese(k,l)
                   Hmn((MOTOA_3(N,I)-1)*4+k,(J-1)*4+l)=Hehe(l,k)
                   Smn((MOTOA_3(N,I)-1)*4+k,(J-1)*4+l)=Sese(l,k)
 100            CONTINUE
 200         CONTINUE
 202      continue
C
C     B(n)
         DO 303 K=1,NVOL(N)*4
            DO 304 L=1,NVOL(N)*4
               Smn(K,L)=Smn(K,L)*RESUB(N,K)*RESUB(N,L)
 304        continue
 303     continue
         B(N)=0.0
         DO 305 K=1,NVOL(N)*4
            DO 306 L=1,NVOL(N)*4
               B(N)=B(N)+Smn(K,L)
 306        continue
 305     continue
C     Bn終了
C     Anの計算
         ARESU(N)=0.0
         DO 199 K=1,NVOL(N)*4
            DO 99 L=1,NVOL(N)*4
               Hmn(K,L)=Hmn(K,L)*
     #              RESUB(N,K)*
     #              RESUB(N,L)/B(N)
               ARESU(N)=ARESU(N)+
     #              Hmn(K,L)
 99         CONTINUE
 199     CONTINUE
C          ARESU(N)=ARESU(N)/(B(N))
C
C     Nについて終了
 201  continue
C      DO 110 N=2,NSIZE-1
C         BHsum(N)=BHsum(N)/BHsum(N-1)
C 110  continue
C      DO 111 N=1,NSIZE-1
C         AHsum(N)=AHsum(N)/BHsum(N)
C 111  continue

      open(24,FILE='ANres.dat')
      DO 121 N=0,NSIZE
         write(24,*) N,ARESU(N)
 121  continue
      close(24,STATUS='keep')
      open(25,FILE='BNres.dat')
      DO 122 N=1,NSIZE
```

```
         write(25,*) N,sqrt(B(N))
 122   continue
       close(25,STATUS='keep')

c      Er での Green 関数を求める
       open(26,FILE='C60mitsu.plt')
       DO 30 WIDTH=(-WIDTH_L),WIDTH_L
          ZE = Er*(WIDTH) + ZEi
          ZBnn=(0.0,0.0)
          ZBnn=ZBnn+(B(NSIZE)/(ZE-ARESU(NSIZE)))
          DO 31 N=1,NSIZE-1
             ZBnn=(B(NSIZE-N))/(ZE-ARESU(NSIZE-N)-ZBnn)
 31       continue
          RE(WIDTH_L+WIDTH+1)=(-imag(1.0/(ZE-ARESU(0)-ZBnn)))/pi
          write(26,*) Er*(WIDTH),4.0*RE(WIDTH_L+WIDTH+1)
 30    continue
       close(26,STATUS='keep')

       STOP
       END
c
c


c      ------------------------------------------------
c      AN を計算するプログラム
c
       subroutine Hij(R_xyz,N_xyz,RLEN_2,Hehe,Sese)
c
       parameter(
      #      Hss=(-6.769),
      #      Hsp=(-5.580),
      #      Hsg=(-5.037),
      #      Hpi=(-3.033),
      #      Sss=0.212,
      #      Ssp=0.102,
      #      Ssg=0.146,
      #      Spi=0.129)
c
c      型宣言
       implicit real*8(a-z)
       dimension N_xyz(3),R_xyz(3),Hehe(4,4),Sese(4,4)
c
c
       rx=R_xyz(1)
       ry=R_xyz(2)
       rz=R_xyz(3)
       n1x=N_xyz(1)
       n1y=N_xyz(2)
       n1z=N_xyz(3)
       ZERO=0.0
       ONE=1.0
c
c      -------- 計算式 -------
c
c      |Rn-R|


       R1L_1=sqrt(RLEN_2)
c
c
c       COSx （N 側の成分からみて R 方向の COS）
       cosx = CSTH(rx,n1x,R1L_1)
       cosxx = cosx**2
c       COSy
       cosy = CSTH(ry,n1y,R1L_1)
       cosyy = cosy**2
c       COSz
       cosz = CSTH(rz,n1z,R1L_1)
       coszz = cosz**2
c
c      H の計算式
c
c      Hsp,Ssp
c
c      値の代入
       Hspx = Hsp * cosx
```

```
      Hspy = Hsp * cosy
      Hspz = Hsp * cosz
c
      Sspx = Ssp * cosx
      Sspy = Ssp * cosy
      Sspz = Ssp * cosz
c
c     Hpapa,Spapa
c
c     値の代入
c     Hpxpx
      Hpxpx1 = Hsg * (-(cosxx))
C
      Hpxpx2 = Hpi * (1 - cosxx)
c
c     Hpypy
      Hpypy1 = Hsg * (-(cosyy))
C
      Hpypy2 = Hpi * (1 - cosyy)
c
c     Hpzpz
      Hpzpz1 = Hsg * (-(coszz))
C
      Hpzpz2 = Hpi * (1 - coszz)
c
      Hpxpx = Hpxpx1 + Hpxpx2
      Hpypy = Hpypy1 + Hpypy2
      Hpzpz = Hpzpz1 + Hpzpz2
      Spxpx = Ssg * (-(cosxx)) + Spi * (1 - cosxx)
      Spypy = Ssg * (-(cosyy)) + Spi * (1 - cosyy)
      Spzpz = Ssg * (-(coszz)) + Spi * (1 - coszz)
c
c     Hpapb
c
c     値の代入
c
c     Hpxpy
      Hpxpy1 = Hsg * (-(cosx * cosy))
c
      Hpxpy2 = Hpi*nejire(rx,ry,rz,n1x,n1y,n1z,R1L_1,
     #     ONE,ZERO,ZERO,ZERO,ONE,ZERO)
c
c     Hpxpz
      Hpxpz1 = Hsg * (-(cosx * cosz))
c
      Hpxpz2 = Hpi*nejire(rx,ry,rz,n1x,n1y,n1z,R1L_1,
     #     ONE,ZERO,ZERO,ZERO,ZERO,ONE)
c
c     Hpypz
      Hpypz1 = Hsg * (-(cosy * cosz))
c
      Hpypz2 = Hpi*nejire(rx,ry,rz,n1x,n1y,n1z,R1L_1,
     #     ZERO,ONE,ZERO,ZERO,ZERO,ONE)
c
c
      Hpxpy = Hpxpy1 + Hpxpy2
      Hpxpz = Hpxpz1 + Hpxpz2
      Hpypz = Hpypz1 + Hpypz2
      Hpypx = Hpxpy
      Hpzpx = Hpxpz
      Hpzpy = Hpypz
      Spxpy = Hpxpy1*Ssg/Hsg + Hpxpy2*Spi/Hpi
      Spxpz = Hpxpz1*Ssg/Hsg + Hpxpz2*Spi/Hpi
      Spypz = Hpypz1*Ssg/Hsg + Hpypz2*Spi/Hpi
      Spypx = Spxpy
      Spzpx = Spxpz
      Spzpy = Spypz
c
c          Hsasa
      Hehe(1,1) = Hss
c
      Hehe(1,2) = Hsp * cosx
c          Hspy
      Hehe(1,3) = Hsp * cosy
c          Hspz
      Hehe(1,4) = Hsp * cosz
```

```
c           Hpxs
      Hehe(2,1) = -(Hehe(1,2))
c           Hpxpx
      Hehe(2,2) = Hpxpx
c           Hpxpy
      Hehe(2,3) = Hpxpy
c           Hpxpz
      Hehe(2,4) = Hpxpz
c           Hpys
      Hehe(3,1) = -(Hehe(1,3))
c           Hpypx
      Hehe(3,2) = Hpypx
c           Hpypy
      Hehe(3,3) = Hpypy
c           Hpypz
      Hehe(3,4) = Hpypz
c           Hpzs
      Hehe(4,1) = -(Hehe(1,4))
c           Hpzpx
      Hehe(4,2) = Hpzpx
c           Hpzpy
      Hehe(4,3) = Hpzpy
c           Hpzpz
      Hehe(4,4) = Hpzpz
      Sese(1,1) = Sss
      Sese(1,2) = Sspx
      Sese(1,3) = Sspy
      Sese(1,4) = Sspz
      Sese(2,1) = -(Sese(1,2))
      Sese(2,2) = Spxpx
      Sese(2,3) = Spxpy
      Sese(2,4) = Spxpz
      Sese(3,1) = -(Sese(1,3))
      Sese(3,2) = Spypx
      Sese(3,3) = Spypy
      Sese(3,4) = Spypz
      Sese(4,1) = -(Sese(1,4))
      Sese(4,2) = Spzpx
      Sese(4,3) = Spzpy
      Sese(4,4) = Spzpz
c
c
c
      return
      END
c     ------------------------------------------------
c     C60 の Bn を計算するプログラム
c
      subroutine Hb(R_xyz,N_xyz,RLEN_2,Hehe)
c
      parameter(
     #     Hss=(-6.769),
     #     Hsp=(-5.580),
     #     Hsg=(-5.037),
     #     Hpi=(-3.033))
c
c     型宣言
      implicit real*8(a-z)
      dimension R_xyz(3),N_xyz(3),Hehe(4,4)
C,tete(4)
c
c
c
c
c
      rx=R_xyz(1)
      ry=R_xyz(2)
      rz=R_xyz(3)
      n1x=N_xyz(1)
      n1y=N_xyz(2)
      n1z=N_xyz(3)
      ZERO=0.0
```

```
      ONE=1.0
c
c
c     -------- 計算式 -------
c
c     |Rn-R|


      R1L_1=sqrt(RLEN_2)
c      write(*,*) R1L_1
c
c
c
c      COSx （N 側の成分からみて R 方向の COS）
      cosx = CSTH(rx,n1x,R1L_1)
      cosxx = cosx**2
c      COSy
      cosy = CSTH(ry,n1y,R1L_1)
      cosyy = cosy**2
c      COSz
      cosz = CSTH(rz,n1z,R1L_1)
      coszz = cosz**2
c
c
c     H の計算式
c
c
c
c     Hsp,Ssp
c
c     値の代入
      Hpxs = Hsp * (-cosx)
C
      Hpys = Hsp * (-cosy)
C
      Hpzs = Hsp * (-cosz)
C
c
c     Hpapa,Spapa
c
c     値の代入
c     Hpxpx
      Hpxpx = Hsg * (-(cosxx)) + Hpi * (1 - cosxx)
c
c     Hpypy
      Hpypy = Hsg * (-(cosyy)) + Hpi * (1 - cosyy)
c
c     Hpzpz
      Hpzpz = Hsg * (-(coszz)) + Hpi * (1 - coszz)
C
c
c     Hpapb
c
c     値の代入
c
c     Hpxpy
      Hpxpy1 = Hsg * (-(cosx * cosy))
c
      Hpxpy2 = Hpi*nejire(rx,ry,rz,n1x,n1y,n1z,R1L_1,
     #     ONE,ZERO,ZERO,ZERO,ONE,ZERO)
c
c     Hpxpz
      Hpxpz1 = Hsg * (-(cosx * cosz))
c
      Hpxpz2 = Hpi*nejire(rx,ry,rz,n1x,n1y,n1z,R1L_1,
     #     ONE,ZERO,ZERO,ZERO,ZERO,ONE)
c
c     Hpypz
      Hpypz1 = Hsg * (-(cosy * cosz))
c
      Hpypz2 = Hpi*nejire(rx,ry,rz,n1x,n1y,n1z,R1L_1,
     #     ZERO,ONE,ZERO,ZERO,ZERO,ONE)
c
c
      Hpxpy = Hpxpy1 + Hpxpy2
      Hpxpz = Hpxpz1 + Hpxpz2
      Hpypz = Hpypz1 + Hpypz2
```

```
      Hpypx = Hpxpy
      Hpzpx = Hpxpz
      Hpzpy = Hpypz
      Hehe(1,1) = Hss
      Hehe(1,2) = Hspx
      Hehe(1,3) = Hspy
      Hehe(1,4) = Hspz
      Hehe(2,1) = -(Hehe(1,2))
      Hehe(2,2) = Hpxpx
      Hehe(2,3) = Hpxpy
      Hehe(2,4) = Hpxpz
      Hehe(3,1) = -(Hehe(1,3))
      Hehe(3,2) = Hpypx
      Hehe(3,3) = Hpypy
      Hehe(3,4) = Hpypz
      Hehe(4,1) = -(Hehe(1,4))
      Hehe(4,2) = Hpzpx
      Hehe(4,3) = Hpzpy
      Hehe(4,4) = Hpzpz
c
      return
      END
c
c     よく出てくる cos,sin の関数の定義
c------------ 副関数 --------------
      real*8 function CSTH(a,b,c)
      real*8 a,b,c
      if(abs(c).lt.1.0D-14) THEN
         CSTH = 0
         write(*,*) 'エラー:function CSTH'
         STOP
      ELSE
         CSTH = (a-b)/c
      END IF
      return
      end

c     ねじれ (π成分)
      real*8 function Nejire(R1x,R1y,R1z,Nx,Ny,Nz,LEN,
     #     RPx,RPy,RPz,NPx,NPy,NPz)
      implicit real*8 (a-z)
c      real*8 NNAI,RNAI,R1,R1x,R1y,R1z,Nx,Ny,Nz,NJ,RJ,NK,RK,
c     #     LEN,NPx,NPy,NPz,RPx,RPy,RPz
      dimension R1(3),NJ(3),RJ(3),NK(3),RK(3)
      parameter(
     #     Hpi=(-3.033))
c
c     R->N ベクトル
      R1(1)=Nx-R1x
      R1(2)=Ny-R1y
      R1(3)=Nz-R1z
c
c     NP の NR 方向の成分の大きさ + 正負
      NNAI=(R1(1)*NPx + R1(2)*NPy + R1(3)*NPz)/LEN
C      write(*,*) R1(1)*NPx
c
c     NP の NR 方向の成分
      NJ(1)=(R1(1)/LEN)*NNAI
      NJ(2)=(R1(2)/LEN)*NNAI
      NJ(3)=(R1(3)/LEN)*NNAI
c
c     NP の残りの成分
      NK(1)=NPx-NJ(1)
      NK(2)=NPy-NJ(2)
      NK(3)=NPz-NJ(3)
c
c     RP の NR 方向の成分の大きさ + 正負
      RNAI=(R1(1)*RPx + R1(2)*RPy + R1(3)*RPz)/LEN
c
c     RP の NR 方向の成分
      RJ(1)=(R1(1)/LEN)*RNAI
      RJ(2)=(R1(2)/LEN)*RNAI
      RJ(3)=(R1(3)/LEN)*RNAI
c
c     RP の残りの成分
```

```
      RK(1)=RPx-RJ(1)
      RK(2)=RPy-RJ(2)
      RK(3)=RPz-RJ(3)
c
      Nejire = (NK(1)*RK(1) + NK(2)*RK(2) + NK(3)*RK(3))
C      write(*,*) nejire
      return
      end
c
```