2000 年度 卒業論文

C_{60} 内包 カーボンナノチューブ の構造

電気通信大学 電子工学科 電子デバイス工学講座 4 年

9610115　中村 俊哉

指導教官 齋藤 理一郎 助教授

提出日 平成 13 年 2 月 8 日

目次

謝辞

本研究及び論文作成に当たり、御指導を賜わりました指導教官の齋藤理一郎助教授に厚く御礼の言葉を申しあげます。また研究室セミナー等にてさまざまな御指導を賜わりました、木村忠正教授、湯郷成美助教授に感謝致します。

そして、勉強や遊びに一緒にすごしてきた、木村・湯郷研究室の学生の皆様に感謝の意を表したいと思います。また、数々の助言を頂いた院生の皆様に感謝致します。そして、経済的援助と生活を支えくださった私の両親に感謝申し上げます。

2001 年 2 月

中村 俊哉

第 1 章

序論

1.1 カーボンナノチューブ

1.1.1 カーボンナノチューブの発見

カーボンナノチューブ (CNT:Carbon Nano tube 以下チューブまたは NT) は、グラファイトシートを継目が無いように筒状に巻いたような構造である。巻き方の違いによりさまざまな直径や螺旋度を持つ CNT が存在する。ナノチューブの構造を一般的に表現する方法として, カイラルベクトルという 2 つの整数値の指数がもちいられる。

そして、その構造を決める指数 (カイラル指数) で電子的性質 (金属・半導体) を一意に決められることが理論計算により明らかにされている [1][2][3]。

チューブには二種類のものがあり、太い直径を持つ NT の中に細い NT、更に細い NT というふうに何重もの入れ子構造の NT は、多層カーボンナノチューブ (MWNT:Multi-Wall Carbon Nanotube) と呼ばれる。対して一層のものを単層カーボンナノチューブ (SWNT:Single-Wall Carbon Nanotube) と呼ぶ。

MWNT は 1991 年に日本電気基礎研究所の飯島らのグループによって発見された。飯島らは C_{60} の生成過程において陰極先端に堆積した塊の中から偶然にも入れ子構造をとった MWNT を発見した。そしてこの時発見されたチューブは先端が丸く閉じており、巨大なフラーレンのような閉殻構造をとっていた。このことは幾何学的な考えから六員環のみならず五員環が炭素ネットワークを構成していることがわかる。

1.1.2 単層カーボンナノチューブの発見

MWNT の発見から二年後の 1993 年、鉄 [Fe] やコバルト [Co] などの磁性金属の超微粒子をグラファイトで包んだナノカプセルを合成する過程で、思いがけず SWNT が発見された。その後触媒作用をもつ金属としてニッケル [Ni] も挙げられた。

SWNT の長さと直径は金属触媒の種類に依存し、長いものは数 μm あり、直径は典型的には 1nm から 3nm までのものを得ることができる。最も細い直径で C_{60} とほぼ同じ [0.7nm] ものまである。これらの特徴は MWNT とは明らかに異なり、 SWNT がフラーレンにより近い構造を持っていることを示している。 MWNT 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、 SWNT はその物性が六員環ネットワークのトポロジーにより支配される特異な性質をもつ。また、分子とバルクの中間にある一次元物質として注目されている新物質でもある。

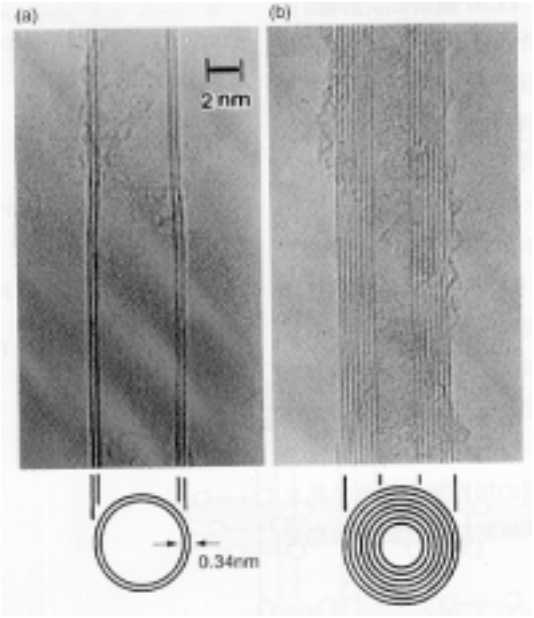

図 1.1 多層ナノチューブの SEM 像
Nature 354(1991)323 より引用[1]

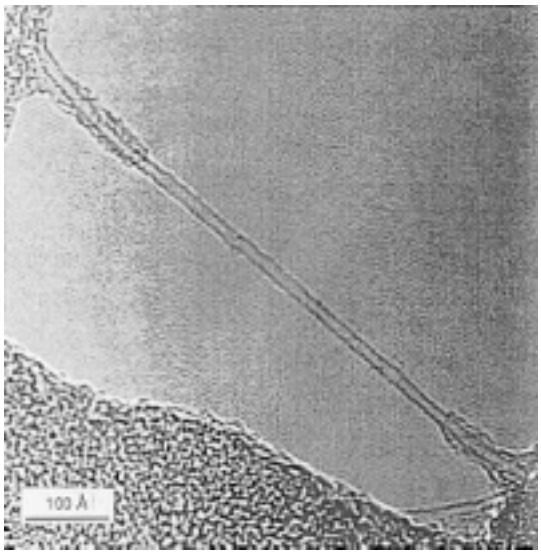

図 1.2 単層ナノチューブの SEM 像
Nature 363(1993)606 より引用[2]

[1]図 1.1 = u00naka/eps/mwcnt.eps

[2]図 1.2 = u00naka/eps/swnt.eps

1.2 SWNT の生成

単層ナノチューブの高密度、大量合成を可能にしたアーク放電法とレーザー蒸発法についてここで簡単に触れておく。どちらの方法においても炭素と同時に触媒となる金属を気相に供給する必要がある。[12]

1.2.1 アーク蒸発法

SWNT の合成に用いるアーク放電装置は他のチューブ以外のフラーレンを合成するときに用いるものとおなじものである。異なる点は用いる電極に SWNT の成長を促す金属触媒を含んだ炭素棒を用いることである。この炭素棒を不活性ガス (主として He) 中で蒸発させると金属 / 炭素混合蒸気のおよそ半分は気相中で凝集して煤を生成する。残りの半分は反対側の陰極先端に堆積する。SWNT はこれらの煤の中に含まれている。

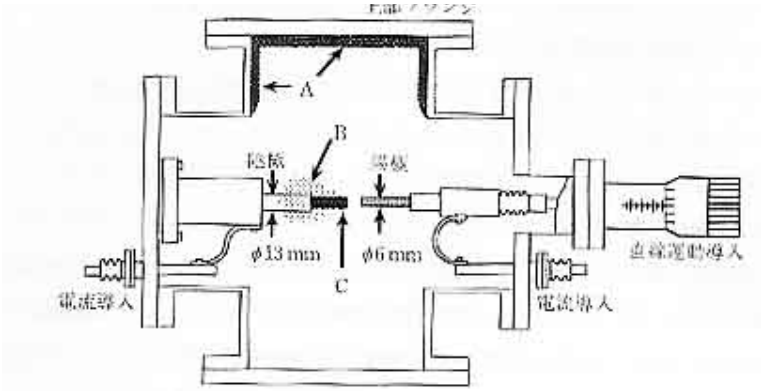

図 1.3 アーク放電法によるフラーレン / ナノチューブ合成装置
”カーボンナノチューブの基礎 ”より引用 [3]

SWNT 生成の触媒能を有することが明らかにされた金属元素は次の 3 つのグループに別けられる。

	族	主な触媒元素
1	鉄	Fe,Co,Ni
2	白金	Pd,Rh,Pt
3	希土類	La,Y

[3]図 1.3 = u00naka/eps/ark2.eps

ここで SWNT 生成の触媒として最もメジャーな Ni,Co といった鉄族触媒を用いた合成法に少し触れておく。 SWNT がはじめて合成されたのもこの触媒下であった。これら鉄族触媒を用いて生成される SWNT の直径は 0.7nm から 1.5nm の範囲にあり、C_{60} などのフラーレンのサイズと同じオーダーである。そして生成された SWNT は非常に長く、図 1.4に示すようにハイウェイジャンクションのようにチューブ (あるいは束) が互いに絡み合っている。チューブの生成過程を観察するのは困難なことだが、Ni 触媒では Ni 微粒子 (粒径 20-50 nm) から多数のチューブが放射状に生え出していることが観測されている。 [10]

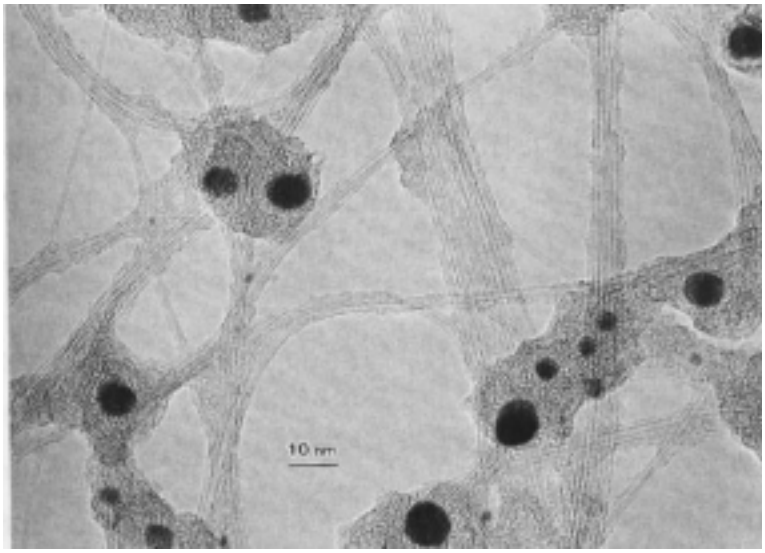

図 1.4 鉄族触媒下で合成された SWNT
”カーボンナノチューブの基礎 ”より引用 [4]

1.2.2 レーザー蒸発法

Smalley の研究グループは高温ガス中レーザー蒸発法を用いて混合触媒を使用し、70% 以上という高収率で SWNT を得ることに成功した。 [13] このような高い収率で SWNT を得るためには次の 2 点が必要な条件と考えられる。

1. チューブ成長空間の温度が 1,200 度 と非常に高いこと。

2. 炭素を均一に蒸発させること。

この方法で生成した SWNT の特徴として、次のようなことが挙げられる。

1. 直径約 1.3 nm を中心とした狭い範囲に分布

[4]図 1.4 = u00naka/eps/junction1.eps

2. アームチェア型チューブが多い

3. 規則正しく三角格子を組んで配列、結晶化したロープ (ナノロープ) を形成

レーザー蒸発法、アーク放電法ともに触媒金属を変えることによりそれぞれのチューブの直径を変化させることができる。また、おなじ触媒でも電気炉 (チューブ成長空間) の温度を下げることによってもチューブ直径を変化させることができる。しかしこの場合、 SWNT の収率は下がる。

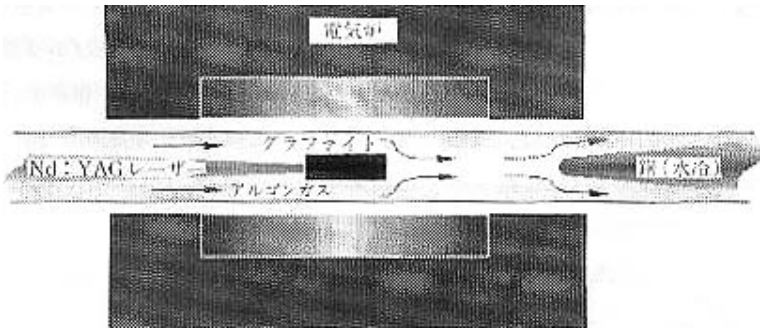

図 1.5 レーザー蒸発法によるフラーレン / ナノチューブ合成装置
”カーボンナノチューブの基礎 ”より引用 [5]

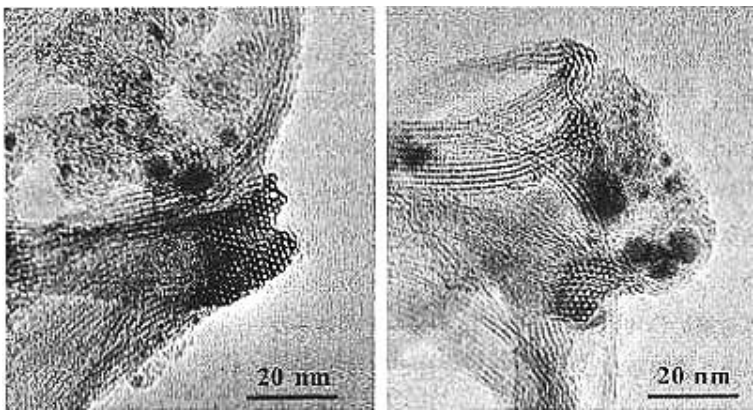

図 1.6 結晶化した SWNT ロープの TEM 像
”カーボンナノチューブの基礎 ”より引用 [6]

[5]図 1.5 = u00naka/eps/rezar2.eps

[6]図 1.6 = u00naka/eps/lope2.eps

1.3 フラーレン

フラーレンとは、C_{60} を代表とする炭素ネットワーク状物質である。Kroto や Smally らによって 1985 年に発見された。

1.3.1 フラーレンの構造

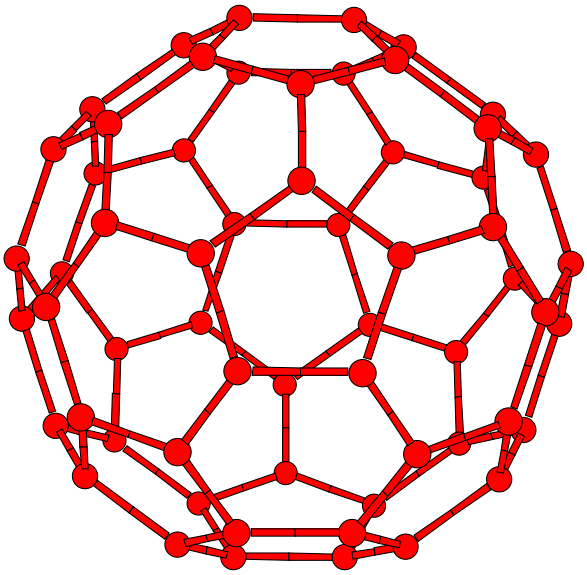

図 1.7 C_{60} の原子模型
R.Matsuo 修士論文 (1999) より引用 [7]

図 1.7に C_{60} の x-mol 像を示す。NMR による測定によると５角形と６角形の接する辺の結合 (5-6 bond) の長さは 1.447Å、６角形同士が接する辺の結合 (6-6 bond) の長さは 1.40Å である。5-6 の結合は 1 重結合的なので single-bond 、6-6 の結合は２重結合的なので double-bond と呼ばれる。この物質の構造を理論計算で表現することができる。構造最適化計算には第一原理による方法や、タイトバインディングによる方法などがあるが、それらの計算による結合長は実験結果と良く一致している。これらは計算量が原子の個数の３乗に比例 (Order N^3) かそれ以上であるが、岡田等は計算量が $O(N)$ になるよう経験ポテンシャルを用いての構造最適で C_{60} を表現することができる事を示した。[4]。

[7]図 1.7 = u00naka/eps/c60.eps

1.4 カーボンナノチューブの構造

カーボンナノチューブの構造は カイラルベクトルと 2 つの整数値のペアで指定することが出来る [1]。カイラルベクトル (C_h) を指定すると、チューブの直径 R やカイラル角 θ、チューブの並進ベクトル T、単位格子あたりの原子数 N を計算で求めることが出来る。

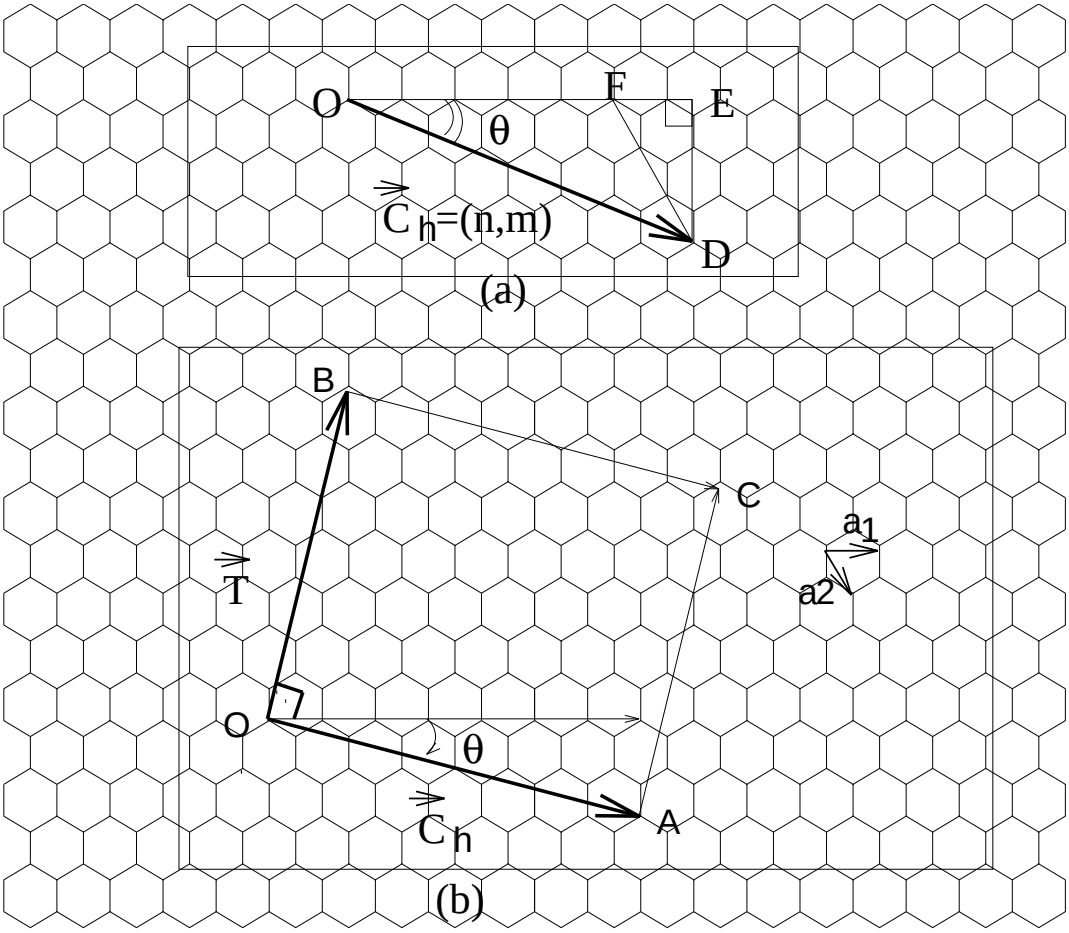

図 1.8 チューブの展開図 (T.Takeya 卒業論文 (1997) より引用 [5])
(a)C_h=(5,3),(b)C_h=(6,2) 半径 R=2.8Å,$|T|$ = Å,N = 104, θ=13.9 度 [8]

カイラルベクトル : C_h

カイラルベクトルにより NT の筒構造を表すことが出来る。カイラルベクトルは 2 つの整数値 (n, m) $(0 \leq |m| \leq n)$ の組で指定でき、グラファイトシートの切りかたを表現している。カイラルベクトル $\mathbf{C_h}$ は グラファイトの基本格子ベクトル $\mathbf{a}_1$,$\mathbf{a}_2$ を用

[8]図 1.8 = u00naka/eps/tenkai.eps

いて表現される。

$$\mathbf{C_h} = n\mathbf{a_1} + m\mathbf{a_2} \equiv (n, m) \qquad (n, m\text{は整数}, 0 \leq |m| \leq n). \tag{1.4.1}$$

直径：$\mathbf{d}_t$

また $\mathbf{C_h}$ によって示された線分がチューブの円周になる。点 O· 点 A を通り 線分 OA に垂直な直線でグラファイトシートを切り、切った線を合わせるように繋げることによりチューブ構造が出来る。炭素原子距離 a_{c-c} が 1.42Å とすると、
a=$|\mathbf{a_1}|$=$|\mathbf{a_2}|$=$\sqrt{3}a_{c-c}$=2.46Å を用いて、
チューブの円周は、

$$|\mathbf{C_h}| = L = \sqrt{OE^2 + ED^2} = a\sqrt{n^2 + m^2 + nm}, \tag{1.4.2}$$

で与えられ、またチューブの直径 d_t は $d_t = \dfrac{L}{\pi}$ で与えられる。

カイラル角：θ

$\mathbf{a_1}$ と $|\mathbf{C_h}|$ のなす角をカイラル角 θ とよぶ、

$$\theta = \tan^{-1}\frac{\sqrt{3}m}{2n + m}. \tag{1.4.3}$$

また $(0 \leq |m| \leq n)$ の条件より、 $|\theta| \leq 30\,^\circ$ が与えられる。
付録**??**に d_t と θ と C_h を d_t の小さい順に並べた。

チューブの並進ベクトル：$\mathbf{T}$

図 1.8(b) において、 **O** から $\mathbf{C_h}$ に垂直な方向ににある **O** と最初に等価な格子点を **B** とおく。チューブの並進ベクトル **T** はベクトル **OB** である。 **T** は $\mathbf{a_1}$、 $\mathbf{a_2}$ を用いて次式で表される。

$$\mathbf{T} = t_1\mathbf{a_1} + t_2\mathbf{a_2} \equiv (t_1, t_2) \quad (\text{ただし} t_1, t_2 \text{は互いに素}) \tag{1.4.4}$$

ここで、 t_1,t_2 は $\mathbf{C_h}$ と **T** は垂直なことをもちいて内積の関係 $\mathrm{C}_h \cdot T = 0$ から、、以下のように表される。

$$t_1 = \frac{2m + n}{d_R},\ t_2 = -\frac{2n + m}{d_R}, \tag{1.4.5}$$

ここで d_R は、 (2m+n) と (2n+m) の最大公約数である。

チューブのユニットセルと原子数： $2N$

チューブのユニットセルは図 1.8で $\mathbf{C_h}$ と $\mathbf{T}$ からなる長方形 $\mathbf{OABC}$ である。このユニットセル内の六員環の数 N は面積 $|\mathbf{C_h}\times\mathbf{T}|$ を六員環 1 個の面積 $(|\mathbf{a_1}\times\mathbf{a_2}|)$ で割ると、求められ次式のようになる。

$$N = 2\frac{(n^2+m^2+nm)}{d_R} \tag{1.4.6}$$

これよりチューブのユニットセル内の炭素原子の数は、$2N$ となる。

1.5 フラーレン内包カーボンナノチューブ

ナノチューブは内部に空洞を持つので、内部に他のフラーレンや更に直径の小さいナノチューブが入りそうなことは容易に予想できる。実際、最初に発見されたナノチューブはそのような 多層ナノチューブであった [6]。

1.5.1 内包型フラーレン

フラーレンはその内部に異種原子が十分収まる空間を持っている。現在までに様々な種類の金属原子を内包したフラーレンが Smally らの実験から明らかにされている。金属原子を取り込むフラーレン籠は主に C_{82} である。多量に合成される C_{60} や C_{70} ではなく、空のフラーレンとしてははるかにマイナーな C_{82} が金属原子を内包しているのが特徴である。このとき、内側に取り込まれる原子は一個だけでなく、複数 (2、3 個) が一緒に内包されていることもある。フラーレンが内包型であることをあらわす場合には、"@"(アットマーク) を使って書き表す。例えば、C_{82} 分子に M 原子が n 個内包されていることを、$M_n@C_{82}$ と書く。ナノチューブでもこれと同じことが言え、カイラルベクトル $(10,10)$ のチューブの中に C_{60} が内包されていることを、$C_{60}@(10,10)$ と書き表す。

1.5.2 C_{60} 内包カーボンナノチューブ

B.W.Smith らのグループは 1998 年、ハロゲン金属の触媒下において、炭素原料からパルスレーザー蒸発法を用いて生成した SWNT 中に C_{60} などのフラーレンが内包されていることを報告した [7][8][9]。

1998 年後半、 Jelemy Sloan らのグループは透過型電子顕微鏡を用い、 SWNT 中に内包されているフラーレンの種類、形状を観察し、その直径から何であるかを特定した。同時にフラーレン内包 SWNT の生成予測も行っている。[10] 1998 年に先立ってこの種の研究が盛んに行われなかった背景には、フラーレンを内包した SWNT の産出量が少量だったため (アーク放電法を用いていた。) と考えられる。

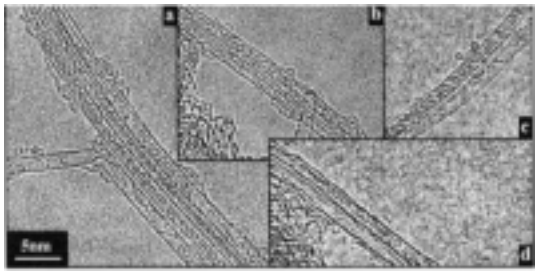

図 1.9 束で存在する C_{60} 内包 SWNT の SEM 像
Chemical Physics Letters**316**(2000) 191. より引用[9]

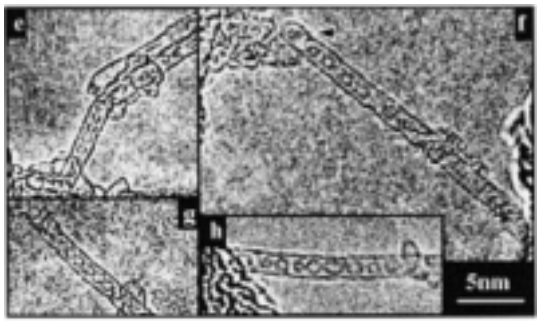

図 1.10 単体で存在する C_{60} 内包 SWNT の SEM 像
Chemical Physics Letters**316**(2000) 191. より引用[10]

また、 Jelemy Sloan らはこの時同時にハロゲン金属触媒下において電子線の照射により内包フラーレンが崩壊及び重合することも確認している。もっと最近では単に温度を上昇させる (約 1200 度) ことによって同様のことが起きることが S.Bandow らによって確認されている [14]。

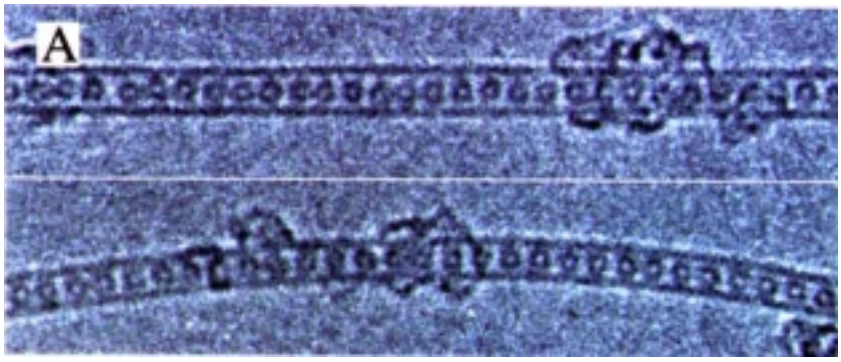

図 1.11 C_{60} 内包カーボンナノチューブの SEM 像
[14] より引用 [11]

[9]図 1.9 = u00naka/eps/sw1.eps

[10]図 1.10 = u00naka/eps/sw2.eps

[11]図 1.11 = u00naka/eps/naihou1.eps

1.6 本研究の目的及び方法

C_{60} 内包 SWNT の安定構造を解析する。99 年度卒業の松尾氏が作製した構造最適化プログラムを用いて C_{60} 内包 SWNT と単体の SWNT をそれぞれ構造最適化し、そのときのエネルギーの差 ΔE を計る。この ΔE が大きいほど SWNT が C_{60} を内包したときの方がエネルギーが得、つまり C_{60} を内側に取り込みやすいといえる。

そして、構造最適化プログラムを改良し新たに温度を設定できるようにする。そして Jelemy Sloan らの研究 [10] に関して興味深かった内包フラーレンの重合及び破壊の過程を時間を追って観察 (アニメーション表示) する。

1.7 本論文の構成

この論文の構成を述べる。第 2 章では C_{60} 内包カーボンナノチューブの構造最適化を行った。構造最適化プログラムを C_{60} SWNT に活用し、フラーレン内包カーボンナノチューブの安定構造をもとめた。第 3 章では C_{60} 内包 SWNT に任意の温度を設定し、そのときのチューブの構造や内包分子の変化をシュミレーションして観察した。先に述べた松尾氏のプログラムに手を加え、温度を考慮して構造最適化を行えるように改良した。第 4 章で得られた結論を述べる。付録として計算データ、各種ソフト、プログラムの使い方、プログラムソースを収める。

第 2 章

C_{60} 内包 SWNT の構造最適化

2.1 方法

99 年度卒業の松尾氏の修士論文では、構造最適化プログラムを作製し、 MWNT の最適構造を解析した [11]。この構造最適化プログラムを C_{60} 内包 SWNT に応用し、様々なカイラルベクトルのチューブについて最適化をおこなった。最適化の手法として松尾氏は、共役勾配法 (Conjugate Graduate Method)、分子動力学的手法 (Molecular Dynamical Method)、という二つの手法を紹介している。
ここで簡単にその二つの手法を紹介する。

1. 共役勾配法 (CG 法)

   - n 元 2 次関数の極小点を $O(N)$ で求める手法である。構造最適化に応用するには、一般の 2 次関数でないポテンシャルを 2 次関数で近似して、その場合の極小点を求めることを繰り返すことによってエネルギーの最小の点を求めることができる。また 1 回共役勾配法を適用する場合の計算量が $O(N)$ であるので計算量が削減できる。

2. 分子動力学的手法 (MD 法)

   - 原子間にかかる力を原子間ポテンシャルを数値微分することにより原子間にかかる力を計算し、ある瞬間の位置・速度の変化をもとめて、速度を一定の減衰率で減らすことにより、系の安定構造を求める手法である。 MD 法は 共役勾配法に必要な 2 次微分がない分有利であるが、系に適合するパラメータを必要とする分汎用性が低い。

2.2 分子動力学的手法 (Molecular dynamical Method)

分子動力学的手法による構造最適化とは、原子間にかかる力を原子間ポテンシャルを数値微分することにより原子間にかかる力を計算し、ある瞬間の位置・速度の変化をもとめて、速度を一定の減衰率で減らすことにより、系の安定構造を求める手法である。

2.2.1 系の運動方程式

系の原子数を N 、原子の質量を m、各原子の座標を $\mathbf{r}_n$、各原子の速度を $\mathbf{v}_n$、各原子にかかる力を $\mathbf{F}_n$、系の全エネルギーを U とする。

$$\mathbf{r}_n = (x_n, y_n, z_n), \quad (1 \leq n \leq N), \tag{2.2.1}$$

原子の座標は位置ベクトルであらわされる。

$$U = U(\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \cdots, \mathbf{r}_N), \tag{2.2.2}$$

系のエネルギー U は、位置ベクトル $\mathbf{r}_n$ の関数である。

$$\mathbf{F}_n = -(\frac{\partial U}{\partial x_n}, \frac{\partial U}{\partial y_n}, \frac{\partial U}{\partial z_n}), \tag{2.2.3}$$

各原子がポテンシャルにより受ける力は系のエネルギー U を原子の位置において微分する事により求められる。

$$\frac{d\mathbf{v}_n(t)}{dt} = \frac{\mathbf{F}_n}{m}, \tag{2.2.4}$$

$$\frac{d\mathbf{r}_n(t)}{dt} = \mathbf{v}_n, \tag{2.2.5}$$

そして、運動方程式により 位置と速度と力が関係づけられている。

2.2.2 動力学シミュレーション

先の方程式を微小な時間 Δt 間隔で区切って考える。 N_{step} をシミュレーションのステップ数、Tを系における時間とする。

$$T = \Delta t N_{step} \tag{2.2.6}$$

$$\mathbf{v}_n(T) = \mathbf{v}_n(T - \Delta t) + \frac{\mathbf{F}_n(T)}{m}\Delta t \tag{2.2.7}$$

$$\mathbf{r}(T) = \mathbf{r}(T - \Delta t) + \mathbf{v}_n(T)\Delta t \tag{2.2.8}$$

Δt が十分に小さい場合、この式で原子の運動を近似することができる。

2.2.3 構造最適化に適用

式 (2.2.7) にパラメータ ε $(0 \leq \varepsilon \leq 1)$ を加えることにより、系の原子の運動速度を次第に減衰させることができる。

$$\mathbf{v}_n(T) = \varepsilon \mathbf{v}_n(T - \Delta t) + \frac{\mathbf{F}_n(T)}{m}\Delta t \tag{2.2.9}$$

$\varepsilon = 1$ の場合、系のエネルギー（ポテンシャルエネルギー＋運動エネルギー）は保存されるが、$(0 \leq \varepsilon < 1)$ の場合、ポテンシャルエネルギーは運動エネルギーへ移り極小に向かい、運動エネルギーは 0 に向かう。このこと利用して構造最適化が行える。実際の計算に用いる Δt と ε に関して述べる。Δt は大きいと原子の移動量が大きくなり最適化が早まるが、原子の移動量が大きくなり過ぎると、原子の位置が発散する。本研究で用いた 炭素の原子ポテンシャルの場合は $3.0[\mathrm{fs}]$ を用いた場合にしばしば原子の位置が発散した。$\Delta t = 2.5[fs]$ で収束速度に関して良好な計算結果を得た。

99 年度卒業の松尾氏によると、” 大きな有限の長さのカーボンナノチューブの構造最適化の手法は共役勾配法 (CG 法) より分子動力学的手法 (MD 法) を用いる方が原子数が多い場合計算量において有利” ということが分かった。また、この研究では、系に適合するパラメータは一定であり、後述の温度設定プログラムで速度のパラメータを利用するため、 MD 法を用いて構造最適化をおこなった。

構造最適化を実行すると各チューブは規則的な収縮を繰り返し、エネルギーの値はグラフ図 2.12のように一定の値に収束しようとしていく。 MD 法では各原子が振動しつつ最適化が行なわれる。例えて言うならチューブを揺すりながら弛緩させ、各原子を安定位置に収めている状態である。

最適化を実行するにあたり、繰り返しの回数を指定する必要がある。 2.5[fs] ごとにプログラムを一回行なう設定になっている。松尾氏のプログラムでの初期値は 2,000 回であるが、様々なカイラルベクトルのナノチューブのエネルギーの収束値は 200 回 (500[fs]) 行なえばフラットになることを確認したので、この研究での繰り返し回数は 200 回を目安に行なった。

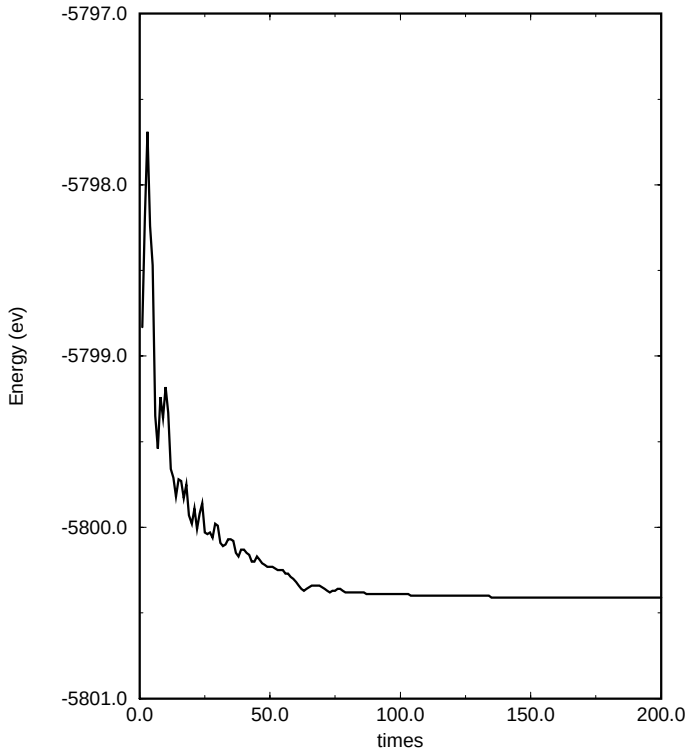

図 2.12 (10,10) チューブのエネルギー収束値と繰り返し回数との関係グラフ [1]

[1]図 2.12 = u00naka/eps/1010eng.eps

2.3 構造最適化プログラムの実行

構造最適化プログラムを用いて C_{60} 内包カーボンナノチューブの構造を最適化する。このプログラムを実行する。(参照 A.1) ホストコンピュータは *wire* を用いる。(高速演算が可能。) まず、プログラムをコンパイルする。コマンドラインで

% f77 md5.f

と打ち込むとコンパイルされ、実行可能ファイル

a.out

が出来上がる。実行には、チューブ座標のファイル (tube.xyz) が必要である。(チューブ座標の作り方は巻末付録 Aに掲載。) 任意の出力ファイルを指定する。(例:kekka.xyz) コマンドラインで

% a.out tube.xyz ▷ kekka.xyz

と打ち込むと計算が開始され、数秒の後出力ファイルに結果の座標が出力される。繰り返し回数などのデータを変更する場合、巻末付録 B.1、 B.3 に示した方法で PARAMETER ファイルなどを変更する必要がある。

2.4 C_{60} 内包時のエネルギー

表 1: エネルギーの収束値と半径との関係

C_h	2N	半径 [nm]	エネルギーの収束値 [eV]			
			差 /atm.	差	チューブ単体	C_{60} 内包チューブ
(12,2)	748	0.513	—	—	—	—
(10,5)	1040	0.517	—	—	—	—
(11,4)	724	0.527	-1.2×10^{2}	-8.7×10^{4}	-5235.7893	-5521.9406
(13,1)	792	0.530	-8.6×10^{1}	-6.8×10^{4}	-5301.1831	-5618.5516
(12,3)	900	0.538	-8.5×10^{1}	-7.7×10^{4}	-6089.7615	-6408.1745
(9,7)	832	0.544	-3.6×10^{-1}	-3.0×10^{2}	-5595.5523	-5999.3609
(14,1)	904	0.569	-2.6×10^{-1}	-2.4×10^{2}	-6113.5482	-6517.3629
(9,8)	928	0.577	-3.0×10^{-4}	-2.8×10^{-1}	-6360.7947	-6704.6110
(10,7)	936	0.579	-3.0×10^{-4}	-2.8×10^{-1}	-6360.0619	-6763.8782
(15,1)	1024	0.608	-2.4×10^{-3}	-2.5×10^{1}	-6996.3288	-7400.1431
(9,9)	780	0.610	-6.2×10^{-4}	-4.8×10^{-1}	-5215.5141	-5619.3301
(11,7)	1048	0.615	-2.2×10^{-4}	-2.3×10^{-1}	-7178.8718	-7582.6883
(15,2)	1096	0.630	6.0×10^{-5}	6.6×10^{-2}	-7524.1282	-7927.9448
(10,9)	1144	0.644	-1.5×10^{-8}	-1.7×10^{-5}	-7888.8742	-8292.6908
(11,8)	1152	0.647	6.0×10^{-9}	6.9×10^{-6}	-7948.1378	-8351.9544
(12,7)	1168	0.651	-7.6×10^{-8}	-8.9×10^{-5}	-8061.6762	-8465.4929
(10,10)	860	0.678	1.9×10^{-8}	1.6×10^{-5}	-5800.4069	-6204.2236
(12,9)	1392	0.714	1.1×10^{-9}	1.5×10^{-6}	-9703.9321	-10107.7488
(18,2)	788	0.747	9.2×10^{-9}	7.2×10^{-6}	-5244.9578	-5648.7744
(18,4)	884	0.795	5.3×10^{-9}	4.7×10^{-6}	-5945.5268	-6349.3435
(20,2)	1244	0.825	1.4×10^{-8}	1.7×10^{-5}	-8602.0991	-9005.9157
(18,6)	996	0.847	-1.4×10^{-8}	-1.4×10^{-5}	-6764.3625	-7168.1791
(14,12)	1076	0.882	1.7×10^{-9}	1.8×10^{-6}	-7365.8599	-7769.6766

C_{60} の半径：約 0.36 [nm]

C_{60} のエネルギー収束値：-403.8166[eV]

表 1 に C_{60} 内包ナノチューブを構造最適化したときの半径とそのときの系のエネルギー (ポテンシャルエネルギー + 運動エネルギー) の収束値の関係を示す。

図 2.13 に原子 1 個あたりのエネルギー (単体、内包各チューブのエネルギー収束値を 2N で割ったもの) と半径の関係をグラフにあらわす。図中丸い点で表したものが (単体チューブ + 単体 C_{60}) の最適化後のエネルギー、星印で示したものが (C_{60} 内包チューブ) の最適化後のエネルギーの値である。

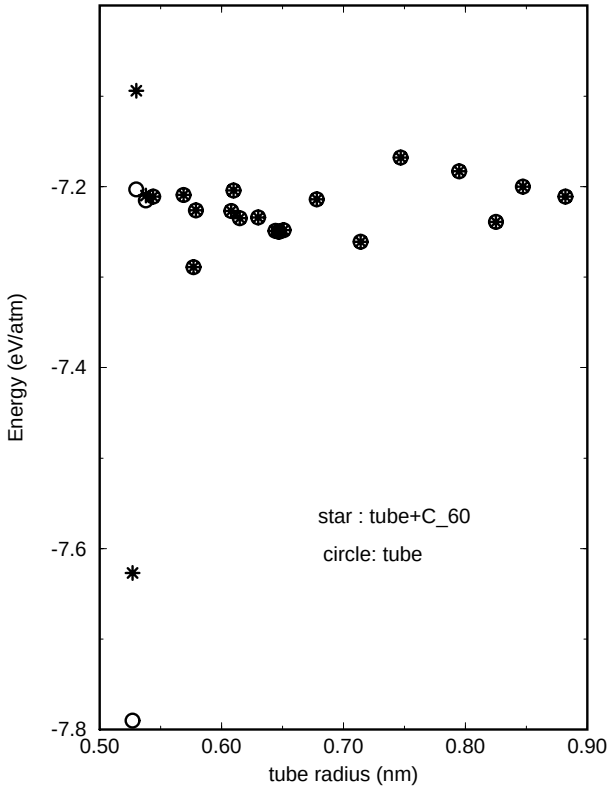

図 2.13 原子 1 個あたりのエネルギーと半径の関係 [2]

この計算では、 C_{60} を内包しているチューブのエネルギー収束値とチューブ単体のエネルギー収束値、 C_{60} のエネルギー収束値をたしたものとを比較してそのエネルギーの差 ΔE をとったものである。

$$(n,n)tube+C_{60} \rightarrow C_{60}@(n,n)\text{-}\Delta E$$

この Δ E が大きいほどカーボンナノチューブが C_{60} を内包したときの方がチューブ単体のときよりもエネルギーが得、つまり安定した構造である、ということが言える。図 2.14に ΔE と半径の関係を示す。

[2]図 2.13 = u00naka/eps/c60eng5.eps

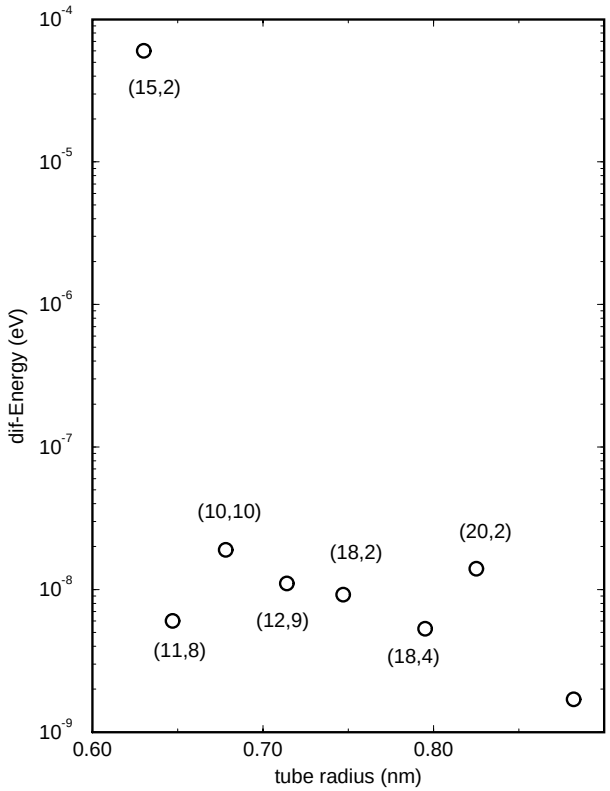

図 2.14 C_{60} 内包 SWNT のエネルギーの差と半径の関係 [3]

図 2.13から、外チューブの半径が 0.526 [nm] から 0.538 [nm] の間、カイラルベクトル (11,4),(13,1),(12,3) のものでは明らかにエネルギーが損をしている。これらのチューブと内包 C_{60} との間隔はおおよそ 0.158 [nm] から 0.178 [nm] となっている。図 2.14からチューブ径 0.63 [nm] 、(15,2) チューブで C_{60}@(15,2) と (15,2)tube + C_{60} のエネルギー差が最大、得をしている。つまり C_{60} をチューブ内に取り込みやすいといえる。これよりチューブ径が大きくなるにつれ、このエネルギーの差は小さくなっていく。(15,2) チューブの C_{60} とチューブ内壁間の距離はおよそ 0.270 [nm] であり、この距離が小さすぎると C_{60} はチューブ内に取り込まれにくく、また大きすぎてもエネルギー差が限りなく 0 に近づき、構造最適化を行なってもチューブ、C_{60} それぞれが互いに無関係な振る舞いをしてしまうものと考えられる。

また、チューブの半径 0.526 [nm]、カイラルベクトル (11,4) より小さいものでは、内包 C_{60} 分子が外チューブの分子と結合してしまい、最適化することができなくなってしまう (図 2.15)。

[3]図 2.14 = u00naka/eps/c60eng10.eps

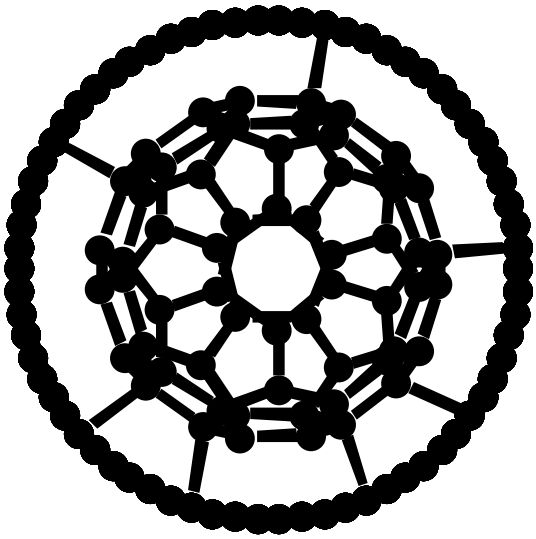

図 2.15 C_{60}@(10,5) チューブ [4]

次ページに SWNT の x-mol 像を掲載する。

[4]図 2.15 = u00naka/eps/tube105.eps

2.4.1 C_{60} 内包 SWNT

この章で計算した C_{60} 内包 SWNT の x-mol 像をここに掲載する。

† 構造最適化実行前 †

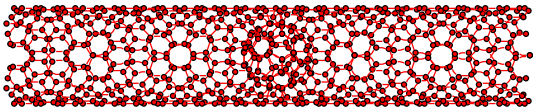

図 2.16 (12,2) 原子数 748[5]

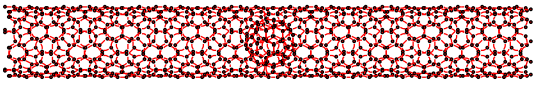

図 2.17 (10,5) 原子数 1040[6]

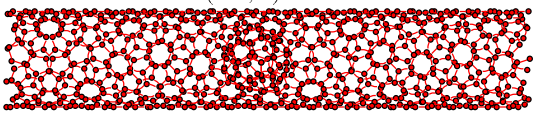

図 2.18 (11,4) 原子数 724[7]

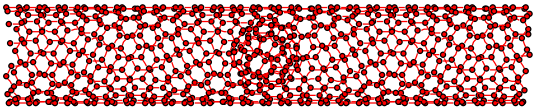

図 2.19 (13,1) 原子数 1,036[8]

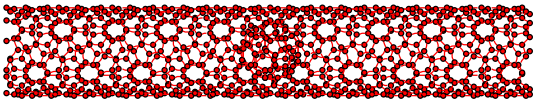

図 2.20 (12,3) 原子数 900[9]

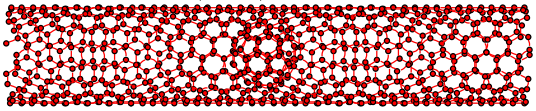

図 2.21 (9,7) 原子数 832[10]

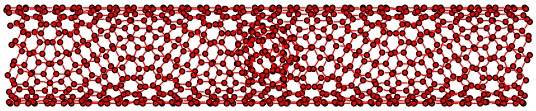

図 2.22 (14,1) 原子数 904[11]

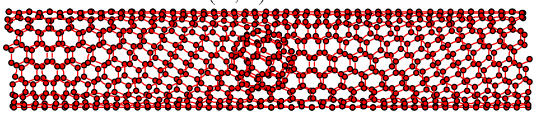

図 2.23 (9,8) 原子数 928[12]

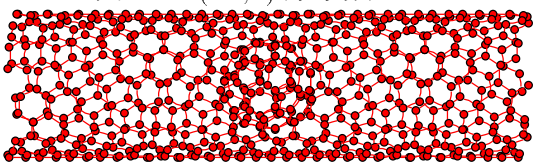

図 2.24 (10,7) 原子数 936[13]

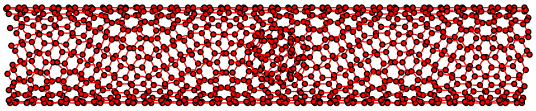

図 2.25 (15,1) 原子数 1,024[14]

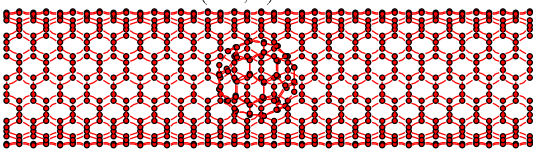

図 2.26 (9,9) 原子数 780[15]

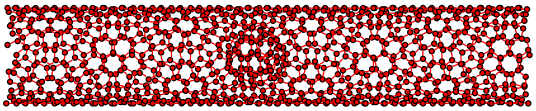

図 2.27 (11,7) 原子数 1,048[16]

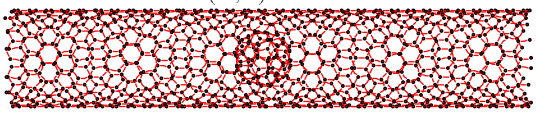

図 2.28 (15,2) 原子数 1,096[17]

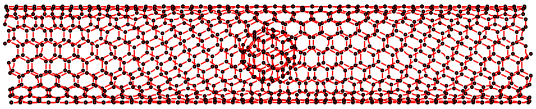

図 2.29 (10,9) 原子数 1,144[18]

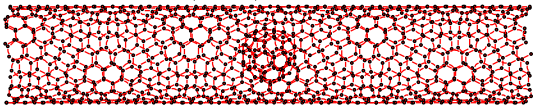

図 2.30 (11,8) 原子数 1,152[19]

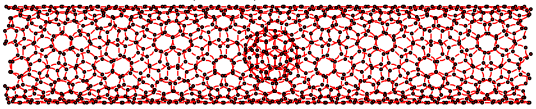

図 2.31 (12,7) 原子数 1,168[20]

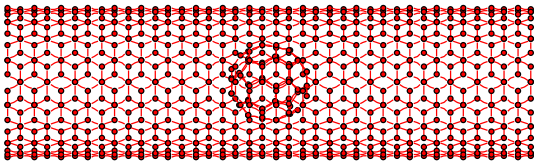

図 2.32 (10,10) 原子数 860[21]

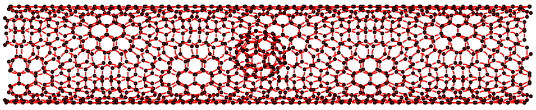

図 2.33 (12,9) 原子数 1,392[22]

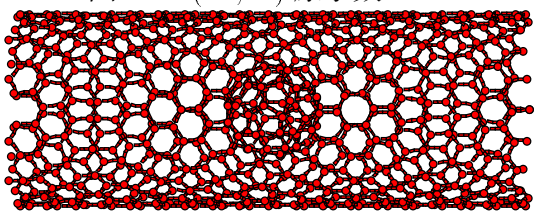

図 2.34 (18,2) 原子数 788[23]

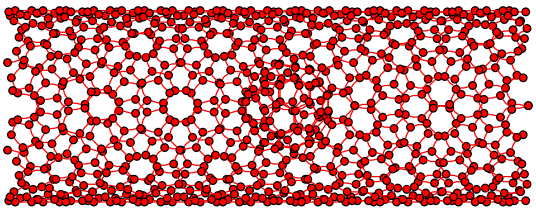

図 2.35 (18,4) 原子数 884[24]

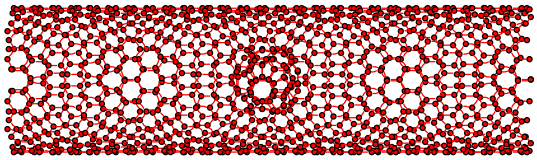

図 2.36 (20,2) 原子数 1,244[25]

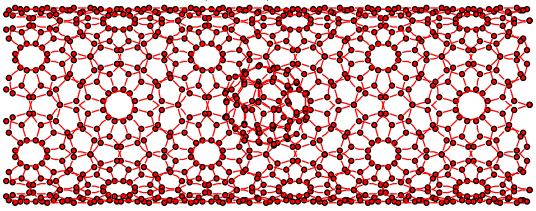

図 2.37 (18,6) 原子数 996[26]

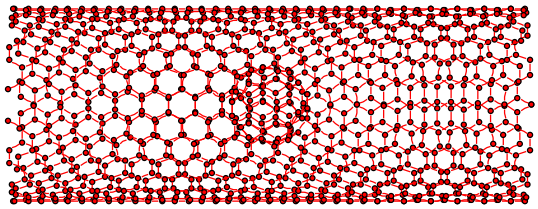

図 2.38 (14,12) 原子数 1,076[27]

[5]図 2.16 = u00naka/eps/122.eps

[6]図 2.17 = u00naka/eps/105.eps

[7]図 2.18 = u00naka/eps/114.eps

[8]図 2.19 = u00naka/eps/131.eps

[9]図 2.20 = u00naka/eps/123.eps

[10]図 2.21 = u00naka/eps/97.eps

[11]図 2.22 = u00naka/eps/141.eps

[12]図 2.23 = u00naka/eps/98.eps

[13]図 2.24 = u00naka/eps/107.eps

[14]図 2.25 = u00naka/eps/151.eps

[15]図 2.26 = u00naka/eps/99.eps

[16]図 2.27 = u00naka/eps/117.eps

[17]図 2.28 = u00naka/eps/152.eps

[18]図 2.29 = u00naka/eps/109.eps

[19]図 2.30 = u00naka/eps/118.eps

[20]図 2.31 = u00naka/eps/127.eps

[21]図 2.32 = u00naka/eps/1010.eps

[22]図 2.33 = u00naka/eps/129.eps

[23]図 2.34 = u00naka/eps/182.eps

[24]図 2.35 = u00naka/eps/184.eps

[25]図 2.36 = u00naka/eps/202.eps

[26]図 2.37 = u00naka/eps/186.eps

[27]図 2.38 = u00naka/eps/1412.eps

† 構造最適化実行後 †

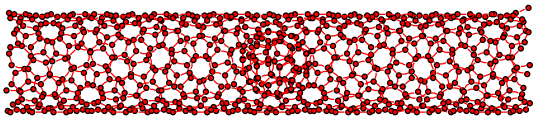

図 2.39 (11,4) 原子数 724[28]

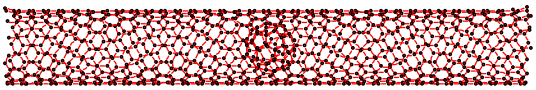

図 2.40 (13,1) 原子数 1,036[29]

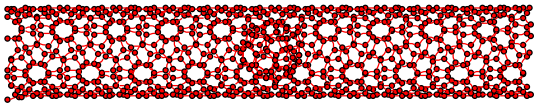

図 2.41 (12,3) 原子数 900[30]

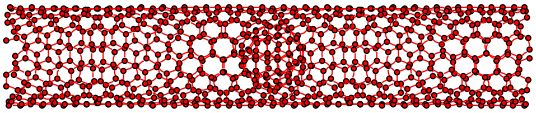

図 2.42 (9,7) 原子数 832[31]

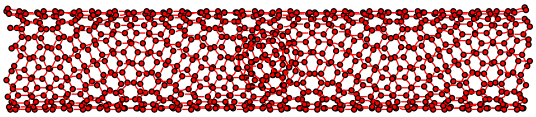

図 2.43 (14,1) 原子数 904[32]

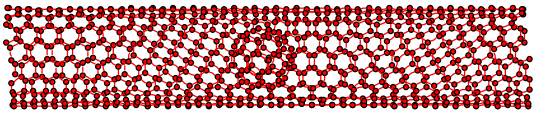

図 2.44 (9,8) 原子数 928[33]

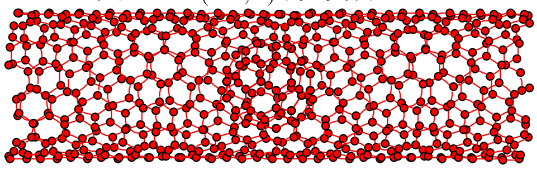

図 2.45 (10,7) 原子数 936[34]

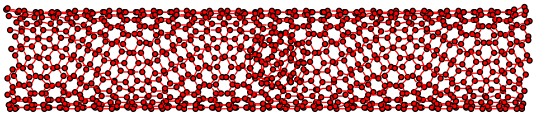

図 2.46 (15,1) 原子数 1,024[35]

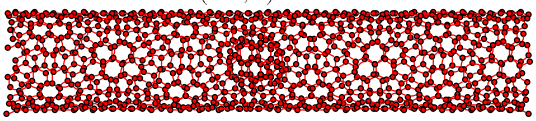

図 2.47 (11,7) 原子数 1,048[36]

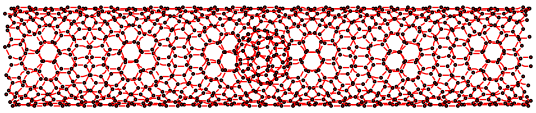

図 2.48 (15,2) 原子数 1,096[37]

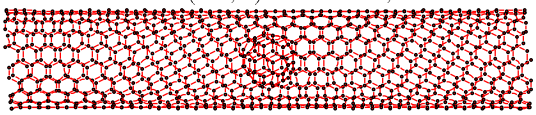

図 2.49 (10,9) 原子数 1,144[38]

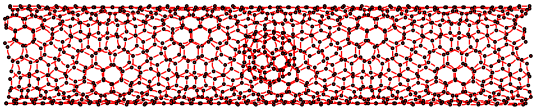

図 2.50 (11,8) 原子数 1,152[39]

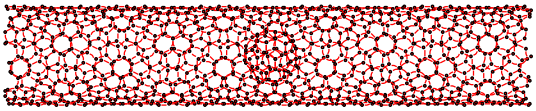

図 2.51 (12,7) 原子数 1,168[40]

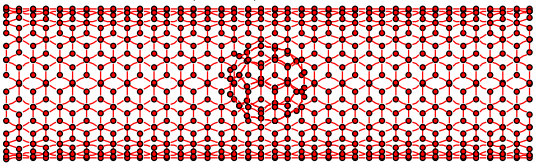

図 2.52 (10,10) 原子数 860[41]

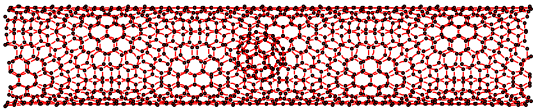

図 2.53 (12,9) 原子数 1,392[42]

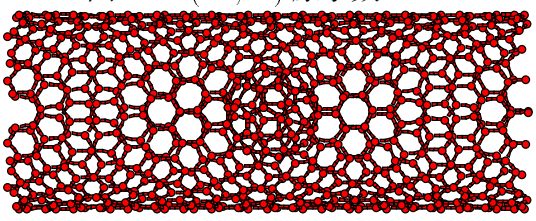

図 2.54 (18,2) 原子数 788[43]

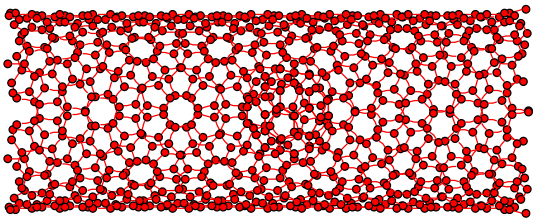

図 2.55 (18,4) 原子数 884[44]

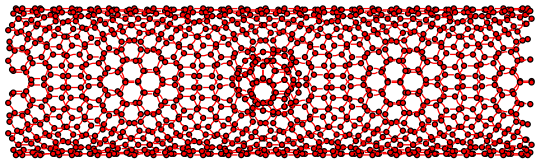

図 2.56 (20,2) 原子数 1,244[45]

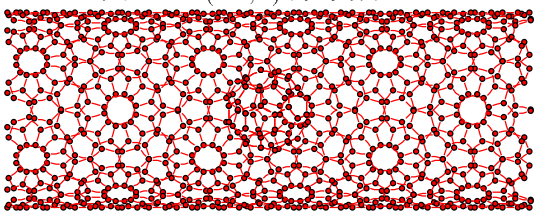

図 2.57 (18,6) 原子数 996[46]

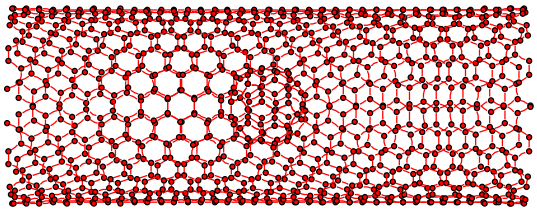

図 2.58 (14,12) 原子数 1,076[47]

[28]図 2.39 = u00naka/eps/k114.eps
[29]図 2.40 = u00naka/eps/k131.eps
[30]図 2.41 = u00naka/eps/k123.eps
[31]図 2.42 = u00naka/eps/k97.eps
[32]図 2.43 = u00naka/eps/k141.eps
[33]図 2.44 = u00naka/eps/k98.eps
[34]図 2.45 = u00naka/eps/k107.eps
[35]図 2.46 = u00naka/eps/k151.eps
[36]図 2.47 = u00naka/eps/k117.eps
[37]図 2.48 = u00naka/eps/k152.eps
[38]図 2.49 = u00naka/eps/k109.eps
[39]図 2.50 = u00naka/eps/k118.eps
[40]図 2.51 = u00naka/eps/k127.eps
[41]図 2.52 = u00naka/eps/k1010.eps
[42]図 2.53 = u00naka/eps/k129.eps
[43]図 2.54 = u00naka/eps/k182.eps
[44]図 2.55 = u00naka/eps/k184.eps
[45]図 2.56 = u00naka/eps/k202.eps
[46]図 2.57 = u00naka/eps/k186.eps
[47]図 2.58 = u00naka/eps/k1412.eps

第 3 章

温度を考慮した構造最適化

Jelemy Sloan らは 1998 年後半、ハロゲン金属触媒下において電子線の照射により内包フラーレンが破壊及び重合することを実験から確認している。もっと最近では単に温度を上昇させることによっても同様のことが起きることが確認されている。[14] そこで今までの構造最適化プログラムを任意の温度を設定することができるように改良し、それを用いて内包フラーレンが破壊、重合していく様子を観察する。

3.1 温度設定プログラム

このプログラムをつくる際の考え方を説明する。

原子一つがもっている速度エネルギーの平均 $[\frac{1}{2}mv^2]$ はボルツマン定数 $[k_B]$ とその速度での絶対温度 $[T]$ との積に等しいから、

$$\frac{1}{2}mv^2 = \frac{3}{2}k_BT = \frac{3}{2}Nk_BT \qquad (\text{原子}N\text{個より}) \tag{3.1.1}$$

$$\left(\begin{array}{l} k_B : \text{ボルツマン定数[J/K]} \\ T : \text{その速度での絶対温度[K]} \\ m : \text{炭素の質量[Kg]} \end{array} \right) \tag{3.1.2}$$

これより

$$T = \frac{mv^2}{3Nk_B} \tag{3.1.3}$$

ここで、この速度がもっていると期待される絶対温度 T_1 [K] は

$$T_1 = \frac{\frac{1}{2}m\sum_{i=1}^{3N} v_i^2}{\frac{3}{2}Nk_B} = \frac{m\sum_{i=1}^{3N} v_i^2}{3Nk_B} \tag{3.1.4}$$

いま、任意に T を与えることによって原子の速度 v_i が α 倍に変化する。$(v_i \rightarrow \alpha\ v_i)$
この変化の割合を α とすると、

$$\alpha^2 = \frac{T}{T_1} = \frac{3Nk_BT}{m\sum_{i=1}^{3N} v_i^2} \tag{3.1.5}$$

$$\alpha = \sqrt{\frac{3Nk_BT}{m\sum_{i=1}^{3N} v_i^2}} \tag{3.1.6}$$

この α をプログラム中に組み込み、温度の変数を与えることができるようにする。
全プログラムソースは巻末付録 Bとして掲載した。

各定数表

ボルツマン定数	k_B [J/K]	$1.38\ \times 10^{-23}$
炭素の質量	m [kg]	$1.99\ \times\ 10^{-26}$
	$\sqrt{\frac{3k_B}{m}}$	2.08×10^{-7}

† プログラムソース (抜粋)†

```
c   温度入力
c
c


      VV2 = 0.0
      VV21 = 0.0
      DO J = 180*B+1 , N*3
      VV2 = VV2 + VELO(J)**2
      END DO

      DO J = 1 , 180*B
```

```
        VV21 = VV21 + VELO(J)**2
        END DO

        WRITE(*,*)'N',N
c VV2 の画面表示 (表示させると処理が遅くなる。)
        WRITE(*,*)'VV2',VV2
c
c ALPHA=SQRT(3*N*Kb*T/m*VV2)
c       =SQRT(3*N*1.38D-23*T/1.99D-26*VV2)
c       =SQRT(2.08D3*N*T/VV2)
c   m/s  と A/fs を比較して、係数を 10D-10 倍する。
c
        ALPHA = SQRT(2.08D-7*REAL(N)*H/VV2)
c H = temperature(画面から読み込み。)
        DO J = 1 , N*3
        VELO(J) = ALPHA*VELO(J)
        END DO

c ALPHA の画面表示
        WRITE(*,*)'ALPHA',ALPHA
        TEMP1 = 0.48D7*VV2*ALPHA1/REAL(N-60*B)
        TEMP2 = 0.48D7*VV21*ALPHA2/REAL(60*B)
        WRITE(*,*)'TEMP=',TEMP1,TEMP2
```

3.2 プログラムの実行

このプログラムを実行する。ホストコンピュータは *wire* を用いる。(高速演算が可能。) まず、プログラムをコンパイルする。コマンドラインで

```
% f77 md5temp5.f
```

と打ち込むとコンパイルされ、実行可能ファイル

$$a.out$$

が出来上がる。実行には、チューブ座標のファイル ($tube.xyz$) が必要である。(チューブ座標の作り方は巻末付録 A に掲載。)
コマンドラインで

% a.out tube.xyz

と打ち込むと繰り返し回数、 C_{60}、 tube それぞれの設定温度、内包 C_{60} の個数、温度入力の間隔、
を聞いてくるのでそれぞれを入力し、計算が開始される。

3.3 温度入力時の C_{60} 内包チューブの変化

様々なカイラルベクトルの C_{60} 内包チューブを任意の温度を設定して構造最適化したときの形状の変化について観察した。

3.3.1 (10,10) チューブの温度設定

まずカイラルベクトル (10,10) のチューブ、(半径 0.678 [nm]、原子数 880 個) を 1,000K ずつ温度を上げていった。繰り返し回数はいずれも 200 回。

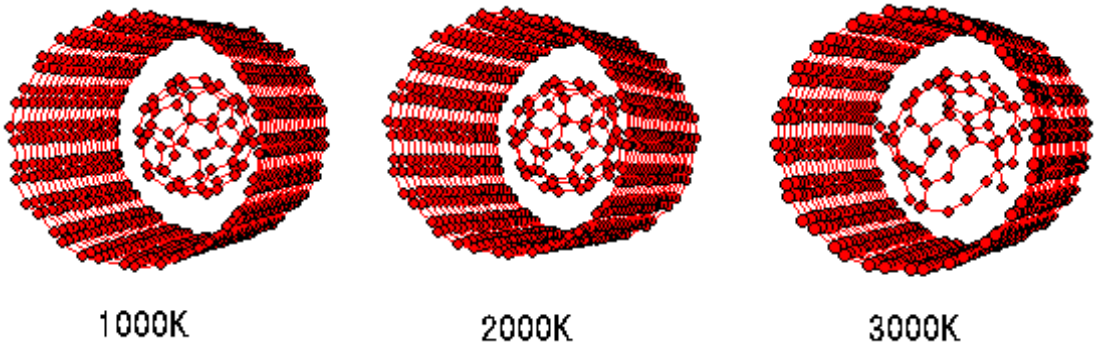

図 3.59 (10,10) チューブの温度入力時の変化 1 [1]

[1]図 3.59 = u00naka/ps/1010-a.ps

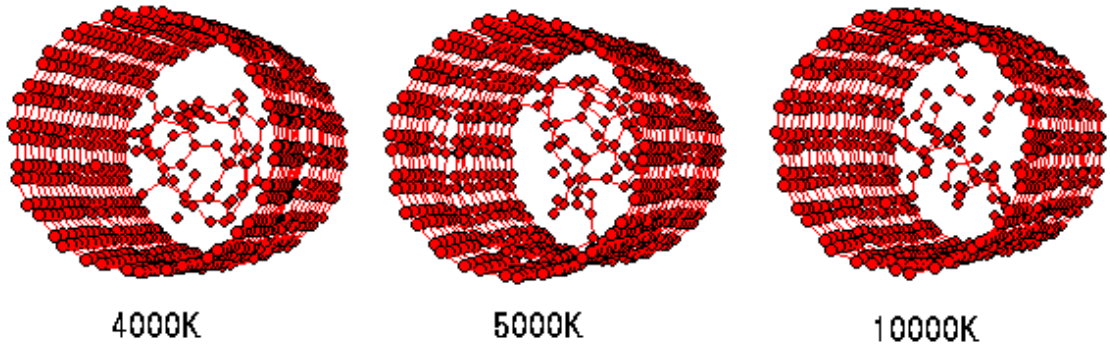

図 3.60 (10,10) チューブの温度入力時の変化 2 [2]

この結果 3,000K を過ぎると、内包 C_{60} 分子が破壊していく様子を見ることができた。この時、構造最適化のプロセスは 200 回 (500[fs]) で行なっているのだが、設定温度 3,000K の場合最適化 40 回目 (100[fs]) を過ぎた頃から内包分子の破壊が始まった。図 3.61に最適化時の回数と温度の変化グラフを示す。この図より、内包分子が崩壊したとき急激に温度が上昇していることが分かった。図中 250[fs] あたりの上昇がそれに当たる。参照 3.3.2

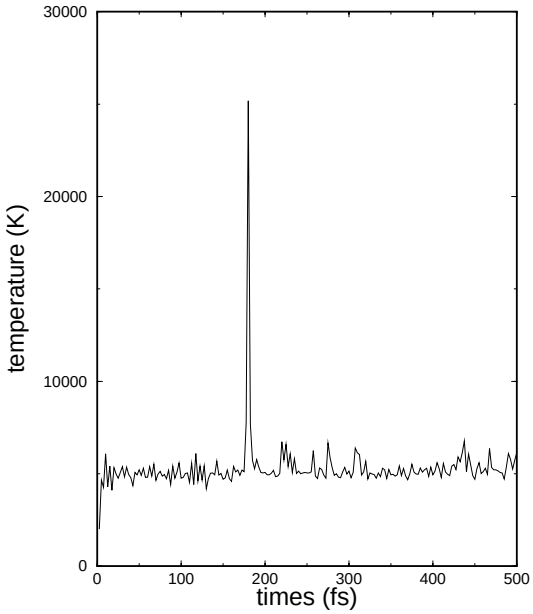

図 3.61 (10,10) チューブ、設定温度 3,000K での温度グラフ [3]

ここで、先述の (10,10) チューブについて、 2,000K から 3,000K までの間のチューブの形状変化を 100K ごとに上昇させ、観察した。

結果 2,400K を越えた頃から内包分子の破壊が始まることを確認した。

[2]図 3.60 = u00naka/ps/1010-b.ps

[3]図 3.61 = u00naka/eps/10103k-xvgr.eps

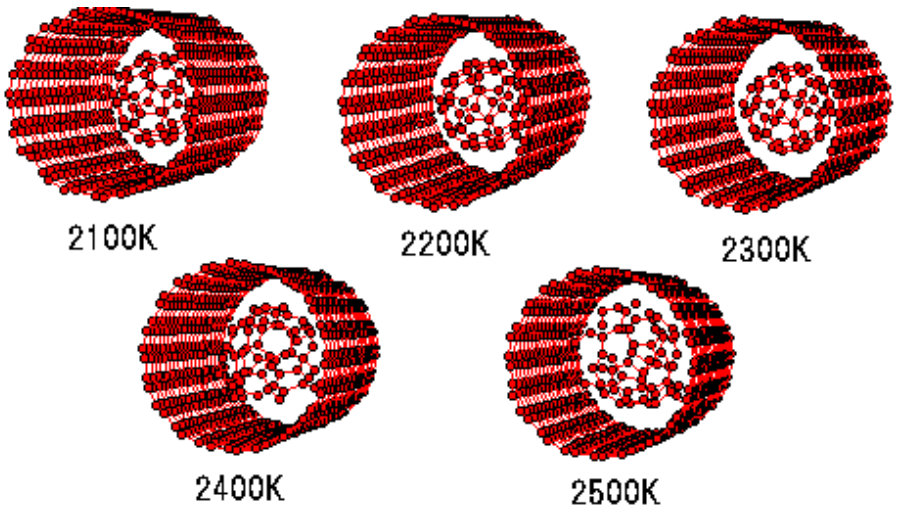

図 3.62 100K ずつ温度を上昇させたときの (10,10) チューブ [4]

3.3.2 温度入力グラフ

章 3.3.1で計算した C_{60}@(10,10) の温度設定時の回数における温度の変化をプロットしたグラフをここに掲載する。 2,300[K] までのものについては C_{60} の崩壊はみられなかった。

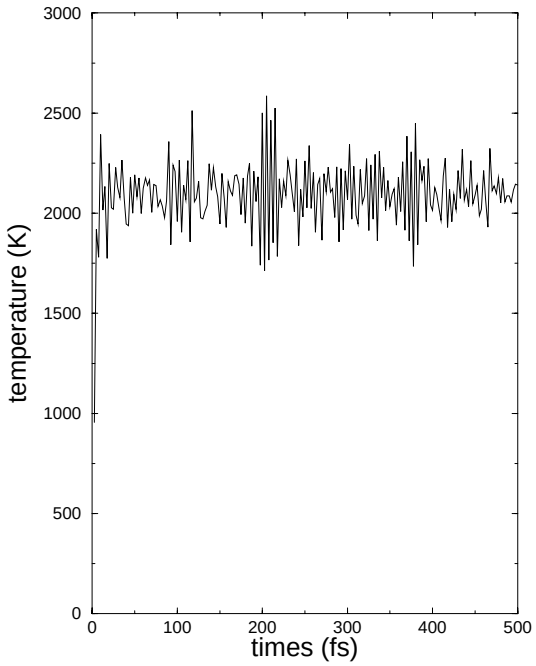

図 3.63 (10,10) 2100(K)[5]

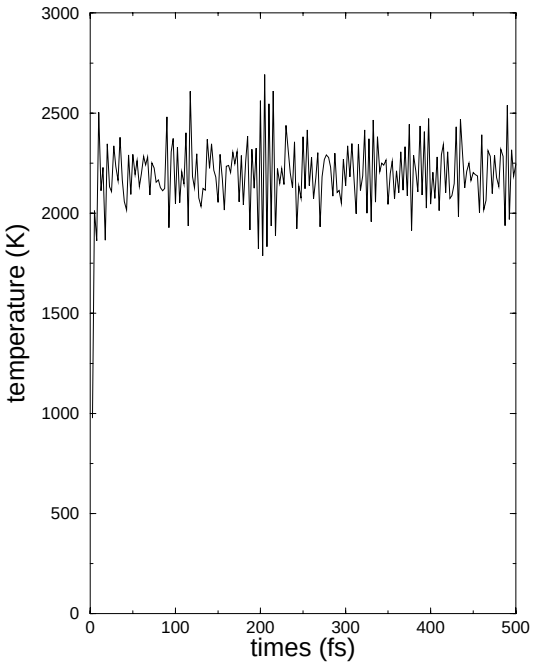

図 3.64 (10,10) 2200(K)[6]

[4]図 3.62 = u00naka/ps/1010ondo.ps

[5]図 3.63 = u00naka/eps/1010-2100.eps

[6]図 3.64 = u00naka/eps/1010-2200.eps

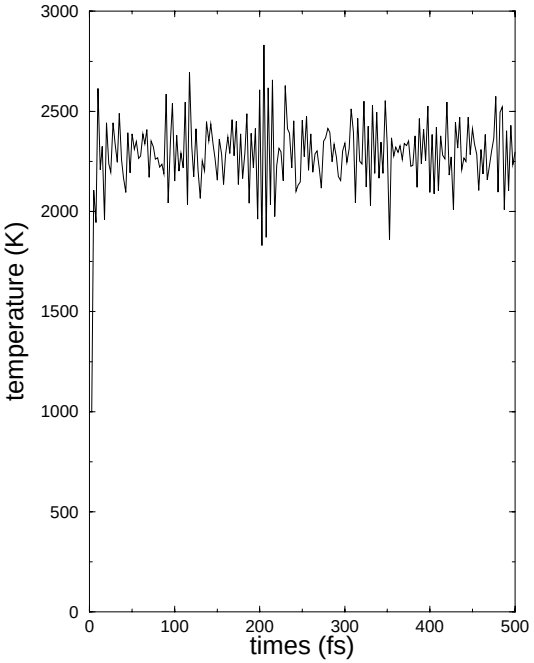

図 3.65 (10,10) 2300(K)[7]

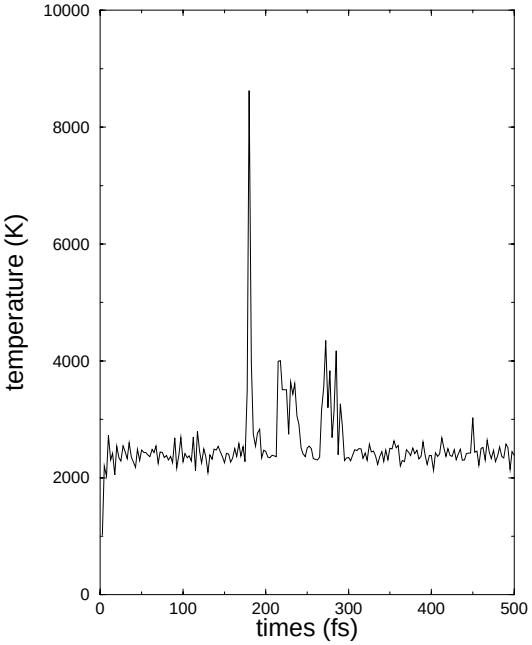

図 3.66 (10,10) 2400(K)[8]

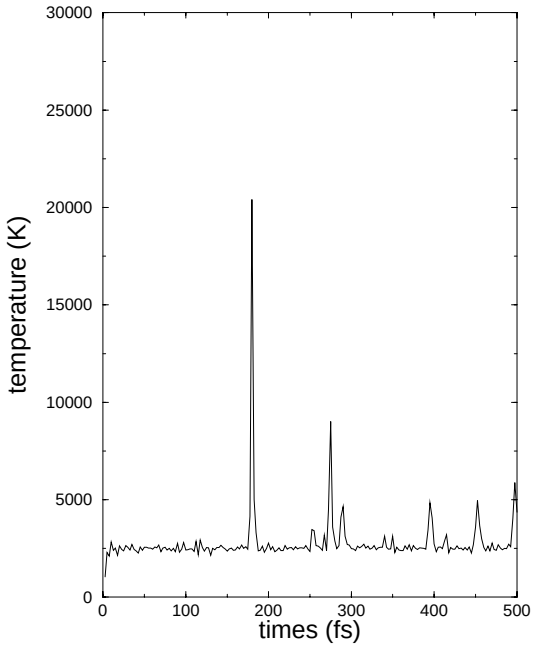

図 3.67 (10,10) 2500(K)[9]

[7]図 3.65 = u00naka/eps/1010-2300.eps

[8]図 3.66 = u00naka/eps/1010-2400.eps

[9]図 3.67 = u00naka/eps/1010-2500.eps

3.3.3 C_{60} を複数個内包した (10,10) チューブ

先の計算では、(10,10) チューブに C_{60} を一個内包したものの温度入力、構造最適化を行なったが、次に C_{60} を複数個内包したものについて同様のシュミレーションをおこなった。

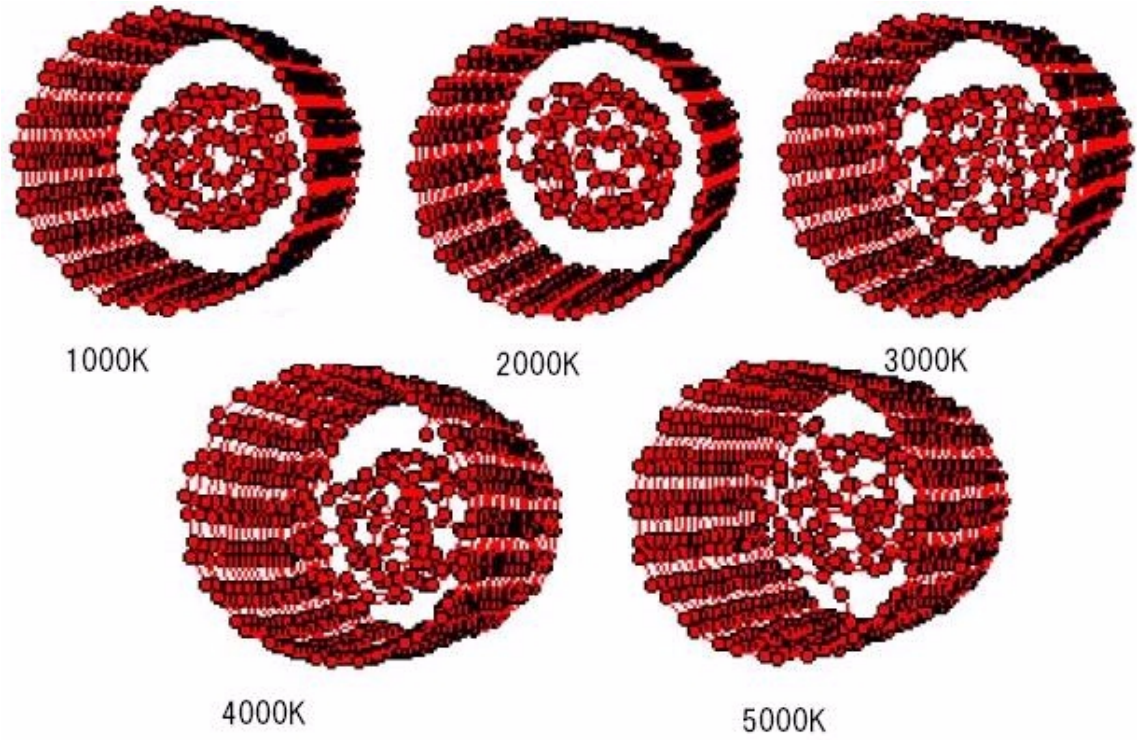

図 3.68 C_{60}×2(10,10) チューブ [10]

この結果、設定温度 2,000K ですでに内包 C_{60} 分子の破壊が始まっていた。

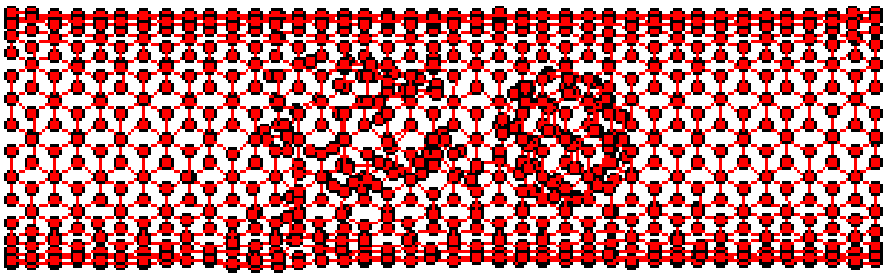

図 3.69 C_{60}×2(10,10) チューブ
設定温度 5,000K [11]

[10]図 3.68 = u00naka/ps/1010a2-a.ps

[11]図 3.69 = u00naka/ps/1010a2-a5k.ps

C_{60} を 2 個入れたときの壊れ方は z 軸の 0 から遠い方の C_{60} 分子がまず崩れていき、0 に近い方の C_{60} 分子はその形状を保ったままということになった。
また、期待していた C_{60} 同士の重合を確認することはできなかった。

この他 C_{60} 3 個から 5 個内包のものも温度を設定し計算をおこなった。いずれも内包 C_{60} 同士の間隔は 1[nm]、プログラムの繰り返しの回数は 200 回 で実行した。
結果、 C_{60} × 3@(10,10)、 C_{60} × 4@(10,10) のフラーレン内包チューブでは 2,000[K] から 4,000[K] の間で内包 C_{60} の破壊が確認できた。 C_{60} × 5@(10,10) チューブでは繰り返し回数 200 回 (約 500[fs]) では内包 C_{60} のみの破壊は見ることができなかった。設定温度を上げていけば破壊することはするのだが、外チューブも一緒に壊れてしまう。プログラムの実行に時間をかければ (繰り返し 300 回、約 750[fs]) 内包 C_{60} が破壊することが確認できた。ここでもまたフラーレン同士の重合を確認することはできなかった。

C_{60} ×	C_{60} が壊れた温度 [K]	そのときの壊れ方
3	4,000	Z 軸の 0 より遠い方から
4	2,000	Z 軸 0 より遠い方から 2 番目から
5		内包 C_{60} の破壊は確認できず
5	6,000	Z 軸 0 に近い方から 2 番目から (繰り返し 300 回)

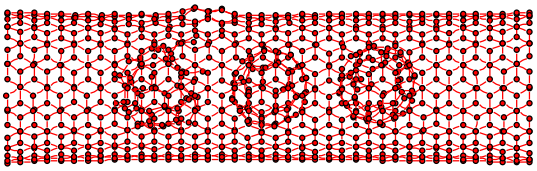

図 3.70 C_{60}×3@(10,10)
4,000[K][12]

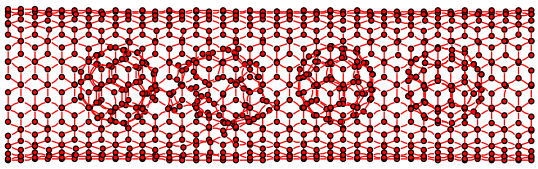

図 3.71 C_{60} ×4@(10,10)
3,000[K][13]

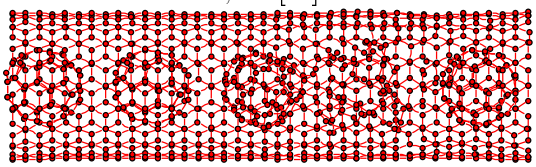

図 3.72 C_{60} ×5@(10,10)
6,000[K][14]

[7]図 3.70 = u00naka/eps/1010a3-4k.eps
[8]図 3.71 = u00naka/eps/1010a4-3k.eps
[9]図 3.72 = u00naka/eps/1010a5-6k.eps

この他 (10,10) 以外の C_{60} 内包チューブについても同様のシュミレーションをおこなった。結果、カイラルベクトル (11,4)、 (半径 0.527[nm]) からカイラルベクトル (14,12)、(半径 0.882[nm]) までのチューブではその半径が大きくなるほど内包 C_{60} が崩壊する温度は高くなっていく。これは外側の筒が大きいと内包分子が安定し、温度の変化による影響を受けにくいためと考えられる。また、半径の小さいチューブでは 3,000[K] 付近を越えるとチューブ全体が爆発した。これはやはりチューブ半径が小さいと安定構造がとれず、何かの拍子に原子間の結合が切れ、そのときの高温状態という条件も加わって、原子に急激に運動エネルギーが加わる (図 3.75)。それが連鎖的に反応し、全体の崩壊につながるのではないかと考える。

チューブ半径 (c_h)	C_{60} が壊れた温度 [K]
- 0.527 (11,4)	200 より小
0.530 (13,1) -	2,000 - 3,000
0.577 (9,8) -	3,000 - 5,000

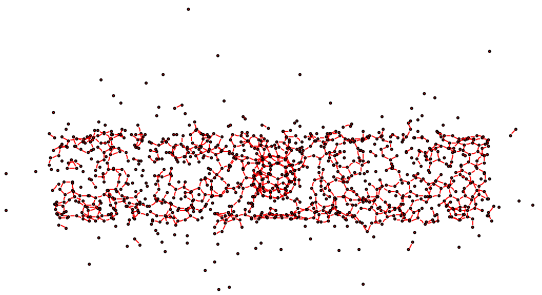

図 3.73 C_{60}@(9,7) チューブ、 4,000K での爆発[15]

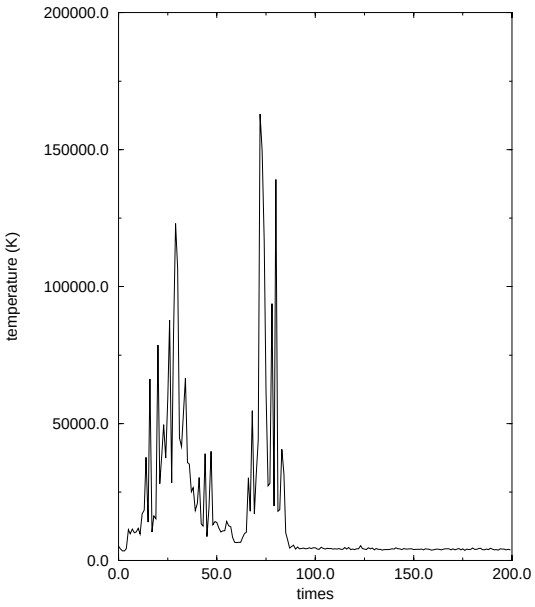

図 3.74 (9,7)、 4,000K での温度グラフ[16]

[15]図 3.73 = u00naka/eps/b97.eps

[16]図 3.74 = u00naka/eps/974k-xvgr.eps

3.3.4 結果

改良した温度設定プログラムでは、設定温度を入力するとその温度を保つようにできている。例えるならサーモスタットのようなものである。一連のシュミレーションでは、内包 C_{60} が崩れ始めるとき、一時的に温度が急激に上昇する。ということは、そのとき各原子の平均の速度 (V_i) は急激に加速されている。 (図 3.75) その結果、 C_{60} の原子間の結合を破り破壊してしまう。これが半径の小さいチューブになると C_{60} が破壊すると同時に外チューブの原子の結合も壊れ、全体的に破裂してしまう。しかし、なぜ分子が崩れ始めるときに速度が急激に加速されるのかについての解明までには至らなかった。

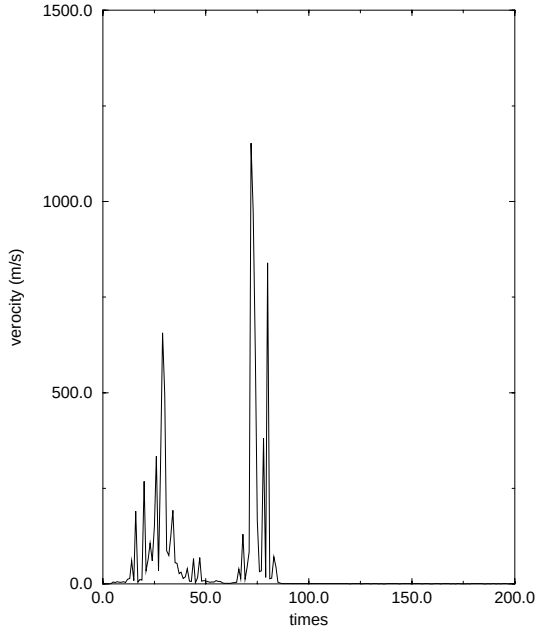

図 3.75 (9,7)、 4,000K での V_i - 回数グラフ [17]

[17]図 3.75 = u00naka/eps/974k-vi.eps

第 4 章

結論

この研究では C_{60} 内包 SWNT について、構造最適化プログラムを用いた安定構造の解析、またそのプログラムを改良し、任意の温度設定時の C_{60} 内包 SWNT の内包分子の崩壊、重合について解析した。
その結果、以下の結論を得た。

1. カーボンナノチューブは半径 0.630 [nm] 、 (15,2) チューブ一番 C_{60} 分子を内側に取り込みやすい。

2. チューブ半径が小さすぎると SWNT は C_{60} を内側に取り込みにくい。しかしチューブ径が大きいからといって C_{60} を取り込みやすいともいえない。チューブ内壁と C_{60} との距離が鍵を握っている。

3. C_{60} 内包カーボンナノチューブに任意の温度を与えると、内包 C_{60} 分子は崩れ、破壊していく。チューブ径が大きくなるにつれこの時の温度は高温である。

今回改良したプログラムでは内包 C_{60} 同士の重合を観察することはできなかった。C_{60} を内包したカーボンナノチューブの研究は歴史が浅く、発展段階にある。この研究ではほんの一部にだけ的を絞ったが、この分野での今後の発展に期待したい。

付録 A

各種ソフトの使い方

ここでは、ナノチューブ座標の作り方、座標から図を起こす方法 (x-mol)、グラフの書き方など、いろいろなソフトの使い方を説明する。全てのソフトは UNIX 上で使うことができる。

A.1 ナノチューブ座標の作り方

任意のカイラルベクトルを入力し、そのチューブの座標を作る方法。プロセスとして、「ユニットセルを作る」 → 「つなげる」 という手順を踏む。

準備

- (A.1.1) nakamura/for/tube/tube-xyz.f
- (A.1.2) nakamura/for/tube/nanbai.f
- (A.1.3) nakamura/for/tube/idou-x.f
- (A.1.3) nakamura/for/tube/idou-z.f

をそれぞれ自分のディレクトリにコピーする。

† この時後々のことを考えて、同じように /for/tube というディレクトリを作っておくことをおすすめする。

A.1.1 ユニットセルを作る

用いるプログラム	tube-xyz.f
入力ファイル	なし
出力ファイル	tube.xyz

1. まず、このプログラムをコンパイル (実行可能な形に書き換える) する。コマンドラインで

   f77 tube-xyz.f

   と入力し、コンパイルをする。この時に出来上がったファイル

   a.out

   を使ってこのプログラムを実行する。

2. コマンドラインで　a.out　と打ち込むと、

   カイラルベクトル $C_h =$

   ときいてくるので、たとえば (10,10) を作りたいのであれば、数字と数字の間をスペースで区切り、

   10 10

   と入力する。すると半径や原子数などの情報が画面に表示され、出力ファイル

   tube.xyz

   に座標データが出力される。

A.1.2 チューブをつなげる

用いるプログラム	nanbai.f
入力ファイル	tube.xyz
出力ファイル	nanbai-kekka.xyz

1. まず、このプログラムをコンパイルする。前述のようにコマンドラインで

   f77 nanbai.f

   と入力し、コンパイルする。この時に出来上がったファイル

   a.out

   を使ってこのプログラムを実行する。

2. コマンドラインで a.out と打ち込むと、

   チューブを何倍する? n=

   と聞いてくるので、整数で打ち込む。

3. 出力ファイル nanbai-kekka.xyz に何倍かされたチューブの座標が出力される。

A.1.3 チューブ、又は C_{60} を軸方向に移動する。

	X 軸方向	Z 軸方向
用いるプログラム	idou-x.f	idou-z.f
入力ファイル	tube.xyz	tube.xyz
出力ファイル	tube-idox.xyz	tube-idoz.xyz

この操作を行うことによって、チューブ、 C_{60} それぞれの軸方向に任意の距離だけ移動することができる。 C_{60} 内包チューブを作るときに内包 C_{60} の位置を変えることができる。
このプログラムの実行、コンパイルは *wire*以外 を用いること。

1. まず、これらのプログラムをコンパイルする。前述のようにコマンドラインで
   f77 idou-x.f
   と入力し、コンパイルする。この時に出来上がったファイル
   a.out
   を使ってこのプログラムを実行する。

2. コマンドラインで a.out と打ち込むと、
   x 方向にどれだけ動かす? dx=
   と聞いてくるので、整数で打ち込む。

3. 出力ファイル tube-idox.xyz に移動されたチューブの座標が出力される。

A.2 x-mol の使い方

x-mol とは Research Equipment Inc. 及び dba Minnesota Supercomputer Center,Inc. が製作した分子描画ソフトである。各原子の座標を与えると、拡大、縮小、回転、が自由に行なえる。また、原子間距離や二角面を自動的に計算してくれる。さらにアニメーション形式のデータを与えれば原子の振動などの動画の表示も可能である。このソフトを使えば前述のチューブの座標データから図を起こすことができる。その方法をここで述べる。
このプログラムを実行するホストコンピュータは wire を用いることをおすすめする。(高速でデータ読み込み可能なため。)

A.2.1 描画方法

1. コマンドラインで xmol & と打ち込む。すると下のような画面が現われる。

   - この時エラーメッセージとともに画面が出てこない場合は出力ディスプレィを指定する必要がある。
     - 例えば、ディスプレィ 14 に出力したい場合は

       xmol -display 172.21.88.14:0.0

       と入力する。

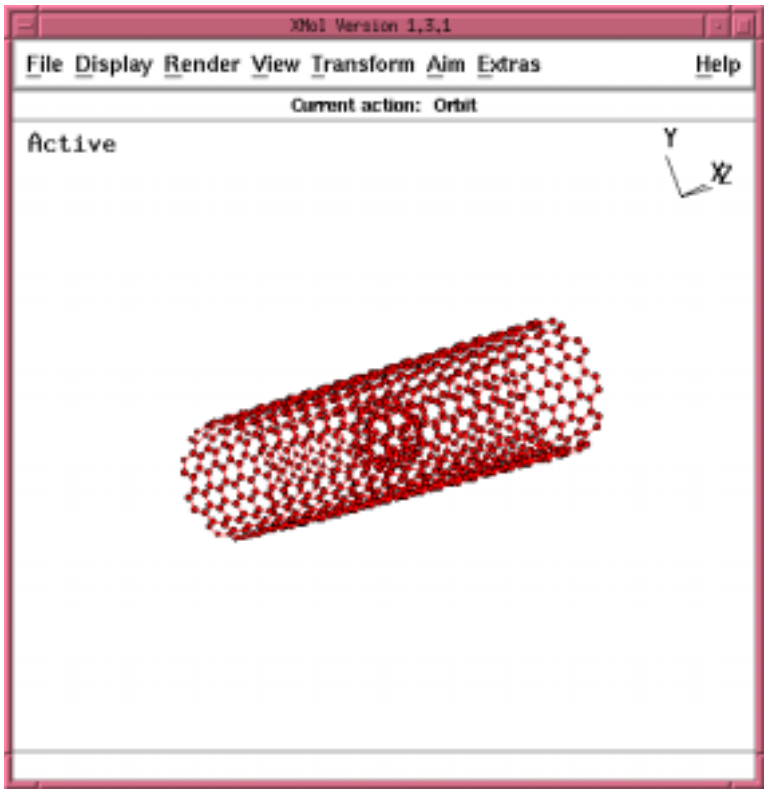

図 1.76 x-mol ファイル表示画面 [1]

2. 画面が立ち上がったら、左上の FILE (READ) をクリックし、読み込みたいデータのファイル名を指定する。

A.2.2 アニメーション表示

アニメーション表示に対応したデータを x-mol で見る場合、フレームを 1 コマずつ動かしてアニメーション表示することができる。

[1]図 1.76 = u00naka/ps/xmol.ps

アニメーション表示のデータとはあらゆる方向に動かした各座標データを同じファイルに順順に書き込んでいったものである。使い方は、表示させたいファイルを前述の方法で表示させたあと、画面右上 Extras → Animete を選択し、表示させる。
(この時データ量が重いと数分かかる時があるので注意。)

A.3 xvgr の使い方

xvgr というソフトを使って x,y 平面のグラフを書き起こす方法を説明する。
注) このプログラムを使用するホストコンピュータは wire 以外 を使用すること。

1. コマンドラインで xvgr & と打ち込む。すると下のような画面が現われる。

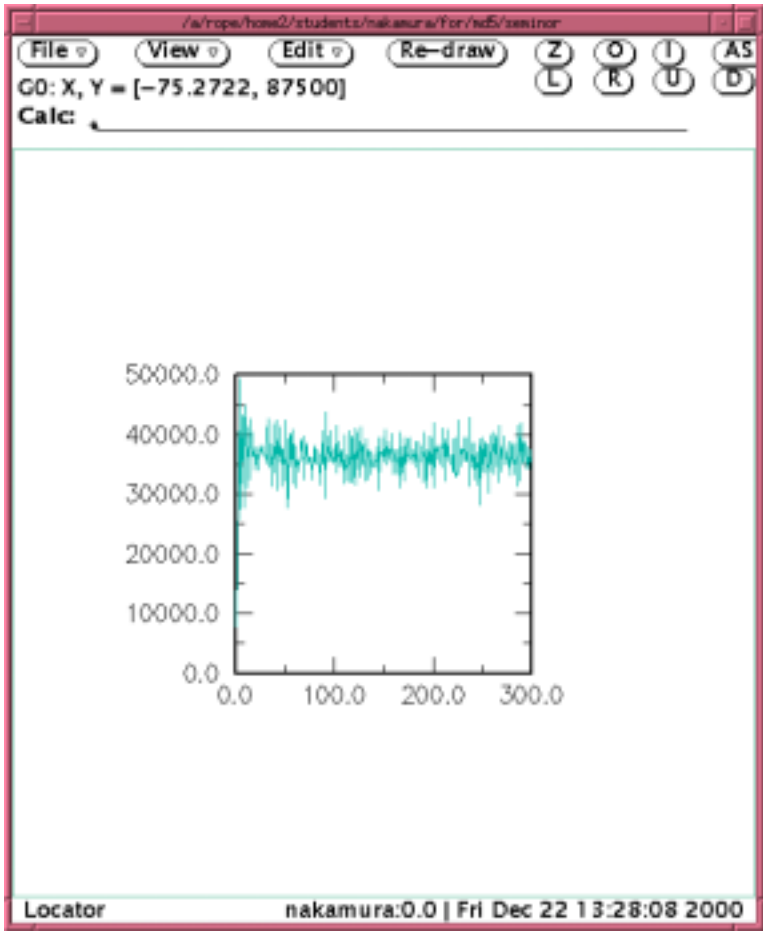

図 1.77 xvgr ファイル表示画面 [2]

2. 左上の FILE を左クリックすると、ファイル選択の画面が現われるので表示させたいデータのファイルを選択する。

[2]図 1.77 = u00naka/ps/xvgr.ps

付録 B

プログラムソース

この論文で作製、改良使用したプログラムソースを掲載する。

B.1 構造最適化プログラム

このプログラムは、炭素構造の３次元座標のファイルを読み込み、エネルギーの極小値を与える３次元座標を求め、標準出力に出力する。原子間ポテンシャルには、Tersoff potential Lennard Jones potential を用いている、また最適化手法に共役勾配法と分子動力学的手法を用いる事ができる。

計算の繰り返し回数は l 86 付近、

```
      K = 0
      DO 41 I = 1 , 2000
```

の 2000 を変える。

共役勾配法を用いる場合は、l 109 付近、 ２微分を計算するところがコメントになっているので、コメント指示を外し、また１次構造最適化をコメントアウトする。

```
      DO 38 J = 1 , MAX_ATOM3
         DIFF2(0,J) = 0
         DD(J) = 0.0D0
 38   CONTINUE

       CALL CALC_DIFF2(N,ZAHYO,LIST,LIST2,KYORI,CUTOFF,DIFF2,DIFF2_DATA)
```

```
        CALL CALC_DIFF2_VDW
     #       (N,ZAHYO,LIST_VDW,KYORI,CUTOFF,DIFF2,DIFF2_DATA)



       CALL CONJGRAD(N,ZAHYO,DIFF1,DIFF2,DIFF2_DATA)


C       DO 39 J = 1 , N*3
C            VELO(J) = VELO(J) + DIFF1(J) * CONV_VELO
C           ZAHYO(J) = ZAHYO(J) + VELO(J)
C            VELO(J) = VELO(J) * 0.9D0
C
C 39     CONTINUE
```

計算を実行するには

初期座標データ tube.xyz を用意して、

```
a.out tube.xyz > kekka.xyz
```

とする。このプログラムの出力データは、アニメーション表示に対応して いない。

なお、このプログラムは次で述べる「温度を考慮した構造最適化プログラム」と重複するのでソースは掲載しない。

B.1.1 構造最適化プログラムの実行

nakamura/for/md5/pro の中にあるプログラム md5.f と PARAMETER を自分のディレクトリにコピーして使うこと。コンパイルの方法などは各セクションに記載してある。

B.2 温度を考慮した構造最適化プログラム

任意の温度を入力し、その温度でナノチューブの構造最適化を実行するプログラム。このプログラムでは、結果が出力されるファイルはあらかじめ決まっており (result.xyz)、アニメーション表示に対応したデータが出力される。

```
c
c     Tersoff potential for Carbon system
c     (1998/Jun/8)   By  R.Hatsuo .
c     (2000/Nov/8)   restored by T.Nakamura
c  入力ファイル .xyz 出力ファイル result.xyz
c  このプログラムは別枠のパラメーターファイルを必要とする。
c
c  温度を任意のステップごとに入力し、計算させるプログラム。
c
```

```
      PROGRAH HD
      INCLUDE 'PARAHETER'

C     原子座標の FILE 名変数
      CHARACTER*50 FILENAHE
      CHARACTER*50 FILENAHE2

C     原子数の保持変数
      INTEGER N

C     loop 用変数
      INTEGER I,J,K

C     温度の変数 (k)
      REAL*8 H,G
      REAL*8 TEHP1
      REAL*8 TEHP2

c     繰り返し変数 (steps)
      INTEGER A,B,H

C     ALPHA、 VV2 の宣言 vv2(2) 1: C60 2: nanotube
      REAL*8 ALPHA1,ALPHA2
      DIHENSION VV2(2)
      REAL*8 VV2
      REAL*8 SQRT

C     座標用配列
      REAL*8  ZAHYO(HAX_ATOH3)
      CHARACTER*8  AN(HAX_ATOH)

C     近接リスト配列
      REAL*8 LIST(0:HAX_LIST_N2,HAX_ATOH)

C     第二 2 近接リスト配列
      REAL*8 LIST2(0:HAX_LIST2_N,HAX_ATOH)

C     近接リスト配列
      REAL*8 LIST_VDW(0:HAX_LIST_VN2,HAX_ATOH)

C     近接データ配列 IDX でリストから参照される
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)

      REAL*8  DIFF1(HAX_ATOH3)
      REAL*8  VELO(HAX_ATOH3)
      REAL*8  DD(HAX_ATOH3)


      INTEGER DIFF2(0:HAX_DIFF2_LIST,HAX_ATOH3)
      REAL*8  DIFF2_DATA(HAX_DIFF2_LIST,HAX_ATOH3)

C     TERSOFF 関数の型
      REAL*8 TERSOFF

C     系のエネルギー
      REAL*8 ENERGY , ENERGY_VDW
      INTEGER IDX

c     itest = 1 debug

      itest = 0

C     FILE 名を取得する
      CALL GETFILENAHE(FILENAHE)
      open(STATUS='OLD',unit=20,file='tempmd.log')
      write(20,25) FILENAHE

 25   format(a50)

C     XYZ 座標を読み込む
      CALL READ_ZAHYO(FILENAHE, N, I,ZAHYO,AN)
      write(*,*)'I = ',I
      DO 29 J = 1 , HAX_ATOH3
         VELO(J) = 0.0D0

 29   CONTINUE

      IDX = 0
      CALL HAKE_LIST(N,ZAHYO,LIST,KYORI,CUTOFF,IDX)
      CALL HAKE_LIST2(N,LIST,LIST2)

      CALL HAKE_LIST_VDW(N,ZAHYO,LIST_VDW,KYORI,CUTOFF,IDX,LIST,
     #LIST2,AN)

      ENERGY = TERSOFF(N,ZAHYO,LIST,KYORI,CUTOFF)
      ENERGY_VDW = VDW(N,ZAHYO,LIST_VDW,KYORI,CUTOFF)
      ENERGY = ENERGY + ENERGY_VDW


      WRITE(*,*)' 何回繰り返す? A='
```

```
      READ(*,*) A
      WRITE(*,*)'TEHPARATURE of C_60 (Real)? H(K)='
      READ(*,*) H
      WRITE(*,*)'TEHPARATURE of NANO (Real)? G(K)='
      READ(*,*) G
      WRITE(*,*)'N*3',N*3
      WRITE(*,*)'C_60 幾つ入ってる B= ?'
      READ(*,*) B
      WRITE(*,*)' 何回ごとに温度入力 H= ?'
      READ(*,*) H

      K = 0
      DO 41 I = 1 , A

      write(*,*)'I=',I
C     近接リストを生成
      IDX = 0
      CALL HAKE_LIST(N,ZAHYO,LIST,KYORI,CUTOFF,IDX)

C     第二近接リストを生成
      CALL HAKE_LIST2(N,LIST,LIST2)

c      WRITE(*,*) IDX
c      WRITE(*,*) 'make_list_vdw'
      CALL HAKE_LIST_VDW(N,ZAHYO,LIST_VDW,KYORI,CUTOFF,IDX,LIST,
     #LIST2,AN)

c      WRITE(*,*) IDX
c      WRITE(*,*) 'make_newkyori'
      CALL HAKE_NEWKYORI(N,ZAHYO,LIST,KYORI,CUTOFF)

c      WRITE(*,*) 'make_newkyoriv'
      CALL HAKE_NEWKYORIV(N,ZAHYO,LIST_VDW,KYORI,CUTOFF)

c      WRITE(*,*) 'calc_diff'
      CALL CALC_DIFF1(N,ZAHYO,LIST,KYORI,CUTOFF,DIFF1)

c      WRITE(*,*) 'calc_diff_vdw'
      CALL CALC_DIFF1_VDW(N,ZAHYO,LIST_VDW,KYORI,CUTOFF,DIFF1)

c  共役勾配法
c      DO 38 J = 1 , MAX_ATOM3
c         DIFF2(0,J) = 0
c         DD(J) = 0.0D0
c 38   CONTINUE

c       CALL CALC_DIFF2(N,ZAHYO,LIST,LIST2,KYORI,CUTOFF,DIFF2,DIFF2_DATA)
c       CALL CALC_DIFF2_VDW
C          #    (N,ZAHYO,LIST_VDW,KYORI,CUTOFF,DIFF2,DIFF2_DATA)


c      CALL CONJGRAD(N,ZAHYO,DIFF1,DIFF2,DIFF2_DATA)
c
c  分子動力学法
      DO 39 J = 1 , N*3
          VELO(J) = VELO(J) + DIFF1(J) * CONV_VELO
        ZAHYO(J) = ZAHYO(J) + VELO(J)

 39   CONTINUE
ccc
c  温度入力
ccc
c
c  c_60 と nanotube を別けてそれぞれに温度を振り分ける。
c  読み込む ファイル の上から 60 番目までが c_60
c  JJ=1 C60, JJ=2 NANOTUBE
c  c_60
c
c  ここで H 回ごとにこのループに入るように条件 IF 文を書く。
c   IF 文と MOD 構文を用いる。
c
       IF (MOD(I,H).EQ.0) THEN

       JJ=1
       NNC60=60*B
       NNC603=NNC60*3
       VV2(JJ) = 0.0
       DO J = 1 , NNC603
       VV2(JJ) = VV2(JJ) + VELO(J)**2
       END DO
       WRITE(*,*)'VV2 OF C60 = ',VV2(JJ)

C
       ALPHA1 = SQRT(2.08D-7*REAL(NNC60)*H/VV2(JJ))
C       --- 計算から求めた alpha をここで入力する。 ---
C       ここでは C_60 用の入力画面。
C
       DO J = 1 , NNC603
       VELO(J) = ALPHA1*VELO(J)
       END DO
C
C 次は nanotube
```

```
C
       JJ=2
       VV2(JJ)=0.0
       DO J = NNC603+1 , N*3
       VV2(JJ) = VV2(JJ) + VELO(J)**2
       END DO
       WRITE(*,*)'VV2 of nanotube = ',VV2(JJ)
       ALPHA2 = SQRT(2.08D-7*REAL(N-NNC60)*G/VV2(JJ))
C
C       --- nanotube 用にここで alpha を入力する。 ---
C
       DO J = NNC603+1 , N*3
       VELO(J) = ALPHA2*VELO(J)
       END DO

c
c ALPHA=SQRT(3*N*Kb*T/m*VV2)
c      =SQRT(3*N*1.38D-23*T/1.99D-26*VV2)
c      =SQRT(2.08D3*N*T/VV2)
c   m/s   と A/fs を比較して、係数を 10D-10 倍する。
c
c ALPHA の画面表示
       WRITE(*,*)'alpha1,2',alpha1,alpha2
       TEHP1 = 0.48D7*VV2(1)*ALPHA1/REAL(NNC60)
       TEHP2 = 0.48D7*VV2(2)*ALPHA2/REAL(N-NNC60)
       WRITE(*,*)'TEHP1,2 =',TEHP1,TEHP2

       ELSE
       END IF

c ファイルへの出力 (.log)
       OPEN(STATUS='UNKNOWN',UNIT=30,FILE='tempC_60.log')
       write(30,*)TEHP1
       OPEN(STATUS='UNKNOWN',UNIT=31,FILE='temp_NANO.log')
       write(31,*)TEHP2
       OPEN(STATUS='UNKNOWN',UNIT=32,FILE='alphaC_60.log')
       write(32,*)alpha1
       OPEN(STATUS='UNKNOWN',UNIT=33,FILE='alpha_NANO.log')
       write(33,*)alpha2
       OPEN(STATUS='UNKNOWN',UNIT=34,FILE='VV2C_60.log')
       write(34,*)VV2(1)
       OPEN(STATUS='UNKNOWN',UNIT=35,FILE='VV2_NANO.log')
       write(35,*)VV2(2)

       ENERGY = TERSOFF(N,ZAHYO,LIST,KYORI,CUTOFF)
       ENERGY_VDW = VDW(N,ZAHYO,LIST_VDW,KYORI,CUTOFF)
       ENERGY = ENERGY +  ENERGY_VDW

       WRITE(20,40) I , ENERGY

 40    format(i6,f20.7)

      CALL PRINT_ZAHYO(N,J,ZAHYO,I,ENERGY,DIFF1,AN)

 41   CONTINUE

C 温度読み込みの確認画面表示
      WRITE(*,*)'TEHP=',T

      close(20)

      STOP
      END

c
c This program is for md to run on Dec Fortran
c

CC
C     座標データの FILE 名を取得する サブルーチン
CC
 101  SUBROUTINE GETFILENAHE(FILENAHE)
      CHARACTER*50 FILENAHE
      INTEGER I
      I = IARGC()
      IF (I .NE. 1) then
         WRITE(*,*) 'Usage: opt hoge.xyz'
         GOTO 110
      ENDIF

      CALL GETARG(1,FILENAHE)

      RETURN
 110  STOP
      END


 121  SUBROUTINE GETARGU(LINE,I,ARGU)
      CHARACTER*80 LINE
      CHARACTER*80 DUH
      CHARACTER*80 ARGU
      INTEGER I
```

```
      INTEGER J
      INTEGER K
      INTEGER lenline

      DUH = LINE


      do 128 K = 1 , I
         lenline = len(DUH)
         DO 122 J = 1 , lenline

            IF ( DUH(J:J) .EQ. ' ') THEN
               GOTO 122
            ENDIF
            IF ( DUH(J:J) .EQ. char(9) ) THEN
               GOTO 122
            ENDIF
            IF ( DUH(J:J) .EQ. char(0) ) THEN
               GOTO 122
            ENDIF
            GOTO 123
 122     CONTINUE

 123     DUH = DUH(J:lenline)

         lenline = len(DUH)
         DO 124 J = 2 , lenline
            IF ( DUH(J:J) .EQ. ' ') THEN
               GOTO 125
            ENDIF
            IF ( DUH(J:J) .EQ. char(9) ) THEN
               GOTO 125
            ENDIF
            IF ( DUH(J:J) .EQ. char(0) ) THEN
               GOTO 125
            ENDIF
 124     CONTINUE
 125     ARGU = DUH(1:J-1)

         DUH=DUH(J:lenline)

 128  continue

      RETURN

c 129  STOP
      END

 130  REAL*8 FUNCTION ATOR(STR)
      CHARACTER*80 STR
      INTEGER I
      INTEGER J
      INTEGER S
      INTEGER D
      INTEGER DCU
      INTEGER*4 R
      INTEGER*4 E

      I = 1
      S = 1

      R = 0
      EC = 0
      E = 0

      IF ( STR(1:1) .EQ. '+') THEN
         S = 1
         I = I + 1
      ENDIF
      IF ( STR(1:1) .EQ. '-') THEN
         S = -1
         I = I + 1
      ENDIF

      DO 140 J = I , 80
         IF ( STR(J:J) .EQ. '.' ) THEN
            EC = 1
            GOTO 140
         ENDIF

         D = ICHAR(STR(J:J)) - ICHAR('0')

         IF ( (D .GT. 9) .OR. (D .LT. 0)) THEN
         GOTO 145
      ENDIF
      R = R * 10 + D
      E = E + EC

      IF ( R .GT. 9999999 ) THEN
         GOTO 145
      ENDIF
```

```
 140  CONTINUE

 145  ATOR = 1.0D0  * S * R / (10.0D0**E)

      RETURN

c 149  STOP
      END



CC
C     座標データ取り込み サブルーチン
CC

 201  SUBROUTINE READ_ZAHYO(FILENAHE, N, I,ZAHYO,AN)
      INCLUDE 'PARAHETER'

      CHARACTER*50 FILENAHE

C     原子数を保持する変数
      INTEGER N
      INTEGER N3

C     FILE IO チェック用変数
      INTEGER IOCHECK

C     loop variable
      INTEGER I

C     炭素原子の座標用配列
      REAL*8 ZAHYO(HAX_ATOH3)

C     元素名用変数
      CHARACTER*8 AN(HAX_ATOH)

      CHARACTER*80 LINE,ARGU

C     FILE OPEN
      OPEN(60,FILE=FILENAHE,STATUS='OLD',IOSTAT=IOCHECK)

C     FILE OPEN のエラーチェック
      IF ( IOCHECK ) THEN
         WRITE(*,*) 'File open error.'
         GOTO 299
      ENDIF

C     原子数の制限チェック
      READ(60,*) N

      IF ( HAX_ATOH .LT. N ) THEN
         WRITE(*,*) "Too many atoms."
         GOTO 299
      ENDIF

      N3 = N * 3

      READ(60,209) LINE
 208  format(f14.8)
 209  format(a80)

C     座標読み込み 単位はÅ
      DO 210 I = 1 , N3 , 3
         READ(60,209) LINE
         CALL GETARGU(LINE,1,ARGU)
         AN(I/3+1)=ARGU
         CALL GETARGU(LINE,2,ARGU)
         ZAHYO(I) = ATOR(ARGU)
         CALL GETARGU(LINE,3,ARGU)
         ZAHYO(I+1) = ATOR(ARGU)
         CALL GETARGU(LINE,4,ARGU)
         ZAHYO(I+2) = ATOR(ARGU)

C      読み込んだ座標の画面表示（確認）
        itest=0
         if(itest.eq.1) then
         WRITE(*,*)'X,Y,Z=',ZAHYO(I),ZAHYO(I+1),ZAHYO(I+3)
         endif
C         READ(60,*) AN(I/3+1) , ZAHYO(I) ,ZAHYO(I+1) ,ZAHYO(I+2)
 210  CONTINUE

      CLOSE(60,STATUS='KEEP')
      RETURN

 299  CLOSE(60,STATUS='KEEP')
      STOP
      END
cc
C     結果出力サブルーチン 1
cc

 301  SUBROUTINE PRINT_ZAHYO(N,J,ZAHYO,T,ENERGY,DIFF1,AN)
```

```
      INCLUDE 'PARAHETER'
      INTEGER N,J,N3
      REAL*8 TEHP
      REAL*8 ZAHYO(HAX_ATOH3)
      CHARACTER*8 AN(HAX_ATOH)
      REAL*8 DIFF1(HAX_ATOH3)
      REAL*8 ENERGY
      INTEGER I


c 結果のファイルへの出力 (result.xyz)
c result.xyz は一回ずつの原子の座標をプロットする。アニメーションで確認できる。
      OPEN(1,STATUS='OLD',FILE='result.xyz')
      WRITE(1,*) N
C      WRITE(1,*) 'TIHE=',T*TIHE_DIV*1.0D15,'fs  ENERGY =',ENERGY
c      WRITE(1,*)'fs  ENERGY =',ENERGY,'TEHP(K)=',TEHP

       WRITE(1,304)'fs  ENERGY (eV)=',ENERGY

      N3 = N * 3

      DO 311 I = 1 , N3 , 3
         WRITE(1,305) AN(I/3+1) , ZAHYO(I) , ZAHYO(I+1) , ZAHYO(I+2)
     #    , DIFF1(I) *10.0 , DIFF1(I+1) * 10.0 , DIFF1(I+2) * 10.0
 304     FORHAT(a20,f15.2)
 305     FORHAT(a5,f15.7,f15.7,f15.7,f15.7,f15.7,f15.7)
 311  CONTINUE

C
C 結果出力サブルーチン 2
C
c
c 302  SUBROUTINE PRINT_ZAHYO2(N,ZAHYO,T,ENERGY,DIFF1,AN)
c      INCLUDE 'PARAHETER'
c      INTEGER N,N3
c      INTEGER T
c      REAL*8 ZAHYO(HAX_ATOH3)
c      CHARACTER*8 AN(HAX_ATOH)
c      REAL*8 DIFF1(HAX_ATOH3)
c      REAL*8 ENERGY
c      INTEGER I
c
c 結果のファイルへの出力 (result1.xyz)
c      OPEN(STATUS='OLD',UNIT=N,FILE='result1.xyz')
c      WRITE(N,*) N
C      WRITE(N,*) 'TIHE=',T*TIHE_DIV*1.0D15,'fs  ENERGY =',ENERGY
c      WRITE(N,*)'fs  ENERGY =',ENERGY,'TEHP(K)=',T
c
C

      N3 = N * 3

      DO 312 I = 1 , N3 , 3
         WRITE(N,306) AN(I/3+1) , ZAHYO(I) , ZAHYO(I+1) , ZAHYO(I+2)
     #    , DIFF1(I) *10.0 , DIFF1(I+1) * 10.0 , DIFF1(I+2) * 10.0

 306     FORHAT(a5,f12.7,f12.7,f12.7,f12.7,f12.7,f12.7)
 312  CONTINUE


 398  RETURN
c 399  STOP
      END

CC
C     最近接原子の LIST を作成するサブルーチン
CC

C
C     LIST(0,N) : 原子番号 N の近接原子の数
C     LIST(I,N) : 原子番号 N の I 番目の近接原子インデックス
C     LIST(I+HAX_LIST_N ,N) : 原子番号 N の I 番目の近接原子番号

 401  SUBROUTINE HAKE_LIST(N,ZAHYO,LIST,KYORI,CUTOFF,IDX)
      INCLUDE 'PARAHETER'
      INTEGER N,N3

      REAL*8  ZAHYO(HAX_ATOH3)
      INTEGER LIST(0:HAX_LIST_N2,HAX_ATOH)
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)
      INTEGER IDX

      REAL*8  HAX_X,HAX_Y,HAX_Z
      REAL*8  HIN_X,HIN_Y,HIN_Z

      INTEGER I,K,J
      INTEGER E,F,G
      INTEGER L,H

      INTEGER I3
```

```
      INTEGER HESH_X,HESH_Y,HESH_Z
      INTEGER TX,TY,TZ
      INTEGER TI,TJ,TI3,TJ3

      INTEGER HESH(0:HAX_HESH_ATOH,
     #       0:HAX_HESH_X+1,0:HAX_HESH_Y+1,0:HAX_HESH_Z+1)

      REAL*8 THP_R,THP_CUTOFF

      REAL*8 SIN

      N3 = N * 3

C     リスト配列の初期化
      DO 405 I = 1 , HAX_ATOH
         LIST(0,I) = 0
 405  CONTINUE

C     HESH 配列の初期化
      DO 406 I = 0 , HAX_HESH_X+1
         DO 406 J = 0 , HAX_HESH_Y+1
            DO 406 K = 0 , HAX_HESH_Z+1
               HESH(0,I,J,K) = 0
 406  CONTINUE

C     系の座標の最大値 - 最小値を検索する
      HAX_X = ZAHYO(1)
      HAX_Y = ZAHYO(2)
      HAX_Z = ZAHYO(3)
      HIN_X = ZAHYO(1)
      HIN_Y = ZAHYO(2)
      HIN_Z = ZAHYO(3)

      DO 410 I = 4 , N3 , 3
         IF ( HAX_X .LT. ZAHYO(I) ) THEN
            HAX_X = ZAHYO(I)
         ENDIF
         IF ( HAX_Y .LT. ZAHYO(I+1) ) THEN
            HAX_Y = ZAHYO(I+1)
         ENDIF
         IF ( HAX_Z .LT. ZAHYO(I+2) ) THEN
            HAX_Z = ZAHYO(I+2)
         ENDIF
         IF ( HIN_X .GT. ZAHYO(I) ) THEN
            HIN_X = ZAHYO(I)
         ENDIF
         IF ( HIN_Y .GT. ZAHYO(I+1) ) THEN
            HIN_Y = ZAHYO(I+1)
         ENDIF
         IF ( HIN_Z .GT. ZAHYO(I+2) ) THEN
            HIN_Z = ZAHYO(I+2)
         ENDIF
 410  CONTINUE
c  最大、最小の画面表示（確認）
      WRITE(*,*)'HAX_',HAX_X,HAX_Y,HAX_Z
      WRITE(*,*)'HIN_',HIN_X,HIN_Y,HIN_Z
      WRITE(*,*) ' D_.', REAL(HESH_D)

C     一度、大まかに原子をグルーピングする。
C     最大値最小値から、メッシュの個数を決める

      HESH_X = 1 + ( ( HAX_X - HIN_X ) / REAL(HESH_D) )
      HESH_Y = 1 + ( ( HAX_Y - HIN_Y ) / REAL(HESH_D) )
      HESH_Z = 1 + ( ( HAX_Z - HIN_Z ) / REAL(HESH_D) )

C     HESH 用の配列 に収まらない場合 STOP
      IF ( HESH_X .LE. HAX_HESH_X ) THEN
         GOTO 415
      ENDIF
      IF ( HESH_Y .LE. HAX_HESH_Y ) THEN
         GOTO 415
      ENDIF
      IF ( HESH_Z .LE. HAX_HESH_Z ) THEN
         GOTO 415


      ENDIF

      WRITE(*,*) 'HESH is overflow: HAX_HESH.',  HESH_X,HESH_Y,HESH_Z
      WRITE(*,*) 'HESH is overflow: HAX_.', HAX_X,HAX_Y,HAX_Z
      WRITE(*,*) 'HESH is overflow: HIN_.', HIN_X,HIN_Y,HIN_Z
      STOP


C     全ての原子を HESH に グルーピングする。

 415  DO 420 I = 1 , N
         I3 = I * 3 - 2
         TX = 1 + ( ( ZAHYO(I3) - HIN_X ) / HESH_D )
         TY = 1 + ( ( ZAHYO(I3+1) - HIN_Y ) / HESH_D )
         TZ = 1 + ( ( ZAHYO(I3+2) - HIN_Z ) / HESH_D )
```

```
C     HESH 配列があるれる場合 STOP
         IF ( HESH(0,TX,TY,TZ) .EQ. MAX_HESH_ATOH ) THEN
            WRITE(*,*) 'HESH_ATOH is overflow: MAX_HESH_ATOH.'
            STOP
         ENDIF

         HESH(0,TX,TY,TZ) =  HESH(0,TX,TY,TZ) + 1
         HESH( HESH(0,TX,TY,TZ) , TX,TY,TZ) = I

 420  CONTINUE


C     HESH のブロックの 近接内 (27 ブロック)
C     で 距離を計算し 近接 LIST を作成する
C     ついでに カットオフ関数の値も計算する

      DO 443 I = 1 , HESH_X
         DO 442 J = 1 , HESH_Y
            DO 441 K = 1 , HESH_Z

               IF ( HESH(0,I,J,K) .EQ. 0 ) THEN
                  GOTO 441
               ENDIF

               DO 433 E = I - 1  , I + 1
                  DO 432 F = J - 1 ,  J + 1
                     DO 431 G = K - 1  , K + 1

                        IF ( HESH(0,E,F,G) .EQ. 0 ) THEN
                           GOTO 431
                        ENDIF

                        DO 425 L = 1 , HESH(0,I,J,K)
                           DO 424 H = 1 , HESH(0,E,F,G)

                              TI = HESH(L,I,J,K)
                              TJ = HESH(H,E,F,G)

                              IF ( TI .LE. TJ ) THEN
                                 GOTO 424
                              ENDIF

                              TI3 = TI*3-2
                              TJ3 = TJ*3-2
C
C     原子間距離の計算
C
      THP_R =
     #     ( ( ZAHYO(TJ3  ) - ZAHYO(TI3  ) )**2 +
     #       ( ZAHYO(TJ3+1) - ZAHYO(TI3+1) )**2 +
     #       ( ZAHYO(TJ3+2) - ZAHYO(TI3+2) )**2  )**0.5d0

C
C     近接でない場合場合 処理を飛ばす
C
      IF ( THP_R .GT. (PRH_CR + PRH_CD) ) THEN
         GOTO 424
      ENDIF

C     LIST の配列があふれる時 STOP

      IF ( ( LIST(0,TI) .EQ. MAX_LIST_N ) .OR.
     #     ( LIST(0,TJ) .EQ. MAX_LIST_N ) ) THEN
         WRITE(*,*) 'LIST_N is overflow: MAX_LIST_N.'
         STOP
      ENDIF

      IDX = IDX + 1
      KYORI(IDX) = THP_R

      LIST(0,TI) = LIST(0,TI) + 1
      LIST( LIST(0,TI) ,TI ) = IDX
      LIST( LIST(0,TI) + MAX_LIST_N ,TI ) = TJ

      LIST(0,TJ) = LIST(0,TJ) + 1
      LIST( LIST(0,TJ) ,TJ ) = IDX
      LIST( LIST(0,TJ) + MAX_LIST_N ,TJ ) = TI

C
C     カットオフ関数を計算
C

      THP_CUTOFF = 1.0D0
      IF ( THP_R .GE. ( PRH_CR - PRH_CD) ) THEN
         THP_CUTOFF = 0.5D0 *
     #   ( 1.0D0 - SIN( PI * ( THP_R - PRH_CR ) / (2.0D0 * PRH_CD ) ))
      ENDIF
      IF ( THP_R .GE. ( PRH_CR + PRH_CD) ) THEN
         THP_CUTOFF = 0.0D0
      ENDIF

      CUTOFF(IDX) = THP_CUTOFF
```

```
 424  CONTINUE
 425  CONTINUE


 431  CONTINUE
 432  CONTINUE
 433  CONTINUE

 441  CONTINUE
 442  CONTINUE
 443  CONTINUE

C      DO 452 I = 1 , N
C         write(*,*) "--",I
C         DO 451 J = 1 , LIST(0,I)
C            WRITE(*,*) LIST(J + HAX_LIST_N , I)
C 451     CONTINUE
C 452   CONTINUE

 469  RETURN
c 499  STOP
      END

C
C     i 原子に関する Tersoff ポテンシャル
C

 501  REAL*8 FUNCTION TERSOFF_I(I,ZAHYO,LIST,KYORI,CUTOFF)
      INCLUDE 'PARAHETER'
      INTEGER I

      REAL*8  ZAHYO(HAX_ATOH3)
      INTEGER LIST(0:HAX_LIST_N2,HAX_ATOH)
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)

      INTEGER J,K,L,H
      INTEGER I3,J3,K3
      INTEGER IDXJ,IDXK

      REAL*8 G,GCOS
      REAL*8 ZETA
      REAL*8 I_X,I_Y,I_Z
      REAL*8 J_X,J_Y,J_Z
      REAL*8 K_X,K_Y,K_Z
      REAL*8 Bij

      REAL*8 EXP

      REAL*8 THP

      I3 = I*3 - 2
      I_X = ZAHYO(I3)
      I_Y = ZAHYO(I3+1)
      I_Z = ZAHYO(I3+2)

      THP = 0.0D0

      DO 520 L = 1 , LIST(0,I)
         IDXJ = LIST(L,I)
         J = LIST(L+HAX_LIST_N,I)
         J3 = J*3 - 2
         J_X = ZAHYO(J3)
         J_Y = ZAHYO(J3+1)
         J_Z = ZAHYO(J3+2)

         ZETA = 0.0D0

         DO 510 H = 1 , LIST(0,I)
            IF ( L .EQ. H ) THEN
               GOTO 510
            ENDIF

            IDXK = LIST(H,I)
            K = LIST(H+HAX_LIST_N,I)
            K3 = K*3 - 2
            K_X = ZAHYO(K3)
            K_Y = ZAHYO(K3+1)
            K_Z = ZAHYO(K3+2)

            GCOS =
     #           (
     #           (J_X - I_X ) * (K_X - I_X )+
     #           (J_Y - I_Y ) * (K_Y - I_Y )+
     #           (J_Z - I_Z ) * (K_Z - I_Z )
     #           ) / KYORI(IDXJ) / KYORI(IDXK)

            IF ( KYORI(IDXK) .EQ. 0.0D0 ) THEN
               WRITE(*,*) KYORI(IDXK) ,I,J,K
            ENDIF
```

```
            G = ( PRH_C * PRH_C ) / ( PRH_D * PRH_D ) -
     #           ( PRH_C * PRH_C ) /
     #           ( PRH_D * PRH_D + ( PRH_H - GCOS ) * (PRH_H - GCOS ))

            G = PRH_A * ( G + 1.0D0 )
            ZETA = ZETA + CUTOFF(IDXK) * G

 510     CONTINUE

         Bij = ( 1.0 + ( ZETA )**( PRH_ETA) )**( -PRH_DELTA )

         THP = THP +
     #   CUTOFF(IDXJ) *
     #   (
     #    PRH_EA * EXP( -PRH_LUHBDA1 * KYORI(IDXJ) ) +
     #    (PRH_EB) * Bij * EXP( -PRH_LUHBDA2 * KYORI(IDXJ))
     #   )

 520  CONTINUE

      TERSOFF_I = THP

      RETURN
c 599  STOP
      END

 601  REAL*8 FUNCTION TERSOFF(N,ZAHYO,LIST,KYORI,CUTOFF)
      INCLUDE 'PARAHETER'
      INTEGER I
      INTEGER N
      REAL*8  ZAHYO(HAX_ATOH3)
      INTEGER LIST(0:HAX_LIST_N2,HAX_ATOH)
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)
      REAL*8  TERSOFF_I

      TERSOFF = 0.0D0

      DO 610 I = 1 , N
         TERSOFF = TERSOFF +
     #        TERSOFF_I(I,ZAHYO,LIST,KYORI,CUTOFF)

 610  CONTINUE

      TERSOFF = TERSOFF / 2.0D0

      RETURN
c 699  STOP
      END

 701  SUBROUTINE CALC_DIFF1(N,ZAHYO,LIST,KYORI,CUTOFF,DIFF1)
      INCLUDE 'PARAHETER'
      INTEGER N,N3
      REAL*8  ZAHYO(HAX_ATOH3)
      INTEGER LIST(0:HAX_LIST_N2,HAX_ATOH)
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)

      REAL*8  DIFF1(HAX_ATOH3)

      INTEGER I,ID3
      REAL*8  ZAHYO_ORG
      REAL*8 TERSOFF_LI

      REAL*8 THP

      N3 = N * 3

      DO 710 I = 1 , N3
         ID3 = ( I + 2 ) / 3
         ZAHYO_ORG = ZAHYO(I)

         ZAHYO(I) = ZAHYO_ORG + EPS1
         CALL RECALC(ID3,ZAHYO,LIST,KYORI,CUTOFF)
         THP = TERSOFF_LI(ID3,ZAHYO,LIST,KYORI,CUTOFF)

         ZAHYO(I) = ZAHYO_ORG - EPS1
         CALL RECALC(ID3,ZAHYO,LIST,KYORI,CUTOFF)
         THP = THP -
     #        TERSOFF_LI(ID3,ZAHYO,LIST,KYORI,CUTOFF)

         ZAHYO(I) = ZAHYO_ORG
         CALL RECALC(ID3,ZAHYO,LIST,KYORI,CUTOFF)

         THP = THP / ( 2.0 * EPS1 )

         DIFF1(I) = -THP

 710  CONTINUE

      RETURN
```

```
c 799  STOP
      END

 801  SUBROUTINE RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
      INCLUDE 'PARAHETER'
      INTEGER I,I3
      REAL*8  ZAHYO(HAX_ATOH3)
      INTEGER LIST(0:HAX_LIST_N2,HAX_ATOH)
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)

      INTEGER J,J3
      INTEGER IDXJ
      INTEGER H

      REAL*8 THP_R
      REAL*8 THP_CUTOFF

      REAL*8 SIN
      REAL*8 SQRT

      I3 = I*3-2

      DO 810 H = 1 , LIST(0,I)
         IDXJ = LIST(H,I)
         J = LIST(H+HAX_LIST_N,I)
         IDXJ = LIST(H,I)

         J3 = J*3-2
          THP_R =
     #    SQRT    ( ( ZAHYO(J3) - ZAHYO(I3) )**2 +
     #        ( ZAHYO(J3+1) - ZAHYO(I3+1) )**2 +
     #        ( ZAHYO(J3+2) - ZAHYO(I3+2) )**2  )


         KYORI(IDXJ) = THP_R

         THP_CUTOFF = 1.0D0
         IF ( THP_R .GE. ( PRH_CR - PRH_CD) ) THEN
         THP_CUTOFF = 0.5D0 *
     #   ( 1.0D0 - SIN( PI * ( THP_R - PRH_CR ) / (2.0D0 * PRH_CD ) ))
         ENDIF
         IF ( THP_R .GE. ( PRH_CR + PRH_CD) ) THEN
            THP_CUTOFF = 0.0D0
         ENDIF

         CUTOFF(IDXJ) = THP_CUTOFF

 810  CONTINUE

      RETURN
c 899  STOP
      END

 901  REAL*8 FUNCTION TERSOFF_LI(I,ZAHYO,LIST,KYORI,CUTOFF)
      INCLUDE 'PARAHETER'
      INTEGER I
      REAL*8  ZAHYO(HAX_ATOH3)
      INTEGER LIST(0:HAX_LIST_N2,HAX_ATOH)
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)
      REAL*8  TERSOFF_I
      INTEGER L

      INTEGER J

      TERSOFF_LI = TERSOFF_I(I,ZAHYO,LIST,KYORI,CUTOFF)

      DO 910 L = 1 , LIST(0,I)
         J = LIST(L + HAX_LIST_N,I)
         TERSOFF_LI = TERSOFF_LI +
     #       TERSOFF_I(J,ZAHYO,LIST,KYORI,CUTOFF)

 910  CONTINUE

      RETURN

c 999  STOP
      END


1001 SUBROUTINE HAKE_LIST2(N,LIST,LIST2)
      INCLUDE 'PARAHETER'
      INTEGER N
      INTEGER LIST(0:HAX_LIST_N2,HAX_ATOH)
      INTEGER LIST2(0:HAX_LIST2_N,HAX_ATOH)

      INTEGER I,J,K,L,H

      DO 1010 I = 1 , N
         LIST2(0,I) = 0
```

```
         DO 1005 J = 1 , LIST(0,I)
            L = LIST(J+HAX_LIST_N,I)

            DO 1003 K = 1 , LIST(0,L)
               H = LIST(K+HAX_LIST_N,L)
               IF ( H .EQ. I) THEN
                  GOTO 1003
               ENDIF
               IF ( H .EQ. L) THEN
                  GOTO 1003
               ENDIF

               IF ( LIST2(0,I) .EQ. HAX_LIST2_N ) THEN
                  WRITE(*,*) 'HAX_LIST2_N is overflow'
     & ,i,LIST2(0,i),HAX_LIST2_N,HAX_ATOH
                  STOP
               ENDIF

               LIST2(0,I) = LIST2(0,I) + 1
               LIST2(LIST2(0,I),I) = H

 1003       CONTINUE
 1005    CONTINUE
 1010 CONTINUE

      RETURN
c 1099 STOP
      END

 1101 SUBROUTINE CALC_DIFF2
     #(N,ZAHYO,LIST,LIST2,KYORI,CUTOFF,DIFF2,DIFF2_DATA)
      INCLUDE 'PARAHETER'

      INTEGER N,N3
      INTEGER LIST(0:HAX_LIST_N2,HAX_ATOH)
      INTEGER LIST2(0:HAX_LIST2_N,HAX_ATOH)

      REAL*8  ZAHYO(HAX_ATOH3)
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)

      INTEGER DIFF2(0:HAX_DIFF2_LIST,HAX_ATOH3)
      REAL*8  DIFF2_DATA(HAX_DIFF2_LIST,HAX_ATOH3)

      INTEGER I,J,K
      INTEGER L

      N3 = N * 3

C      DO 1110 I = 1 , HAX_ATOH3
C         DIFF2(0,I) = 0
C 1110 CONTINUE

      DO 1150 I = 1 , N
C     同一原子の座標系
         CALL CALC_DIFF2_1
     #        (I,ZAHYO,LIST,KYORI,CUTOFF,DIFF2,DIFF2_DATA)

C     隣合う原子の座標系
         DO 1145 L = 1 , LIST(0,I)
            J = LIST(L+HAX_LIST_N,I)

            CALL CALC_DIFF2_2
     #           (I,J,ZAHYO,LIST,KYORI,CUTOFF,DIFF2,DIFF2_DATA)
 1145    CONTINUE


C     第二隣接の座標系

         DO 1146 L = 1 , LIST2(0,I)
            J = LIST2(L,I)
            CALL CALC_DIFF2_3
     #           (I,J,ZAHYO,LIST,KYORI,CUTOFF,DIFF2,DIFF2_DATA)
 1146       CONTINUE

 1150 CONTINUE

      RETURN
c 1199 STOP
      END

 1201 SUBROUTINE CALC_DIFF2_1
     #     (I,ZAHYO,LIST,KYORI,CUTOFF,DIFF2,DIFF2_DATA)

      INCLUDE 'PARAHETER'
      INTEGER I
      INTEGER LIST(0:HAX_LIST_N2,HAX_ATOH)

      REAL*8  ZAHYO(HAX_ATOH3)
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)
```

```
      INTEGER DIFF2(0:HAX_DIFF2_LIST,HAX_ATOM3)
      REAL*8  DIFF2_DATA(HAX_DIFF2_LIST,HAX_ATOM3)
      REAL*8  TERSOFF_LI

      INTEGER I3 , J3
      INTEGER II,JJ
      INTEGER I3II,J3JJ
      REAL*8 X,Y

      REAL*8 THP
      REAL*8 DATA

      I3 = I*3 - 3
      J3 = I3

      DO 1230 II = 1 , 3
         DO 1220 JJ = 1 , 3
            I3II = I3 + II
            J3JJ = J3 + JJ

            IF ( I3II .NE. J3JJ ) THEN

               X = ZAHYO(I3II)
               Y = ZAHYO(J3JJ)

               ZAHYO(I3II) = X + EPS2
               ZAHYO(J3JJ) = Y + EPS2
               CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
               THP = TERSOFF_LI(I,ZAHYO,LIST,KYORI,CUTOFF)

               ZAHYO(I3II) = X - EPS2
               ZAHYO(J3JJ) = Y - EPS2
               CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
               THP = THP +
     #              TERSOFF_LI(I,ZAHYO,LIST,KYORI,CUTOFF)

               ZAHYO(I3II) = X + EPS2
               ZAHYO(J3JJ) = Y - EPS2
               CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
               THP = THP -
     #              TERSOFF_LI(I,ZAHYO,LIST,KYORI,CUTOFF)

               ZAHYO(I3II) = X - EPS2
               ZAHYO(J3JJ) = Y + EPS2
               CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
               THP = THP -
     #              TERSOFF_LI(I,ZAHYO,LIST,KYORI,CUTOFF)

               DATA = THP / ( 8.0D0 * EPS2 * EPS2 )

               ZAHYO(I3II) = X
               ZAHYO(J3JJ) = Y

            ELSE

               X = ZAHYO(I3II)

               ZAHYO(I3II) = X + 2.0D0*EPS2
               CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
               THP = TERSOFF_LI(I,ZAHYO,LIST,KYORI,CUTOFF)

               ZAHYO(I3II) = X - 2.0D0*EPS2
               CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
               THP = THP +
     #              TERSOFF_LI(I,ZAHYO,LIST,KYORI,CUTOFF)

               ZAHYO(I3II) = X + EPS2
               CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
               THP = THP -
     #              TERSOFF_LI(I,ZAHYO,LIST,KYORI,CUTOFF)

               ZAHYO(I3II) = X - EPS2
               CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
               THP = THP -
     #              TERSOFF_LI(I,ZAHYO,LIST,KYORI,CUTOFF)

               DATA = THP / ( 3.0D0 * EPS2 * EPS2 )

               ZAHYO(I3II) = X

            ENDIF

            CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
            CALL ADD_DIFF2( DATA , I3II , J3JJ , DIFF2 ,DIFF2_DATA)

 1220    CONTINUE
 1230 CONTINUE

      RETURN
c 1299 STOP
      END
```

```
 1301 SUBROUTINE ADD_DIFF2( DATA , I3II , J3JJ , DIFF2 , DIFF2_DATA )
      INCLUDE 'PARAHETER'

      INTEGER I3II , J3JJ
      INTEGER DIFF2(0:HAX_DIFF2_LIST,HAX_ATOH3)
      REAL*8  DIFF2_DATA(HAX_DIFF2_LIST,HAX_ATOH3)
      REAL*8  DATA

      IF ( DIFF2(0,I3II) .EQ. HAX_DIFF2_LIST ) THEN
         WRITE(*,*) 'HAX_DIFF2_LIST is overflow.'
         STOP
      ENDIF

      DIFF2(0,I3II) = DIFF2(0,I3II) + 1
      DIFF2(DIFF2(0,I3II),I3II) = J3JJ
      DIFF2_DATA(DIFF2(0,I3II),I3II) = DATA
C      WRITE(*,*) I3II , J3JJ, DATA

      RETURN
c 1349 STOP
      END

 1351 SUBROUTINE ADD2_DIFF2( DATA , I3II , J3JJ , DIFF2 , DIFF2_DATA )
      INCLUDE 'PARAHETER'

      INTEGER I3II , J3JJ
      INTEGER DIFF2(0:HAX_DIFF2_LIST,HAX_ATOH3)
      REAL*8  DIFF2_DATA(HAX_DIFF2_LIST,HAX_ATOH3)
      REAL*8  DATA
      INTEGER I


      DO 1360 I = 1 , DIFF2(0,I3II)
         IF ( DIFF2(I,I3II) .EQ. J3JJ ) THEN
            DIFF2_DATA(I,I3II) = DIFF2_DATA(I,I3II) + DATA
            GOTO 1380
         ENDIF
 1360 CONTINUE

      WRITE(*,*) 'error ADD2_DIFF2'
      STOP
C     WRITE(*,*) I3II , J3JJ, DATA



 1380 RETURN
c 1399 STOP
      END




 1401 SUBROUTINE CALC_DIFF2_2
     #     (I,J,ZAHYO,LIST,KYORI,CUTOFF,DIFF2,DIFF2_DATA)

      INCLUDE 'PARAHETER'
      INTEGER I,J
      INTEGER LIST(0:HAX_LIST_N2,HAX_ATOH)
      INTEGER LIST_IJ(0:HAX_LIST_IJ)

      REAL*8  ZAHYO(HAX_ATOH3)
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)

      INTEGER DIFF2(0:HAX_DIFF2_LIST,HAX_ATOH3)
      REAL*8  DIFF2_DATA(HAX_DIFF2_LIST,HAX_ATOH3)
      REAL*8  TERSOFF_I
      REAL*8  TERSOFF_LIJ

      INTEGER I3 , J3
      INTEGER II,JJ
      INTEGER I3II,J3JJ
      REAL*8 X,Y

      REAL*8 THP
      REAL*8 DATA

      CALL HAKE_LIST_IJ( I,J,LIST_IJ,LIST )

      I3 = I*3 - 3
      J3 = J*3 - 3

      DO 1430 II = 1 , 3
         DO 1420 JJ = 1 , 3
            I3II = I3 + II
            J3JJ = J3 + JJ

               X = ZAHYO(I3II)
               Y = ZAHYO(J3JJ)

               ZAHYO(I3II) = X + EPS2
               ZAHYO(J3JJ) = Y + EPS2
```

```
              CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
              CALL RECALC(J,ZAHYO,LIST,KYORI,CUTOFF)
              THP = TERSOFF_I(I,ZAHYO,LIST,KYORI,CUTOFF)
     #              +TERSOFF_I(J,ZAHYO,LIST,KYORI,CUTOFF)
     #              +TERSOFF_LIJ(LIST_IJ,ZAHYO,LIST,KYORI,CUTOFF)

              ZAHYO(I3II) = X - EPS2
              ZAHYO(J3JJ) = Y - EPS2
              CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
              CALL RECALC(J,ZAHYO,LIST,KYORI,CUTOFF)

              THP = THP
     #              +TERSOFF_I(I,ZAHYO,LIST,KYORI,CUTOFF)
     #              +TERSOFF_I(J,ZAHYO,LIST,KYORI,CUTOFF)
     #              +TERSOFF_LIJ(LIST_IJ,ZAHYO,LIST,KYORI,CUTOFF)

              ZAHYO(I3II) = X + EPS2
              ZAHYO(J3JJ) = Y - EPS2

              CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
              CALL RECALC(J,ZAHYO,LIST,KYORI,CUTOFF)

              THP = THP
     #              -TERSOFF_I(I,ZAHYO,LIST,KYORI,CUTOFF)
     #              -TERSOFF_I(J,ZAHYO,LIST,KYORI,CUTOFF)
     #              -TERSOFF_LIJ(LIST_IJ,ZAHYO,LIST,KYORI,CUTOFF)

              ZAHYO(I3II) = X - EPS2
              ZAHYO(J3JJ) = Y + EPS2
              CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
              CALL RECALC(J,ZAHYO,LIST,KYORI,CUTOFF)

              THP = THP
     #              -TERSOFF_I(I,ZAHYO,LIST,KYORI,CUTOFF)
     #              -TERSOFF_I(J,ZAHYO,LIST,KYORI,CUTOFF)
     #              -TERSOFF_LIJ(LIST_IJ,ZAHYO,LIST,KYORI,CUTOFF)

              DATA = THP / ( 8.0D0 * EPS2 * EPS2 )

              ZAHYO(I3II) = X
              ZAHYO(J3JJ) = Y

              CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
              CALL RECALC(J,ZAHYO,LIST,KYORI,CUTOFF)

              CALL ADD_DIFF2( DATA , I3II , J3JJ , DIFF2 ,DIFF2_DATA)


 1420    CONTINUE
 1430 CONTINUE

      RETURN
c 1499 STOP
      END

 1501 SUBROUTINE HAKE_LIST_IJ( I,J,LIST_IJ,LIST )
      INCLUDE 'PARAHETER'

      INTEGER I,J
      INTEGER LIST_IJ(0:HAX_LIST_IJ)
      INTEGER LIST(0:HAX_LIST_N2,HAX_ATOH)

      INTEGER L,H

      LIST_IJ(0) = 0

      DO 1540 L = 1 , LIST(0,I)
         DO 1530 H = 1 , LIST(0,J)
            IF ( LIST(L+HAX_LIST_N,I)
     #           .NE. LIST(H+HAX_LIST_N,J)) THEN
               GOTO 1530
            ENDIF

            IF ( LIST_IJ(0) .EQ. HAX_LIST_IJ )  THEN
               WRITE(*,*) 'HAX_LIST_IJ is overflow.'
            ENDIF

            LIST_IJ(0) = LIST_IJ(0) + 1
            LIST_IJ(LIST_IJ(0)) = LIST(L+HAX_LIST_N,I)

 1530    CONTINUE
 1540 CONTINUE

      RETURN
c 1599 STOP
      END

 1601 REAL*8 FUNCTION TERSOFF_LIJ(LIST_IJ,ZAHYO,LIST,KYORI,CUTOFF)
      INCLUDE 'PARAHETER'
      INTEGER LIST_IJ(0:HAX_LIST_IJ)
      REAL*8  ZAHYO(HAX_ATOH3)
      INTEGER LIST(0:HAX_LIST_N2,HAX_ATOH)
```

```
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)
      REAL*8  TERSOFF_I

      INTEGER I,L

      TERSOFF_LIJ = 0.0D0

      DO 1610 L = 1 , LIST_IJ(0)

         I = LIST_IJ(L)

         TERSOFF_LIJ = TERSOFF_LIJ +
     #        TERSOFF_I(I,ZAHYO,LIST,KYORI,CUTOFF)

 1610  CONTINUE

      RETURN

c 1699  STOP
      END

 1701 SUBROUTINE CALC_DIFF2_3
     #     (I,J,ZAHYO,LIST,KYORI,CUTOFF,DIFF2,DIFF2_DATA)

      INCLUDE 'PARAHETER'
      INTEGER I,J
      INTEGER LIST(0:HAX_LIST_N2,HAX_ATOH)
      INTEGER LIST_IJ(0:HAX_LIST_IJ)

      REAL*8  ZAHYO(HAX_ATOH3)
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)

      INTEGER DIFF2(0:HAX_DIFF2_LIST,HAX_ATOH3)
      REAL*8  DIFF2_DATA(HAX_DIFF2_LIST,HAX_ATOH3)
      REAL*8  TERSOFF_LIJ

      INTEGER I3 , J3
      INTEGER II,JJ
      INTEGER I3II,J3JJ
      REAL*8 X,Y

      REAL*8 THP
      REAL*8 DATA

      CALL HAKE_LIST_IJ( I,J,LIST_IJ,LIST )

      I3 = I*3 - 3
      J3 = J*3 - 3

      DO 1730 II = 1 , 3
         DO 1720 JJ = 1 , 3
            I3II = I3 + II
            J3JJ = J3 + JJ

            X = ZAHYO(I3II)
            Y = ZAHYO(J3JJ)

            ZAHYO(I3II) = X + EPS2
            ZAHYO(J3JJ) = Y + EPS2
            CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
            CALL RECALC(J,ZAHYO,LIST,KYORI,CUTOFF)


            THP = TERSOFF_LIJ(LIST_IJ,ZAHYO,LIST,KYORI,CUTOFF)

            ZAHYO(I3II) = X - EPS2
            ZAHYO(J3JJ) = Y - EPS2
            CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
            CALL RECALC(J,ZAHYO,LIST,KYORI,CUTOFF)

            THP = THP
     #           +TERSOFF_LIJ(LIST_IJ,ZAHYO,LIST,KYORI,CUTOFF)

            ZAHYO(I3II) = X + EPS2
            ZAHYO(J3JJ) = Y - EPS2
            CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
            CALL RECALC(J,ZAHYO,LIST,KYORI,CUTOFF)

            THP = THP
     #           -TERSOFF_LIJ(LIST_IJ,ZAHYO,LIST,KYORI,CUTOFF)

            ZAHYO(I3II) = X - EPS2
            ZAHYO(J3JJ) = Y + EPS2
            CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
            CALL RECALC(J,ZAHYO,LIST,KYORI,CUTOFF)

            THP = THP
     #           -TERSOFF_LIJ(LIST_IJ,ZAHYO,LIST,KYORI,CUTOFF)

            DATA = THP / ( 8.0D0 * EPS2 * EPS2 )
```

```
            ZAHYO(I3II) = X
            ZAHYO(J3JJ) = Y

            CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
            CALL RECALC(J,ZAHYO,LIST,KYORI,CUTOFF)

            CALL ADD_DIFF2( DATA , I3II , J3JJ , DIFF2 ,DIFF2_DATA)

 1720    CONTINUE
 1730 CONTINUE

      RETURN
c 1799 STOP
      END

 1801 SUBROUTINE CONJGRAD(N,ZAHYO,DIFF1,DIFF2,DIFF2_DATA)
      INCLUDE 'PARAHETER'
      INTEGER N
      REAL*8 ZAHYO(HAX_ATOH3)
      REAL*8 DIFF1(HAX_ATOH3)
      INTEGER DIFF2(0:HAX_DIFF2_LIST,HAX_ATOH3)
      REAL*8  DIFF2_DATA(HAX_DIFF2_LIST,HAX_ATOH3)

      INTEGER N3

      REAL*8 P(HAX_ATOH3)
      REAL*8 R(HAX_ATOH3)
      REAL*8 AP(HAX_ATOH3)
      REAL*8 X(HAX_ATOH3)

      REAL*8 NORH_R2,NORH_R2_

      REAL*8 PAP

      REAL*8 A

      REAL*8 C

      INTEGER I,J,K
      INTEGER L

      N3 = N*3
      NORH_R2 = 0.0D0

      DO 1810 I = 1 , N3
         X(I) = 0.0D0
         R(I) = DIFF1(I)
         P(I) = DIFF1(I)
         NORH_R2 = NORH_R2 + R(I) * R(I)
 1810 CONTINUE

C      WRITE(*,*) 0,NORH_R2
      DO 1860 K = 1 , N3

         IF ( NORH_R2 .LT. 1.0D-6 ) THEN
            GOTO 1865
         ENDIF

         DO 1820 I = 1 , N3
            AP(I) = 0.0D0
            DO 1815 L = 1 , DIFF2(0,I)
               J = DIFF2(L,I)
               AP(I) = AP(I) + P(J) * DIFF2_DATA(L,I)

 1815       CONTINUE
 1820    CONTINUE

         PAP = 0.0D0
         DO 1830 I = 1 , N3
            PAP = PAP + P(I) * AP(I)
 1830    CONTINUE

         A = NORH_R2 / PAP
C         WRITE(*,*) 'A=',A

         NORH_R2_ = 0.0D0

         DO 1840 I = 1 , N3
            X(I) = X(I) + A * P(I)
            R(I) = R(I) - A * AP(I)
            NORH_R2_ = NORH_R2_ + R(I) * R(I)
 1840    CONTINUE

C         WRITE(*,*) K,NORH_R2_

         C = NORH_R2_ / NORH_R2
         IF ( C .GT. 1.0D0 ) THEN
            GOTO 1865
         ENDIF
```

```
         NORH_R2 = NORH_R2_
         DO 1850 I = 1 , N3
            P(I) = R(I) + C * P(I)
 1850    CONTINUE

 1860 CONTINUE

 1865 DO 1870 I = 1 , N3
         ZAHYO(I) = ZAHYO(I) + X(I)
 1870 CONTINUE

      RETURN
c 1899 STOP
      END

 1901 SUBROUTINE MAKE_LIST_VDW(N,ZAHYO,LIST_VDW,KYORI,CUTOFF,IDX
     #,LIST,LIST2,AN)
      INCLUDE 'PARAMETER'
      INTEGER N,N3

      REAL*8  ZAHYO(MAX_ATOM3)
      CHARACTER*8  AN(MAX_ATOM)
      INTEGER LIST_VDW(0:MAX_LIST_VN2,MAX_ATOM)
      REAL*8  KYORI(MAX_DATA_IDX)
      REAL*8  CUTOFF(MAX_DATA_IDX)
      INTEGER IDX

      INTEGER LIST(0:MAX_LIST_N2,MAX_ATOM)
      INTEGER LIST2(0:MAX_LIST2_N,MAX_ATOM)


      INTEGER I,K,J
      INTEGER E,F,G
      INTEGER L,M

      INTEGER I3

      INTEGER TX,TY,TZ
      INTEGER TI,TJ,TI3,TJ3

      REAL*8 THP_R,THP_CUTOFF

C      REAL*8 SIN

C      INTEGER IDX

      N3 = N * 3

C     リスト配列の初期化
      DO 1905 I = 1 , MAX_ATOM
         LIST_VDW(0,I) = 0
 1905  CONTINUE


C     系の座標の最大値 - 最小値を検索する


C     一度、大まかに原子をグルーピングする。
C     最大値最小値から、メッシュの個数を決める



C     全ての原子を MESH に グルーピングする。


DO 1925 TI = 1 , N
          DO 1924 TJ = 1 , N
                IF ( TI .LE. TJ ) THEN
                GOTO 1924
                 ENDIF

                       TI3 = TI*3-2
                       TJ3 = TJ*3-2
C
C     原子間距離の計算
C
      THP_R =
     #     ( ( ZAHYO(TJ3  ) - ZAHYO(TI3  ) )**2 +
     #       ( ZAHYO(TJ3+1) - ZAHYO(TI3+1) )**2 +
     #       ( ZAHYO(TJ3+2) - ZAHYO(TI3+2) )**2  )**0.5d0

C
C     範囲外のとき飛ばす。

      IF ( THP_R .GT. (PRH_VCR2 + PRH_VCD2) ) THEN
      GOTO 1924
      ENDIF

C     同一分子内は 無視
      IF ( AN(TI) .EQ.  AN(TJ) ) THEN
      GOTO 1924
```

```
      ENDIF

C WRITE(*,*) AN(TI) , AN(TJ)

C     LIST の配列があふれる時 STOP

      IF ( ( LIST_VDW(0,TI) .EQ. MAX_LIST_VN ) .OR.
     #     ( LIST_VDW(0,TJ) .EQ. MAX_LIST_VN ) ) THEN
         WRITE(*,*) 'LIST_VN is overflow: MAX_LIST_VN.'
         STOP
      ENDIF

      IDX = IDX + 1
      KYORI(IDX) = THP_R

      LIST_VDW(0,TI) = LIST_VDW(0,TI) + 1
      LIST_VDW( LIST_VDW(0,TI) ,TI ) = IDX
      LIST_VDW( LIST_VDW(0,TI) + MAX_LIST_VN ,TI ) = TJ

      LIST_VDW(0,TJ) = LIST_VDW(0,TJ) + 1
      LIST_VDW( LIST_VDW(0,TJ) ,TJ ) = IDX
      LIST_VDW( LIST_VDW(0,TJ) + MAX_LIST_VN ,TJ ) = TI

C
C     カットオフ関数を計算
C

      THP_CUTOFF = 0.0D0
      IF ( THP_R .GE. ( PRM_VCR - PRM_VCD) ) THEN
         THP_CUTOFF = 0.5D0 *
     #   ( 1.0D0 + SIN( PI * ( THP_R - PRM_VCR ) / (2.0D0 * PRM_VCD ) ))
      ENDIF
      IF ( THP_R .GE. ( PRM_VCR + PRM_VCD) ) THEN
         THP_CUTOFF = 1.0D0
      ENDIF
      IF ( THP_R .GE. ( PRM_VCR2 - PRM_VCD2) ) THEN
         THP_CUTOFF = 0.5D0 *
     #  ( 1.0D0 - SIN( PI * (THP_R - PRM_VCR2 )/(2.0D0 * PRM_VCD2 ) ))
      ENDIF
      IF ( THP_R .GE. ( PRM_VCR2 + PRM_VCD2) ) THEN
         THP_CUTOFF = 0.0D0
      ENDIF

      CUTOFF(IDX) = THP_CUTOFF

 1924  CONTINUE
 1925  CONTINUE



C      DO 1952 I = 1 , N
C         write(*,*) "--",I
C         DO 1951 J = 1 , LIST_VDW(0,I)
C            WRITE(*,*) LIST_VDW(J + MAX_LIST_VN , I)
C 1951     CONTINUE
C 1952  CONTINUE

 1969  RETURN
c 1999  STOP
      END

C
C 原子 I に関した Van Der Waals ポテンシャル
C
 2001 REAL*8 FUNCTION VDW_I(I,ZAHYO,LIST_VDW,KYORI,CUTOFF)
      INCLUDE 'PARAHETER'
      INTEGER I

      REAL*8  ZAHYO(MAX_ATOM3)
      INTEGER LIST_VDW(0:MAX_LIST_VN2,MAX_ATOM)
      REAL*8  KYORI(MAX_DATA_IDX)
      REAL*8  CUTOFF(MAX_DATA_IDX)

      REAL*8  R,U
      REAL*8 VDW_FUNC

      INTEGER L,IDXJ

      U = 0.0D0

      DO 2010 L = 1 , LIST_VDW(0,I)
         IDXJ = LIST_VDW(L,I)
         R = KYORI(IDXJ)
         U = U + CUTOFF(IDXJ) * VDW_FUNC(R)

 2010 CONTINUE

      VDW_I = U
C      write(*,*) "VDW" , U , R

      RETURN
```

```
c 2049 STOP
      END

 2050 REAL*8 FUNCTION VDW_FUNC(R)
      INCLUDE 'PARAHETER'
      REAL*8 R,SR
      REAL*8 THP_CUTOFF
      REAL*8 SIN


      SR = 3.407D0 / R


      THP_CUTOFF = 0.0D0
      IF ( R .GE. ( PRH_VCR - PRH_VCD) ) THEN

      THP_CUTOFF = 0.5D0 *
     #  ( 1.0D0 + SIN( PI * ( R - PRH_VCR ) / (2.0D0 * PRH_VCD ) ))
      ENDIF
      IF ( R .GE. ( PRH_VCR + PRH_VCD) ) THEN
         THP_CUTOFF = 1.0D0
      ENDIF
      IF ( R .GE. ( PRH_VCR2 - PRH_VCD2) ) THEN
         THP_CUTOFF = 0.5D0 *
     #   ( 1.0D0 - SIN( PI * ( R - PRH_VCR2 ) / (2.0D0 * PRH_VCD2 ) ))
      ENDIF
      IF ( R .GE. ( PRH_VCR2 + PRH_VCD2) ) THEN
         THP_CUTOFF = 0.0D0
      ENDIF


      VDW_FUNC =
     #     4.0D0 * 0.002964D0 *
     #     ( SR**12.0D0 - SR**6.0D0 )
     #     * THP_CUTOFF

      RETURN
c 2099 STOP
      END


 2101 SUBROUTINE CALC_DIFF1_VDW(N,ZAHYO,LIST_VDW,KYORI,CUTOFF,DIFF1)
      INCLUDE 'PARAHETER'
      INTEGER N,N3
      REAL*8  ZAHYO(HAX_ATOH3)
      INTEGER LIST_VDW(0:HAX_LIST_VN2,HAX_ATOH)
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)

      REAL*8  DIFF1(HAX_ATOH3)

      INTEGER I,ID3
      REAL*8  ZAHYO_ORG
      REAL*8 VDW_I

      REAL*8 THP

      N3 = N * 3

      DO 2110 I = 1 , N3

         ID3 = ( I + 2 ) / 3
         ZAHYO_ORG = ZAHYO(I)

         ZAHYO(I) = ZAHYO_ORG + EPS1
         CALL RECALC_VDW(ID3,ZAHYO,LIST_VDW,KYORI,CUTOFF)
         THP = VDW_I(ID3,ZAHYO,LIST_VDW,KYORI,CUTOFF)

         ZAHYO(I) = ZAHYO_ORG - EPS1
         CALL RECALC_VDW(ID3,ZAHYO,LIST_VDW,KYORI,CUTOFF)
         THP = THP -
     #        VDW_I(ID3,ZAHYO,LIST_VDW,KYORI,CUTOFF)

         ZAHYO(I) = ZAHYO_ORG
         CALL RECALC_VDW(ID3,ZAHYO,LIST_VDW,KYORI,CUTOFF)

         THP = THP / ( 2.0 * EPS1 )
C         WRITE(*,*) , I , THP
         DIFF1(I) = DIFF1(I) - THP
C         DIFF1(I) = - THP



 2110  CONTINUE

      RETURN
c 2199 STOP
      END
 2201 SUBROUTINE RECALC_VDW(I,ZAHYO,LIST_VDW,KYORI,CUTOFF)
      INCLUDE 'PARAHETER'
      INTEGER I,I3
      REAL*8  ZAHYO(HAX_ATOH3)
```

```
      INTEGER LIST_VDW(0:MAX_LIST_VN2,MAX_ATOM)
      REAL*8  KYORI(MAX_DATA_IDX)
      REAL*8  CUTOFF(MAX_DATA_IDX)

      INTEGER J,J3
      INTEGER IDXJ
      INTEGER M

      REAL*8 THP_R
      REAL*8 THP_CUTOFF

C      REAL*8 SIN
      REAL*8 SQRT

      I3 = I*3-2

      DO 2210 M = 1 , LIST_VDW(0,I)
         IDXJ = LIST_VDW(M,I)
         J = LIST_VDW(M+MAX_LIST_VN,I)

         J3 = J*3-2
          THP_R =
     #    SQRT    ( ( ZAHYO(J3) - ZAHYO(I3) )**2 +
     #        ( ZAHYO(J3+1) - ZAHYO(I3+1) )**2 +
     #        ( ZAHYO(J3+2) - ZAHYO(I3+2) )**2  )


         KYORI(IDXJ) = THP_R

         THP_CUTOFF = 1.0D0
C         IF ( THP_R .GE. ( PRM_CR - PRM_CD) ) THEN
C         THP_CUTOFF = 0.5D0 *
C     #   ( 1.0D0 - SIN( PI * ( THP_R - PRM_CR ) / (2.0D0 * PRM_CD ) ))
C         ENDIF
C         IF ( THP_R .GE. ( PRM_CR + PRM_CD) ) THEN
C            THP_CUTOFF = 0.0D0
C         ENDIF

         CUTOFF(IDXJ) = THP_CUTOFF

 2210 CONTINUE

      RETURN
c 2299 STOP
      END
 2301  REAL*8 FUNCTION VDW(N,ZAHYO,LIST_VDW,KYORI,CUTOFF)
      INCLUDE 'PARAMETER'
      INTEGER I
      INTEGER N
      REAL*8  ZAHYO(MAX_ATOM3)
      INTEGER LIST_VDW(0:MAX_LIST_VN2,MAX_ATOM)
      REAL*8  KYORI(MAX_DATA_IDX)
      REAL*8  CUTOFF(MAX_DATA_IDX)
      REAL*8  VDW_I

      VDW = 0.0D0

      DO 2310 I = 1 , N
         VDW = VDW +
     #        VDW_I(I,ZAHYO,LIST_VDW,KYORI,CUTOFF)

 2310  CONTINUE

      VDW = VDW / 2.0D0

      RETURN
c 2399  STOP
      END

 2401 SUBROUTINE CALC_DIFF2_VDW
     #(N,ZAHYO,LIST_VDW,KYORI,CUTOFF,DIFF2,DIFF2_DATA)
      INCLUDE 'PARAMETER'

      INTEGER N,N3
      INTEGER LIST_VDW(0:MAX_LIST_VN2,MAX_ATOM)

      REAL*8  ZAHYO(MAX_ATOM3)
      REAL*8  KYORI(MAX_DATA_IDX)
      REAL*8  CUTOFF(MAX_DATA_IDX)

      INTEGER DIFF2(0:MAX_DIFF2_LIST,MAX_ATOM3)
      REAL*8  DIFF2_DATA(MAX_DIFF2_LIST,MAX_ATOM3)

      INTEGER I,J,K
      INTEGER L

      N3 = N * 3

C      DO 2410 I = 1 , MAX_ATOM3
C         DIFF2(0,I) = 0
C 2410 CONTINUE
```

```
      DO 2450 I = 1 , N

C     隣合う原子の座標系
         DO 2445 L = 1 , LIST_VDW(0,I)
            J = LIST_VDW(L+MAX_LIST_VN,I)

            CALL CALC_DIFF2_VDW_1
     #           (I,J,ZAHYO,LIST_VDW,KYORI,CUTOFF,DIFF2,DIFF2_DATA)
 2445    CONTINUE


 2450 CONTINUE

      RETURN
c 2499 STOP
      END

 2501 SUBROUTINE CALC_DIFF2_VDW_1
     #     (I,J,ZAHYO,LIST_VDW,KYORI,CUTOFF,DIFF2,DIFF2_DATA)

      INCLUDE 'PARAMETER'
      INTEGER I,J
      INTEGER LIST_VDW(0:MAX_LIST_VN2,MAX_ATOM)

      REAL*8  ZAHYO(MAX_ATOM3)
      REAL*8  KYORI(MAX_DATA_IDX)
      REAL*8  CUTOFF(MAX_DATA_IDX)

      INTEGER DIFF2(0:MAX_DIFF2_LIST,MAX_ATOM3)
      REAL*8  DIFF2_DATA(MAX_DIFF2_LIST,MAX_ATOM3)

      REAL*8  VDW_FUNC

      INTEGER I3 , J3
      INTEGER II,JJ
      INTEGER I3II,J3JJ
      REAL*8 X,Y

      REAL*8 THP
      REAL*8 DATA

      REAL*8 R,DIST

      REAL*8 GI(3)
      REAL*8 GJ(3)

      I3 = I*3 - 3
      J3 = J*3 - 3

      GI(1) = ZAHYO(I3 + 1)
      GI(2) = ZAHYO(I3 + 2)
      GI(3) = ZAHYO(I3 + 3)

      GJ(1) = ZAHYO(J3 + 1)
      GJ(2) = ZAHYO(J3 + 2)
      GJ(3) = ZAHYO(J3 + 3)


      DIST = CALC_R(GI,GJ)

      DO 2530 II = 1 , 3
         DO 2520 JJ = 1 , 3


               X = GI(II)
               Y = GJ(JJ)

               GI(II) = X + EPS2
               GJ(JJ) = Y + EPS2
               R = CALC_R(GI,GJ)

               THP = VDW_FUNC(R)

               GI(II) = X - EPS2
               GJ(JJ) = Y - EPS2
               R = CALC_R(GI,GJ)


               THP = THP +  VDW_FUNC(R)


               GI(II) = X + EPS2
               GJ(JJ) = Y - EPS2
               R = CALC_R(GI,GJ)

               THP = THP -  VDW_FUNC(R)



               GI(II) = X - EPS2
               GJ(JJ) = Y + EPS2
               R = CALC_R(GI,GJ)
```

```
              THP = THP - VDW_FUNC(R)

              DATA = THP / ( 8.0D0 * EPS2 * EPS2 )


              IF ( DIST .LT. (PRH_CR+PRH_CD) ) THEN
             CALL ADD2_DIFF2( DATA , I3+II , J3+JJ , DIFF2 ,DIFF2_DATA)
              ELSE
             CALL ADD_DIFF2( DATA , I3+II , J3+JJ , DIFF2 ,DIFF2_DATA)
              ENDIF
              GI(II) = X
              GJ(JJ) = Y


 2520    CONTINUE
 2530 CONTINUE

      RETURN
c 2599 STOP
      END


 2601 REAL*8 FUNCTION CALC_R(GI,GJ)

      REAL*8 GI(3)
      REAL*8 GJ(3)
      REAL*8 SQRT


      CALC_R = SQRT(
     #     (GI(1) - GJ(1))**2+
     #     (GI(2) - GJ(2))**2+
     #     (GI(3) - GJ(3))**2 )



      RETURN
c 2699 STOP
      END

2701  SUBROUTINE HAKE_NEWKYORI(N,ZAHYO,LIST,KYORI,CUTOFF)
      INCLUDE 'PARAHETER'
      INTEGER N,I
      REAL*8  ZAHYO(HAX_ATOH3)
      INTEGER LIST(0:HAX_LIST_N2,HAX_ATOH)
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)

DO 2710 I = 1 , N
CALL RECALC(I,ZAHYO,LIST,KYORI,CUTOFF)
 2710  CONTINUE

      RETURN
c 2799  STOP
      END

2801  SUBROUTINE HAKE_NEWKYORIV(N,ZAHYO,LIST_VDW,KYORI,CUTOFF)
      INCLUDE 'PARAHETER'
      INTEGER N,I
      REAL*8  ZAHYO(HAX_ATOH3)
      INTEGER LIST_VDW(0:HAX_LIST_VN2,HAX_ATOH)
      REAL*8  KYORI(HAX_DATA_IDX)
      REAL*8  CUTOFF(HAX_DATA_IDX)

DO 2810 I = 1 , N
CALL RECALC_VDW(I,ZAHYO,LIST_VDW,KYORI,CUTOFF)
 2810  CONTINUE

      RETURN
c 2899  STOP
      END
```

B.3 経験ポテンシャルのパラメータファイル

ここで示すファイルは 経験ポテンシャルで用いられる係数をパラメータとして定義する物である。
fortran のプログラムの各 function や subroutine から参照される。
計算において変更する必要があるのは

```
c       プログラムで用いる炭素原子の最大個数
c
        INTEGER MAX_ATOM
        PARAMETER (MAX_ATOM = 5000)
```

もちいる分子の大きさを考慮して、 1 で 役 1.5Å の大きさと考えておくとよい。

```
c       list を作るときの メッシュの最大値
        INTEGER MAX_MESH_X
        PARAMETER (MAX_MESH_X = 25)
        INTEGER MAX_MESH_Y
        PARAMETER (MAX_MESH_Y = 25)
        INTEGER MAX_MESH_Z
        PARAMETER (MAX_MESH_Z = 50)
```

ここでメッシュの最大値をあまり大きくしすぎると計算機に過剰な負荷をかけすぎ、他のユーザにも支障を来たしてしまうので注意されたい。
目安としては、各メッシュの最大値が 100 未満の間で調整すると良い。

```
C     変数型 は STRICT
C      IHPLICIT LOGICAL ( A-Z )
IHPLICIT REAL*8 ( A-Z )


c     Tesoff ポテンシャルの
c     パラメータ ( 炭素原子 )

      REAL*8 PRH_EA
      REAL*8 PRH_EB

      REAL*8 PRH_LUHBDA1
      REAL*8 PRH_LUHBDA2
      REAL*8 PRH_LUHBDA3

      REAL*8 PRH_ALPHA
      REAL*8 PRH_DELTA
      REAL*8 PRH_ETA

      REAL*8 PRH_A
      REAL*8 PRH_C
      REAL*8 PRH_D
      REAL*8 PRH_H

      REAL*8 PRH_CR
      REAL*8 PRH_CD
```

```
      PARAHETER (PRH_EA =  1393.6D0)
      PARAHETER (PRH_EB =  -346.74D0)

C normal

C      PARAHETER (PRH_LUHBDA1 = 3.4879D0)
C      PARAHETER (PRH_LUHBDA2 = 2.2119D0)

C Graphite saigen
       PARAHETER (PRH_LUHBDA1 = 3.6012997D0)
       PARAHETER (PRH_LUHBDA2 = 2.28381399D0)

C C60 saigen

C      PARAHETER (PRH_LUHBDA1 = 3.6219D0)
C      PARAHETER (PRH_LUHBDA2 = 2.2969D0)


      PARAHETER (PRH_LUHBDA3 = 0.0D0)

      PARAHETER (PRH_ALPHA = 0.0D0)
      PARAHETER (PRH_DELTA = 0.687276D0)
      PARAHETER (PRH_ETA   = 0.72752D0)

      PARAHETER (PRH_A  =  1.5724D-7)
      PARAHETER (PRH_C  =  38049.0D0)
      PARAHETER (PRH_D  =  4.3484)
C     PARAHETER (PRH_D  =  4.384)
      PARAHETER (PRH_H  = -0.57058)

      PARAHETER (PRH_CR  = 1.95D0)
      PARAHETER (PRH_CD  = 0.15D0)


c     円周率π
      REAL*8 PI
      PARAHETER (PI  = 3.14159265358979323846D0)

c     プログラムで用いる炭素原子の最大個数
c
      INTEGER HAX_ATOH
      PARAHETER (HAX_ATOH = 5000)

c     炭素原子の最大個数の３倍
c
      INTEGER HAX_ATOH3
      PARAHETER (HAX_ATOH3 = HAX_ATOH * 3)

c     list を作るときの メッシュの最大値
      INTEGER HAX_HESH_X
      PARAHETER (HAX_HESH_X = 25)
      INTEGER HAX_HESH_Y
      PARAHETER (HAX_HESH_Y = 25)
      INTEGER HAX_HESH_Z
      PARAHETER (HAX_HESH_Z = 50)


c     一つのメッシュの最大個数
      INTEGER HAX_HESH_ATOH
      PARAHETER (HAX_HESH_ATOH = 50)

c     一つの list の最大個数
      INTEGER HAX_LIST_N
      PARAHETER (HAX_LIST_N = 20)

c     一つの list の最大個数 の２倍
      INTEGER HAX_LIST_N2
      PARAHETER (HAX_LIST_N2 = HAX_LIST_N * 2)

c     一つの list2 の最大個数
      INTEGER HAX_LIST2_N
C      PARAHETER (HAX_LIST2_N = 40)
      PARAHETER (HAX_LIST2_N = 20)



      INTEGER HAX_DIFF2_LIST
C      PARAHETER (HAX_DIFF2_LIST = 400)
      PARAHETER (HAX_DIFF2_LIST = 1)

      INTEGER HAX_LIST_IJ
      PARAHETER (HAX_LIST_IJ = 10)

c     距離 カットオフのデータ領域の大きさ
      INTEGER HAX_DATA_IDX
      PARAHETER (HAX_DATA_IDX = 500000)

c     メッシュ間隔
      INTEGER HESH_D
      PARAHETER ( HESH_D = ( PRH_CR + PRH_CD ) * 1.01D0 )
```

```
      PARAHETER (EPS1  = 0.000001D0)
      REAL*8 EPS2
      PARAHETER (EPS2  = 0.00001D0)

      REAL*8 TIHE_DIV
      PARAHETER (TIHE_DIV = 2.5D-15)
C 2.5
      REAL*8 Na
      PARAHETER (Na = 6.022045D+23)

      REAL*8 HassC
      PARAHETER (HassC = 12.0D0 * 1.6605655D-27)

      REAL*8 CONV_VELO
      PARAHETER (CONV_VELO =
     # 1.6021892D-19/HassC*TIHE_DIV*TIHE_DIV*1.0D+20)

      REAL*8 PRH_VCR
      REAL*8 PRH_VCD

      PARAHETER (PRH_VCR  = 2.1D0)
      PARAHETER (PRH_VCD  = 0.1D0)

      REAL*8 PRH_VCR2
      REAL*8 PRH_VCD2

      PARAHETER (PRH_VCR2  = 17.5D0)
      PARAHETER (PRH_VCD2  = 1.0D0)

      REAL*8 HESH_D_VDW
      PARAHETER ( HESH_D_VDW = ( PRH_VCR2 + PRH_VCD2 ) * 1.01D0 )

c     vdw の list を作るときの メッシュの最大値
      INTEGER HAX_HESH_V_X
      PARAHETER (HAX_HESH_V_X = 7)
      INTEGER HAX_HESH_V_Y
      PARAHETER (HAX_HESH_V_Y = 7)
      INTEGER HAX_HESH_V_Z
      PARAHETER (HAX_HESH_V_Z = 20)

c     vdw 一つのメッシュの最大個数
      INTEGER HAX_HESH_V_ATOH
      PARAHETER (HAX_HESH_V_ATOH = 2100)

c     一つの list の最大個数
      INTEGER HAX_LIST_VN
      PARAHETER (HAX_LIST_VN = 2100)

c     一つの list の最大個数 の２倍
      INTEGER HAX_LIST_VN2
      PARAHETER (HAX_LIST_VN2 = HAX_LIST_VN * 2)
```

参考文献

[1] H. Hamada, S. Sawada, and A. Oshiyama

[2] J. M. Mintmire, B. I. Dunlap, and C. T. White, Phys. Rev. Lett. **68**, 631 (1992).

[3] R. Saito, M. fujita, G. Dresselhause, and M. S. Dresselhause, Phys. Rev. B **46**, 1804 (1992).

[4] Susumu Okada, Electronic and Geometric Structure of Fullerlens and Polymerized C_{60}, (Nov 1997). 東京工業大学博士論文.

[5] 竹谷 隆男, "カーボンナノチューブの構造", 電子工学専攻 1998 年度 修士論文.

[6] S. Iijima, Nature **354**, 56 (1991)

[7] B. W. Smith, M. Monthoux, D. E. Luzzi, Nature **396** (1998) 323.

[8] A. G. Rinzler, J. Liu, H. Dai, et. al, Appl. Phys. A **67** (1998) 29.

[9] C. Journet, W. K. Maser, P. Bernier, et. al, Nature **388** (1997) 756.

[10] Jelemy Sloan, Rafal E. Dunin-Borkowski, John L. Hutchison, et al, Chemical Physics Letters **316** (2000) 191.

[11] 松尾 竜馬, "カーボンナノチューブの面間相互作用に関する研究", 電子工学専攻 1999 年度 修士論文.

[12] 齋藤 弥八、坂東 俊治 共著 "カーボンナノチューブの基礎", コロナ社 (1998) 23.

[13] R. E. Smalley, et al, Science **273** (1996) 483.

[14] S. Bandow, M. Takizawa, et al, "Raman scattering study of double-wall carbon-nanotubes derived from the chains of fullerens in single-wall carbon nanotubes"