1998 年度 修士論文

ドナー及び、アクセプター型ドープ微小黒鉛クラスターの電子状態

電気通信大学 大学院 電気通信学研究科 電子工学専攻

9730052　八木 将志

指導教官 齋藤 理一郎 助教授

木村 忠正 教授

提出日 平成 11 年 2 月 3 日

目次

第 1 章

序論

本章では、まず本研究で扱った黒鉛についての構造、特徴等を示し、次に、背景となるドナー及び、アクセプター型ドープの実験結果を述べる。そして、本研究の目的を述べる。

1.1　グラファイト

グラファイトとは、炭素原子が六角形の 2 次元の格子を作っているものであり、この格子のことを六方格子という。この格子がグラファイト層をつくり、層が積み重なって黒鉛をつくっている。また、六角形の一辺の長さは 1.42 Å、層間の距離は 3.35 Å である。

そして、グラファイトの層間に不純物をドープし、新たな材料として用いられている。例えば、Li をドープしたグラファイトを 2 次電池の電極に用いた物は製品化されている。

近年、特に直径約数十Å のグラファイトが注目されてきている。このグラファイトの特徴は終端部分の占める割合が非常に多いということである。このことによると思われる興味深い実験結果が数多く報告されている。

また、本研究室でも過去に中平らによって Li ドープグラファイトの研究 [1][2] が行われており、それらを参照した例を以下に示す。

1.1.1　グラファイト層間化合物

層状物質であるグラファイトに不純物をドープした場合、層間に異種物質が侵入し、新しい化合物を生成する。このことをインターカレーション (intercalation) といい、

それによって出来る化合物を黒鉛層間化合物 (Graphite Intercalation Compounds:GIC) と呼ぶ [4]。また、 GIC では、層間に挿入される物質が数枚のグラファイト層を隔て、規則正しい積層構造をとる。これをステージの存在と呼び、グラファイト層 n 枚ごとに挿入物質があるとき、第 n ステージ GIC と呼ぶ。ステージ数の異なる化合物は生成反応条件の制御によって生成され、それぞれ異なる物性を持つ。

1.1.2 微小黒鉛

本論文では、直径数十Åm のグラファイトを微小黒鉛とよび、研究を行った。作製方法は、ポリパラフェニレンや、ポリアセンのような有機物を熱処理である。そして、1000 〜 3000 $m^2 \cdot g^{-1}$ の極めて大きい比表面積を持つという特徴がある。特に、グラファイトの端の割合が多いというのが注目されている。

代表的なものとして、 1.4.1に示すピッチと呼ばれる炭素水素化物等がある。

1.2 Li ドープナノグラファイト

グラファイトの層間に最も多く Li が入っているときのドープ位置は、図 1.1のように、 $\sqrt{3} \times \sqrt{3}$ 構造と呼ばれる位置関係を持って存在する。第 1 ステージ GIC の場合で、組成比は Li:C=1:6 になることが良く知られている。

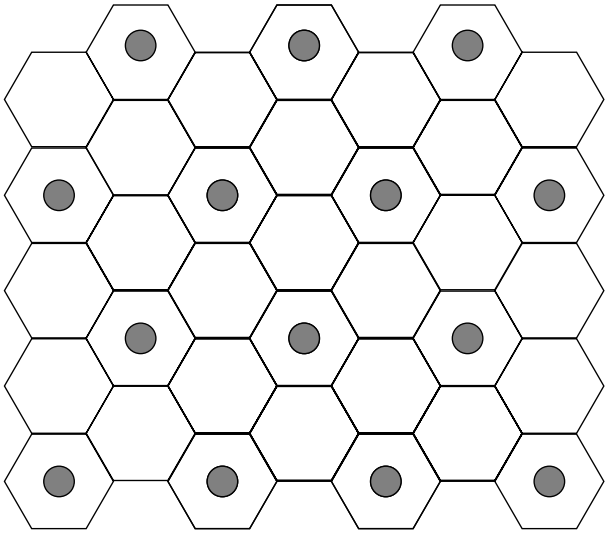

図 1.1: 第 1 ステージ Li GIC の面内構造[1]

そこで以下に、 Li ドープの実験結果、過去の研究成果について述べ、研究課題などを述べる。

[1] Li-GIC.eps

1.2.1 2 次電池への応用

最近では、実際に陽極に $LiCoO_2$ を用い、陰極にグラファイトを用いた 2 次電池が製品化されている [5]。この電池では、陽極で式 (1.2.2) のような反応が進行し、陰極では式 (1.2.3) のような反応が進行する。全体では式 (1.2.3) のような反応が進行する。

$$\mathbf{LiCoO_2} \iff \mathbf{Li_{1-x}CoO_2} + x\mathbf{Li^+} + x\mathbf{e^-} \tag{1.2.1}$$

$$6\mathbf{C} + x\mathbf{Li^+} + x\mathbf{e^-} \iff \mathbf{Li}_x\mathbf{C}_6 \tag{1.2.2}$$

$$\mathbf{LiCoO_2} + 6\mathbf{C} \iff \mathbf{Li_{1-x}CoO_2} + \mathbf{LiC}_6 \tag{1.2.3}$$

充電するときは陰極のグラファイトの層間に Li が入り、放電するときは、グラファイト層間の Li が抜け出し、陽極の $LiCoO_2$ の層間に入り込む。

しかし、陰極に GIC を用いた電池では理論上第 1 ステージ GIC での放電容量以上は望めないので、さらに高容量化するために、様々な形態の炭素について研究がなされている。

1.2.2 GIC 以外の陰極材料

最近、ポリパラフェニレン (PPP) やポリアセン (PAS) のような有機物を熱処理し、グラファイト微結晶が集まったような形状を持ったもの (図 1.2参照) を陰極に使用した場合、リチウムと炭素の組成比が Li:C=1:2 になり、第 1 ステージ GIC(Li:C=1:6) よりも 3 倍もの Li をドープすることができ、グラファイト以上の放電容量を示すという報告がなされた [6]-[8]。

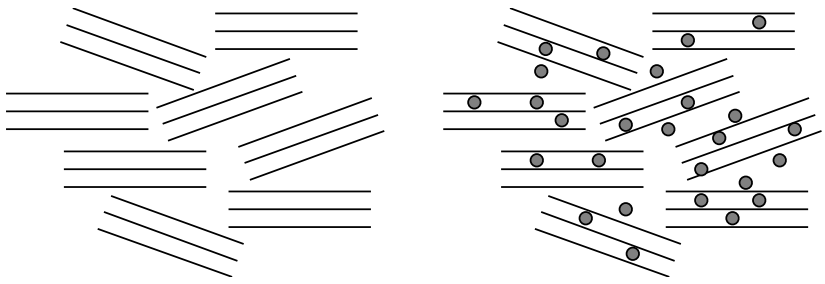

図 1.2: グラファイト微結晶[2] Li ドープグラファイト微結晶 摸式図[3]

また、遠藤らは様々な炭素材料について放電容量を測定し、グラファイトの結晶子の厚さ L_{c002} との関連性を示した [9](図 1.3参照)。図 1.3を見ると結晶性が未発達な低

[2] /home9/students/yagi/tex/m98yagi/eps/ppp-model.eps, 以下 directory は全て同じ

[3] Li-dope-ppp.eps

結晶性炭素材料 (ZoneIII) と結晶性が発達している高結晶性炭素材料 (ZoneI) の放電容量が大きく、その中間的な結晶性をもつ炭素材料 (ZoneII) の放電容量が小さくなっており、全体として U 字型になっている。L_{c002} が小さい低結晶性炭素材料では Li のドープ反応であり、結晶性が低いほど Li がドープしやすくなる。ここでドープ反応とは、グラファイトと Li が共有結合又は、イオン結合する反応である。そして徐々に結晶性が高くなり、L_{c002} =100Å 程度まではドープ量が徐々に低下する。一方、L_{c002} が大きい高結晶性炭素材料では Li のインターカレーション反応であり、結晶性が高い (つまり黒鉛化度が高い) ほど理論容量の 372mAh/g の LiC_6 に近い組成まで充電が可能であると考えられる。これらに対し中間的な結晶性をもつ炭素材料ではドープ反応、インターカレーション反応が起きづらく容量の低下が起こると考えられる。また、低結晶性炭素材料の中でも特に、PPP700(700°C で熱処理を施した PPP) の放電容量が大きく、約 680mAh/g でグラファイトの理論的な容量の約 2 倍もの値を示している。

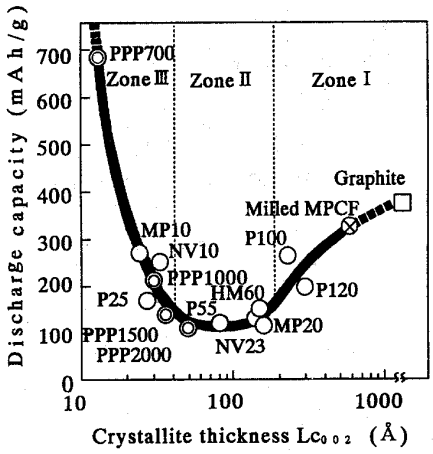

図 1.3: 放電容量とグラファイトの結晶サイズ依存性 [9] [4]

このように Li が過剰にドープされる原因を検証するために、幾つかの実験がなされており、佐藤らは ^{7}Li NMR のナイトシフトの実験 (図 1.5) から、PPP 焼成体にドープされた Li に、イオン化している状態と、共有結合している状態の 2 つの状態が存在していることを発見し、このことが Li が過剰にドープされる要因であると考え、そのドープモデルを考案している [6](図 1.6)。このモデルとは、図の白丸のイオン結合してる Li の周りに、黒丸の共有結合した Li が吸着されるというものである。また、共有結合の ^{7}Li NMR シグナル (図 1.5、A) は充電の始めから現れ、イオン結合の ^{7}Li

[4] size-effect.eps

NMR シグナル (図 1.5、B) は充電の途中から現れるので、佐藤らは以下のように考えた。充電するとき、まず最初に 共有結合サイトの Li が入り、その次にイオン化サイトの Li が入る。また、入る場所は図 1.6のように、GIC の $\sqrt{3} \times \sqrt{3}$ 構造のサイトにイオン化したものが入り、その周りを取り囲む様に Li_2 分子が入り込んでいると提案した。また、充放電特性は (図 1.4参照) 左右対称になっていないことは、充電と放電の機構が異なっていることを示している。

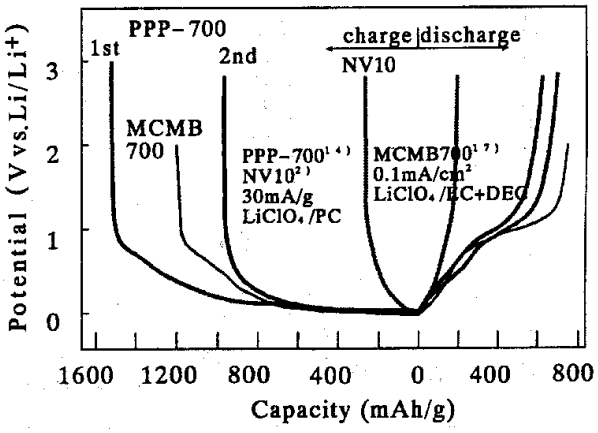

図 1.4: Li イオン 2 次電池の充放電特性 [9][5]

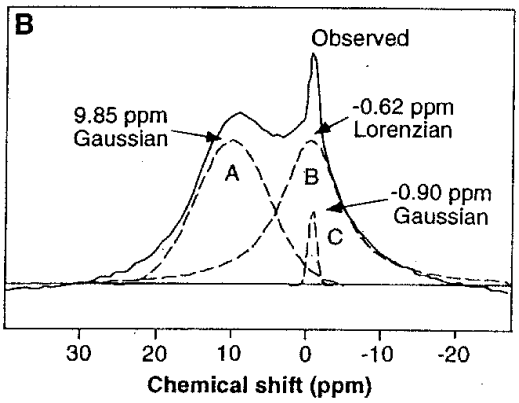

図 1.5: ^{7}LiNMR スペクトル [6][6]

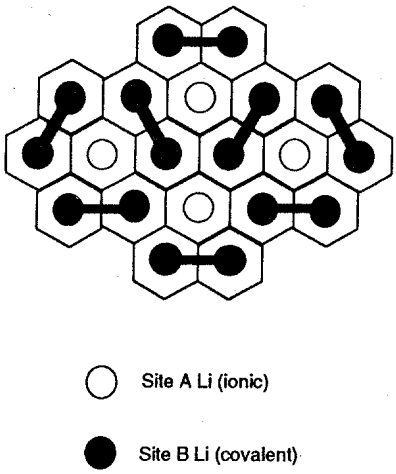

図 1.6: PPP への Li ドープモデル [6][7]

しかし、このモデル (図 1.6) は、Li　同士の反発が考えられ、現実的ではない。

[5] discharge.eps

[6] nmrsp.eps

[7] ppp-Li-p.eps

1.2.3　過去の研究経過

リチウムドープの研究は我々の研究グループで過去に中平によって行なわれており、その結果を図 1.7に示す。

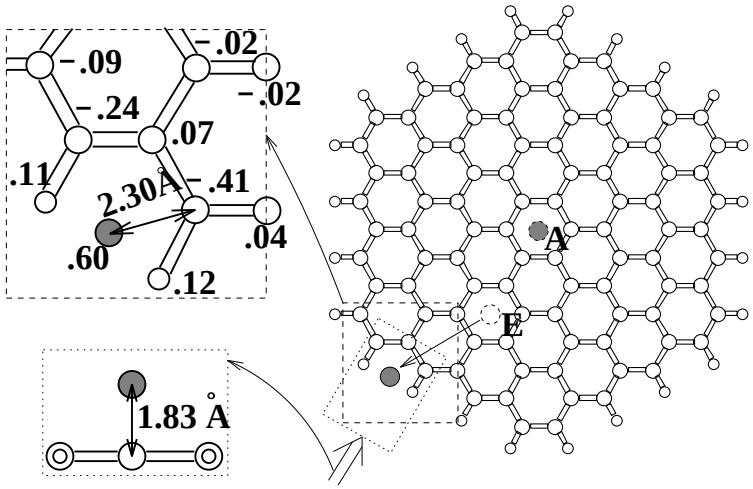

図 1.7: $C_{96}H_{24}$ のエッジ部分の Li の吸着サイト [1] [8]
(修士論文「グラファイトクラスターの Li 過剰吸着とラマン強度」中平政男、 1996 年度 電通大 電子工学専攻 修士論文より引用)

結論によると、グラファイトの上と端に Li が吸着でき、水素終端された端にイオン結合できることによって、大量ドープが可能であると報告した。

しかし、 Li を大量ドープしたグラファイトの構造最適化及び、電子状態の計算はしていない。

1.3　F ドープナノグラファイト

フッ素グラファイト層間化合物は 100℃以下においてフッ化水素や金属フッ化物の存在下でフッ素ガスと反応させることによって生成する。この化合物は炭素ーフッ素間の結合は電荷移動を含むイオン性の結合から半イオン結合 (部分共有結合) 的なものにわたると考えられる。

それに対して、共有結合性のフッ化グラファイトも存在する。フッ化グラファイトはグラファイトとフッ素との直接反応によって生成する層間化合物で、その化学式は $(CF_x)_n$ で表され、 x の値は $0 < x < 1.25$ の範囲の不定比組成をとりやすい。

[8] m96naka.eps

以下に、東工大の榎教授らによって行なわれたフッ化グラファイトの作成の実験と、物性測定について高井らの修士論文 [3] 等を参考にして要点のみ説明する。

1.3.1　フッ化グラファイトの作成

高井らの実験では、直径約 30 Å のグラファイトを用い、合成前に 150℃または 260 ℃で加熱真空脱気を行なった後、合成を行う。反応条件はフッ素圧力 1 atom 、反応時間 1 週間 で 100℃、 150℃、 200℃の各温度で行っている。また、より浅いフッ素化として、フッ素圧力 0.1　気圧、反応時間 30 分 、室温及び、フッ素圧力 0.1 気圧 、反応時間 5 日 、室温で反応させたものも作成する。また、全くフッ化させていない試料も作成している。

この時のグラファイトの基本構造は、図 1.8のような微小グラファイトが 3 枚ほど積層したものとみなしている。

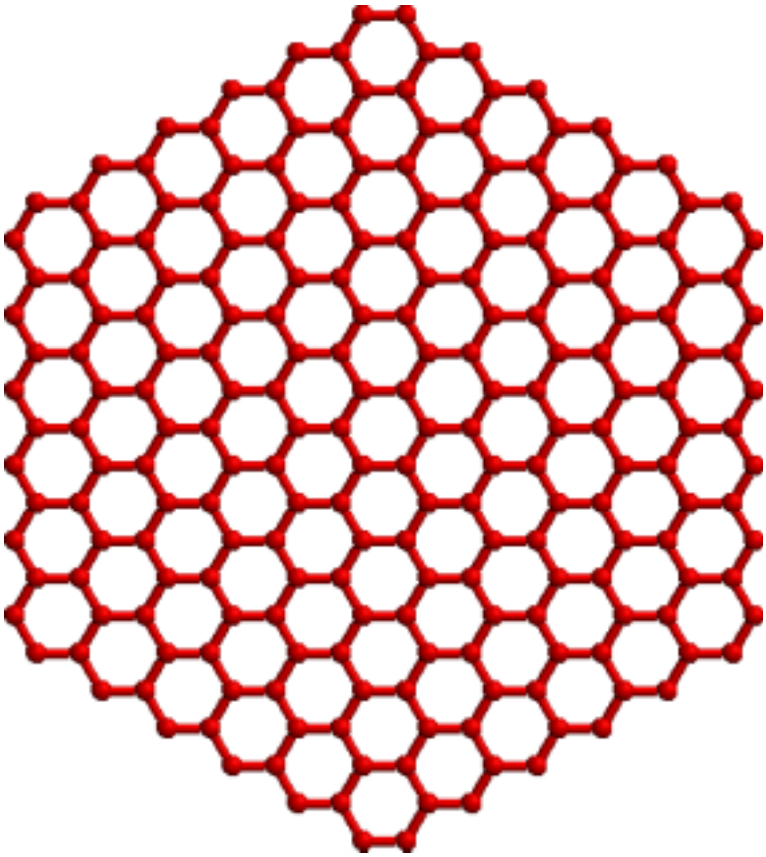

図 1.8: 実験に用いたグラファイトのモデル図 [9]

このグラファイトは炭素原子数 216 個の分子で、直径約 25Å である。ベンゼン環の

[9] A-C216.eps

数は 91 個である。ここで、いくつのベンゼン環の構成原子になっているかで、炭素原子を区分してみる。

(a) 1 個の六員環の構成原子。終端部分に存在

(b) 2 個の六員環の構成原子。終端部周辺に存在

(c) 3 個の六員環の構成原子。内部に存在

と区分すると、(a) は 36 原子、(b) は 30 原子、(c) は 150 原子である。

ドープの結果、試料の炭素フッ素濃度比は 0 〜 1.2 となった。黒鉛結晶でのフッ化グラファイトの組成比の理論値 1 であるが、この微少グラファイトを用いた結果ではこの値を超えている。これは、終端部である、区分 (c) の炭素原子に CF_2 の結合が生じているためと考えられる。そして、1.2 という組成比は、この実験のモデルとして考えられた図 1.8のグラファイトにおける理論上の組成比 1.17 とよく一致している。

また、この理論値とのわずかな差は、グラファイトの形状には多くの種類があり、図**??**の形が周辺部を少なく見積もった形状であったり、これより小さいグラファイトの存在等が考えられる。しかし、炭素フッ素濃度比は現実としてはほぼ誤差の範囲であり、グラファイトの大きさはこのモデルでほぼ正しいといえる。

1.3.2　フッ化グラファイトの測定

この試料に対して、スピン密度を測定したところ炭素フッ素濃度比の変化によって大きく変化することがを高井らは示した。そのグラフを図 1.9に示す。

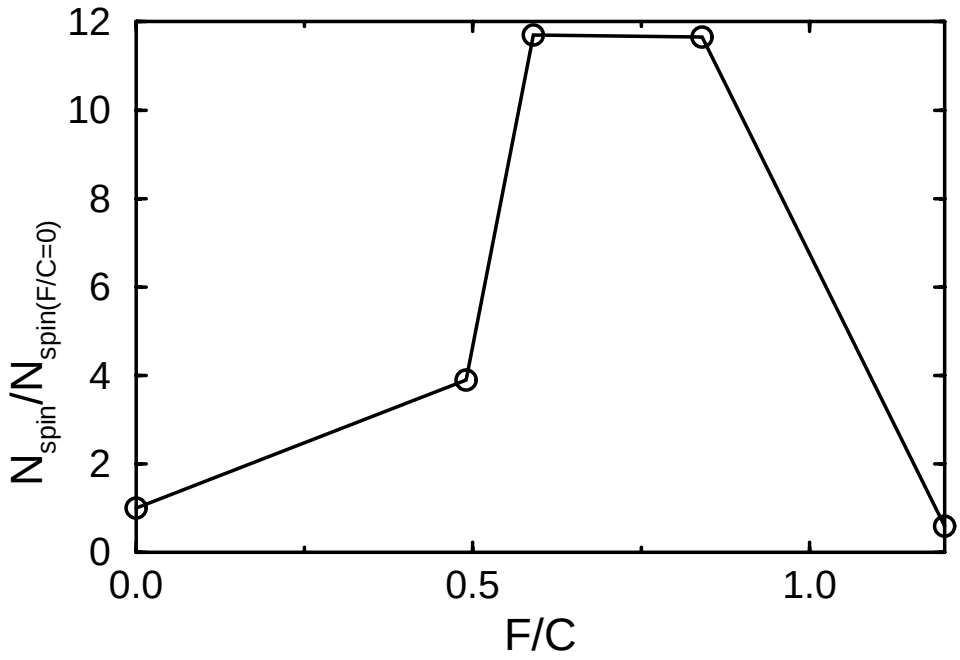

図 1.9: フッ化グラファイトのスピン密度変化 [10]

横軸は炭素フッ素濃度比、縦軸はノンドープ時を 1 としたスピン濃度の相対量を示している。この図より、スピン密度が炭素フッ素濃度比約 0.5 〜 0.9 でピークを持ち、最大ドープされたとき、最小値をとることがわかる。また、静磁化率等の測定結果に興味深い結果が現われており、今後、フッ化グラファイトの現象等が明らかになれば、磁化をもった微小材料に応用されることが期待されている。

[10] Fspin.eps

1.4 Iドープナノグラファイト

ピッチと呼ばれる炭素水素化物にヨウ素をドープすることによって低音 (100 °C) でグラファイト化が進むということを東工大の安田教授、田邊助教授グループによって報告された。この実験内容について安田教授の院生である宮島氏による実験結果の要点を説明する。

1.4.1 ピッチ

ピッチとは、石油、石炭、木材などの有機物質の乾留によって得られるタールを蒸留したときの釜残油の総称で、コールタールピッチ、石油ピッチ、木タールピッチなどがある。

普通は軟化温度 60 〜 75℃であるが、アントラセン油の蒸留終点を低くおさえるか、高くするかによって、軟質ピッチ (軟化温度 50 〜 60 ℃)、硬質ピッチ (軟化温度 90 ℃以上)、が得られる。また、乾留によって生じるピッチコークス (得量約 60%) は灰分 (無機分) を含まず、良質の電極材料となる。ピッチを水蒸気分留して約 15 % を留出することができ、これからピレン、クリセンなどを採取することができる。乾留で留出する油 (ピッチ油) もそれらを含んでいる。

ピッチの用途は、練炭や電極の粘結剤、防水材料、鉄材、木材などの防水、防さび、防腐などのための塗料、また、上記の残油を配合して舗装タールや防水塗料に使われている。また、石油ピッチは、アスファルトより高温で石油を分解したときに残る粘性のない炭化物であり、不純物を多く含むので良質の化学原料とはならず、燃料や電気絶縁材料などに用いられる。基本的にピッチは、良質の原料としては用いることのできないものである [10]。

1.4.2 グラファイト化

グラファイト化とは、微少なグラファイトが、融点である 3500 °C よりも低い温度で、より大きなグラファイトへ成長していくことである。この反応は、ピッチに 650 〜 700°C の高温を与えることでも可能である。つまり、 650 〜 700°C の温度領域では水素が黒鉛から離脱しダングリングボンドが生成する。このダングリングボンド同士が結合することでより大きなグラファイトに成長可能である。また、 2000 °C を超える温度では C−C 間の結合が離れることによって直接グラファイトの結晶が成長することが知られている。

しかし、田邊らの実験では、ヨウ素処理が 100 °C 程度の温度で行い、その後 300 °C ぐらいの処理過程をでもグラファイト化が進むことを見出した。つまり、このような低温でもヨウ素処理を行うことによって、水素を黒鉛から離脱させることが可能である。この機構として安田らはIと黒鉛の電荷移動錯体と提案しているがその機構は明確ではない。

そこで、低温でのヨウ素の結合、及び水素、ヨウ素の分離が考えられるが理論的計算によって、この反応の解析が望まれている。

1.5 研究目的

ここに、本研究の目的を示す。

本研究では、グラファイトにリチウム、フッ素、ヨウ素のそれぞれをドープした実験結果に対して、理論的な検証を行ない、反応等の様子を探り、新たな反応機構の提案をすることを目的としている。

次に、それぞれのドープ原子についての細かい目的を示す。

1.5.1 リチウムドープ

電池に応用されているリチウムドープグラファイトのドープ量は、大量にドープされている。しかし、大量ドープの機構はまだ明らかにされていない。以前の、中平による研究でも大量ドープという観点では行なっておらず、本研究では大量ドープ時の Li の吸着位置、電荷について重点的に研究することを目的とする。

1.5.2 フッ素ドープ

フッ素ドープグラファイトの実験結果では、スピン密度に特長のある結果を得ている。しかし、今回用いた計算方法である MOPAC93 の非制限ハートリーフォック法では、スピン密度に関して信用のおける結果を得ることができない事が知られている。

そのため、本研究ではドープ時にの構造最適化の計算より、反応時の様子を探り、フッ化グラファイトの特徴、及びフッ素原子の微小グラファイトの電子状態での影響等について計算する。

1.5.3 ヨウ素ドープ

ヨウ素の反応は動的反応座標 (DRC) の計算方法を用い、ヨウ素ドープによるグラファイト化のメカニズムを解明する。また、この実験ではグラファイトの温度が重要になっている。そのため、反応を起こす温度を MOPAC の DRC 計算において設定するプログラムを開発、作成し、それを用いて化学反応解析を行うことを目的とする。

第 2 章

計算方法

本章では、計算方法及び計算モデルについて述べる。クラスターモデルの電子状態、構造最適化及び基準振動モードの計算は半経験的分子軌道法プログラム (MOPAC93) を用いた。

2.1 MOPAC93

MOPAC93 は半経験的分子軌道法プログラムである。半経験的分子軌道法プログラムとは、幾つかの積分項を経験的なパラメーターを用いるため、全ての積分項について基底関数を用いて計算を行なう $ab\ initio$ よりもはるかに計算が速い、という特徴を持っている。従って、本研究のように原子数の多い系、又、クラスターのカンプル数の多い系の特徴を把握する計算を行なうのに適している。以下では、 MOPAC93 の概要を述べる。 [12][13]

2.1.1 MOPAC の概要

MOPAC93 では分子の全電子エネルギー、分子軌道等が計算できる。最適化構造計算では、原子間ポテンシャルとして 4 つの計算方法、 MINDO/3(Modified Intermediate Neglect of Differential Overlap)、 MNDO(Modified Neglect of Diatomic Overlap)、 AM1(Austin Model 1)、 PM3 (Parametric Method 3) 法があり、それぞれに特徴があり、その一つを選ぶことができる。又、それぞれの原子間ポテンシャル、用いているパラメータやハミルトニアンが異なっており、使用できる原子にも違いがある。本研究では、この MOPAC93 の 最もよい精度で、構造最適化、生成エネルギーが求まる様に、多くの分子のデータを再現する様に最小 2 乗法でパラメータを決めて

いる PM3 法を用いて電子状態、基準振動、及び構造最適化の計算を行なった。

以下では、 MOPAC の計算原理を簡単に述べる。

MOPAC は、次のような Hartree-Fock-Roothaan 方程式を SCF(Self-Consistent-Field) 計算により求める。

$$\mathbf{FC}_i = \varepsilon_i \mathbf{SC}_i \tag{2.1.1}$$

ここで $\mathbf{C}_i$ は固有ベクトルで、分子軌道 φ_i を原子軌道 χ_q の線形結合で表した (LCAO 近似; Linear Combination of atomic orbitals) 式

$$\varphi_i = \sum_{q=1}^{n} \chi_q C_{qi} \tag{2.1.2}$$

の行列 C_{qi} の列ベクトルである。また、 ε_i は固有値、 $\mathbf{S}$ は重なり積分で

$$S_{pq} = < \chi_p | \chi_q > \tag{2.1.3}$$

で表される行列である。 $\mathbf{F}$ はフォック行列で 一電子部分、 $\mathbf{H}$、 二電子部分、 $\mathbf{P}$ の和、

$$\mathbf{F} = \mathbf{H} + \mathbf{P} \tag{2.1.4}$$

で表せる。ここで $\mathbf{H}$、 $\mathbf{P}$ はそれぞれ、

$$\begin{aligned} H_{pq} &= < \chi_p | \hat{h} | \chi_q > \\ P_{pq} &= \sum_{r,s} T_{pq,rs} \cdot D_{rs} \\ D_{rs} &= \sum_{j=1}^{n} 2 C_{rj} C_{sj} \end{aligned}$$

$$T_{pq,rs} = (\chi_p \chi_q | \chi_r \chi_s) - \frac{1}{2} (\chi_p \chi_s | \chi_r \chi_q)$$

である。ここで 2 電子積分項 $(\chi_p \chi_s | \chi_r \chi_q)$ は

$$(\chi_p \chi_s | \chi_r \chi_q) = \iint \chi_p(\mathbf{r}) \chi_s(\mathbf{r}) \frac{1}{(\mathbf{r}' - \mathbf{r})} \chi_r(\mathbf{r}') \chi_q(\mathbf{r}') d\mathbf{r} d\mathbf{r}'$$

以上の様な Hartree-Fock-Roothaan 方程式を SCF 計算によって解く。ここで、 SCF の計算とは、まず適当に固有ベクトル $\mathbf{C}$ を仮定して $\mathbf{F}$ 行列を作り、これを対角化して Hartree-Fock-Roothaan 方程式を解く。これで、固有値 ε_i と固有ベクトル $\mathbf{C}$ が求まる。ここで求まった固有ベクトルと先に D_{rs} で仮定したものが、許容範囲で一致す

るならばSCF は収束したことになり計算を終える。もし、許容範囲外ならば求めた固有ベクトルを使って、いまの行程を繰り返す。

半経験的分子軌道法とは、フォック行列の中の幾つかの積分項をあらかじめ与えられた原子間距離の関数としての原子毎のパラメーターに置き換えたものである。

さらに、?? で述べるように、重要な積分以外は一切無視するという近似を行っている。これに対する補正は、経験的なパラメータの中に含まれていると考えることがこの計算の考え方である。

2.1.2 PM3 法

次に、本研究で用いたMOPAC のPM3 法について述べる。又、 MNDO 法との比較も説明する。 PM3 法、 MNDO 法は、どちらもNDDO 近似 (Neglect of Diatomic Differential Overlap) を用いている。これはある電子についての 2 原子間にわたる微分重なりの値をすべて 0 とする近似である。この様な近似を行なうと、近似によって 0 とされた積分項の他に、被積分関数の積の持つ空間対称性のため、幾つかの 2 電子積分項が 0 となる。これにより計算する量を大幅に減らすことができ、計算時間を短縮することができる。

以下に、この様な近似を使ったPM 3 法と MNDO 法に共通なフォック行列を記す。但し、原子を A、 B などと表し、原子軌道を μ、 ν、 λ、 σ などと表記する。また、内殻電子と原子核を含めてコア (Core) と呼ぶ。

PM3 法、 MNDO 法に共通なフォック行列は、

$$
\begin{aligned}
F_{\mu\mu} \quad & \underline{U_{\mu\mu} + \sum_{B} V_{\mu\mu,B}} + \sum_{\nu}^{A} P_{\nu\nu} \left[(\mu\mu|\nu\nu) - \frac{1}{2}(\mu\nu|\mu\nu) \right] + \sum_{B} \sum_{\lambda,\sigma}^{B} P_{\lambda\sigma}(\mu\mu|\lambda\sigma) \\
F_{\mu\nu} \quad & \underline{\sum_{B} V_{\mu\nu,B}} + \frac{1}{2} P_{\mu\nu} \left[3(\mu\nu|\mu\nu) - (\mu\mu|\nu\nu) \right] + \sum_{B} \sum_{\lambda,\sigma}^{B} P_{\lambda\sigma}(\mu\nu|\lambda\sigma) \\
F_{\mu\lambda} \quad & \underline{\beta_{\mu\lambda}} - \sum_{\lambda}^{A} \sum_{\sigma}^{B} P_{\nu\sigma}(\mu\nu|\lambda\sigma)
\end{aligned}
\qquad (2.1.5)
$$

ただし、

$U_{\mu\mu}$: 1 中心電子コアエネルギー (A 原子上の原子軌道の積 $\chi_\mu\chi_\mu$ で記述される 1 個の電子の運動エネルギー、及び原子 A のコアとの吸引に基づくポテンシャルエネルギーの和)

$(\mu\mu|\nu\nu)$：1 中心 2 電子反発積分 (クーロン積分) $(= g_{\mu\nu})$

$(\mu\nu|\mu\nu)$：1 中心交換積分 $(= h_{\mu\nu})$

$V_{\mu\nu,B}$ ： 2 中心 1 電子吸引エネルギー (原子 A 上の原子軌道の積 $\chi_\mu\chi_\nu$ で表される電子と原子 B とのコアとの静電気に基づくポテンシャルエネルギー)

$\beta_{\mu\nu}$：2 中心 1 電子コア共鳴積分 (各原子軌道に固有な結合パラメーター (bounding parameter)β_μ,β_λ から次式で計算する)

$$\beta_{\mu\lambda} = \frac{1}{2} S_{\mu\lambda}(\beta_\mu^A + \beta_\lambda^B) \tag{2.1.6}$$

$(\mu\nu|\lambda\sigma)$：2 中心 2 電子反発積分

$P_{\lambda\sigma}$：結合次数 $(= 2\sum_i^{occ} C_{\lambda i}C_{\sigma i})$

である。$U_{\mu\mu}$,$G_{\mu\nu}$,$h_{\mu\nu}$,β_μ,β_λ などは原子ごとに定めたパラメーターである。計算したい分子の構造と原子種に応じて $F_{\mu\nu}$ が決まると、**F** の対角化を繰り返し、SCF(Self-Consistent Field) となったところで固有値 ε_D と固有ベクトル **C** が求まる。全電子エネルギー E_{el} は、

$$E_{el} = \frac{1}{2}\sum_\mu\sum_\nu P_{\mu\nu}(H_{\mu\nu} + F_{\mu\nu}) \tag{2.1.7}$$

として求まる。ここに $H_{\mu\nu}$ はコアハミルトニアンと呼ばれ、$F_{\mu\nu}$ の 1 電子部分 ((2.1.5) 式の下線つきの部分：” 電子の運動エネルギー” と” 電子と殻のポテンシャルエネルギー” の和) のことである。

分子の全エネルギー E_{tot} は、E_{el} にコア − コア間の反発エネルギー E_{AB}^{core} の総和を加えて、

$$E_{tot} = E_{el} + \sum_{A<B} E_{AB}^{core} \tag{2.1.8}$$

となる。

コア A と B の間の静電反発エネルギー項 E_{AB}^{core} の計算式が、PM3 法と MNDO 法とでは異なり以下のように与えられる。

<MNDO 法 >

$$E_{AB}^{core} = Z_A Z_B(s^A s^A|s^B s^B)\{1 + f_{AB}e^{-\alpha_A R_{AB}} + e^{-\alpha_B R_{AB}}\} \tag{2.1.9}$$

<PM3 法 >

$$
\begin{aligned}
E_{AB}^{core} &= Z_A Z_B (s^A s^A | s^B s^B)\{1 + f_{AB} e^{-\alpha_A R_{AB}} + e^{-\alpha_B R_{AB}}\} \\
&\quad + \frac{Z_A Z_B}{R_{AB}} \Bigg\{ \sum_{k=1}^{2} a_{kA} \exp[-b_{kA}(R_{AB} - c_{kA})^2] \\
&\quad + \sum_{k=1}^{2} a_{kB} \exp[-b_{kB}(R_{AB} - c_{kB})^2] \Bigg\}
\end{aligned}
\tag{2.1.10}
$$

但し、

Z_A : コア A の有効電荷

s^A : 原子 A の s 軌道

$Z_A Z_B (s^A s^A | s^B s^B)$ は、それぞれ s 軌道の大きさに広がっている電荷球として近似した場合のコア A とコア B の間の静電反発エネルギーを表す。

f_{AB} : A 原子が N か O で B 原子が H の場合は R_{AB}、その他の場合は 1

R_{AB} : コア A とコア B との間の距離

$\alpha_A, \alpha_B, a_{kA}, b_{kA}, c_{kA}$: 原子ごとに決めたパラメーターである

MNDO 法の式 (2.1.9) は、経験式である。式 (2.1.9)~ 式 (2.1.10) を比較すると、 MNDO 法と PM3 法との違いは、式 (2.1.10) の下線つきの部分があるかないかだけの違いである。下線の部分は、 MNDO 法では van der Waals 距離付近の原子間反発エネルギーを過大評価しているとされているので、これを補正するための項である。ただし、この補正をしてもグラファイトの層間距離である 3.35 Å は再現できない。

また、 MNDO 法と PM3 法ではパラメーターの決め方が異なっており、 MNDO 法が 32 個の分子の各諸量の実験値を再現するようにパラメーターが決められているのに対し、 PM3 法は 763 個の分子の生成エネルギーの実験値とのずれが最小になるようにパラメーターが最適化されている。従って PM3 法は MNDO 法と比べ精度が良く、多くの分子を再現することが知られている。ここで精度とは、結合距離でおよそ 0.01 Å 程度である。

以上のように、 MOPAC ではフォック行列や原子間反発エネルギーを求める式の中に、幾つかのパラメーターを使用している点で半経験的な計算である。

また、MOPAC の分子軌道計算では、扱う軌道は最外殻の原子軌道だけであり、残りはコアに含める。本研究では炭素原子とリチウム原子、及び水素原子であるので、扱う軌道は $1s$(水素の場合)、又は $2s$、$2p_x$、$2p_y$、$2p_z$(炭素及びリチウムなどの場合)である。MOPAC では RHF 計算 (制限 Hartree-Fock) と UHF 計算 (非制限 Hartree-Fock) が扱える。RHF は up スピンと down スピンの入る軌道は同じ関数で表され、UHF では違う関数で表される。よって得られる準位は UHF では RHF の倍となる。本研究では全て、UHF 計算で行なった。UHF 計算では、したがって炭素の場合 1 個の原子あたり $4\ 2 = 8$ 個の原子軌道の係数を最適化することになる。今回の修士論文の原子数は約 80 個程度であり、約 700 軌道の変分空間の電子状態の最適化を行っている。

2.1.3 MOPAC のオプション

MOPAC ではオプションを指定することで、様々な機能が使用できる。以下では、本研究で使用したオプションを列挙し、簡単な説明をする。

- GNORM=n
  エネルギー勾配が n になったら計算を終了させる。
  構造最適化計算終了の判定基準となる。実際に使用した n の値は、粗い精度の場合は 0.5 であり、振動等を求めるときは 0.01 である。通常は 0.1 を用いた。

- PULAY
  SCF を得るために Pulay の強制収束法を使用する。

- SHIFT=n
  SCF の計算の開始に減衰ファクター。実際の計算では 2 を用いた。

- PM3
  近似法として PM3 法を使用する。

- UHF
  非制限ハートリーフォック計算をさせる。何も指定しなければ RHF (制限ハートリーフォック) 計算をする。

- GEO-OK
  構造最適化において幾つかの安全チェックを無視させる。

- SYMMETRY
  対称性を保ちながら計算をさせる。
  対称性を考慮することにより、計算時間を短縮することができる。グラファイトの構造が変形しない、 Li ドープの場合に用いた。

- DRC
  動的反応座標の計算
  DRC = t とした場合 : t > 0 半減期 t [fs] でエネルギーを減らす。
  : t < 0 半増期 −t [fs] でエネルギーを増やす。
  特に、この内容については次項にて説明する。

- T-PRIORITY=t
  DRC 計算で、時間が t [fs] 変化するごとに出力。実際には、 t=2、又は 5 用いた。

- OUTMO
  状態密度の計算用に、固有値、固有ベクトルの出力を行う。出力ファイル名は outmo.dat である。

2.2 動的反応座標

また、本研究では動的反応座標を用いて計算した。動的反応座標とは、化学反応の様子を、実時間毎に分子の構造、力、運動、電子状態構造最適化だけでなく化学反応の様子を振動、回転、の効果も含めた反応座標を求めることができ、反応条件のより詳しい検証をすることが期待できる。

2.2.1 計算原理

この計算はニュートンの運動方程式 $f_i = m_i a_i$ (f_i、m_i、a_i はそれぞれ粒子 i に働く力、粒子 i の質量、加速度) を数値積分することより求める。エネルギー勾配を $g_i(t) = \nabla_i E(t)$ 、 ($E(t)$ は時間 (t) における SCF 計算結果のエネルギー、 ∇_i は粒子 i の座標に関する微分) とすると、

$$-\frac{g_i(t)}{m_i} = v_i'(t) \tag{2.2.1}$$

$$\left(v_i'(t)=\frac{dv_i(t)}{dt}、v_i は粒子 i の速度\right)$$

同様に

$$-\frac{g_i(t-dt_1)}{m_i}=v_i'(t-dt_1)$$
$$=v_i'(t)-v_i''(t)dt_1+\frac{1}{2}\cdot v_i'''(t)(dt_1)^2\cdots \quad (2.2.2)$$
$$-\frac{g_i(t-dt_1-dt_2)}{m_i}=v_i'(t-dt_1-dt_2)$$
$$=v_i'(t)-v_i''(t)(dt_1+dt_2)$$
$$+\frac{1}{2}\cdot v_i'''(t)(dt_1+dt_2)^2\cdots \quad (2.2.3)$$

(dt_1、dt_2はタイムステップ。Taylor展開を行なった。)

となる。

よって、

$$\frac{g_i(t)-g_i(t-dt_1)}{m_i}=-v_i''(t)dt_1+$$
$$\frac{1}{2}\cdot v_i'''(t)(dt_1)^2\cdots \quad (2.2.4)$$
$$\frac{g_i(t)-g_i(t-dt_1-dt_2)}{m_i}=-v_i''(t)(dt_1+dt_2)+$$
$$\frac{1}{2}\cdot v_i'''(t)(dt_1+dt_2)^2\cdots \quad (2.2.5)$$

となる。

$(2.2.4)\times(dt_1+dt_2)-(2.2.3)\times dt$ で、 $v_i''(t)$ に関する項を消去すると、

$$v_i'''(t)=\frac{2}{m_i}$$
$$\times\quad\frac{\{g_i(t)-g_i(t-dt_1)\}(dt_1+dt_2)-\{g_i(t)-g_i(t-dt_1-dt_2)\}dt_1}{(dt_1)^2(dt_1+dt_2)-dt_1(dt_1+dt_2)^2} \quad (2.2.6)$$

$$v_i''(t)=\frac{\dfrac{g_i(t-dt_1)-g_i(t)}{m_i}+\dfrac{1}{2}\cdot v_i'''(t)(dt_1)^2}{dt_1} \quad (2.2.7)$$

となる。

半経験的分子起動法計算から得られた $g_i(t-dt_1-dt_2)$、$g_i(t-dt_1)$、$g_i(t)$ を式 (2.2.1)、(2.2.6)、(2.2.7) に代入して $v_i'(t)$、$v_i''(t)$、$v_i'''(t)$ を求め、時間 dt 後の速度と位置

$$v_i'(t+dt)\sim v_i(t)+v_i'(t)dt+\frac{1}{2}\cdot v_i''(t)(dt)^2+$$

$$\frac{1}{6}\cdot v_i'''(t)(dt)^3$$

$$x_i'(t+dt) \sim x_i(t) + v_i(t)dt + \frac{1}{2}\cdot v_i'(t)(dt)^2 +$$

$$\frac{1}{6}\cdot v_i''(t)(dt)^3 + \frac{1}{24}\cdot v_i'''(t)(dt)^4$$

を求める。

以上の計算を繰り返し行なうことにより DRC を求めることができる。

2.2.2 温度による評価

実際の MOPAC 93 での動的反応座標の計算では、各原子に運動方向と運動速度を与え、計算を実行する。そして、出力される結果は、各反応時間の構造と、各原子の運動方向と速度及び、全体のポテンシャルエネルギーと、運動エネルギーである。つまり、対象分子の温度を直接評価していない。そのために、我々は運動速度から温度を求めるプログラムを作製し、計算結果から反応　分子の温度を測定した。その原理を以下に示す。

一般に高温近似において運動エネルギー K は、1 自由度あたり $1/2kt$ のエネルギーを持つ。即ち、

$$\begin{aligned} K &= \sum \frac{1}{2} m_i v_i^2 \\ &= \frac{3}{2} NkT \end{aligned} \tag{2.2.8}$$

となり、運動速度から温度への変換、及び逆変換が可能となる。しかし、1 つの分子上での反応を考えた場合、重心の並進や、回転運動は反応とは関係ない。そのため、原子の運動の中で、重心の並進、回転運動を取り除く必要がある。

まず、質量 m_i、速度 V_i で動いてる原子 i からなる分子を考える。原子 i の位置を X_i とすると分子の重心は位置 X は

$$X = \frac{1}{M}\sum_i m_i X_i \qquad M = \sum_i m_i \tag{2.2.9}$$

この重心の速度 V は $dX/dt = V_i$ より、

$$V = \frac{dX}{dt} = \frac{1}{M}\sum_i m_i V_i \tag{2.2.10}$$

分子の全体としての重心のまわりの角速度ベクトル ω は

$$\omega = \frac{1}{I}\sum_i m_i r_i \times (V_i - V) \tag{2.2.11}$$

ただし、r_i、I は重心からの位置ベクトル、及び慣性モーメントである。また、ここでは X はベクトルの外積をあらわしている。

$$r_i = X_i - X \tag{2.2.12}$$

$$I = \sum_i m_i r_i \cdot r_i (\text{慣性モーメント}) \tag{2.2.13}$$

となる。したがって、(v, ω) で動く座標系での原子 i の速度を V_i とすると、

$$V_i = V + \omega \times r_i + V_i \tag{2.2.14}$$

と表される。つまり原子の運動は、分子の剛体としての速度である並進、回転速度と、熱力学的速度である振動の速度に分けることができる。

次に、それぞれの速度に対する温度を考える。分子の運動エネルギーを E その運動の自由度を ν とするとき、その分子の運動に対する高温での分子温度 T は、

$$\nu kT = 2E \tag{2.2.15}$$

と、対応づけることができる。ここで k はボルツマン定数である。したがって、無次元化温度 T_r としてはポテンシャルエネルギーの代表値を U_r として

$$T_r = \frac{U_r}{k} \text{あるいは} \frac{\varepsilon}{k} \tag{2.2.16}$$

と選ぶことができる。

(x、 y、 z) 座標系での分子に対する並進温度、$T^{(trans)}$ は、

$$3kT^{(trans)} = 2\left(\frac{1}{2}m{v_x}^2 + \frac{1}{2}m{v_y}^2 + \frac{1}{2}m{v_z}^2\right) \tag{2.2.17}$$

回転温度は、$\nu = 3$ のとき

$$3kT^{(rot)} = 2\left(\frac{1}{2}I_{ix}{\omega_{ix}}^2 + \frac{1}{2}I_{iy}{\omega_{ix}}^2 + \frac{1}{2}I_{iz}{\omega_{ix}}^2\right) \tag{2.2.18}$$

となる。振動温度 $T^{(vib)}$ は運動エネルギーの平均をとって、

$$3kT^{(vib)} = 2\left\langle\left(\frac{1}{2}m{v_{ix}}^2 + \frac{1}{2}m{v_{iy}}^2 + \frac{1}{2}m{v_{iz}}^2\right)\right\rangle_i \tag{2.2.19}$$

となる。

よって、速度ベクトルから温度を求めるには、分子の並進、回転速度を取り除き、残った振動速度より運動エネルギーを求め、$T^{(vib)}$ を求め、それが分子の温度となる。

また、逆に温度から速度ベクトルを求めることもでき、その計算プログラムを作成した。そして、ある温度に対応するそれぞれの運動の速度を求めた。このとき、振動方向は固有振動の方向をとるべきだが、計算の都合上ランダムな方向を与える。実際、高温状態では全く問題ない。

そして、それぞれの温度の元に求められた運動ベクトルを式 (2.2.14) より足し合わせると、原子の速度ベクトルが計算できる。

2.3　計算モデル及び計算条件

次に、本研究で用いた MOPAC の入力データの作製方法、計算モデル、計算条件等の実際の計算方法について説明する。

2.3.1　入力データ

入力データは一つのファイルに記述する。ファイルの名前は filename.dat のように .dat という拡張子をつける。最初の 1 行にオプションのキーワードを、次の 2 行にコメントを書き、 4 行目から分子の構造を記述する。また + オプションでオプション行を増やし、 2 行目、 3 行目にもオプションを書くことができる。構造の記述の仕方は 3 通りある。内部座標形式、 XYZ 座標形式、 GAUSSIAN 形式である。本研究では、 MOPAC で一般的に使われている内部座標形式を用いた。内部座標形式の構造の記述の仕方は、次のようである。

定義した原子の順に番号を付けていくと、i 番目の原子の位置の定義は、定義済みの原子 j、 k、 ℓ によって記述される。i 番目の原子は、 (a)j 番目の原子との距離 r(Å 単位)、 (b) 原子 i、 j、 k でなす結合角 θ(度)、 (c) 原子 i、 j、 k でなす面と原子 j、 k、 ℓ でなす面とのなす 2 面角 ψ(度) で定義される (図 2.1)。

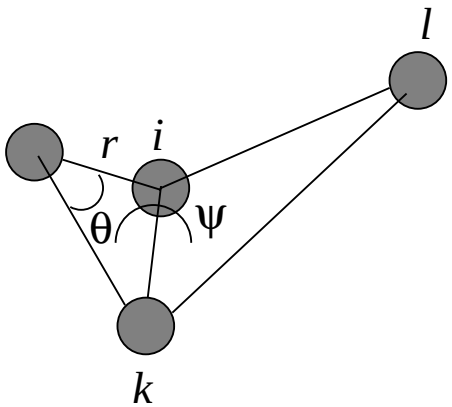

図 2.1: 構造の定義[1]

また、 1 番目の原子はそれ以前に定義済みの原子がないので内部座標は共に 0 とし、 2 番目の原子は 1 番目の原子との距離のみ指定して他は 0 とし、 3 番目の原子は 1、 2

[1] ijkl.eps

番目の原子を参照して原子間距離と結合角を指定して 2 面角は 0 とする。

また、対称性を考慮して構造を定義するには、対称関数を用いる。対称性の指定は、SYMMETRY オプションを指定し、構造データの次に空行を 1 行入れ、その次の行から記述する。記述は参照原子の番号、対称関数、指定原子の順に記述する。対称関数はそれぞれ、1: 指定原子の原子間距離が参照原子と同じ、2: 指定原子の結合角が参照原子と同じ、3: 指定原子の 2 面角が参照原子と同じにするという意味である。また、対称関数は他にもあるがここでは述べない [12]。

以下にメタン CH_4 の入力例を示す。

```
SYMMETRY T=1.0D NOINTER GNORM=0.01 PM3 GEO-OK UHF SHIFT=2 PULAY
CH4
neutral
C  0.00000  0    0.00000  0    0.00000  0   0   0   0
H  1.09000  1    0.00000  0    0.00000  0   1   0   0
H  1.09000  0  109.47000  1    0.00000  0   1   2   0
H  1.09000  0  109.47000  0  120.00000  0   1   2   3
H  1.09000  0  109.47000  0 -120.00000  0   1   2   3

 2  1   3   4   5
 3  2   4   5
```

ここで対称性を指定 (SYMMETRY) しており、C−H の結合距離、∠HCH が共通になるようにしてある。

構造最適化計算での計算条件は、このキーワードを共通して用いた。計算終了条件を示す GNORM=n は、 0.01 〜 0.5 の値を用いた。また、F をドープした計算等、対称性の破れる結果がある場合には SYMMETRY を指定しなかった。

また、動的反応座標計算での入力データは xyz 座標で行う。そして、それぞれの原子の速度ベクトル (x,y,z 方向) を与え、計算を実行する。以下に、メタンの振動をさせるための入力データを例として示す。速度ベクトルの単位は [cm/s] である。

```
T=1.0D NOINTER GNORM=0 PM3 GEO-OK UHF SHIFT=2 PULAY &
VELOCITY T-PRIORITY=5.0 LARGE=-1 DRC
DRC=  0, kine=     0.894 T=  300.0,
   C    0.0000 0       0.0000 0       0.0000 0
   H    1.0900 0       0.0000 0       0.0000 0
   H   -0.3633 0       1.0277 0       0.0000 0
   H   -0.3633 0      -0.5138 0      -0.8900 0
   H   -0.3633 0      -0.5138 0       0.8900 0
```

```
      3.35601        3.21141       -0.51144
     -7.28521   -37369.25310   -32349.09367
  35215.51327    12440.31051    32343.98679
-301153.70763  -203586.99777   240454.20083
 240111.46821   246715.63030   240453.77406
```

そして、このキーワードを共通してすべての計算に用いた。ただし、半減期、半増期を示す DRC=n は、通常 n=0 を用いた。しかし、計算によって $\pm 100 \sim 500$ の値を使用した。

2.3.2 計算に用いた炭素クラスター

本研究で用いたグラファイトのモデルは図 2.2であり、全てのダングリングボンドを水素終端させたものを用いた。グラファイトのダングリングボンドは非常に不安定であり、通常、水素などによって終端されているからである。

また、 3章で詳しく述べるが、フッ素原子は一度炭素原子と結合すると、その位置で移動しないのに対して、リチウム原子はグラファイト上を比較的自由に移動することができる。そのため、リチウムドープでは (a) の炭素原子数 54 個のグラファイトを用い、フッ素ドープでは (b) の炭素原子数 24 個のグラファイトを用いた。また、ヨウ素ドープの計算では、 DRC 計算を用いたため、 (b) のグラファイトを用いて計算を行なった。

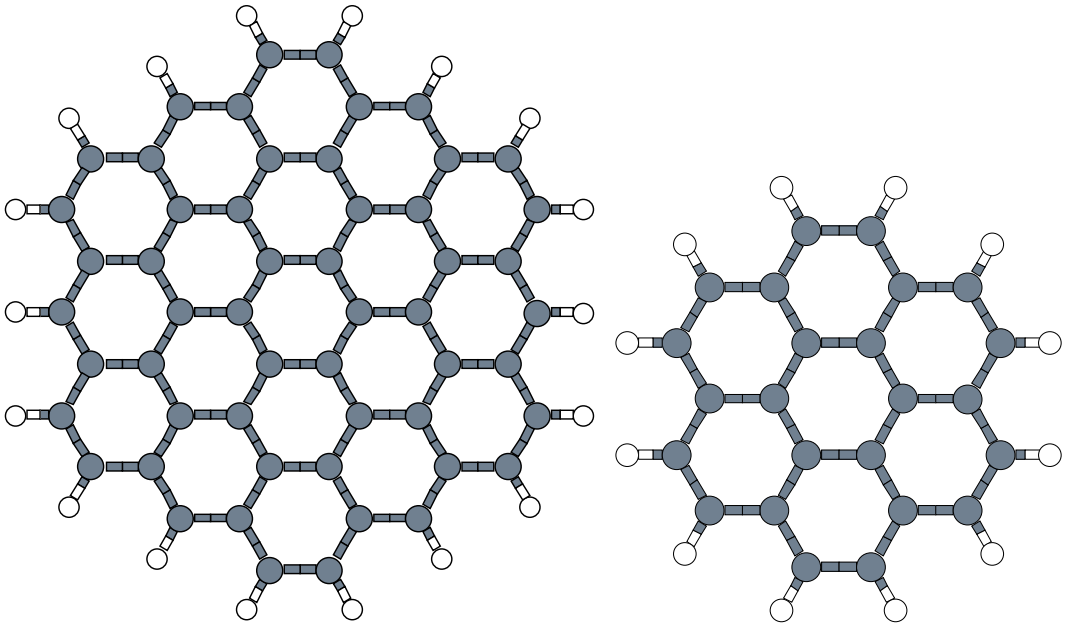

図 2.2: 用いたグラファイトモデル [2][3]

MOPAC では、グラファイト層間の van der Waals 力を正しく求めることができず、グラファイト間の 3.35 Å を再現することができない。そのため、 2 層のグラファイトは用いず、 1 層のグラファイトのみ計算に用いた。

[2] C54H18.eps

[3] C24H12.eps

2.3.3 吸着エネルギーの計算

本研究では、ドープした Li や F の吸着エネルギーを以下の式を用いて求め、構造の安定度の比較を行なった。ここで、ドープした原子を X 原子 と示すと、吸着エネルギー $E_{\mathrm{X}_{ad}}$ は、

$$E_{\mathrm{X}_{ad}} = \frac{E_{\mathrm{C}_m\mathrm{H}_n} + E_{\mathrm{X}} \times k - E_{\mathrm{C}_m\mathrm{H}_n\mathrm{X}_k}}{k} \tag{2.3.20}$$

とあらわされる。ここで、

$$\begin{aligned} E_{\mathrm{X}_{ad}} &: \text{X原子の吸着エネルギー} \\ E_{\mathrm{C}_m\mathrm{H}_n} &: \mathrm{C}_m\mathrm{H}_n\text{の全エネルギー} \\ E_{\mathrm{X}} &: \text{X原子単体の全エネルギー} \\ E_{\mathrm{C}_m\mathrm{H}_n\mathrm{X}_k} &: \mathrm{C}_m\mathrm{H}_n\mathrm{X}_k\text{の全エネルギー} \\ m &: \text{炭素の数} \\ n &: \text{水素の数} \\ k &: \text{X原子の数} \end{aligned}$$

を表している。エネルギーの単位は [kcal/mol] 又は、 [eV] であり、 1 [eV] = 23.0492 [kcal/mol] である。

2.3.4 状態密度の計算

本研究では、クラスターモデルの電荷の移動などを評価するために、 MOPAC の計算結果からクラスターの電子状態の状態密度の計算を行なった。これは、 MOPAC で得られた固有値 (エネルギー準位) の Gaussian 分布の巾をつけ、全ての固有値の分布の和をとるという方法を用いて、個体の状態密度と同じ比較ができる。 Gaussian の巾はクラスターの大きさから期待でき、エネルギー間隔 ΔE は、

$$\Delta E = \frac{W}{N} \qquad (W : \text{エネルギーバンド巾}、N : \text{原子数}) \tag{2.3.21}$$

程度にする。

2.4　作成プログラムの説明と使用方法

本研究では、 DRC の計算用に温度と速度を変換するプログラムや、データ処理等のためのプログラムを作成した。ここで、そのプログラムの具体的な使用方法について説明する。ここで、ファイル名は filename と説明する。

2.4.1　DRC 計算用入力データ作成

DRC 計算を行うためには、入力座標が xyz 座標で、それぞれの原子に運動の速度ベクトルを与えなければならない。そこで、このプログラムによって xyz 座標と、定めた分子群に温度を与えることによって、それに対応する速度ベクトルをつくりだした。プログラムの使用手順を以下に示す。

1. xmol 等より、使用する分子群の xyz 座標を作成する。簡単な方法として、内部座標系で作成した分子を xmol で読み込み、次に、 format を xyz 形式で保存する方法がある。この時、ファイル名は filename.xyz とする。このとき原子の並ぶ順番として、

   (1) 母材分子群

   (2) ドープ分子群

   (3) ドープ分子群

   ⋮

   である必要がある。この時の母材分子群とは、 DRC 計算で振動のみをさせる分子で、並進、回転のない分子である。また、ドープ分子群とは振動、回転、並進運動を行う分子を示している。

   例えば、グラファイトに F 分子をドープする計算の場合、グラファイトが母材分子群、 F 分子がドープ分子群となる。また、分子振動のみを DRC 計算するため、ドープ分子群はなくてもよい。

2. 温度等の必要なデータを入れた input ファイルを作成し、同じディレクトリにおく。ただし、作らなくても実行することは可能である。

1 行目	:	n	: 半減期、半増期の時間。エネルギー保存時は 0 を入力。
2 行目	:	n	: 母材分子群の最終原子の番号
3 行目	:	n	: 母材分子の温度 [K]
4 行目	:	n,m	: ドープ分子の最初と、最後の原子の番号。
5 行目	:	n	: ドープ分子の温度 [K]
6 行目	:	n	: 分子並進方向の入力方法指示。1：xyz 座標 or 2：特定分子へ
7 行目 (1)	:	x,y,z	: 並進方向ベクトル
7 行目 (2)	:	n	: 並進方向先の原子の番号
8 行目	:	x,y,z	: 回転軸ベクトル

分子群が複数のときは、4 〜 8 行目を繰り返す。ただし、分子群が 1 原子のとき、8 行目は省略すること。

3. % xyz2DRC filename input
と実行する。このとき、 input を入力しないときは必要なデータを質問通りに入力することによって作成することができる。入力データが正しければ、以下のファイルが作られる。

filename.dat	:	mopac 入力ファイル
filename.xyz	:	元の xyz 座標に、方向ベクトルを入力
filename.temp	:	filename.xyz に、分子群の重心をダミー原子で追加したもの

計算には、 filename.dat を使用する。また、 filename.xyz は温度などのデータを変更した新たな入力ファイルを作成するときに使用する。 filename.temp は入力データの確認用のファイルであり、ダミー原子の位置で重心のずれがないことを確認し、xmol でベクトル方向や大きさを確認する。

そして、 filename を適当な名前にし、必要ならばまた新たな条件のファイルを作成する。最後に不必要なファイルを消去する。(プログラムソースは付録 A.1参照)

2.4.2 DRC 計算処理

DRC 計算が正しく行われると、 out ファイルに、時間経過ごとの構造、エネルギー等のデータが出力されていくが、そのままでは構造変化の様子や、エネルギー変化等の様子を見ることはできない。そのため、必要な結果を取り出し、構造やエネルギー

の時間変化を簡単に観測するためのプログラムを作成した。これによって、構造変化のアニメーションや、エネルギー変化のグラフを出力させることができる。ここに、このプログラムの使用方法を説明する。またこのプログラムは、計算中にも実行できる。その時出力されるエラーは無視してよい。

この計算結果から得られるデータは、その時間の構造、速度ベクトル、及びエネルギーである。それより、得られる時間変化のデータは、構造変化のアニメーション、特定原子の距離変化、ポテンシャル・運動エネルギー 、及び温度変化である。

このプログラムを実行すると、表2.1の中の必要なデータの種類を聞いてくるので、それに対応する番号を入力する。そして、それぞれの案内通りに必要な指示を行うと、対応する結果のデータファイルが作成される。

出力切り替え番号	データ種類	出力ファイル拡張子	入力指示
1	アニメーション	anm	なし
2	エネルギー変化	energy	なし
3	原子間距離変化	dis	指定原子番号 2 個
4	温度変化	thermo	指定原子番号等

表 2.1: DRC 計算結果編集プログラムの対応表

原子間距離を求めるときには、その 2 原子の原子の番号を指示にしたがって入力する。

ここで、プログラム実行中にも表示されるが、温度変化を求めるとき、このプログラムは同時にいくつも分子群についての求めることができるので、その分子群の数と、それぞれの分子群の最初と最後の原子の番号を入力する。このとき、1 つの原子は複数の分子群の構成要素になることが可能である。(プログラムソースは付録 A.2参照)

2.4.3 状態密度

本研究で用いた状態密度の計算プログラムは、中平 [1] によるプログラムを改良したものを使用した。改良点は、

- Li 原子以外の使用
- すべての軌道の出力
- 1 原子の状態密度の計算

である。

使用方法はまず、 mopac の計算時に OUTMO というキーワードを入力する。計算終了後、 outmo.dat というファイルが作成されるので、 filname.grp と書き換える。

そして、 % grp.out filename と実行することによって、それぞれの軌道の状態密度が作成される。また、 grp-number.out を使用すると、指定した原子のみの状態密度が作成される。

(プログラムソースは付録 A.3参照)

このプログラムを使用してデータ処理を行ったグラフは図 3.16、図 3.17、図 3.19等である。

2.4.4　out ファイル処理

ここでは、 out ファイルよりドープ原子の情報を取り出し、原子位置、電荷、エネルギーの関係から、必要なグラフを作り出すためのプログラムである。

使用方法は、 UNIX 上で、 grep を用いて、 out ファイル内のドープ原子の情報と、エネルギーの情報を読み出す。これは、付録 A.4のコマンドを用いることによって、ディレクトリ内のすべての out ファイルからドープ原子の情報と、その系のエネルギーを取り出し、このプログラムによって、処理、編集、出力することができる。

これによって得られる結果は、ドープ原子ごとの xyz 座標、電荷、及び系ごとの、ドープ数、総電荷、吸着エネルギーである。そこで、任意のデータデータ編集し、出力させることによってグラフを作成させる。

(プログラムソースは付録 A.5参照)

このプログラムを使用してデータ処理を行ったグラフは図 3.8、図 3.9、図 3.14等である。

2.4.5　arc ファイル処理

このプログラムも 付録 A.4 のコマンドを用いて、 arc ファイル内のドープ原子の情報を取り出したものを、処理、編集、出力が可能である。

これによって得られる結果は、ドープ原子ごとの結合距離、電荷、及び系ごとのドープ数である。

任意のデータデータ編集し、出力させることによってグラフを作成させる。

(プログラムソースは付録 A.6参照)

このプログラムを使用してデータ処理を行ったグラフは図 3.25、図 3.28等である。

第 3 章

結果及び考察

本章では計算から得られた結果を示し、その結果について考察をする。3.1では、Li についての結果を示し、考察する。同様に、3.2では F について、3.3では I についての結果を示し、考察する。

3.1　リチウムドープ

$C_{54}H_{18}$ のグラファイトに Li を大量ドープした系の構造最適化と電子状態について計算し、それから得られた結果とそれによる考察を述べる。

3.1.1　Li 吸着の最適化構造

$C_{54}H_{18}$ のグラファイトに Li 原子を 1 〜 24 個ドープし、構造最適化を行った。Li ドープの計算では、Li 原子は初期位置から数Å も移動した安定位置を持つことがある。特に、内側の 6 員環上を初期位置としたものが、エッジサイトに移動し、安定することが多い。そこで便宜上、複数の Li 原子の安定位置の中に、図 3.1のように、内部の六員環で安定している Li が一つでもある系と、図 3.2のように、すべての Li がエッジサイトで安定した系で分けた。

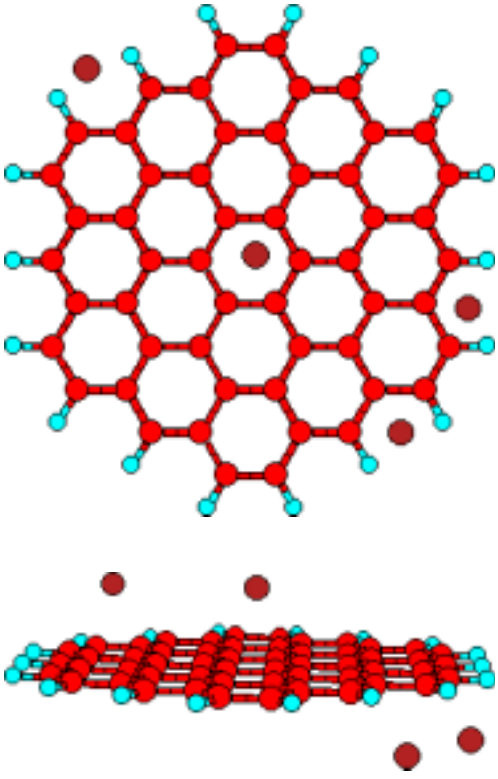

図 3.1: 内部にも吸着した系 [1] [2]

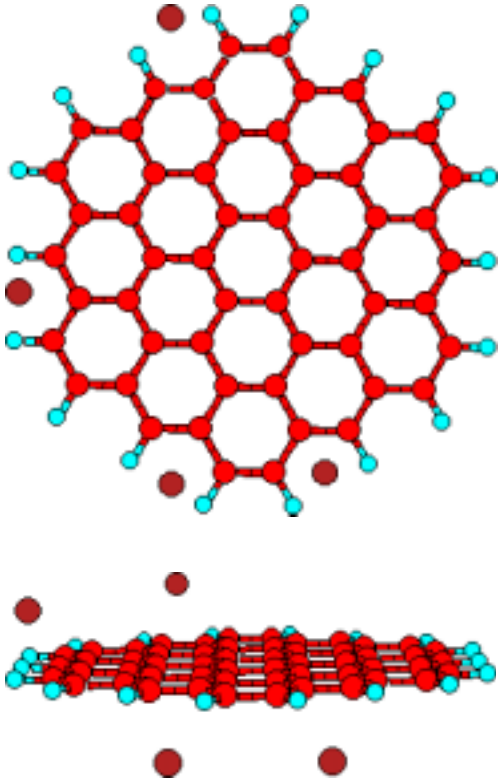

図 3.2: すべてエッジサイトに吸着した系 [3] [4]

中平の報告でもあるとおり、グラファイト上の Li は比較的自由に移動することができる。そこで、図 3.3 のように、グラファイトの中心から A 、 B 方向へ Li 原子をエッジ方向へ少しずつ動かし、それぞれの位置において構造最適化をおこなった。このとき、 Li 原子に与えた自由度はグラファイト面に対し垂直方向のみであり、グラファイトの構造は固定したままである。そして、それぞれの位置での Li 原子の高さと、吸着エネルギーについて比較した。

[1] C54-Li-center.eps

[2] C54-Li-center-2.eps

[3] C54-Li-center.eps

[4] C54-Li-no-center-2.eps

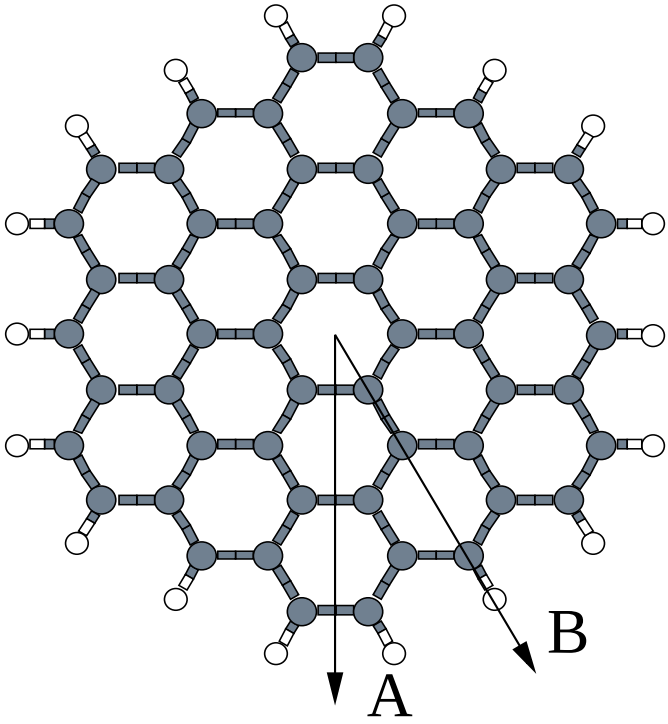

図 3.3: $C_{54}H_{18}$ の中心から端への状態の変化の比較した 2 方向 [5]

それぞれの方向に対する Li 原子の高さと吸着エネルギーの変化を図 3.4、図 3.5に示す。グラフの横軸はグラファイトの中心からの距離である。また、縦軸は吸着エネルギーと、グラファイト平面からの高さを示す。吸着エネルギーは、低い方がより安定である。

それぞれの方向のエッジ部分は、異なる形状であり、通常 A 方向はアームチェア型、 B 方向はジグザグ型といわれるエッジ部分へ進む方向である。また、六角形の中心と最近接の六角形の中心の距離は、 2.46 Å で、グラファイトのボンド長は 1.42 Å である。したがって、 A 方向では 6.15 Å 以上、 B 方向では 5.68 Å 以上が、エッジ領域に対応する。

[5] C54H18-edge.eps

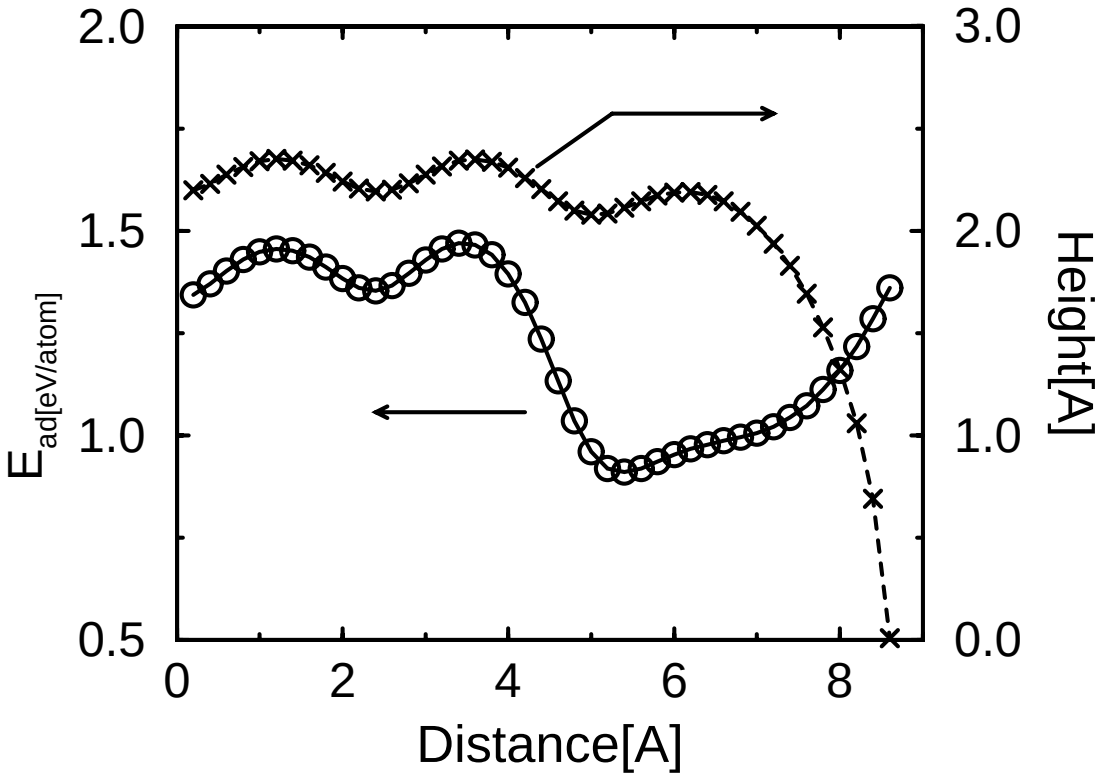

図 3.4: 方向 A における Li 原子の高さと吸着エネルギーの変化 [6]

A 方向には 約 0.12 eV の障壁エネルギーがあり、最外六員環内の上空がもっとも安定になる。そして、エッジ部分では、より外側の位置ほど、より不安定である。

[6] C54H18-Li-A.eps

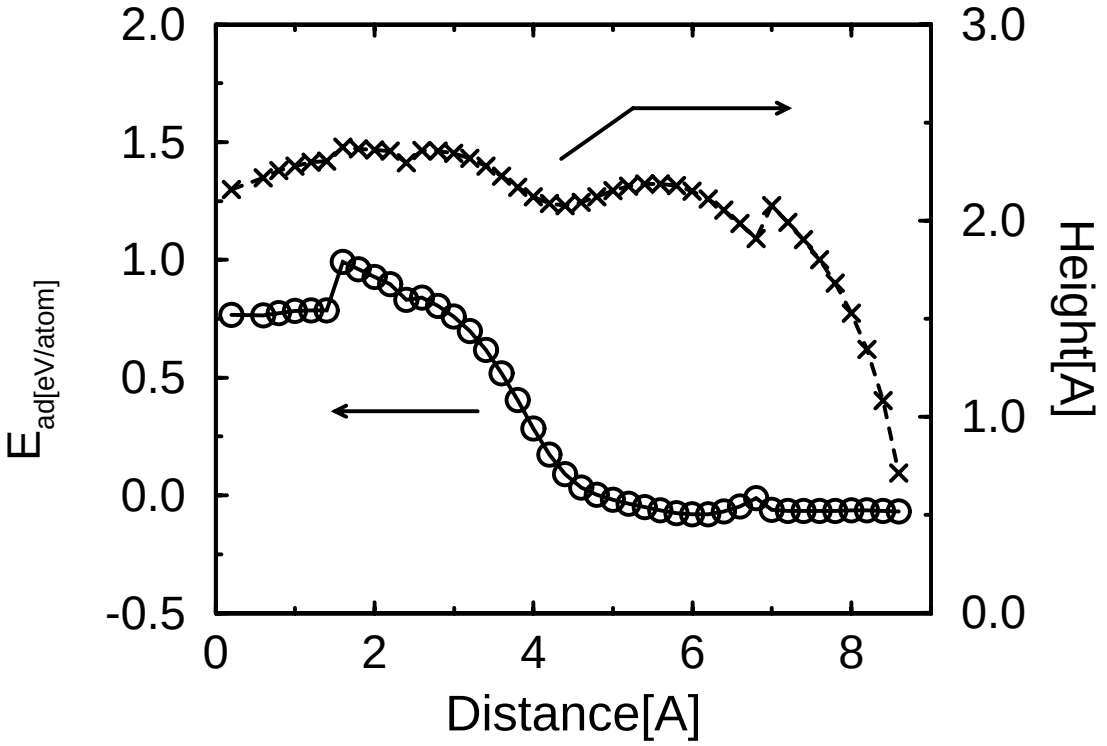

図 3.5: 方向 B における Li 原子の高さと吸着エネルギーの変化 [7]

ジグザグ型の方向は、全体的に外側へ行くほど安定に存在し、エッジ部分がもっとも安定になる。このとき中心付近に約 0.21 eV の障壁エネルギーが存在し、 A 方向の値より大きい。しかし、全体的に A 方向の吸着エネルギーより B 方向の吸着エネルギーが低く、 B 方向への移動は容易におこなわれると考えられる。

また、 3.5で 中心からの距離が約 1.4Å と 約 6.8 Å の位置で、縦軸のグラファイト平面からの高さや、吸着エネルギーのデータが、とびとびの値をとっている。これは、グラファイトの中心からの距離、 約 1.4 〜 2.8 Å では、 Li が、グラファイトのボンド上に存在している。また、グラファイトの中心からの距離、約 6.8 Å は、終端された水素原子の上空であるためであると、それぞれ考えられる。そして、エッジの形状によって Li のグラファイト上の移動の容易さが異なると考えられる。

ここで用いたグラファイト構造は 1 種類であり、まだ検証の余地が多い。例えば、グラファイトのエッジ部分に、アームチェア型の構造の割合を多くした構造や、ドープする Li の原子数を 2 個などの複数についての計算が残っている。

[7] C54H18-Li-A.eps

3.1.2　Li の微小グラファイトの電荷移動

電池の応用を考えるため、グラファイトから Li への電荷移動について調べる。グラファイト上の Li の電荷には、図 3.6のように正、負それぞれの値をとるものがある。図内の数値は電荷を示しており、 0 は省略してある。

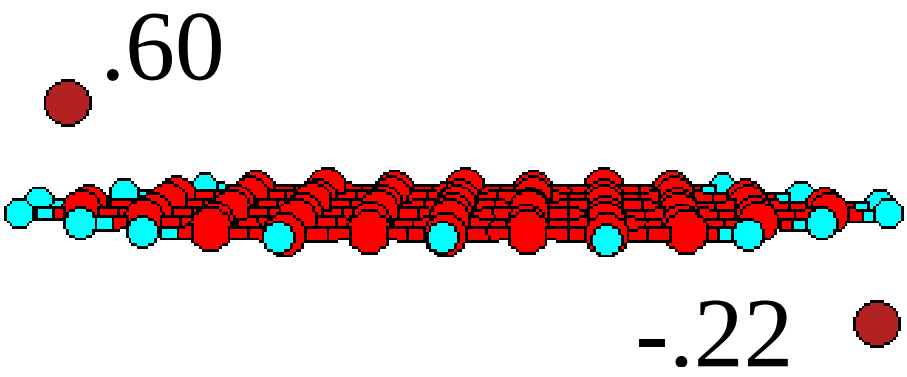

図 3.6: $C_{54}H_{18}$ グラファイト上の正と負の電荷を持った Li 原子 [8]

このとき、ドープ原子数が 1 個である場合は、必ず正の電荷をとる。しかし、ドープ数が 2 個以上になると、ドープ数、ドープ位置、 Li 同士の位置関係等によって、 $+0.6$ から、 -0.3 まで、 0 を含む連続したどれかの値の電荷を持つ。

大量にドープさせたときの構造、及び電荷を図 3.7に示す。ドープ原子数は 14 個で、図内の数値は電荷を示している。

[8] Li-2.eps

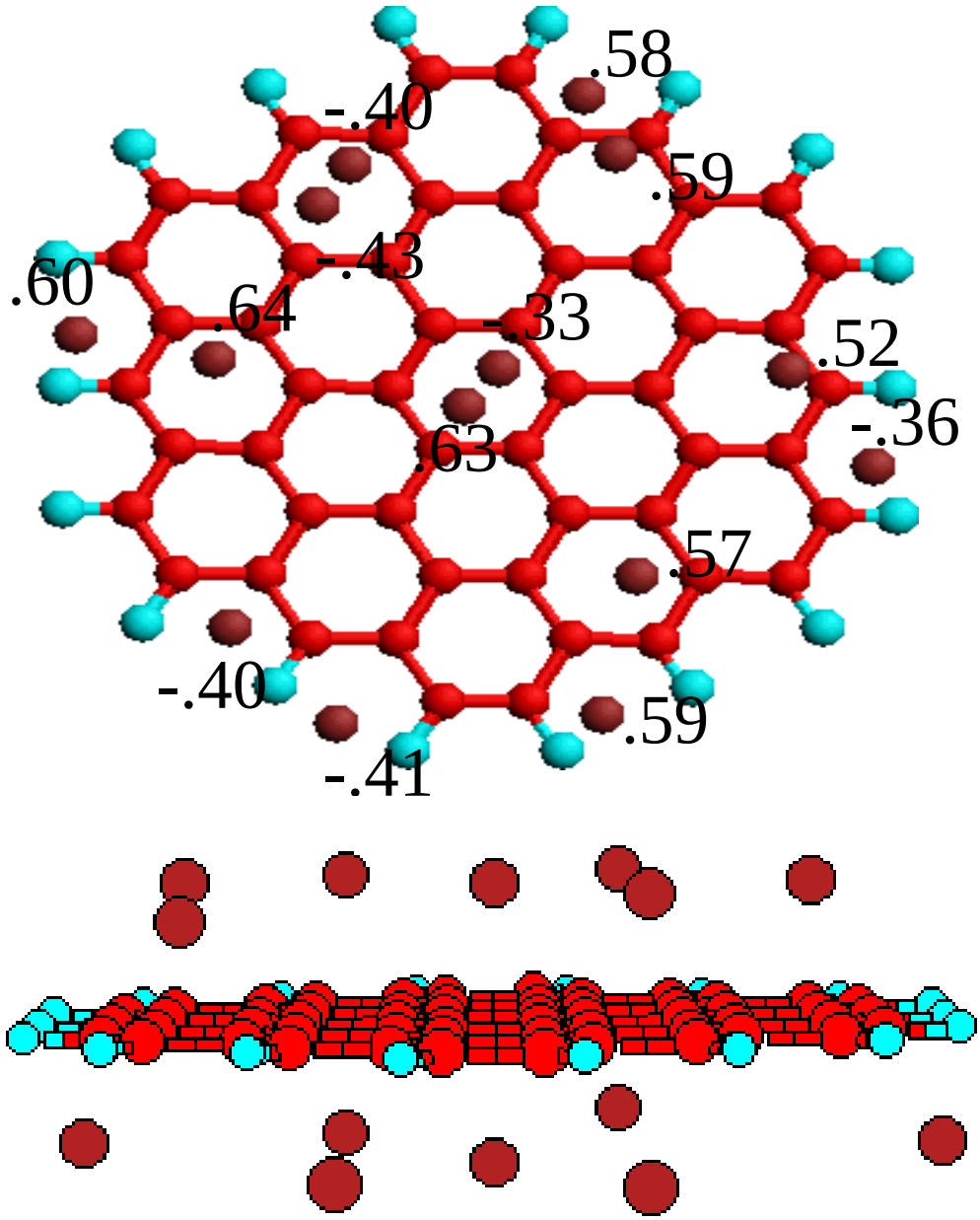

図 3.7: $C_{54}H_{12}Li_{14}$ の構造と電荷 [9] [10]

ドープした Li 原子は、グラファイトの六員環の上空、又はジグザグ型のエッジ部分の上空に安定に存在する。そこで、複数の Li 原子をドープした系の計算を行い、それぞれの原子の位置と電荷の関係、及びそれぞれの系における総電荷移動等について調べ、Li 原子のドープ位置と電荷についての関係について考察を行った。

[9] Li-14.eps

[10] Li-14-b.eps

まず、 Li の安定位置を調べる。図 3.1、図 3.2の構造のように、 Li はグラファイトの上空と、端の位置に吸着しているのがわかる。そこで、安定している Li 原子の位置を、グラファイト中心からの距離と、グラファイト平面からの高さとの関係について、 $C_{54}H_{18}Li_x(x = 1, \cdots, 25)$ の 115 個のクラスターをまとめて表示したのが図 3.8である。

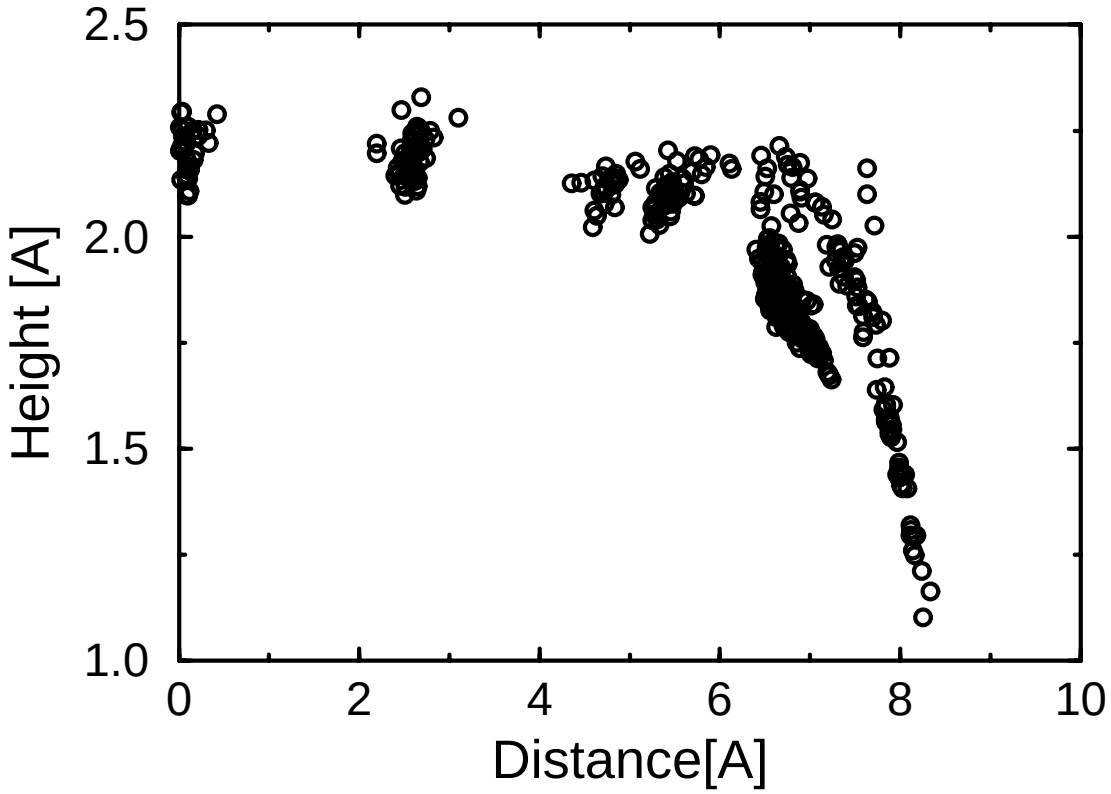

図 3.8: Li の結合位置のグラファイト中心からの距離と平面からの高さ [11]

内部領域の Li の安定位置は六員環の中心付近に集まっており、グラファイト平面からの高さは比較的高い位置 (約 2.2 〜 2.3 Å) で吸着している。それに対して、端の安定位置のグラファイト平面からの高さは比較的 (1.1Å) から連続的に存在している。

[11] C54-Li-l-z.eps

次に、Li 原子の吸着位置と電荷の関係を調べる。まず、グラファイトの中心からの距離と電荷の関係を図 3.9に示す。

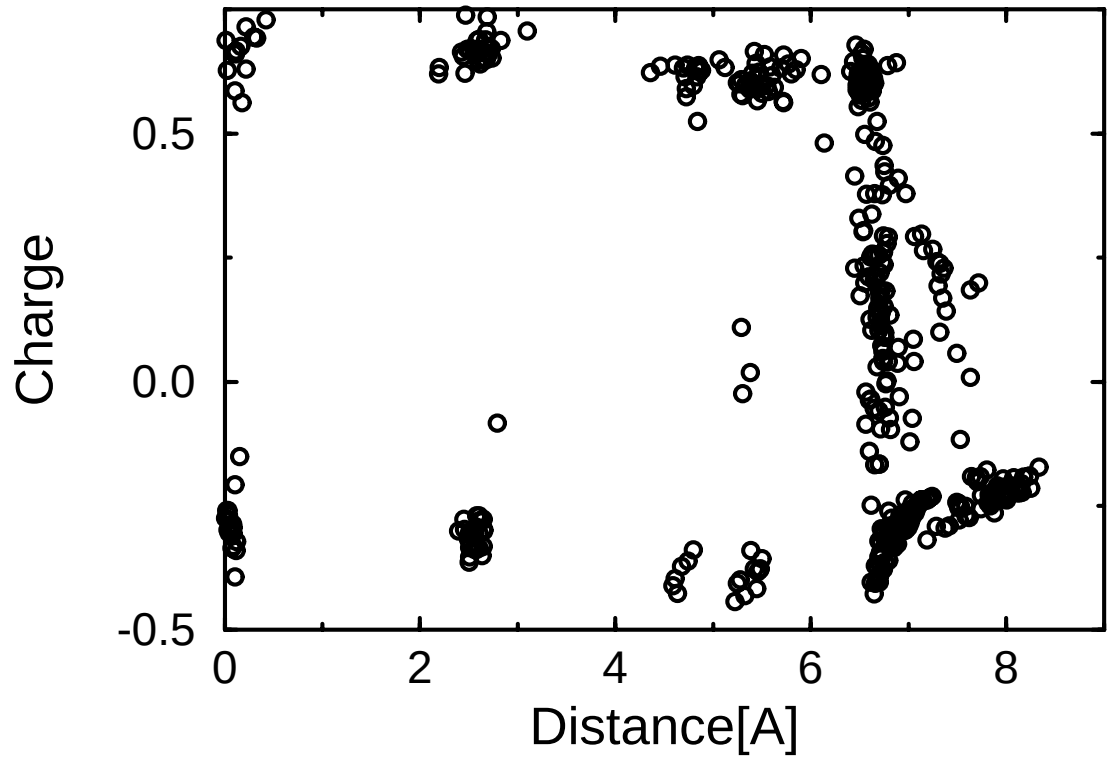

図 3.9: Li のグラファイト中心からの距離と電荷の関係 [12]

Li の吸着位置がグラファイト内部の場合、電荷が $+0.6$ と、-0.3 のあたりに分布している。また、端に吸着した Li は、$+0.6$ 〜 -0.3 まで、広く分布している。

[12] C54-Li-l-charge.eps

そこで、図 3.8、図 3.9より、グラファイト平面からの高さと、電荷の関係を調べ、図 3.10に示す。

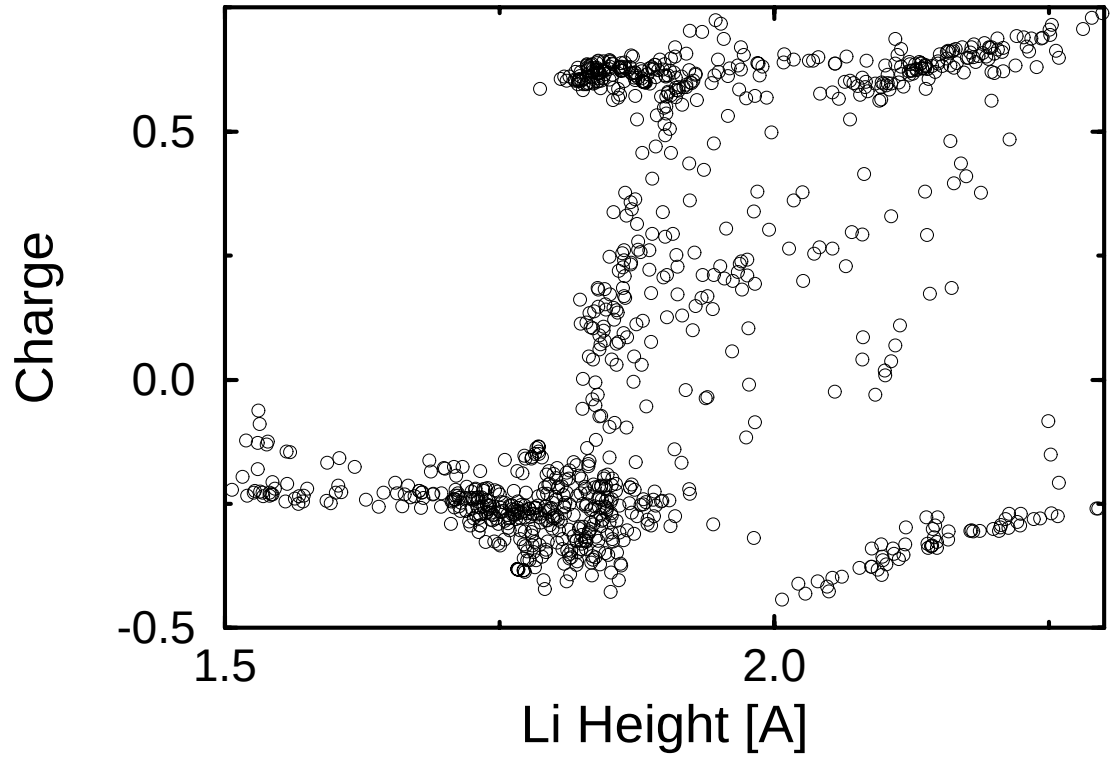

図 3.10: Li のグラファイト平面からの距離と電荷の関係 [13]

端の位置にある電荷は、 グラファイト平面からの高さ約 1.7 Å で、電荷が $+0.6$ と -0.3 で大きく変化することがわかる。つまり、グラファイト平面から離れた位置で安定する方が $+$ への電荷移動に適している。

また、内部に吸着した Li は 約 2.3 Å の高さで安定するが、電荷は $+$ と $-$ の原子が存在する。そして、個々の電荷がどのような原因で負電荷が決定されるかについて考察する。

[13] C54-Li-z-charge.eps

まず、グラファイトの内部の Li が負電荷をとる原因について、内部に吸着している Li 原子 2 個を例にして考える。近い位置で吸着している Li 原子が存在する場合、図 3.11の様に $2s$ 軌道が混成すると考えられる。それによって、図 3.12の様に Li_2 クラスターのエネルギー準位をつくる。このとき、低い方のエネルギー準位がグラファイトのエネルギー準位よりも低い場合、Li_2 クラスターに電子が流れ込み、負の電荷をもつと考えられる。

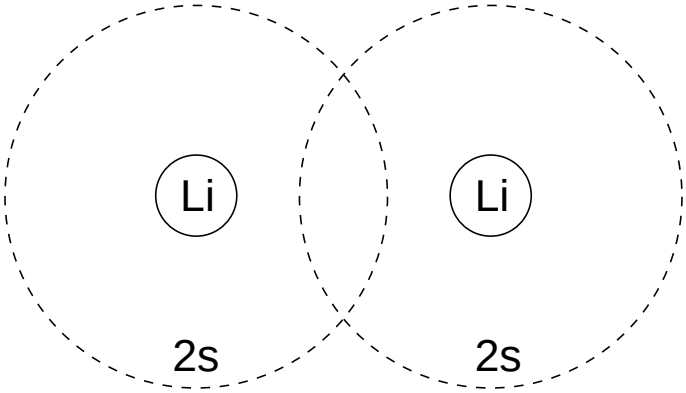

図 3.11: Li $2s$ 軌道の混成 [14]

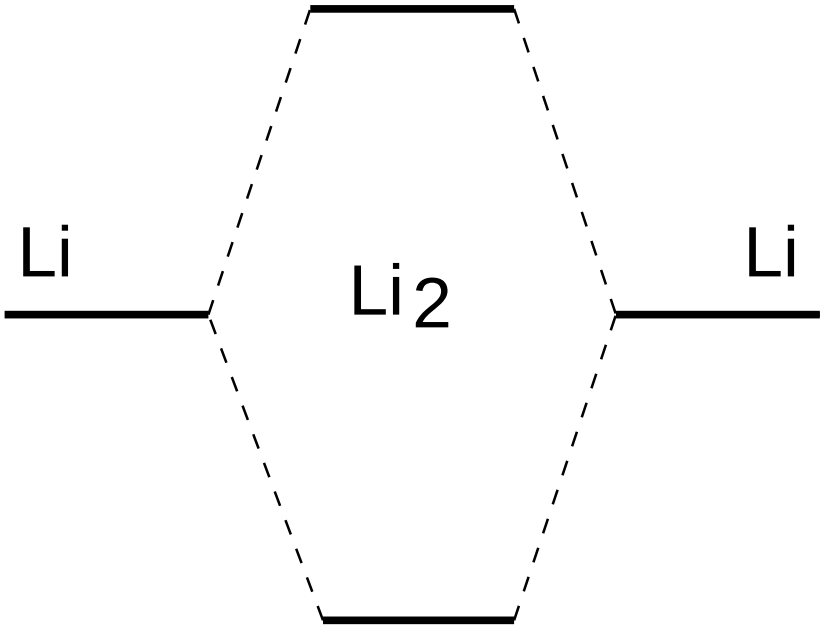

図 3.12: Li_2 クラスターのエネルギー準位 [15]

[14] Li2-atom.eps

[15] Li2-energy.eps

次に、グラファイトの端に吸着したLi が負電荷をとる原因について考察する。まず、グラファイトの端の部分の電荷の様子は、図3.13の様に、 H は 正の電荷 $(+0.11)$、結合している C は 負の電荷 (-0.09) を持っている。 Li ドープ数が一つの場合、図3.13の炭素原子に近い方の Li の位置に吸着し、正の電荷を持つ。

しかし、複数の Li がドープされた場合、 Li^+ 同士のクーロン反発によって、 Li が外側へ弾き出される。このとき、図3.13の様に水素原子の近くまで移動すると、 LiH_2 クラスターをつくり、 Li は負の電荷を持つ。

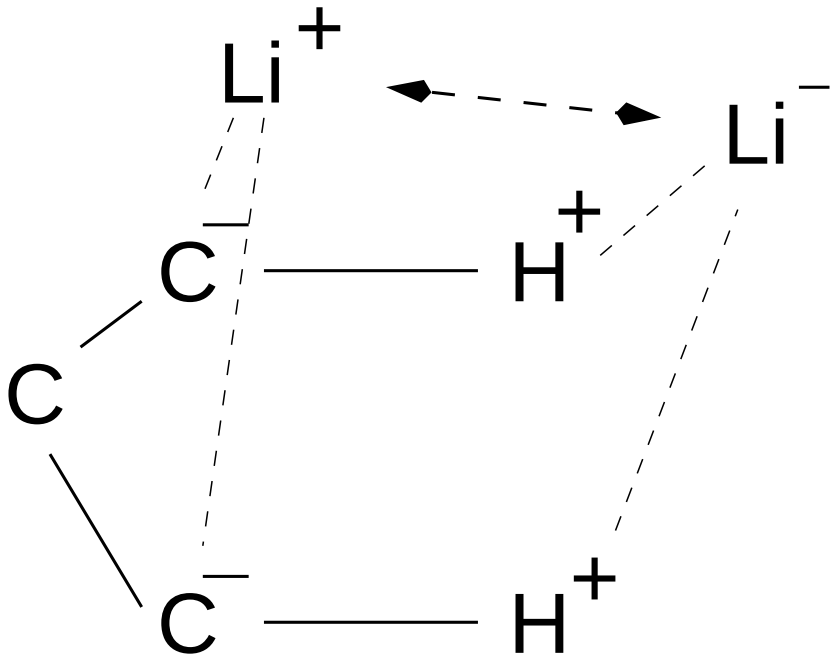

図 3.13: Li_2 クラスターのエネルギー準位 [16]

これらの考察は、 3.1.4の部分状態密度の結果とも一致する。

[16] Li-edge-charge.eps

3.1.3　総電荷移動量

系全体のグラファイトから Li への総電荷移動量を Li のドープ数との関係を図 3.14に示す。

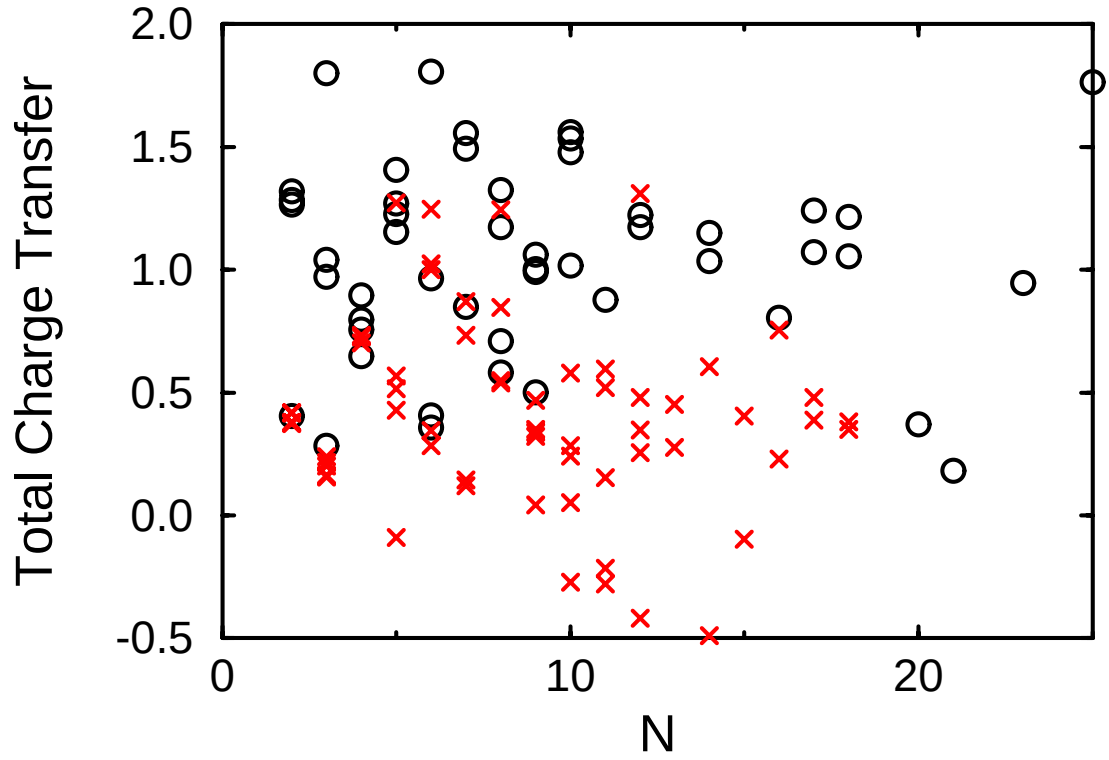

図 3.14: Li のドープ原子数と総電荷移動量の関係 [17]

ここで、○と×は次のような意味がある。

○：Li が一つでも内部に吸着した系

×：Li がすべて端に吸着した系

このグラフより、Li の総電荷移動量はドープ原子数 3 ～ 6 個で最大値をとっており、ドープ原子数を増加してもそれ以上の値はとっていない。

また、グラフの○が全体的に×より上にあることより、内部に吸着している系の方が、総電荷移動に有利であることがわかる。つまり、Li ドープグラファイトの Li への総電荷移動量は吸着位置に影響し、吸着原子数にはほとんど影響しない。

[17] C54-Li-total-charge.eps

次に、総電荷移動量と吸着エネルギーの関係を図 3.15に示す。この図での○と×の意味は、図 3.14と同じである。

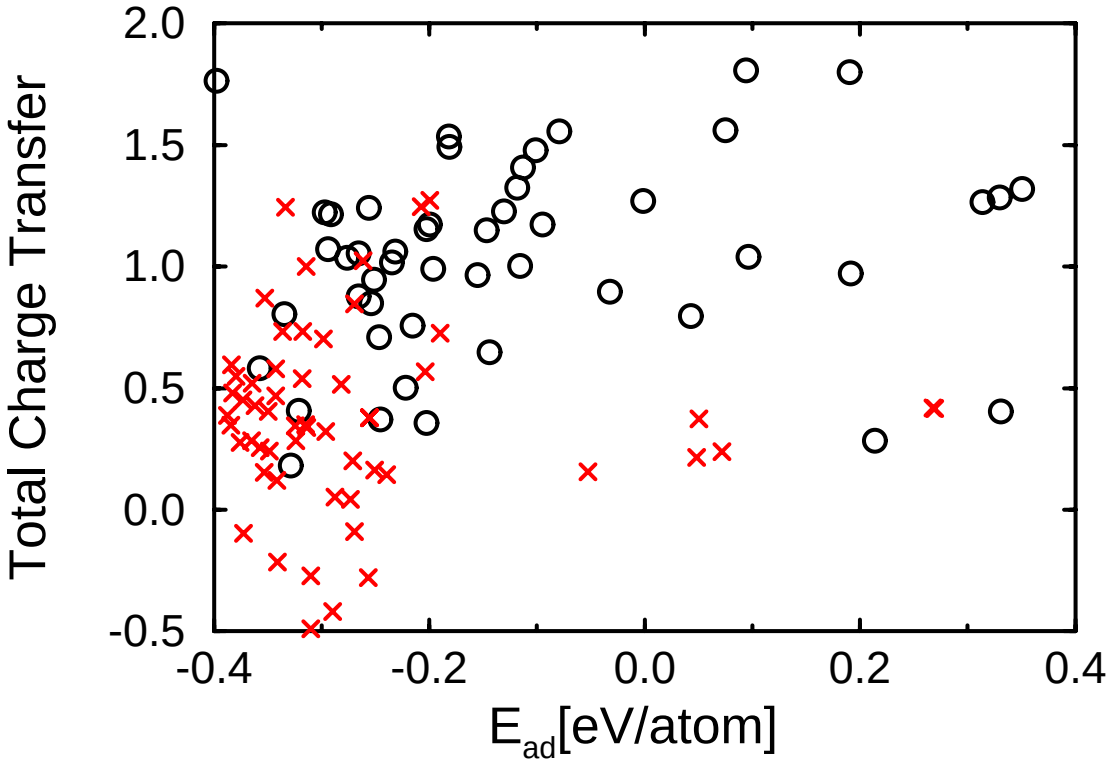

図 3.15: 総電荷移動量と吸着エネルギーの関係 [18]

グラフ上より、端のみに吸着した系は内部にも吸着した系に比べて比較的安定であることがわかる。つまり、内部に吸着するにはある程度のエネルギーが必要であることがわかる。

しかし端のみの吸着というのは、内部にも吸着するに比べてると、ドープ数に限界があるのは想像にたやすい。図 3.14でも、端のみの系では最大吸着数 18 個であるのに対して、内部に吸着させたものは 25 までの原子数が可能であった。

より多い総電荷移動量を得るためには、エッジサイトを飽和させ、内部のサイトに吸着されるまでドープされる必要がある。

[18] C54-Li-charge-energy.eps

3.1.4 状態密度

Li の電荷移動と電子状態の関係を示すために、正の電荷を持つ Li 原子と 負の電荷を持つ Li 原子のそれぞれの部分状態密度を求めた。比較するために図 3.1 の系における 4 個の Li 原子についての結果を、図 3.16、図 3.17に示した。

この系の HOMO (Highest Occupied Molecular Orbital) は約 -5 [eV] にある。そこで、図 3.16、図 3.17の場合の電子の詰まっている量を求め、表 3.1に示す。

	$2s$	$2px$	$2py$	$2pz$
正電荷 Li (図 3.16)	0.07	0.07	0.07	0.09
負電荷 Li (図 3.17)	0.94	0.12	0.08	0.18

表 3.1: Li 原子の電子の占有量

正電荷の Li は、軌道上に電子が平均して 0.08 ほどしか占有していないのに対して、負電荷の Li は特に $2s$ 軌道に電子が入っているのがわかる。

Li $2s$ 軌道が十分にひろがっていて、他の Li の $2s$ 軌道や炭素の分子軌道と共有結合をつくるとこの結合軌道のエネルギーが HOMO のエネルギーより低くなり電子が占有される様になる。もう一つの機構は、端の状態において、Li と H との構造と関連している。

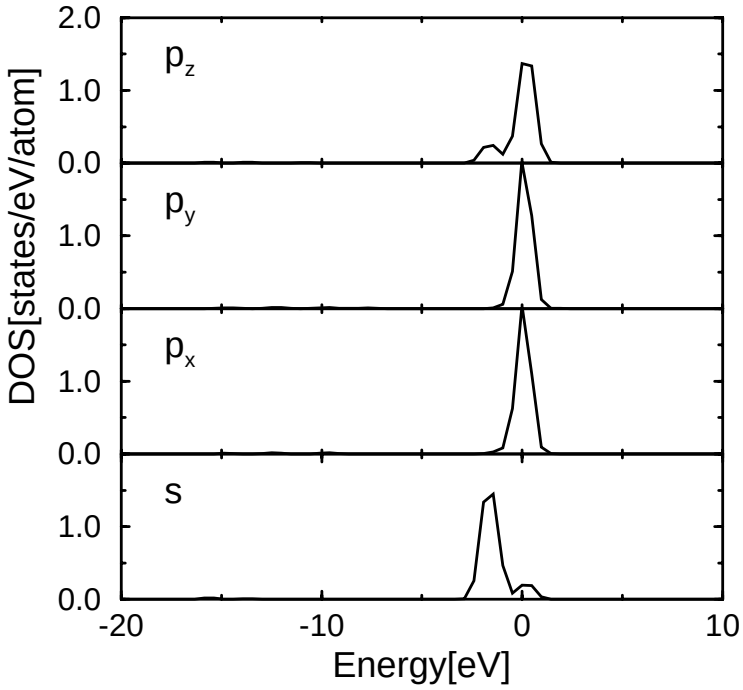

図 3.16: 正電荷を持つリチウム原子の状態密度 [19]

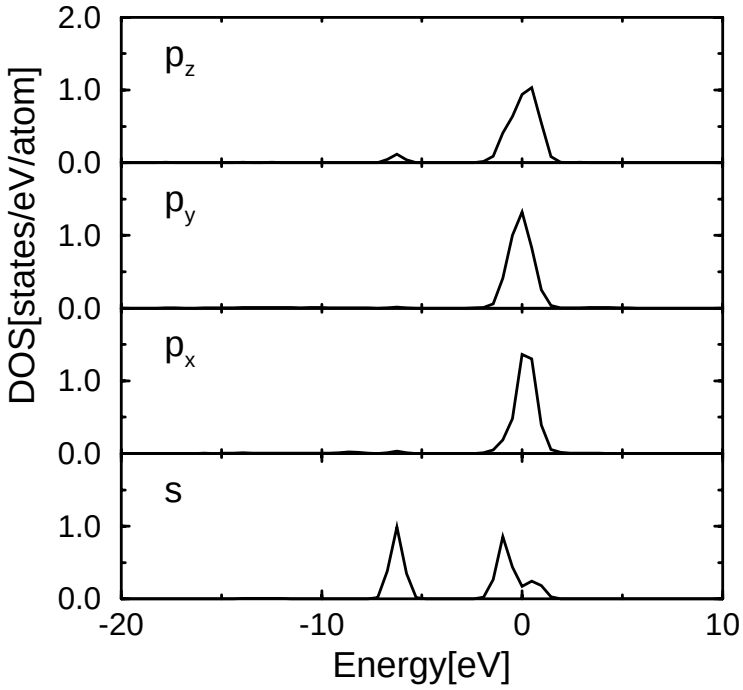

図 3.17: 負電荷を持つリチウム原子の状態密度 [20]

[19] Li+DOS.eps

[20] Li-DOS.eps

3.2　フッ素ドープ

コロネンと呼ばれる $C_{24}H_{12}$ 微小グラファイトに F をドープしたときの構造最適化と電子状態の計算より求められた結果と、それに対する考察を述べる。

3.2.1　ハロゲン原子の比較

$C_{24}H_{12}$ のグラファイトにハロゲン原子 (F、 Cl、 Br) を 1 原子ドープし、構造最適化と電子状態の計算を行なった。

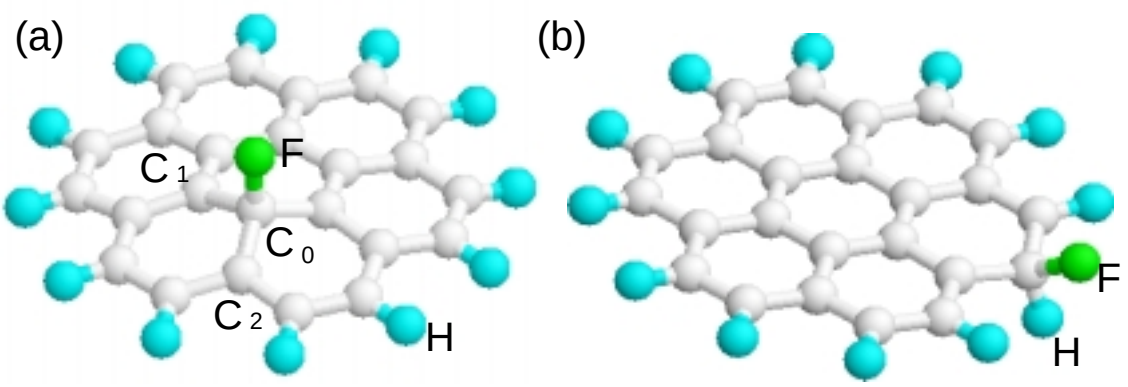

図 3.18: (a) 内部に結合 : (b) 端に結合 [21]

F を C ドープした場合の構造である図 3.18より、グラファイト内部にドープした場合、グラファイトの骨格が変形しているのが分る。また、結合した炭素原子 (図 3.18(a) の C_0) は sp^3 構造を取っていることが分る。

実際、内部に結合した (a) の場合について、他のハロゲン原子 (Cl、 Br) との構造パラメータと電荷の比較を行なった。結果を表 3.2に示す。表内の X はハロゲン原子を示す。

	F	Cl	Br
電荷	−0.15	−0.07	−0.17
X−C_0 [Å]	1.39	1.87	2.13
∠XC_0C_1 [°]	107	104	98
∠$C_1C_0C_2$ [°]	112	115	118

表 3.2: ハロゲン原子の構造の比較

[21] halogen-structure.eps

構造変化は、F ドープの場合が一番大きく、Cl 、Br になるにしたがって小さくなっている。つまり、イオン半径が小さい原子ほど構造変化が大きいということが分った。このことは、イオン半径が大きいとグラファイトの骨格を変形して sp^3 の共有結合エネルギーをかせぐより、他のまわりの炭素原子と結合したほうがエネルギー的に有利であることを示している。特にヨウ素の場合には、変形がなく $C_{24}H_{12}$ のクラスターでは安定な構造をとらなかった。ここで、安定な構造をとらないとは、MOPAC でエネルギー勾配ノルム (GNORM) が 0.5 以下の構造をとらないことである。しかも、F ドープ時の炭素周りの結合角は sp^3 結合の角度 109 ° に非常に近い値になっている。

ここで、それぞれの原子の van der Waals 半径を表 3.3に示す。

	F	Cl	Br	I
van der Waals 半径 [Å]	1.35	1.80	1.95	2.15

表 3.3: ハロゲン原子の van der Waals 半径 [11]

この値と、表 3.2の結合距離はほぼ等しくなっており、原子半径と構造変化に関係があると考えられる。

次に、ハロゲン原子の結合している炭素原子 C_0 の部分状態密度を比較した。

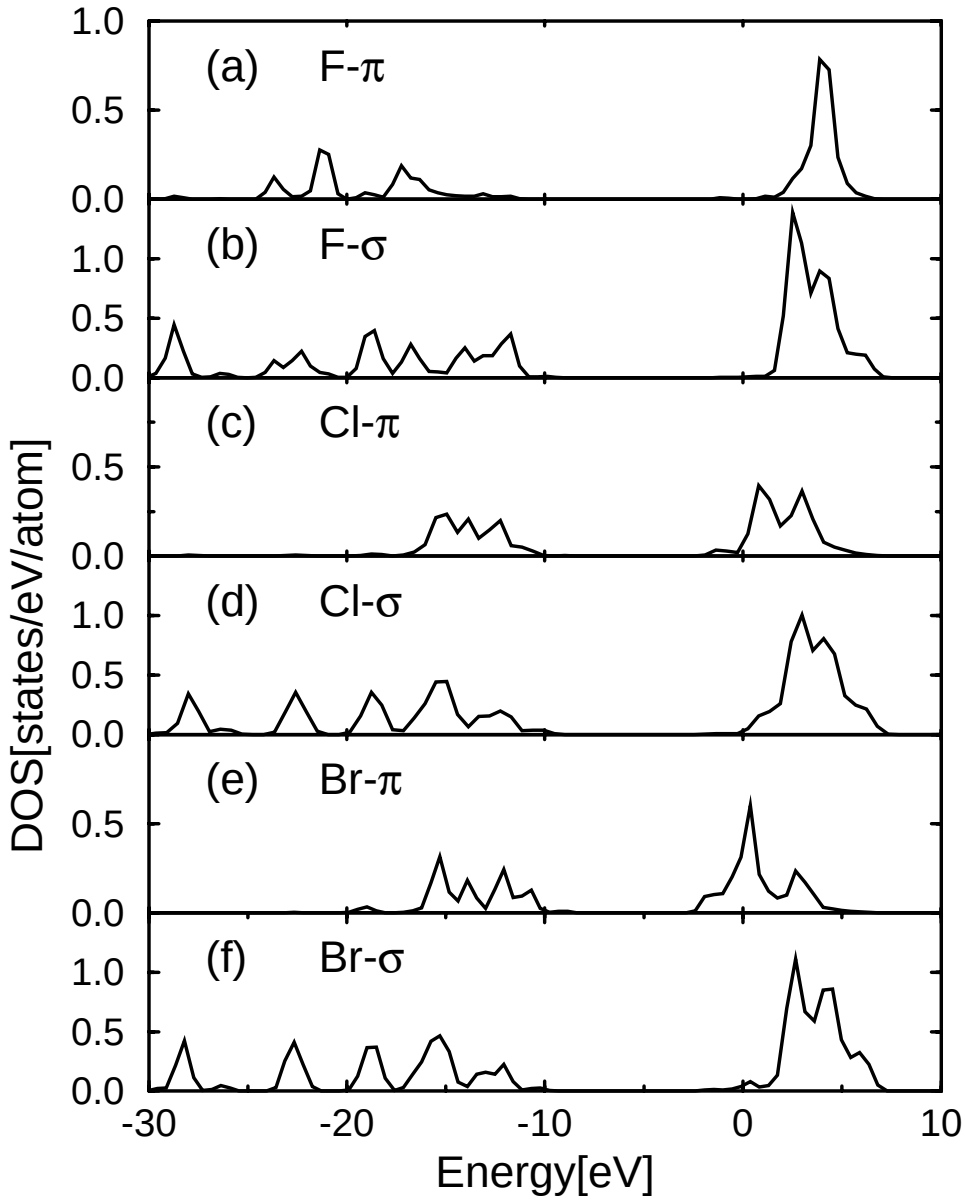

図 3.19: C_0 原子の状態密度 [22]

図 3.19のグラフは、

(a) F と結合した C_0 のπ軌道　(b) F と結合した C_0 のσ軌道
(c) Cl と結合した C_0 のπ軌道　(d) Cl と結合した C_0 のσ軌道
(e) Br と結合した C_0 のπ軌道　(f) Br と結合した C_0 のσ軌道

[22] halogen-C0-DOS.eps

の状態密度を示している。

F を結合した C_0 のπ軌道と、σ軌道はほぼ同じエネルギーを持っており、sp^3 混成軌道を形成していることが分る。それに対してイオン半径の大きい Br を結合した C_0 のπ軌道と、σ軌道は分れており、これは、sp^2 のグラファイトに近い傾向である。また、Cl を結合させた場合のものは、両者の中間的な傾向を示している。

また、C_0 において、変化の大きかったπ軌道の部分状態密度の変化を調べるため、図 3.18の C_1 における、π軌道の状態密度の計算を行った。

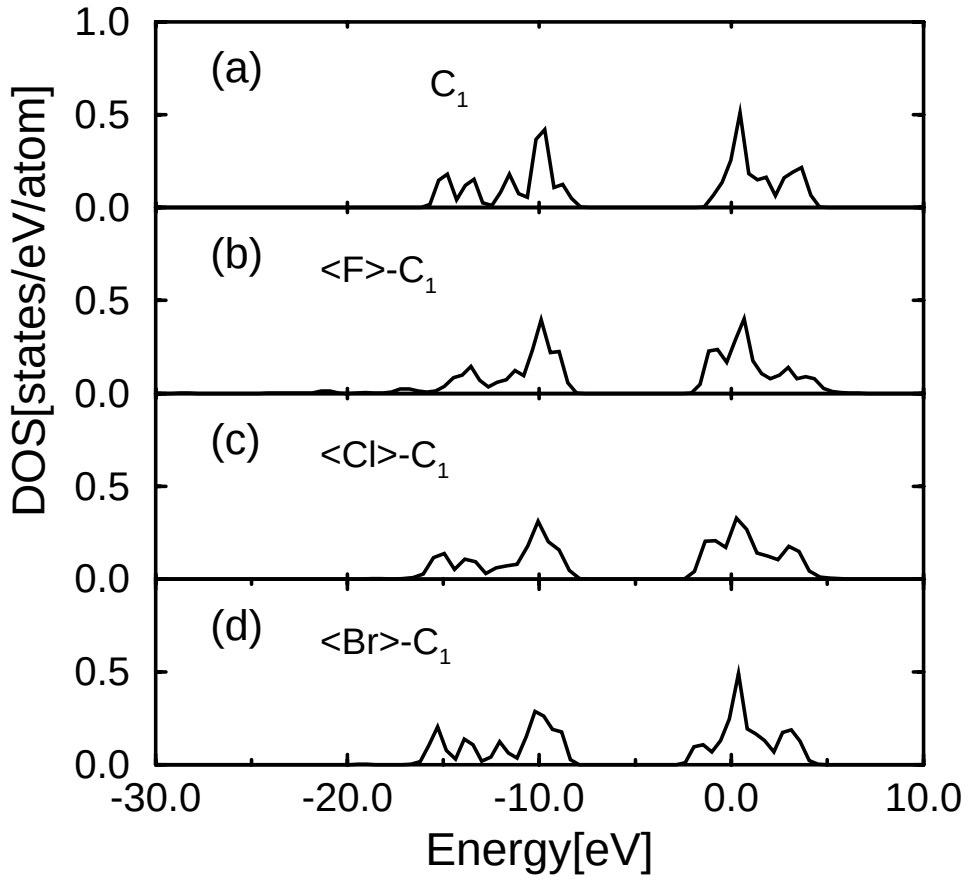

図 3.20: C_1 原子の状態密度 [23]

図 3.20のグラフは、

(a)　ノンドープの C_1 のπ軌道　　(b)　F と結合した C_1 のπ軌道

(c)　Cl と結合した C_1 のπ軌道　　(d)　Br と結合した C_1 のπ軌道

の状態密度を示している。

[23] halogen-C1-DOS.eps

微妙な変化だが、 -15[eV] 付近の変化を比べると、ノンドープのπ軌道と比べて、Br をドープしたときが変化が一番大きく、 F ドープしたときの変化が一番小さい。

つまり、ハロゲン原子においてイオン半径が小さい原子ほど強い sp^3 結合を取る。これは、フッ素のようなイオン半径の小さい原子では、グラファイトとの結合が一つ炭素原子との強い結合で、となりの炭素原子との結合は弱いということが考えられる。この様に、 F の結合は一つの炭素原子に十分に局在していることが理解できる。逆に、臭素のようなイオン半径の大きい原子との結合では、一つの炭素原子だけではなく、グラファイト全体との弱い結合によって、結合していると考えられる。

次に、それぞれの原子において図 3.18のような結合位置とエネルギーの関係を調べた。その結果を表 3.4に示す。

結合位置	F	Cl	Br
内部の C[eV]	-1.50	-0.69	-0.50
端の C[eV]	-2.41	-1.35	-0.96

表 3.4: ハロゲン原子の結合位置と吸着エネルギーの関係

全てのハロゲン原子において、グラファイト端の炭素原子に結合する方がグラファイト内部に結合するより安定であることが分る。これは、ドープ時に端からドープされやすいということを示している。

また、活性化エネルギーを計算した。これは、内部の炭素原子に結合したハロゲン原子が隣の炭素原子に移動するために必要なエネルギーとして計算した。これは Li の場合 (図 3.3)、 A 方向に一つ格子ベクトル進んだときのバリヤーの大きさを活性化エネルギーとして評価したものである。

ハロゲン原子	F	Cl	Br
活性化エネルギー	1.810	0.667	0.249

表 3.5: ハロゲン原子の活性化エネルギー

フッ素の活性化エネルギーは非常に高く、一度結合したあと他の炭素原子へと移動することはほとんど無いと考えてよい。

Cl 、 Br の活性化エネルギーはフッ素に比べると小さいが、図 3.4、図 3.5のリチウムの障壁エネルギー 0.12eV と比べると大きい。つまり、ハロゲン原子の特長として、グラファイトとの結合は比較的強い結合であるといえる。

3.2.2　2 原子ドープ

次に、$C_{24}H_{12}$ のグラファイトにフッ素原子を 2 個を考え得る全ての場合の位置関係にてドープし、構造最適化を行なった。ここで便宜上、グラファイト上の炭素原子を水素終端されている炭素原子と、それ以外の炭素原子とで分け、それぞれ「端」、「内」と示す。この場合、大きく分けて 3 つの場合が考えられ、表 3.6にそれぞれの構造の種類数を示す。

位置関係	内・内	内・端	端・端
種類数 [個]	10	11	9

表 3.6: ドープ原子数 2 個の場合の位置ドープ位置の場合の数

下に、内・内と、端・端の構造の例を示す。

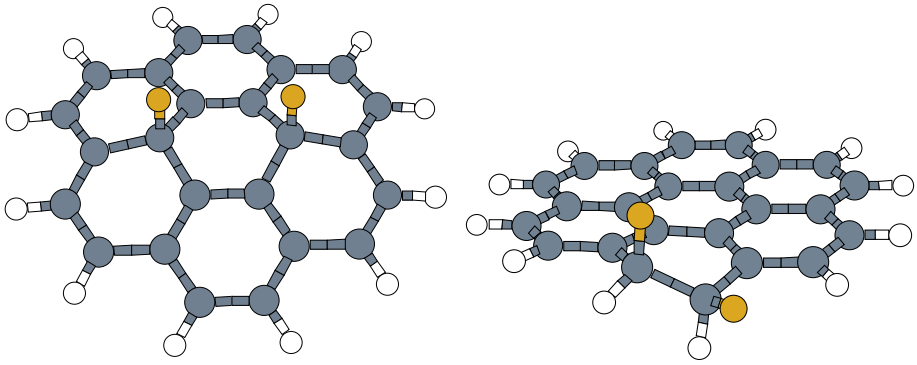

図 3.21: フッ素原子 2 個ドープした構造の例 [24] [25]

そして、その時の吸着エネルギーを図 3.22に示す。このグラフでは、図 3.21の様に 2 個のフッ素原子はグラファイト平面に対して同方向にあるもののみを示している。

ここで、計算手法の中で、 UHF で triplet(s=1) を用いて同様に計算したところ、フッ素の結合した炭素原子の位置関係が偶数近接である場合、 triplet の方が安定であり、奇数近接の場合では singlet(s=0) の方が安定であった。しかし、この時の構造に違いはほとんど見られなかった。

[24] F2-in.eps

[25] F2-edge.eps

そして、この安定だった方の吸着エネルギーを、図 3.22 に示す。このグラフの横軸は、フッ素原子の結合している炭素原子同士の N 次近接を示している。

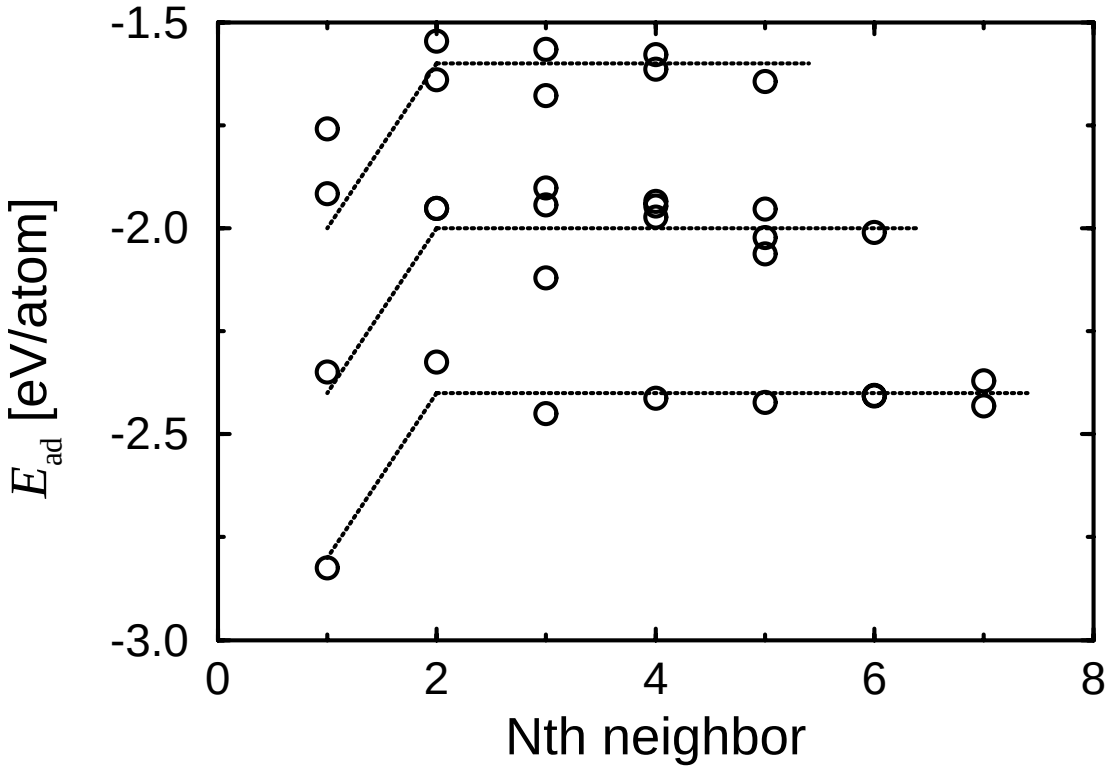

図 3.22: F の結合位置関係と吸着エネルギーの関係 [26]

グラフ上の 3 本の点線は、同じ結合の位置関係の結果についておおよその値を示したものである。それぞれ上から、「内・内」、「内・端」、「端・端」の結合についての結果である。第 1 近接の場合には、他のものに比べて約 0.4[eV] 安定である。これは、フッ素同士に弱い共有結合ができ、そのためにエネルギーが下がると考えられる。逆に考えると、最近接以外では結合したフッ素原子同士はほとんど影響しないといえる。このことは、部分状態密度の結果と一致している。

また、2 原子内部の炭素原子に結合している場合に比べて、端の炭素原子に結合した方が、フッ素原子 1 個あたり 0.4 [eV] 安定である。これは、フッ素原子を 1 個ドープした場合と同様の結果であり、多原子をドープした場合でも同様の結果であると考えられる。つまり、大量のフッ素原子をドープした場合、端の炭素原子から結合していると考えられる。

[26] F2-energy.eps

3.2.3 状態密度

高井らの東工大の実験では、炭素フッ素濃度比は最大約 1.2 と炭素原子数以上にドープしている。そこで、グラファイトに対する最大ドープ原子数を考える。

グラファイト端の炭素には 2 個のフッ素原子を結合させることができる。また、内部の炭素原子には 1 個の炭素原子が結合可能である。計算に用いた炭素原子数 24 個のグラファイトと、この実験のモデルのグラファイト (図 1.8) について、フッ素原子の最大結合数を表 3.7に示す。

	d[Å]	最大ドープ数 [個]			F/C
		端	内部	合計	
C_{24}	7.5	24	12	36	1.5
C_{216}	30	72	180	252	1.17

表 3.7: グラファイトへのドープ最大量

炭素原子数 216 個のグラファイトでの最大ドープ量の理論値は表 3.7の様に、実験値の 1.2 に近い値をとる。この誤差は、グラファイトの構造には同じ炭素原子数でも細長い等、端の炭素原子数の異る構造もあるからであり、このモデルがほぼ正しいといえる。

計算に用いた炭素原子数 24 個のグラファイト (コロネン) では、炭素フッ素濃度比は、 1.5 と実験値の 1.2 と比べて大きいが、微小グラファイトとしての特徴を持っており、計算としては有効な母材と考えられる。

次に、その最大ドープ数を結合させた計算を行ない、その構造を、図 3.23に示す。このとき、それぞれのフッ素原子の上下関係が、交互になるのが安定であり、そのように結合させた。このときの 1 原子あたりの吸着エネルギーは、 -3.56 [eV/atom] であり、安定な構造である。

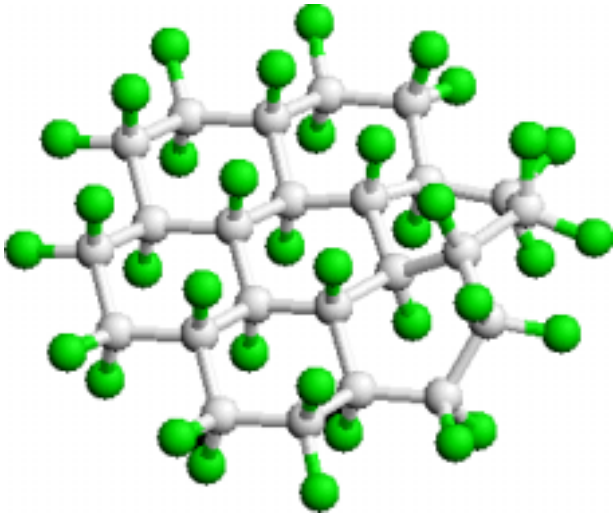

図 3.23: $C_{24}F_{36}$ の構造 [27]

このときの結合角 (108 ～ 112 °)、結合長 (1.58 ～ 1.62Å) はそれぞれ、ダイヤモンド構造 (109.5 °、 1.54Å) に近い値になっており、全ての炭素原子が sp^3 構造になっているのがわかる。この状態密度を、図 3.24に示す。 (a) は炭素全体、 (b) はフッ素全体、 (c) は分子全体の 1 原子あたりの状態密度を示している。

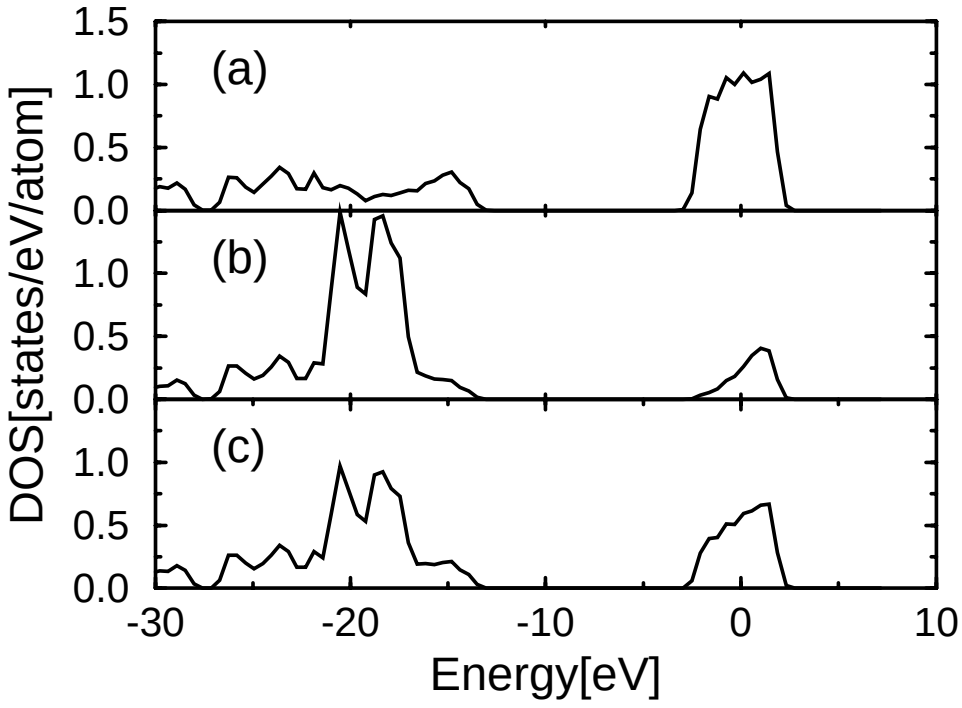

図 3.24: $C_{24}F_{36}$ の状態密度 [28]

[27] A-C24F36.eps

[28] C24F36-DOS.eps

(a)、(b) とも、ほぼ同じ状態をとっており、完全に sp^3 の共有結合になっている。この様に、フッ素を加えるとグラファイトを sp^3 化することができる。このことを利用して、ダイヤモンド核を形成することも可能であると考えられる。

3.2.4　電荷移動

フッ素原子を 1 個づつドープしていき、そのときのグラファイトからフッ素原子への電荷移動について調べた。このとき用いたグラファイトは、ダングリングボンドを全て水素終端させたもので、フッ素原子はグラファイト端には最大 12 個である。

$C_{24}H_{12}$ に F をドープしたときの C と F の結合距離を、図 3.25に示す。

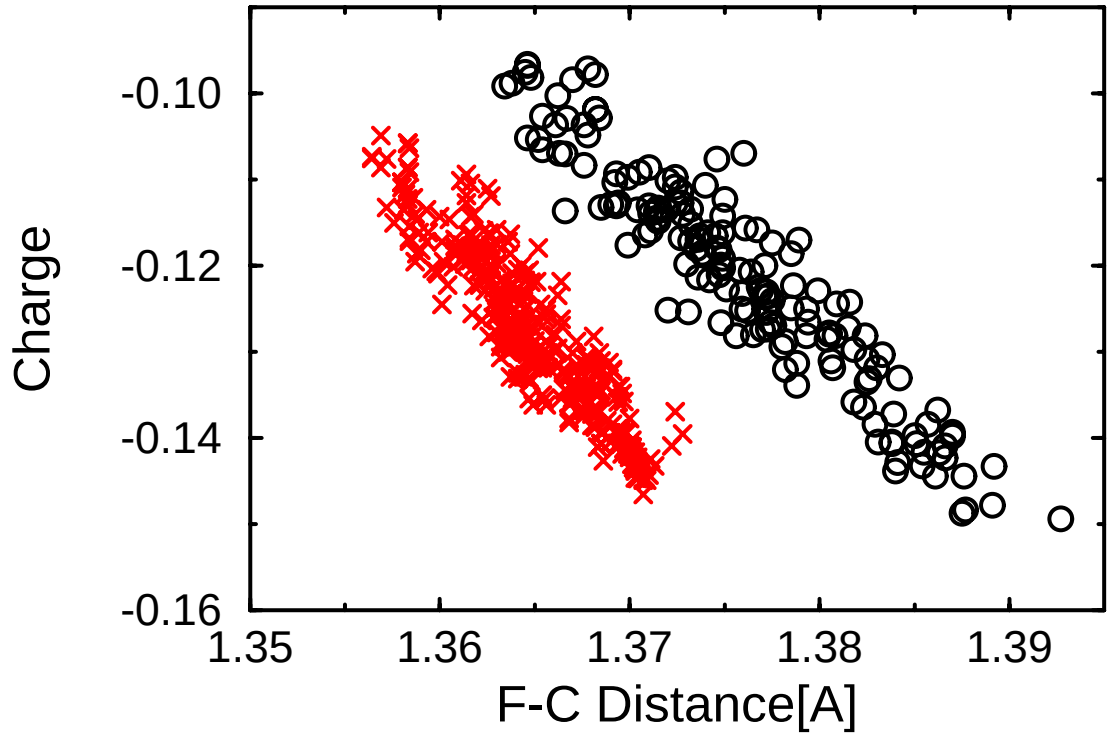

図 3.25: 炭素 – フッ素の原子間距離と電荷の関係 [29]

このグラフは結合距離が大きく 2 つに分かれている。これは、結合距離が短い方の集まり (×) がグラファイト端に結合した F の結合距離、長い方 (○) が内部の炭素原子に結合した集まりである。

この図より、○×いずれの場合でも、F−C 結合距離が増加すると、電荷移動が大きくなることがわかる。つまり、結合距離の増加によってイオン性が増したことを示

[29] CFdistance-charge.eps

している。しかし、 Li の時のように、イオンの符号が変わることはない。これは、共有結合、イオン結合とも電子の移動する方向が同じであることによると考えられる。

ここで、 F の結合距離と電荷について、それぞれドープ原子数との関係を調べ、図 3.26、図 3.27に示す。それぞれのグラフの記号の意味は、

○：内部に結合した原子の平均

×：端に結合した原子の平均

である。

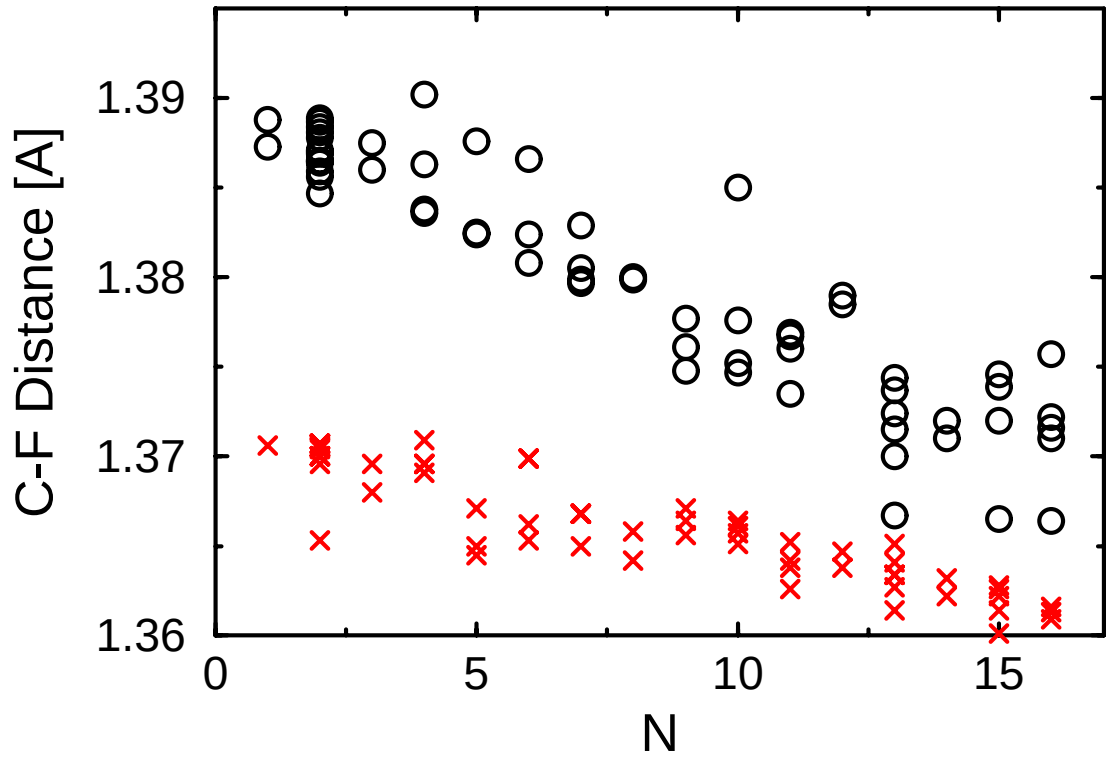

図 3.26: ドープ原子数と平均結合距離の関係 [30]

[30] C-F-length.eps

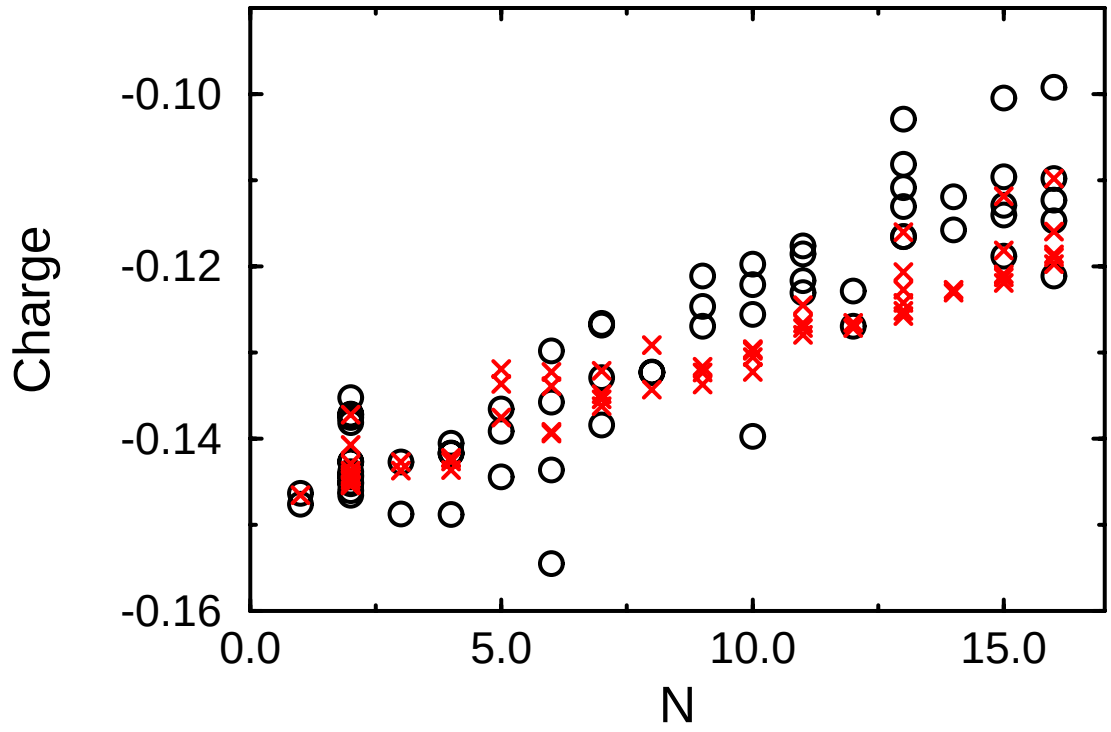

図 3.27: ドープ原子数と平均電荷の関係 [31]

図 3.25に示す様に C−F の結合距離は結合位置に影響するが、平均電荷は、結合位置にはほとんど関係ない。また、F 結合原子数が増えると、C−F 結合の平均距離は短く、F の平均電荷は減少していく。結合距離が短くなっていくのは、それぞれ炭素原子との結合が強くなっていると考えられる。また、電荷の減少については、総電荷移動が少しずつ飽和していると考えられる。そこで次に、グラファイトから F への総電荷移動量を図 3.28に示す。

[31] N-charge.eps

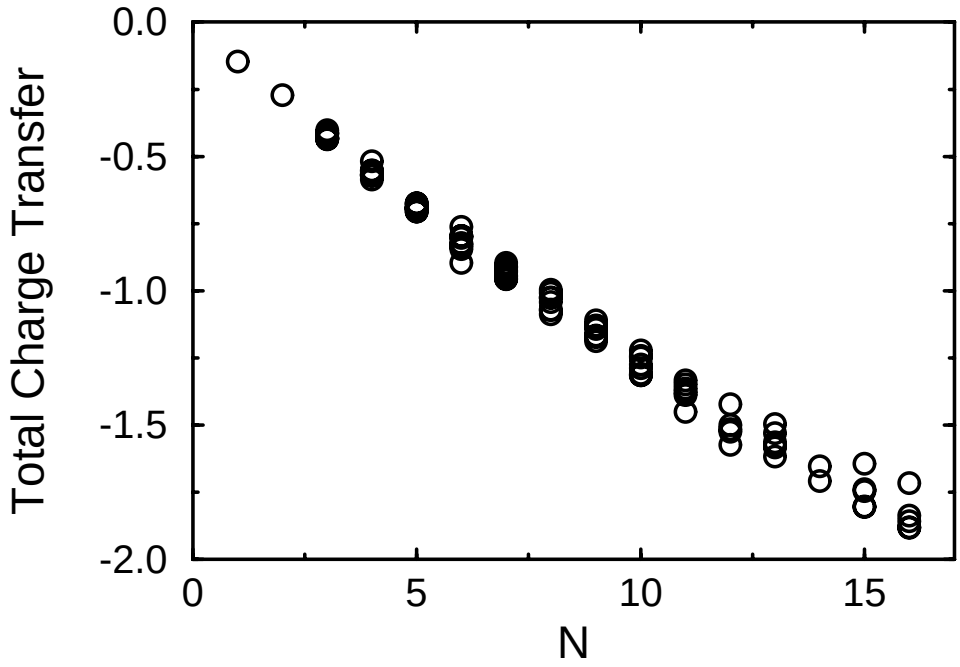

図 3.28: フッ素への総電荷移動量 [32]

総電荷移動量は、ほぼ平均してマイナスへ増加しており、飽和現象はまだ見られない。これは、図 3.14 の Li の結果とは大きく異なる結果である。つまり、 F 原子は Li に比べると ドープ原子同士の影響が非常に小さいということがわかる。

3.2.5 動的反応座標 (DRC) 計算

構造最適化の計算では、ドープ位置は自分で自由に決めることができる。しかし、実際の反応でのドープ位置の確率等はわからない。そこで、実際の反応の様子を確認するために、動的反応座標 (DRC) を用いて、フッ素ドープの計算を行なった。

[32] F-TotalCharge.eps

F 分子をグラファイトのエッジ部分に、グラファイト平面に対して垂直方向からぶつけたときの計算結果を図 3.29に示す。

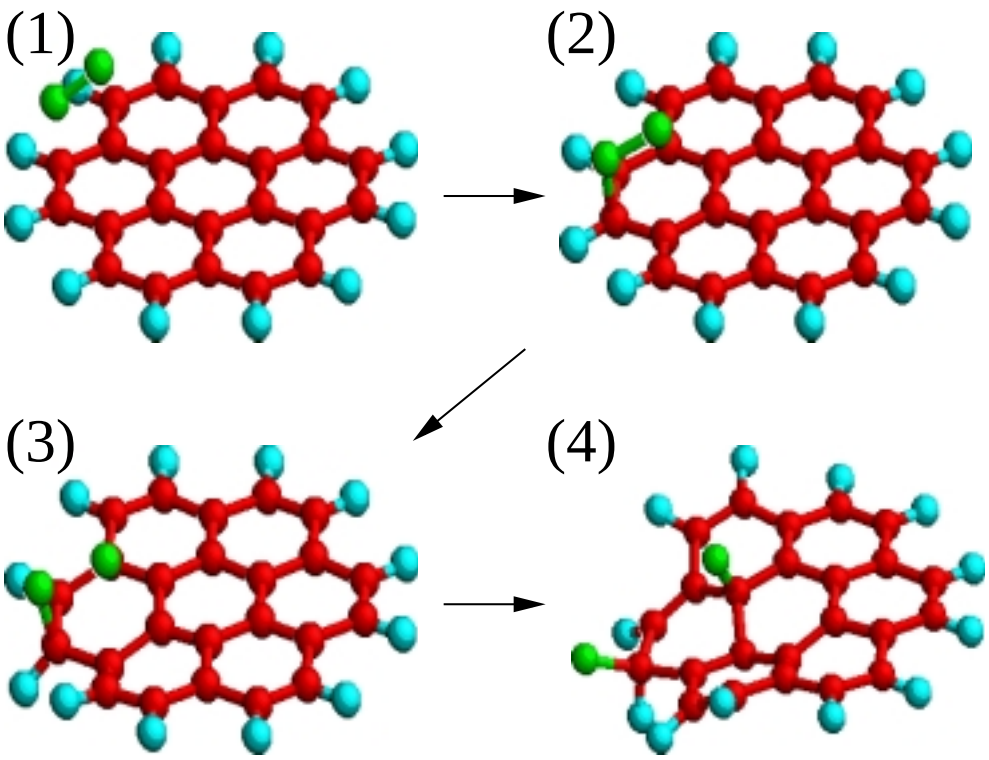

図 3.29: フッ素分子をグラファイト上空からぶつけた DRC 計算 [33]

1. F 分子を上空からドープ
2. F 分子がエッジの炭素原子と結合
3. F 分子の結合が切れる
4. 離れた F 原子が近くの炭素原子と結合

この時与えた運動エネルギーは 260 [kcal/mol] であるが、これより低いエネルギーのときは反応は起こらない。また、内部の炭素原子へぶつけたときも反応しない。

これより、F 分子のドープの場合、反応の障壁エネルギーが高く、内部との反応はより高い。しかし、分子の結合が切れ、F 原子となった場合は、内部の炭素原子とも比較的自由に反応できる。

しかし、このままドープさせると、終端の水素があり、図 3.23のような構造をとることができない。そこで、水素に直接ぶつける反応の計算もおこなった。

[33] DRC-F2.eps

フッ素分子をグラファイトのエッジ部分に、水平方向からぶつけたときの計算結果を図 3.30に示す。

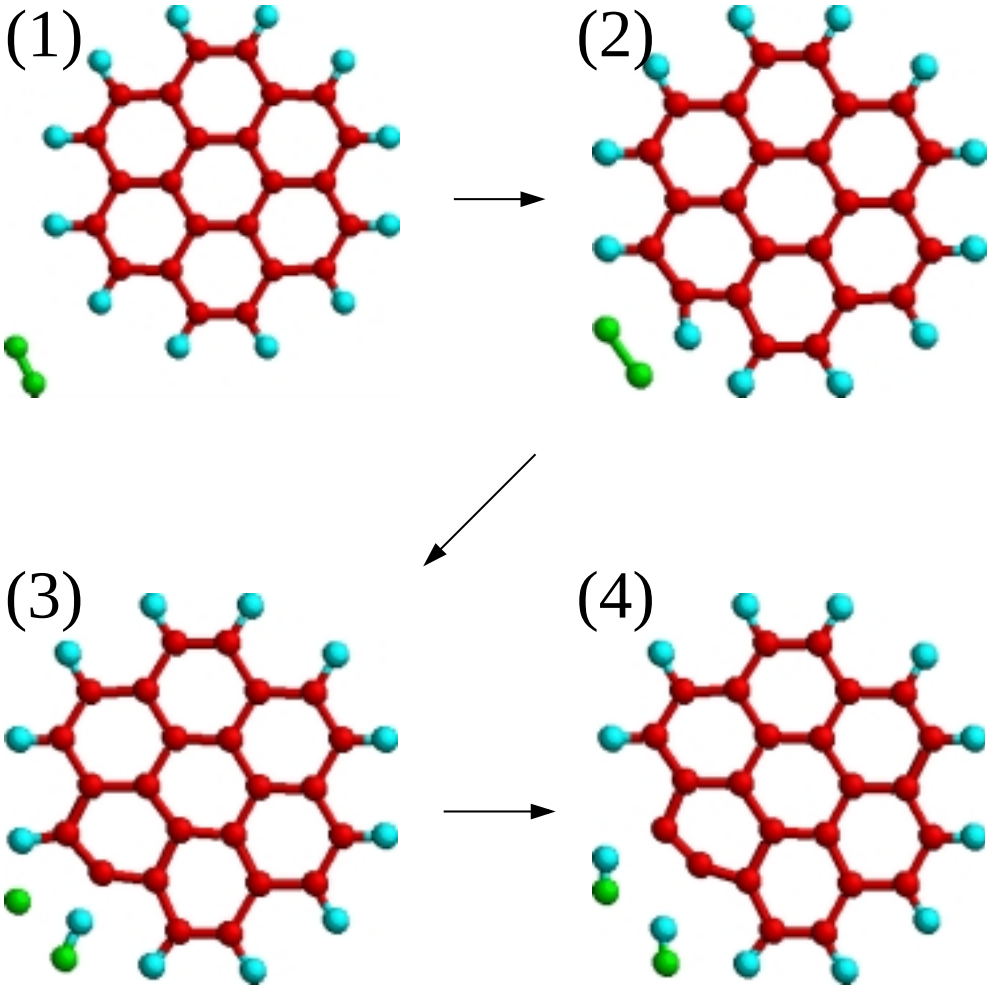

図 3.30: フッ素分子を終端の水素にぶつけた DRC 計算 [34]

1. F 分子を水平方向からドープ
2. F 分子に水素原子が反発
3. HF の構造をとり、 F 分子の結合が切れる
4. もう一方の F も H と反応、振動しながら離れる

このときの F 分子の運動エネルギーは、 120 [kcal/mol] である。この反応より、 F ドープによって、グラファイトに終端されている水素原子などが取り去られることが可能であることがわかる。このことより、図 3.23のように、すべての炭素原子に F をドープすることが可能である。

[34] DRC-F1.eps

3.3　ヨウ素ドープ

1.4に示したように、ピッチと呼ばれる炭化水素化物にヨウ素ドープによって通常よりも低温 (300℃) でグラファイト化が進むことが、東工大の宮嶋らによって報告された。

そこで、I がグラファイト化におけるダングリングボンドの作成機構において、どの様な役割を担っているのか、動的反応座標 (DRC) の計算手法により研究する。

3.3.1　電荷移動錯体

宮嶋らは、このグラファイト化に対して次のようなモデルを示した。まず、ドープされたI は、図 3.31の様に、I_2 〜 I_3 のクラスターとしてグラファイトにドープされる。この時、I クラスターは負の電荷を帯び、グラファイトは正の電荷を帯びると仮定する。そして、H は通常正の電荷をとるため、H とグラファイト間でクーロン反発が起こり、終端の H 原子の離脱が容易になるのではないかというモデルを考えた。

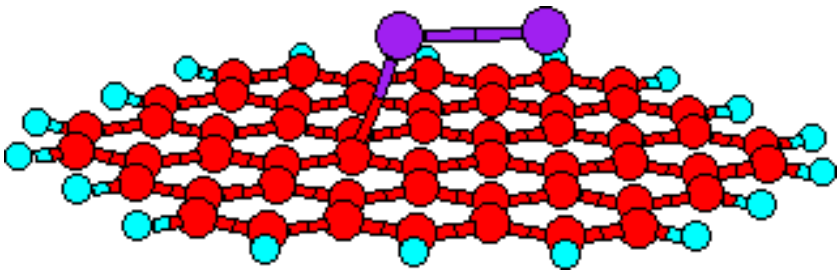

図 3.31: 宮嶋らによるI ドープモデル [35]

しかし、このような構造を MOPAC による構造最適化の計算を行っても安定な構造として求めることができない。つまり、図 3.31のような構造は存在せず、これとは異なるドープ構造が存在していると考えられる。

3.3.2　グラファイト化

ここで、グラファイト化とは、微小なグラファイトが結合しより大きいグラファイトに成長することである。そこで、水素終端のあるグラファイトと、水素終端のないグラファイトのそれぞれ同じもの同士と、異なる種類でのグラファイトを衝突させ、グラファイト化が進むか検証した。その結果を表 3.8に示す。

[35] C54H18I2.eps

水素終端の有無	有・有	無・無
反応の様子	反応無し	炭素同士の結合有り

表 3.8: グラファイト同士の衝突の水素終端の有無による反応の違い

このことより、ダングリングボンドの作製がグラファイト化の十分条件として考えられる。そこで以下から、ヨウ素ドープによるダングリングボンドの生成条件を求める。

3.3.3　ヨウ素ドープ

他のハロゲン原子 (F、 Cl、 Br) と同様に、グラファイトの炭素原子上空にヨウ素原子を置き、構造最適化を行なう。他のハロゲン原子の場合は、図 3.18の様に結合し、結合角等は表 3.2の様になる。しかし、ヨウ素原子をドープしても炭素との sp^3 結合する構造は安定に存在しない。

また、図 3.30のように、F ドープでもダングリングボンドは作られる。しかし、 F は炭素原子とも結合してしまい、グラファイト化には適さないと考えられる。つまり、ヨウ素は炭素原子には結合が弱く水素を取り去る反応を積極的におこなう反応がおきていると考えられる。

そこで、 DRC 計算によって、ヨウ素原子に運動エネルギーを与え、半強制的に反応させることを試みる。

ヨウ素分子をグラファイトのエッジ部分に、水平方向からぶつけたときの構造変化を図 3.32に示す。

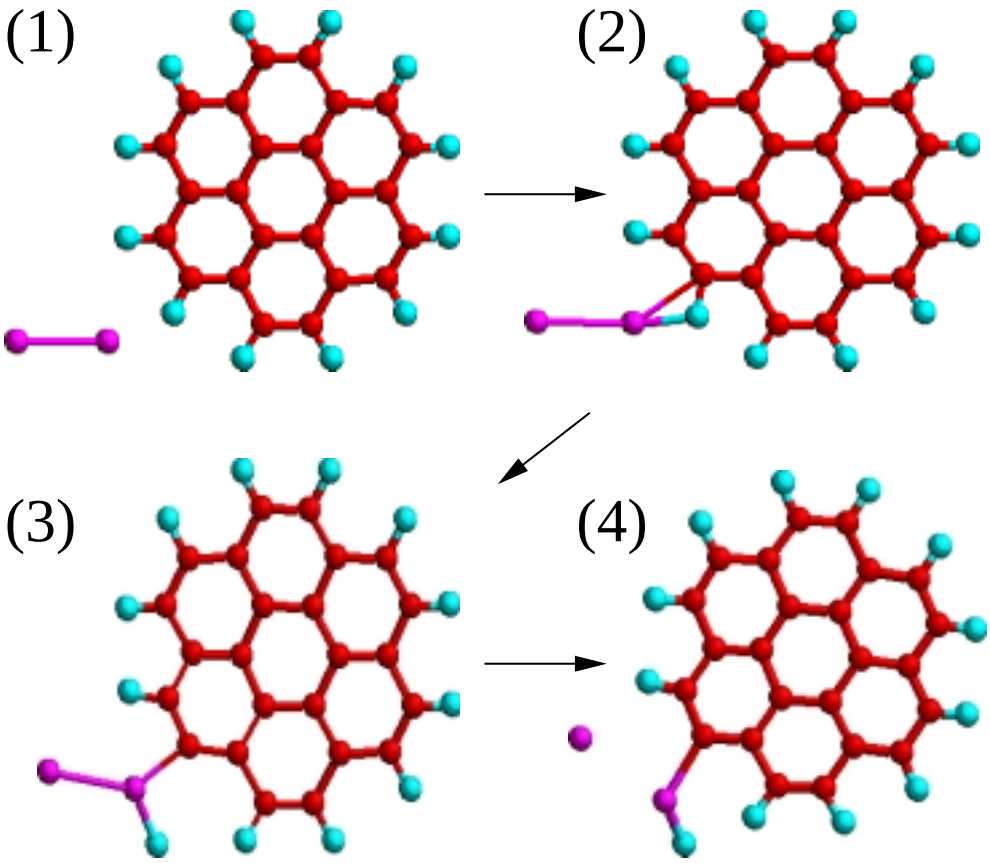

図 3.32: ヨウ素のドープの DRC 計算 [36]

1. I 分子を水平方向からドープ
2. 瞬間的に I 原子が C、 H 原子と結合
3. C–H の結合が切れ、 C–I–H の結合をとる
4. I 分子の結合が切れる

このとき与えた I 分子の運動エネルギーは、 120 [kcal/mol] である。この反応によって C–H の結合が切れるが、新たに C–I–H の結合が生じる。また、計算上の反応のエネルギーが非常に高い。

そこで、図 3.30の F のドープと反応の温度を比較する。この時の温度は計算上において、原子の速度より求めた温度である。そのため、実際の反応温度とは、異なる値

[36] DRC-I1.eps

であり、非現実的な値を示している。しかし、温度の上下関係の比較には使用できると考え、ここに用いた。

F ドープの反応までの F_2 温度は 約 25000 [K] である。それに対して、I_2 の温度は約 10000 [K] と、より低い温度で反応がおこっていることがわかる。そこで、実験での F ドープの反応温度は最大でも 200℃であるので、ヨウ素のドープは、より低い温度で起こるといえる。

3.3.4　ヨウ素の分離

図 3.32の反応によって、ドープされた I 原子は次に示す図 3.33の様に、炭素と水素の間に入って結合すると考える。この時の I 原子の電荷は $+0.49$ である。I に結合している H は -0.22 、 C は、 -0.33 である。

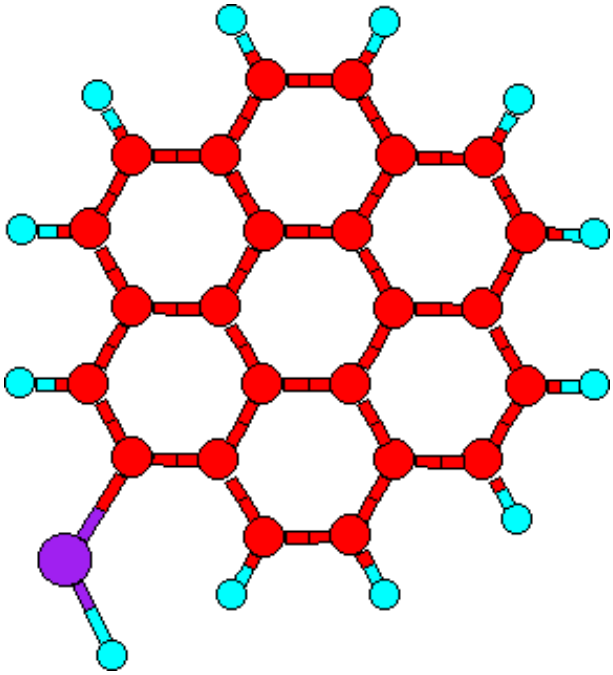

図 3.33: ヨウ素原子のドープされたグラファイト構造 [37]

しかし、ダングリングボンドのないこの構造から直接グラファイト化が進むとは考えにくい。つまり、反応の手順として、水素、ヨウ素が分離し、ダングリングボンドが形成され、その後グラファイト化が進むと考えられる。

そこで、図 3.33の構造を振動させ、その時の温度と、反応の関係を調べた。その結果、表 3.9の様に、温度が上昇するにしたがって、I–H 、 I–C 、 C–H 、 C–C の結合がそれぞれ分解することがわかった。

[37] C24H11-I-H.eps

温度 [K]	900 ～ 1500	2500 ～ 3000	9000 ～ 11000
切れる結合	I–H 結合	C–I 結合	C–H、C–C 結合

表 3.9: 図 3.33の結合の切れる温度

このように、MOPAC 上の DRC 計算では、約 9000[K] から、グラファイトの崩壊が始まったが、通常の C–H の分解は約 850℃起こり、この計算の結果とは異なる。そのため、この温度と実際の温度には差があると考えられるが、反応の起こる順番などは本来の現象と同様な結果を得ているものと考えられる。

よって、ヨウ素ドープによってダングリングボンドの形成が容易になったといえる。そして、これよりグラファイト化が進行する。

第 4 章

まとめ

以下に本研究で得られた結論を述べ、更に今後の課題について述べる。

4.1　Li ドープ

- Li はグラファイトの内部に比べて、エッジサイトに吸着する方が約 0.8[eV] 安定。
- 多数の Li をドープしたとき、正電荷を持つものと、負電荷を持つものが存在し、グラファイト平面からの高さに深い関係がある。負電荷のおきる機構として、
  - リチウム同士の共有結合
  - Li-H_2 クラスター等の影響

  が考えられる。
- Li の総電荷移動量は、ドープ位置によって大きく異なり、少数のドープ量で飽和することもある。しかし、全体的に内部のサイトにドープされる系の方が、より多い総電荷移動量を持つ。
- リチウムの活性化エネルギーは 0.12[eV] 以下と小さく、リチウム原子はグラファイト内を移動しやすい。

4.2 F ドープ

- ハロゲン原子 (F、 Cl 、 Br) は炭素原子と結合し、 sp^3 構造をとる。このとき、イオン半径の小さい原子ほど強い sp^3 結合をとり、結合力も強い。また、どの原子も端に結合する方が安定である。しかし、ヨウ素は炭素原子との sp^3 結合した構造をとることはできない。

- ハロゲン原子の活性化エネルギーは、イオン半径の小さい原子ほど大きい値である。フッ素原子では 1.810[eV] であり、グラファイト内を移動することはほとんどない。

- ドープされたフッ素同士の影響は最近接同士の炭素原子に結合する以外はほとんどなく、 2 個ドープしたとき、端の炭素原子に結合してい数が同じ場合、最近接では他に比べて 0.4 eV 安定になが、それ以外はほとんど変化がない。

- フッ素の電荷は結合位置、結合数にほとんど影響を受けず、総電荷移動量はほぼ比例して増えていく。

- 炭素とフッ素の結合距離は結合位置にのみ影響を受け、内部の炭素原子との結合距離は、端の炭素原子との結合距離に比べて約 0.01 〜 0.02 [Å] 長い。

4.3 ヨウ素ドープ

- ヨウ素ドープによって、エッジ部分の炭素と水素の間にヨウ素が結合される。この反応は、フッ素のドープより低い温度で起こることができる。

- 直接、終端の水素をとるためのエネルギーより、ヨウ素をはさんだ構造からダングリングボンドを形成する方が低いエネルギーで可能である。

謝 辞

本研究及び論文作成にあたり、終始御懇切なる御指導、御鞭撻を賜わりました指導教官である齋藤理一郎助教授に衷心より御礼の言葉を申し上げます。

また、本研究を進めるにあたり、熱心な御指導をいただくとともに種々の御高配を賜わりました木村忠正教授、湯郷成美助教授、一色秀夫助手に深謝の意を表します。

また、本研究にあたりフッ素ドープによる興味深い研究成果について詳しく紹介してくださった、信州大遠藤守信教授、東工大の榎敏明教授、高井和之様、及びヨウ素ドープについての研究に指針を与えてくださった、東工大の安田榮一教授、田邊靖博助教授、宮嶋尚哉様に深謝いたします。

また、本研究では、文部省科学研究費 (特定領域研究 (A) 「カーボンアロイ」) より多くの援助をして頂き、深く感謝致します。

また、研究活動をともにし、多くの援助をいただいた中平政男氏、竹谷隆夫氏、中島瑞樹氏、平原勝久氏、松尾竜馬氏、沼知典氏に深謝いたします。

そして、数々の御援助、御助言をしていただいた中ノ瀬貴生氏、王威氏、山下裕氏、戸田博之氏、はじめ木村 · 齋藤 · 湯郷研究室の大学院生、卒研生の方々に感謝致します。

最後に、研究室の事務業務をして頂いた秘書の山本純子さんに感謝致します。

参考文献

[1] 中平 政男, "グラファイトクラスターの過剰吸着とラマン強度", 1996 年度 修士論文

[2] M. Nakadaira, R. Saito, T. Kimura, G. Dresselhaus, and M. S. Dresselhaus, J. Mater. Res.**12**,1367-1375(1997).

[3] 高井 和之, "sp^2、 sp^3 炭素混合系としての乱雑カーボンの構造、及び磁性、電子物性", 東工大理工学研究科科学専攻 1997 年度修士論文

[4] M. S. Dresselhaus and G. Dresselhaus, Advances in Phys. **30**, 139 (1981).

[5] K. Tozawa, 固体物理 **31**, 925 (1996).

[6] K. Sato, M. Noguchi, A. Demachi, N. Oki, and M. Endo, Sience **264**, 556 (1994).

[7] K. Tanaka, S. Yata, and T. Yamabe, Synthetic Metals **71**, 2147 (1995).

[8] S. Yata, Y. Hato, H. Kinoshita, N. Ando, A. Anekawa, T. Hashimoto, M. Yamaguchi, K. Tanaka, and T. Yamabe, Synthetic Metals **73**, 273 (1995).

[9] M. Endo, Y. Nishimura, T. Takahashi, A. Takamuku, T. Tamaki, 炭素 **172**, 121 (1996).

[10] 化学便覧 応用編, (1980)

[11] 理科年表, 国立天文台編 , 545 , (1993)

[12] J. J. P. Stewart, Fujitsu Limitted Tokyo Japan, (1993). semiempirical quantum chemistry library.

[13] 分子軌道法 MOPAC ガイドブック ー 2 訂版ー, 平野 恒夫 ・ 田辺 和俊編, (1994)

付録 A

プログラムソース

ここに、作成プログラムのソースを示す。

- A.1、 DRC 計算の入力データ作成用プログラム。
- A.2、 DRC 計算の出力データ解析用プログラム。
- A.3、部分状態密度の計算用プログラム。
- A.4、 arc,out ファイル内の、特定原子の情報を取り出すコマンド
- A.5、 out ファイル処理用プログラム。
- A.6、 arc ファイル処理用プログラム。

詳しいプログラムの使用方法は、 2.4を参照のこと。プログラムは、
/home9/students/yagi/script/source のディレクトリにある。

A.1 DRC 入力データ作成プログラム

```
c
c     xyz 座標で 分子の座標を決め、それを DRC 計算するために
c     mopac の入力データに書き換えるプログラム
c
      implicit real*8(a-h,o-z)
      real*8 kine,kineall,kineall2
      integer drc,which
      character*32 jobname,inputdata
      character*50 tmp1
c
      parameter(ntmp=300)
      character*2 atom(ntmp)
      dimension x(ntmp),y(ntmp),z(ntmp)
      dimension xx(ntmp),yy(ntmp),zz(ntmp),r(ntmp)
      dimension ax(ntmp),ay(ntmp),az(ntmp)
      dimension bx(ntmp),by(ntmp),bz(ntmp)
      dimension cx(ntmp),cy(ntmp),cz(ntmp)
      dimension dx(ntmp),dy(ntmp),dz(ntmp)
      dimension v(ntmp),number(ntmp),which(ntmp)
      dimension weight(ntmp),thermo(ntmp),nx(ntmp)
c
c     jobname から 読み込むファイルネームの決定
c
      call getarg(1,jobname)
      j=index(jobname,' ')-1
      open(60,file=jobname(:j)//'.xyz')
c
c   ファイル読み込み
c
      read(60,*) n
      read(60,'(a)') tmp1
c
      do i = 1,n
      read(60,*) atom(i),x(i),y(i),z(i)
      end do
      close(60)
c
c     分子量の決定
c
      do i = 1,n
      if(atom(i).eq."H")  weight(i)=1.008
      if(atom(i).eq."He")  weight(i)=4.002602
      if(atom(i).eq."HE")  weight(i)=4.002602
      if(atom(i).eq."Li") weight(i)=6.941
      if(atom(i).eq."LI") weight(i)=6.941
      if(atom(i).eq."C")  weight(i)=12.011
      if(atom(i).eq."F")  weight(i)=18.998
      if(atom(i).eq."N") weight(i)=14.00674
      if(atom(i).eq."Cl") weight(i)=35.4527
      if(atom(i).eq."CL") weight(i)=35.4527
      if(atom(i).eq."Ar") weight(i)=39.948
      if(atom(i).eq."AR") weight(i)=39.948
      if(atom(i).eq."Br") weight(i)=79.904
      if(atom(i).eq."BR") weight(i)=79.904
      if(atom(i).eq."I")  weight(i)=126.9045
      if(weight(i).eq.0.0d0) stop
      end do
c
c     計算条件の入力
c
      call input(drc,thermo,inputdata,n,
     &nnn,nx,ax,ay,az,which,number,cx,cy,cz)
c
c     母材の重心を求める
c
      xx(1) = x(1)
      yy(1) = y(1)
      zz(1) = z(1)
      sweight1 = weight(1)
c
      do i = 1 , nx(1) - 1
         sweight2 =  sweight1 + weight(i+1)
         xx(1)=(sweight1 * xx(1) + weight(i+1) * x(i+1))/sweight2
         yy(1)=(sweight1 * yy(1) + weight(i+1) * y(i+1))/sweight2
         zz(1)=(sweight1 * zz(1) + weight(i+1) * z(i+1))/sweight2
         sweight1 = sweight2
      end do
c
c     半径を求める
c
      do i = 1 , nx(1)
       r(i) = sqrt((x(i)-xx(1))**2+(y(i)-yy(1))**2+(z(i)-zz(1))**2)
      end do
c
c   乱数を方向ベクトルとして与える
c
      do i = 1 , nx(1)
         dx(i) = rand()
```

```
         dy(i) = rand()
         dz(i) = rand()
      end do
c
c     全て正の値なので、平均の値で全てを引く
c
      heikin = 0.0d0
      do i = 1, nx(1)
         heikin = heikin + dx(i) + dy(i) + dz(i)
      end do
c
      heikin = heikin / nx(1) / 3
c
      do i = 1 , nx(1)
         dx(i) = dx(i) - heikin
         dy(i) = dy(i) - heikin
         dz(i) = dz(i) - heikin
      end do
c
      uengy = 1.5d0 * 1.380658d-23*thermo(1)
c
      call move (x,y,z,xx,yy,zz,r,dx,dy,dz,nx,uengy,weight,n,v)
c
c     単位をそろえる
c
      do i = 1 ,nx(1)
         dx(i) = dx(i)
         dy(i) = dy(i)
         dz(i) = dz(i)
      end do
c
c     全体の振動のエネルギーを等しくする
c
      totalkine = 0.0d0
c
      do i = 1 , nx(1)
         v(i)=dx(i)**2+dy(i)**2+dz(i)**2
         kine = 0.5d0 * weight(i) * v(i)*1.6605402d-27
         totalkine = totalkine + kine
      end do
c
      totalkine = totalkine / nx(1)
c
      t2 = sqrt ( uengy / totalkine)
c
      do i = 1 , nx(1)
         dx(i) = dx(i) * t2 * 100d0
         dy(i) = dy(i) * t2 * 100d0
         dz(i) = dz(i) * t2 * 100d0
      end do
c
c     ここから衝突分子の計算
c
      do  j = 2, nnn+1
c
c   初期化
c
      tt = 1.0d0
      n1 = nx(j-1) + 1
      n2 = nx(j)
c
      do i = n1,n2
         bx(i) = 0.0d0
         by(i) = 0.0d0
         bz(i) = 0.0d0
      end do
c
c     ぶつける原子が 1 つの時は重心をその原子の値にして
c     推進部分の計算へ進む。
c
      if(n1.eq.n2) then
         xx(j) = x(n1)
         yy(j) = y(n1)
         zz(j) = z(n1)
         goto 100
      end if
c
      call roll (x,y,z,xx,yy,zz,ax,ay,az,
     &bx,by,bz,r,weight,n1,n2,n3,j,n)
      write(*,*) ' 重心のベクトル'
      write(*,*) xx(j),yy(j),zz(j)
c
c     v1 : 推進速度 (比を求めるため)
c     v2 : 回転速度 (比を求めるため)
c     tt : 推進速度 / 回転速度の最大値
c
c     2 分子の場合は tt = 1.5 にする。
c
      if(n2-n1.eq.1) then
         tt = 1.5d0
      else
         do i = n1,n2
            v1 = v1 + weight(i) * r(i)**2
         end do
c
         do i = n1 ,n2
            v2 = v2 + weight(i)
```

```
         end do
         tt = sqrt( v1 / v2) / r(n3)
      end if
c
      write(*,*) tt
c
100   call attack (x,y,z,xx,yy,zz,cx,cy,cz,j,tt,n,which,number)
      write(*,*) '推進方向ベクトル'
      write(*,*) cx(j),cy(j),cz(j)
c
      do i = n1,n2
         dx(i) = bx(i) + cx(j)
         dy(i) = by(i) + cy(j)
         dz(i) = bz(i) + cz(j)
      end do
c
      call kinetic (weight,thermo,dx,dy,dz,n1,n2,t1,n,v,j)
c
      end do
c
c     全体の運動エネルギー
c
      kineall = 0.0d0
      do i = 1,n
         v(i) =  dx(i)**2 + dy(i)**2 + dz(i)**2
         kineall = kineall+0.5d0*weight(i)*v(i)*1.6605402d-27
      end do
      kineall = kineall * 1.0d-4 / n
      kineall2 = kineall / 1.60218d-19
      write(*,*) kineall2, '[eV/atom]'
      kineall = kineall  * 6.022d23 * 0.239006 * 1.0d-3
c      kineall = kineall * 1.43862d16
c
c
c   ファイルの書き込み sub-out
c
      j=index(jobname,' ')-1
      open(61,file=jobname(:j)//'.dat')
      write(61,1002)
      if(drc.ne.0) then
         write(61,1003) drc
      else
      write(61,1004)
      end if
      write(61,1007) drc,kineall,(thermo(i),i=1,nnn+1)
c
      do i = 1,n
         write(61,1001) atom(i),x(i),y(i),z(i)
      end do
c
      write(61,*) ' '
c
      do i =1,n
         write(61,1005) dx(i),dy(i),dz(i)
      end do
c
      write(61,*) ' '
      write(61,*) ' '
      close(61)
c
c  .xyz へ、ベクトルを入れて出力
c
      do i = 1,n
         dx(i) = -dx(i) * 0.00001d0
         dy(i) = -dy(i) * 0.00001d0
         dz(i) = -dz(i) * 0.00001d0
      end do
c
      j=index(jobname,' ')-1
      open(60,file=jobname(:j)//'.xyz')
      write(60,*) n
      write(60,*) tmp1
      do i = 1,n
         write(60,1006) atom(i),x(i),y(i),z(i),dx(i),dy(i),dz(i)
      end do
      write(60,*) ' '
      write(60,*) ' '
      close(60)
      write(*,*) ' 推進速度', t1*tt , ' cm/sec'
      write(*,*) ' 運動エネルギー', kineall , ' kcal/mol'
c
c     確認
c
      do i = 1,n
         dx(i) = dx(i) * 100.0d0
         dy(i) = dy(i) * 100.0d0
         dz(i) = dz(i) * 100.0d0
      end do
c
      j=index(jobname,' ')-1
      open(60,file=jobname(:j)//'.temp')
      write(60,*) n+2
      write(60,*) tmp1
      do i = 1,n
         write(60,1006) atom(i),x(i),y(i),z(i),dx(i),dy(i),dz(i)
```

```
      end do
      write(60,1000) xx(1) , yy(1), zz(1)
      write(60,1000) xx(2) , yy(2), zz(2)
      write(60,*) ' '
      write(60,*) ' '
      close(60)
c
c
 1000 format(' XX ',f10.5,f10.5,f10.5)
 1001 format(a5,f9.4,' 0 ',f9.4,' 0 ',f9.4, ' 0' )
 1002 format('T=1.0D NOINTER GNORM=0 PM3 GEO-OK UHF SHIFT=2 PULAY &')
 1003 format('VELOCITY T-PRIORITY=5.0 LARGE=-1 DRC=',I3)
 1004 format('VELOCITY T-PRIORITY=5.0 LARGE=-1 DRC ')
 1005 format(f20.5,f20.5,f20.5)
 1006 format(a5,f9.4,f9.4,f9.4,f11.5,f11.5,f11.5)
 1007 format('DRC=',I3,', kine=',f10.3,' T=',f7.1,',',f7.1,f7.1,f7.1)
 1008 format (f10.5)
      stop
      end
c
      real function rand()
      integer c,k,r,m
      real maxr
      parameter(c=656329,k=163,
     & M=12518383,maxr=12518383.0)
      save R
      data R/12345678/
      r=mod(k*r+c,m)
      rand=real(r)/maxr
      return
      end
c
      subroutine roll (x,y,z,xx,yy,zz,ax,ay,az,
     &bx,by,bz,r,weight,n1,n2,n3,j,n)
      implicit real*8(a-h,o-z)
      parameter(ntmp=300)
      dimension x(n),y(n),z(n),r(n)
      dimension xx(n),yy(n),zz(n)
      dimension ax(n),ay(n),az(n)
      dimension bx(n),by(n),bz(n)
      dimension tx(ntmp),ty(ntmp),tz(ntmp)
      dimension weight(n)
c
c     重心、回転の計算
c
      xx(j)=x(n1)
      yy(j)=y(n1)
      zz(j)=z(n1)
      sweight1 = weight(n1)
c
      do i =n1,n2-1
      sweight2 =  sweight1 +weight(i+1)
      xx(j) = (sweight1 * xx(j) + weight(i+1) * x(i+1))/sweight2
      yy(j) = (sweight1 * yy(j) + weight(i+1) * y(i+1))/sweight2
      zz(j) = (sweight1 * zz(j) + weight(i+1) * z(i+1))/sweight2
      sweight1 = sweight2
      end do
c
c     tx,ty,tz : i 番目の原子からの回転軸への垂線の交点
c     bx,by,bz : i 番目の原子の回転の方向ベクトル
c     r(i)     : i 番目の原子の回転軸からの距離(回転半径)
c
      do i = n1,n2
      t=((x(i)-xx(j))*ax(j)+(y(i)-yy(j))*ay(j)+(z(i)-zz(j))*az(j))
     & /(ax(j)**2+ay(j)**2+az(j)**2)
      tx(i) = ax(j) * t + xx(j)
      ty(i) = ay(j) * t + yy(j)
      tz(i) = az(j) * t + zz(j)
      r(i) = sqrt((tx(i)-x(i))**2+(ty(i)-y(i))**2+(tz(i)-z(i))**2)
      bx(i) = ay(j) * (z(i)-tz(i)) - az(j) * (y(i)-ty(i))
      by(i) = az(j) * (x(i)-tx(i)) - ax(j) * (z(i)-tz(i))
      bz(i) = ax(j) * (y(i)-ty(i)) - ay(j) * (x(i)-tx(i))
      bx(i) = bx(i) * r(i)
      by(i) = by(i) * r(i)
      bz(i) = bz(i) * r(i)
      end do
c
c    正規化しよー
c
c     n3  : 回転半径の最大値の原子の番号
c     rmax : 回転半径の最大値
c     bmax : 回転方向ベクトルの最大値 (正規化用)
c
      rmax = r(n1)
      n3 = n1
      do i = n1+1,n2
      if (r(i).gt.rmax) then
      rmax = r(i)
      n3 = i
      end if
```

```
      end do
c
      bmax=sqrt(bx(n3)**2+by(n3)**2+bz(n3)**2)
      do i=n1,n2
         bx(i) = bx(i) / bmax
         by(i) = by(i) / bmax
         bz(i) = bz(i) / bmax
      end do
c
      return
      end
c
c
c
      subroutine attack (x,y,z,xx,yy,zz,cx,cy,cz,j,tt,n,which,number)
      implicit real*8(a-h,o-z)
      integer which
      dimension x(n),y(n),z(n)
      dimension xx(n),yy(n),zz(n)
      dimension cx(n),cy(n),cz(n)
      dimension which(n),number(n)
c
      if(which(j).eq.2) then
         cx(j) = 0.0d0
         cy(j) = 0.0d0
         cz(j) = 0.0d0
c
         cx(j) = x(number(j)) - xx(j)
         cy(j) = y(number(j)) - yy(j)
         cz(j) = z(number(j)) - zz(j)
      else
         if (which(j).eq.1) then
         else
            stop
         end if
      end if
c
      cmax=sqrt(cx(j)**2+cy(j)**2+cz(j)**2)
      cx(j) = -cx(j) / cmax * tt
      cy(j) = -cy(j) / cmax * tt
      cz(j) = -cz(j) / cmax * tt
c
      return
      end
c
c     ドープ原子の運動エネルギーを求め、
c     速度を求める
c
      subroutine kinetic (weight,thermo,dx,dy,dz,n1,n2,t1,n,v,j)
      implicit real*8(a-h,o-z)
      dimension dx(n),dy(n),dz(n)
      dimension v(n),weight(n),thermo(n)
      real*8 kine
c
      kine = 0.0d0
      do i = n1, n2
         v(i) =  dx(i)**2 + dy(i)**2 + dz(i)**2
         kine = kine+0.5d0*weight(i)*v(i)*1.6605402d-27
      end do
c
      poten=1.5d0*(n2-n1+1)*1.380658d-23*thermo(j)
c
c     t1 : 回転速度の最大値を 1 とした時の
c     実際の速度との比。→ t1 * tt = 推進速度
c
c
      t1 = sqrt ( poten / kine ) * 100d0
c
      do i = n1,n2
         dx(i) = dx(i) * t1
         dy(i) = dy(i) * t1
         dz(i) = dz(i) * t1
      end do
      return
      end
c
c
c
c     1 原子あたりの運動エネルギーより、振動速度を求める
c
      subroutine move (x,y,z,xx,yy,zz,r,dx,dy,dz,nx,uengy,weight,n,v)
      implicit real*8(a-h,o-z)
      parameter(ntmp=300)
      dimension x(n),y(n),z(n)
      dimension xx(n),yy(n),zz(n),r(n)
      dimension dx(n),dy(n),dz(n)
      dimension fx(ntmp),fy(ntmp),fz(ntmp)
      dimension v(n),fv(ntmp)
      dimension weight(n),nx(n)
      real*8 kine
      real*8 mvx,mvy,mvz
      integer count
c
```

```
c      1 原子あたりのエネルギーを "uengy" にする
c      x,y,z 方向それぞれの運動量を "0"へ
c      x,y,z 軸それぞれの回転成分を "0" へ
c
c      初期化
c
 400   mvx = 0.0d0
       mvy = 0.0d0
       mvz = 0.0d0
       wmvx = 0.0d0
       wmvy = 0.0d0
       wmvz = 0.0d0
c
       do i = 1,nx(1)
          v(i) = dx(i)**2 + dy(i)**2 + dz(i)**2
          kine = 0.5d0 * weight(i) * v(i) * 1.6605402d-27
          t2 = sqrt ( uengy / kine)
          dx(i) = dx(i) * t2
          dy(i) = dy(i) * t2
          dz(i) = dz(i) * t2
       end do
c
c      分子全体の運動量を調べ、 0 にする
c
       do i = 1 , nx(1)
          mvx = mvx + weight(i) * dx(i)
          mvy = mvy + weight(i) * dy(i)
          mvz = mvz + weight(i) * dz(i)
       end do
c
       mvx = mvx / nx(1)
       mvy = mvy / nx(1)
       mvz = mvz / nx(1)
c
       do i = 1 , nx(1)
          dx(i) = dx(i) - mvx / weight(i)
          dy(i) = dy(i) - mvy / weight(i)
          dz(i) = dz(i) - mvz / weight(i)
       end do
c
c      回転の成分を消す
c
c      x 軸の回転成分を求める
c
       do i = 1 , nx(1)
          ey = - z(i) + zz(1)
          ez =   y(i) - yy(1)
          fy(i) =  ( (dy(i)*ey + dz(i)*ez) / (ey**2+ez**2)) * ey
          fz(i) =  ( (dy(i)*ey + dz(i)*ez) / (ey**2+ez**2)) * ez
          fv(i) = sqrt( fy(i)**2 + fz(i)**2 )
       end do
c
c      符号の確認
c
       do i = 1 ,nx(1)
c
          if(y(i).lt.yy(1)) then
             if(fz(i).lt.0.0d0) then
                fv(i) = -fv(i)
             else
                if(fz(i).eq.0.0d0) then
                   write(*,*) 'error'
                   stop
                end if
             end if
          else
             if(y(i).gt.yy(1)) then
                if(fz(i).gt.0.0d0) then
                   fv(i) = -fv(i)
                else
                   if(fz(i).eq.0.0d0) then
                      write(*,*) 'error'
                      stop
                   end if
                end if
             else
                if(y(i).eq.yy(1)) then
                   if(fy(i).lt.0.0d0) then
                      fv(i) = -fv(i)

                   end if
                else
                   write(*,*) 'error'
                   stop
                end if
             end if
          end if
       end do
c
c      回転の運動量を求め、 0 になるように修正
c
       do i = 1 , nx(1)
          wmvx = wmvx + weight(i) * fv(i) / r(i)
       end do
c
       wmvx = wmvx / nx(1)
```

```
c
      do i = 1, nx(1)
         dy(i) = dy(i) - wmvx / weight(i) * r(i) / fv(i) * fy(i)
         dz(i) = dz(i) - wmvx / weight(i) * r(i) / fv(i) * fz(i)
      end do
c
c     y 軸の回転成分を求める
c
      do i = 1 , nx(1)
         ez = - x(i) + xx(1)
         ex =   z(i) - zz(1)
         fz(i) =  ( (dz(i)*ez + dx(i)*ex) / (ez**2+ex**2)) * ez
         fx(i) =  ( (dz(i)*ez + dx(i)*ex) / (ez**2+ex**2)) * ex
         fv(i) = sqrt( fz(i)**2 + fx(i)**2 )
      end do
c
c     符号の確認
c
      do i = 1 ,nx(1)
c
         if(x(i).lt.xx(1)) then
            if(fz(i).lt.0.0d0) then
               fv(i) = -fv(i)
            else
               if(fz(i).eq.0.0d0) then
                  write(*,*) 'error'
                  stop
               end if
            end if
         else
            if(x(i).gt.xx(1)) then
               if(fz(i).gt.0.0d0) then
                  fv(i) = -fv(i)
               else
                  if(fz(i).eq.0.0d0) then
                     write(*,*) 'error'
                     stop
                  end if
               end if
            else
               if(x(i).eq.xx(1)) then
                  if(fx(i).lt.0.0d0) then
                     fv(i) = -fv(i)

                  end if
               else
                  write(*,*) 'error'
                  stop
               end if
            end if
         end if
      end do
c
c     回転の運動量を求め、 0 になるように修正
c
      do i = 1 , nx(1)
         wmvy = wmvy + weight(i) * fv(i) / r(i)
      end do
c
      wmvy = wmvy / nx(1)
c
      do i = 1, nx(1)
         dz(i) = dz(i) - wmvy / weight(i) * r(i) / fv(i) * fz(i)
         dx(i) = dx(i) - wmvy / weight(i) * r(i) / fv(i) * fx(i)
      end do
c
c     z 軸の回転成分を求める
c
      do i = 1 , nx(1)
         ex = - y(i) + yy(1)
         ey =   x(i) - xx(1)
         fx(i) =  ( (dx(i)*ex + dy(i)*ey) / (ex**2+ey**2)) * ex
         fy(i) =  ( (dx(i)*ex + dy(i)*ey) / (ex**2+ey**2)) * ey
         fv(i) = sqrt( fx(i)**2 + fy(i)**2 )
      end do
c
c     符号の確認
c
      do i = 1 ,nx(1)
c
         if(x(i).lt.xx(1)) then
            if(fy(i).lt.0.0d0) then
               fv(i) = -fv(i)
            else
               if(fy(i).eq.0.0d0) then
                  write(*,*) 'error'
                  stop
               end if
            end if
         else
            if(x(i).gt.xx(1)) then
               if(fy(i).gt.0.0d0) then
                  fv(i) = -fv(i)
               else
                  if(fy(i).eq.0.0d0) then
                     write(*,*) 'error'
                     stop
                  end if
```

```
               end if
            else
               if(x(i).eq.xx(1)) then
                  if(fx(i).lt.0.0d0) then
                     fv(i) = -fv(i)

                  end if
               else
                  write(*,*) 'error'
                  stop
               end if
            end if
         end if
      end do
c
c     回転の運動量を求め、 0 になるように修正
c
      do i = 1 , nx(1)
         wmvz = wmvz + weight(i) * fv(i) / r(i)
      end do
c
      wmvz = wmvz / nx(1)
c
      do i = 1, nx(1)
         dx(i) = dx(i) - wmvz / weight(i) * r(i) / fv(i) * fx(i)
         dy(i) = dy(i) - wmvz / weight(i) * r(i) / fv(i) * fy(i)
      end do
c
c
c     データ修正の繰り返し
c
c     それぞれ動かした値の合計が "10 cm/sec" で終了
c
      data=abs(mvx)+abs(mvy)+abs(mvz)+abs(wmvx)+abs(wmvy)+abs(wmvz)
c
      count = count + 1
      if(count.lt.100) then
         if(data.gt.10.0d0) goto 400
      end if
      return
      end
c
      subroutine input(drc,thermo,inputdata,n,
     &nnn,nx,ax,ay,az,which,number,cx,cy,cz)
      implicit real*8(a-h,o-z)
      integer drc,which
      character*32 inputdata
      dimension ax(n),ay(n),az(n)
      dimension cx(n),cy(n),cz(n)
      dimension nx(n)
      dimension number(n),which(n),thermo(n)
c
      call getarg(2,inputdata)
      in=index(inputdata,' ')-1
c
      if(in.eq.0) then
         write(*,*) 'DRC = ? 0 ならエネルギー保存'
         read(*,*) drc
         write(*,*) ' 母材の分子群は 1 から何番まで?'
         read(*,*) nx(1)
         write(*,*) ' 母材の温度は?'
         read(*,*) thermo(1)
c
         if(nx(1).eq.n) goto 80
         write(*,*) ' いくつの分子群をぶつけますか?'
         read(*,*) nnn
c
         do j = 2 , nnn+1
            write(*,*) j-1,' 個目の分子群'
            write(*,*) nx(j-1)+1,' 〜何番までの原子を動かしますか?'
            write(*,*) ' 一つしか動さない場合は、その番号を入力'
            read(*,*) nx(j)
            write(*,*) ' 温度 = ? '
            read(*,*) thermo(j)
            if(nx(j-1)+1.eq.nx(j)) goto 70
            write(*,*) ' 回転軸の方向ベクトル (x,y,z)'
            read(*,*) ax(j),ay(j),az(j)
c
 70         write(*,*) ' 推進方向ベクトルを決めます。'
            write(*,*) ' xyz 座標の場合は 1、原子の番号の場合は 2'
            read(*,*) which(j)
            if(which(j).eq.2) then
               write(*,*) ' 何番の原子にぶつけますか'
               read(*,*) number(j)
            else
               if (which(j).eq.1) then
                  write(*,*) ' 推進方向の xyz 座標を書いて下さい '
                  read(*,*) cx(j),cy(j),cz(j)
               else
                  goto 70
               end if
            end if
         end do
```

```
      else
c
         open(70,file=inputdata(:in))
         nnn = 0
         read(70,*) drc
         read(70,*) nx(1)
         read(70,*) thermo(1)
c
         if(nx(1).eq.n) goto 80
         do j = 2, 20
            read(70,*) n1,n2
            if(n1.ne.nx(1)+1) stop
            if(n2.gt.n) stop
            nx(j) = n2
c
            read(70,*) thermo(j)
c
            if(n1.ne.n2) then
               read(70,*) ax(j),ay(j),az(j)
            end if
c
            read(70,*) which(j)
            if(which(j).eq.2) then
               read(70,*) number(j)
            else
               if (which(j).eq.1) then
                  read(70,*) cx(j),cy(j),cz(j)
               else
                  stop
               end if
            end if
            nnn = nnn + 1
            if(n2.eq.n) goto 80
         end do
      end if
 80   write(*,*) 'data-input end'
      close(70)
      return
      end
```

A.2 DRC 出力データ解析プログラム

```
c
c      DRC 計算の out ファイルより、必要なデータを取り出し、
c      解析するプログラム。
c
       implicit real*8(a-h,o-z)
       character*32 jobname
       character*16 tmp1
       character*6 tmp2
       character*60 tmp60
       parameter(nn=3000)
       dimension x(nn),y(nn),z(nn)
       dimension xx(nn),yy(nn),zz(nn)
       dimension dx(nn),dy(nn),dz(nn)
       dimension charge(nn),weight(nn),v(nn)
       dimension mm(nn),r(nn),heat(nn)
       character*16 atom(nn)
       integer data
       real time
c
c     jobname から 読み込むファイルネームの決定
c
       call getarg(1,jobname)
       jo=index(jobname,' ')-1
c
c      出力データの選択
c
 5     write(*,*) 'どのデータが必要ですか？'
       write(*,*) ' アニメーション → 1 '
       write(*,*) ' エネルギー変化 → 2 '
       write(*,*) ' 原子間距離変化 → 3 '
       write(*,*) ' 温度変化 → 4 '
c
       read(*,*) data
       if(data.ge.5)  goto 5
       if(data.eq.1) open(61,file=jobname(:jo)//'.anm')
       if(data.eq.2) open(62,file=jobname(:jo)//'.energy')
       if(data.eq.3) then
          open(63,file=jobname(:jo)//'.dis')
          write(*,*) ' 何番となんばんの距離?? '
          read(*,*) nn1,nn2
       end if
c
       if(data.eq.4) then
          open(64,file=jobname(:jo)//'.thermo')
          write(*,*) 'いくつの分子群の温度を求めますか?'
          read(*,*) many
          do k = 1, many
             write(*,*) many ,'何番から何番までの温度??'
             read(*,*) mm(k),mm(k+1)
          end do
       end if
c
c      データ読み込み開始
c
       open(60,file=jobname(:jo)//'.out')
c
       do i= 1,1000
          read(60,*) tmp1
          if ( tmp1.eq."NUMBER") GOTO 10
       end do
c
c     原子数の確認
c
 10    n = 0
       m = 2
       do i = 1,300
          read(60,'(a)') tmp1
          if ( tmp1.eq."          CARTES") GOTO 20
          n = n+1
       end do
 20    read(60,'(a)') tmp1
       read(60,'(a)') tmp1
       read(60,'(a)') tmp1
       n = n-4
       write(*,*) n
c
c     初期データ読み込み
c
       do i = 1,n
          read(60,*) number,atom(i),x(i),y(i),z(i)
       end do
c
c     初期方向ベクトルの読み込み
c
       do i = 1,1000
          read(60,'(a)') tmp2
          if ( tmp2.eq." INITI") GOTO 30
       end do
c
 30    do i=1,n
```

```
         read(60,*) xx(i),yy(i),zz(i)
      end do
c
      do i = 1,n
         xx(i) = - xx(i) * 1.0d-5
         yy(i) = - yy(i) * 1.0d-5
         zz(i) = - zz(i) * 1.0d-5
      end do
c
c     分子量の計算
c
      do i = 1,n
         if(atom(i).eq."H")  weight(i)=1.008
         if(atom(i).eq."He")  weight(i)=4.002602
         if(atom(i).eq."HE")  weight(i)=4.002602
         if(atom(i).eq."Li") weight(i)=6.941
         if(atom(i).eq."LI") weight(i)=6.941
         if(atom(i).eq."C")  weight(i)=12.011
         if(atom(i).eq."F")  weight(i)=18.998
         if(atom(i).eq."N") weight(i)=14.00674
         if(atom(i).eq."Cl") weight(i)=35.4527
         if(atom(i).eq."CL") weight(i)=35.4527
         if(atom(i).eq."Ar") weight(i)=39.948
         if(atom(i).eq."AR") weight(i)=39.948
         if(atom(i).eq."Br") weight(i)=79.904
         if(atom(i).eq."BR") weight(i)=79.904
         if(atom(i).eq."I")  weight(i)=126.9045
         if(weight(i).eq.0.0d0) stop
         weight(i) = weight(i) * 1.6605402d-27
      end do
c
c  初期入力データの書き込み
c
      if(data.eq.1) then
         write(61,*) n
         write(61,*) '1'
         do i = 1,n
            write(61,1001) atom(i),x(i),y(i),z(i),xx(i),yy(i),zz(i)
         end do
      end if
c
      if(data.eq.3) then
         call dis(n,time,x,y,z,nn1,nn2)
      end if
c
      if(data.eq.4) then
         do k = 1 , many
            n1 = mm(k)
            n2 = mm(k+1)
            call thermo(n,n1,n2,v,time,atom,x,y,z,xx,yy,zz,
     &           weight,heat,k,dx,dy,dz,r)
         end do
         write(64,1004) time,(heat(k),k = 1,many)
      end if
c
c     結果の最初の部分の検索
c
      mcount=1
      do i= 1,1000
         read(60,'(a)') tmp2
         if ( tmp2.eq." FEMTO") then
            read(60,*) time,bbb,energy1,energy2,energy3
            goto 33
         end if
      end do
c
 33   large = 0
      do i= 1,1000
         read(60,'(a)') tmp2
         if ( tmp2.eq."  ATOM") goto 35
         large = large + 1
      end do
c
c     xyz データ読み込み
c
 35   do i = 1,10000
c
      mcount = mcount + 1
      do j = 1,n
         read(60,*) number,atom(j),x(j),y(j),z(j),xx(j),yy(j),zz(j)
         xx(j) = xx(j) * -0.00001d0
         yy(j) = yy(j) * -0.00001d0
         zz(j) = zz(j) * -0.00001d0
      end do
c
c     charge の読み込み
c
      read(60,'(a)') tmp1
      read(60,'(a)') tmp1
      read(60,'(a)') tmp1
```

```
      read(60,'(a)') tmp1
      read(60,'(a)') tmp1
      read(60,'(a)') tmp1
      if(n.gt.99) then
         do j = 1,n
            read(60,*) tmp1
         end do
      else
         do j = 1,n
            read(60,2000) tmp60,charge(j)
         end do
      end if
c
c     データ出力用の切り替えポイント
c     ( 新たに必要なデータがある場合、ここからサブルーチン
c       をつくって、出力させると良い )
c
      if(data.eq.1) then
         call anm(n,time,energy1,energy2,energy3,mcount,
     &atom,x,y,z,xx,yy,zz,charge)
      end if
c
      if(data.eq.2) then
         call energy (time,energy1,energy2,energy3)
      end if
c
      if(data.eq.3) then
         call dis(n,time,x,y,z,nn1,nn2)
      end if
c
      if(data.eq.4) then
         do k = 1 , many
            n1 = mm(k)
            n2 = mm(k+1)
            call thermo(n,n1,n2,v,time,atom,x,y,z,xx,yy,zz,
     &           weight,heat,k,dx,dy,dz,r)
         end do
         write(64,1004) time,(heat(k),k = 1,many)
      end if
c
c
c
c
c   次の検索
c
      do k= 1,1000
         read(60,'(a)') tmp2
         if ( tmp2.eq." FEMTO") then
            do l = 1 , large-1
               read(60,*)
            end do
            read(60,*) time,bbb,energy1,energy2,energy3
            read(60,*)
            read(60,*)
            GOTO 40
         end if
         if ( tmp2.eq." TOTAL") GOTO 50
      end do
c
 40   m = m + 1
      end do
 50   write(*,*) 'end', m
c
 1000 format(a5,f10.5,f10.5,f10.5)
 1001 format(a5,f10.5,f10.5,f10.5,f10.5,f10.5,f10.5,f10.5)
 1002 format(a5,f13.7,I3,f13.7,I3,f13.7,I3,I5,I5,I5,f12.4)
 1003 format(f7.1,a8,' poten.=',f9.2,' kine.=',f9.2,' total=',f9.2,I5)
 1004 format(f10.2,f17.5,f17.5,f17.5,f17.5,f17.5)
 2000 format(a60,f12.4)
      stop
      end
c
c     アニメーションのサブルーチン
c
      subroutine anm (n,time,energy1,energy2,energy3,mcount,
     &atom,x,y,z,xx,yy,zz,charge)
      implicit real*8(a-h,o-z)
      dimension x(n),y(n),z(n)
      dimension xx(n),yy(n),zz(n)
      dimension charge(n)
      character*16 atom(n)
      real time
c
      write(61,*) n
      write(61,1101) time,'[fsec.]',energy1,energy2,energy3,mcount
      do j = 1,n
      write(61,1102) atom(j),x(j),y(j),z(j),charge(j),xx(j),yy(j),zz(j)
      end do
c
 1101 format(f7.1,a8,' poten.=',f9.2,' kine.=',f9.2,' total=',f9.2,I5)
 1102 format(a5,f10.5,f10.5,f10.5,f10.5,f10.5,f10.5,f10.5)
      return
      end
```

```
c
c     エネルギーの表示のサブルーチン
c
      subroutine energy (time,energy1,energy2,energy3)
      implicit real*8(a-h,o-z)
      real time
      write(62,1201) time,energy1,energy2,energy3
 1201 format(f10.2,f12.5,f12.5,f12.5)
      return
      end
c
c     分子間距離の変化の表示のサブルーチン
c
      subroutine dis(n,time,x,y,z,nn1,nn2)
      implicit real*8(a-h,o-z)
      dimension x(n),y(n),z(n)
      real time
c
      r=(x(nn1)-x(nn2))**2+(y(nn1)-y(nn2))**2+(z(nn1)-z(nn2))**2
      r=sqrt(r)
      write(63,1301) time, r
 1301 format(f10.2,f12.5)
      return
      end
c
c     温度変化の表示のサブルーチン
c
      subroutine thermo(n,n1,n2,v,time,atom,x,y,z,xx,yy,zz,
     &     weight,heat,k,dx,dy,dz,r)
      implicit real*8(a-h,o-z)
      dimension x(n),y(n),z(n),xx(n),yy(n),zz(n)
      dimension dx(n),dy(n),dz(n)
      dimension weight(n),v(n),r(n),heat(n)
      character*16 atom(n)
      real time,kine
c
c     初期化及び定数の決定
c
      kine = 0.0d0
      heat(k) = 0.0d0
      bol = 1.380658d-23
c
c     重心を求める (cx,xy,cz)
c
      cx = 0.0d0
      cy = 0.0d0
      cz = 0.0d0
c
      sweight1 = 0.0d0
      do i = n1 ,n2
         sweight1 = sweight1 + weight(i)
      end do

      do i = n1 , n2
         cx = cx + weight(i) * x(i)
         cy = cy + weight(i) * y(i)
         cz = cz + weight(i) * z(i)
      end do

      cx = cx / sweight1
      cy = cy / sweight1
      cz = cz / sweight1
c
      do i = n1 , n2
         r(i) = sqrt((x(i)-cx)**2+(y(i)-cy)**2+(z(i)-cz)**2)
      end do
c
c     回転、並進成分を取り除く
c
      do i = n1 , n2
         dx(i) = xx(i)
         dy(i) = yy(i)
         dz(i) = zz(i)
      end do
c
      if(n1.eq.n2) goto 1500
c
      call remove (x,y,z,xx,yy,zz,r,dx,dy,dz,weight,
     &     n,v,n1,n2,cx,cy,cz)
c
c     運動エネルギーの計算開始
c
 1500 do j = n1 , n2
         v(j) = dx(j)**2 + dy(j)**2 + dz(j)**2
         v(j) = v(j) * 1.0d6
      end do
c
      do j = n1 , n2
         kine = kine + weight(j) * v(j)
      end do
c
c     温度への変換
c
      heat(k) = kine / 3.0d0 / (n2-n1+1) / bol
c
      return
      end
```

```
c
      subroutine remove (x,y,z,xx,yy,zz,r,dx,dy,dz,weight,
     &     n,v,n1,n2,cx,cy,cz)
      implicit real*8(a-h,o-z)
      parameter(ntmp=300)
      dimension x(n),y(n),z(n)
      dimension xx(n),yy(n),zz(n),r(n)
      dimension dx(n),dy(n),dz(n)
      dimension fx(ntmp),fy(ntmp),fz(ntmp)
      dimension v(n),fv(ntmp)
      dimension weight(n)
      real*8 mvx,mvy,mvz
c
c     初期化
c
      mvx = 0.0d0
      mvy = 0.0d0
      mvz = 0.0d0
      wmvx = 0.0d0
      wmvy = 0.0d0
      wmvz = 0.0d0
c
c     分子全体の運動量を調べ、 0 にする
c
      do i = n1 , n2
         mvx = mvx + weight(i) * dx(i)
         mvy = mvy + weight(i) * dy(i)
         mvz = mvz + weight(i) * dz(i)
      end do
c
      mvx = mvx / (n2 - n1 + 1)
      mvy = mvy / (n2 - n1 + 1)
      mvz = mvz / (n2 - n1 + 1)
c
      do i = n1 , n2
         dx(i) = dx(i) - mvx / weight(i)
         dy(i) = dy(i) - mvy / weight(i)
         dz(i) = dz(i) - mvz / weight(i)
      end do
c
c     回転の成分を消す
c
c     x 軸の回転成分を求める
c
      do i = n1 , n2
         ey = - z(i) + cz
         ez =   y(i) - cy
         fy(i) =  ( (dy(i)*ey + dz(i)*ez) / (ey**2+ez**2)) * ey
         fz(i) =  ( (dy(i)*ey + dz(i)*ez) / (ey**2+ez**2)) * ez
         fv(i) = sqrt( fy(i)**2 + fz(i)**2 )
      end do
c
c     符号の確認
c
      do i = n1 ,n2
c
         if(y(i).lt.cy) then
            if(fz(i).lt.0.0d0) then
               fv(i) = -fv(i)
            else
               if(fz(i).eq.0.0d0) then
                  write(*,*) 'x error'
c                  stop
               end if
            end if
         else
            if(y(i).gt.cy) then
               if(fz(i).gt.0.0d0) then
                  fv(i) = -fv(i)
               else
                  if(fz(i).eq.0.0d0) then
                     write(*,*) 'x error'
c                     stop
                  end if
               end if
            else
               if(y(i).eq.cy) then
                  if(fy(i).lt.0.0d0) then
                     fv(i) = -fv(i)
                  end if
               else
                  write(*,*) 'x error'
c                  stop
               end if
            end if
         end if
      end do
c
c     回転の運動量を求め、 0 になるように修正
c
      do i = n1 , n2
         wmvx = wmvx + weight(i) * fv(i) / r(i)
      end do
c
      wmvx = wmvx / ( n2 - n1 + 1 )
```

```
c
      do i = n1, n2
         dy(i) = dy(i) - wmvx / weight(i) * r(i) / fv(i) * fy(i)
         dz(i) = dz(i) - wmvx / weight(i) * r(i) / fv(i) * fz(i)
      end do
c
c     y 軸の回転成分を求める
c
      do i = n1 , n2
         ez = - x(i) + cx
         ex =   z(i) - cz
         fz(i) =  ( (dz(i)*ez + dx(i)*ex) / (ez**2+ex**2)) * ez
         fx(i) =  ( (dz(i)*ez + dx(i)*ex) / (ez**2+ex**2)) * ex
         fv(i) = sqrt( fz(i)**2 + fx(i)**2 )
      end do
c
c     符号の確認
c
      do i = n1 ,n2
c
         if(x(i).lt.cx) then
            if(fz(i).lt.0.0d0) then
               fv(i) = -fv(i)
            else
               if(fz(i).eq.0.0d0) then
                  write(*,*) 'error'
                  stop
               end if
            end if
         else
            if(x(i).gt.cx) then
               if(fz(i).gt.0.0d0) then
                  fv(i) = -fv(i)
               else
                  if(fz(i).eq.0.0d0) then
                     write(*,*) 'error'
                     stop
                  end if
               end if
            else
               if(x(i).eq.cx) then
                  if(fx(i).lt.0.0d0) then
                     fv(i) = -fv(i)

                  end if
               else
                  write(*,*) 'error'
                  stop
               end if
            end if
         end if
      end do
c
c     回転の運動量を求め、 0 になるように修正
c
      do i = n1 , n2
         wmvy = wmvy + weight(i) * fv(i) / r(i)
      end do
c
      wmvy = wmvy / ( n2 - n1 + 1 )
c
      do i = n1, n2
         dz(i) = dz(i) - wmvy / weight(i) * r(i) / fv(i) * fz(i)
         dx(i) = dx(i) - wmvy / weight(i) * r(i) / fv(i) * fx(i)
      end do
c
c     z 軸の回転成分を求める
c
      do i = n1 , n2
         ex = - y(i) + cy
         ey =   x(i) - cx
         fx(i) =  ( (dx(i)*ex + dy(i)*ey) / (ex**2+ey**2)) * ex
         fy(i) =  ( (dx(i)*ex + dy(i)*ey) / (ex**2+ey**2)) * ey
         fv(i) = sqrt( fx(i)**2 + fy(i)**2 )
      end do
c
c     符号の確認
c
      do i = n1 ,n2
c
         if(x(i).lt.cx) then
            if(fy(i).lt.0.0d0) then
               fv(i) = -fv(i)
            else
               if(fy(i).eq.0.0d0) then
                  write(*,*) 'error'
                  stop
               end if
            end if
         else
            if(x(i).gt.cx) then
               if(fy(i).gt.0.0d0) then
                  fv(i) = -fv(i)
               else
                  if(fy(i).eq.0.0d0) then
                     write(*,*) 'error'
                     stop
                  end if
```

```
               end if
            else
               if(x(i).eq.cx) then
                  if(fx(i).lt.0.0d0) then
                     fv(i) = -fv(i)

                  end if
               else
                  write(*,*) 'error'
                  stop
               end if
            end if
         end if
      end do
c
c     回転の運動量を求め、 0 になるように修正
c
      do i = n1 , n2
         wmvz = wmvz + weight(i) * fv(i) / r(i)
      end do
c
      wmvz = wmvz / ( n2 - n1 + 1 )
c
      do i = n1, n2
         dx(i) = dx(i) - wmvz / weight(i) * r(i) / fv(i) * fx(i)
         dy(i) = dy(i) - wmvz / weight(i) * r(i) / fv(i) * fy(i)
      end do
c
      return
      end
c
```

A.3　状態密度計算プログラム

ここには、1 原子のみの状態密度を求めるプログラムをのせる。

```
c
c      .grp から MO data を読み込むルーチン
c      指定した原子の DOS のみを出力
c
       implicit real*8(a-h,o-z)
       real*8 moh
       character*16 title
       character*32 jobname
       parameter(n1=130,nm1=n1*4,n2=131)
       dimension x(n1),y(n1),z(n1),xc(n2)
       dimension yall(n2),yspxpy(n2)
       dimension ys(n2),ypx(n2),ypy(n2),ypz(n2)
       dimension ng(n1),ms(n1),me(n1)
       dimension moh(nm1*nm1),amo(nm1,nm1),bmo(nm1,nm1),aei(nm1),bei(nm1)
       dimension amo2(nm1,nm1),bmo2(nm1,nm1)
c
c
c 指定原子の定義
c
       dimension amoall(nm1),bmoall(nm1),amospxpy(nm1),bmospxpy(nm1)
       dimension amos(nm1),bmos(nm1),amopx(nm1),bmopx(nm1)
       dimension amopy(nm1),bmopy(nm1),amopz(nm1),bmopz(nm1)
       dimension abmoall(nm1),abmospxpy(nm1)
       dimension abmos(nm1),abmopx(nm1)
       dimension abmopy(nm1),abmopz(nm1)
c
       call getarg(1,jobname)
c
c      指定原子の番号を入力
c

       write(*,*) ' Program Start!'
       write(*,*) ' 何番?? '
       read(*,1005) number
c
       j=index(jobname,' ')-1
       open(60,file=jobname(:j)//'.grp')
       read(60,*) title
       read(60,*) n,nm,ne
       if(n1.lt.n) stop 'n1.lt.n'
       read(60,*) (ng(i),i=1,n)
       do i=1,n
          read(60,*) x(i),y(i),z(i)
       end do
       read(60,*) (ms(i),me(i),i=1,n)
       read(60,*) (m,i=1,n*3)
       read(60,*) (moh(i),i=1,nm*nm)
       k=0
       do i=1,nm
          do j=1,nm
             k=k+1
             amo(i,j)=moh(k)
          end do
       end do
       read(60,*) (aei(i),i=1,nm)
       read(60,*) (moh(i),i=1,nm*nm)
       k=0
       do i=1,nm
          do j=1,nm
             k=k+1
             bmo(i,j)=moh(k)
          end do
       end do
       read(60,*) (bei(i),i=1,nm)
       close(60)
       write(*,*) 'Data Read Finish!!'
c
c      ここまでが data 読み込みルーチン
c
c-------------------------------------
c
c      ここから計算ルーチン
c

c
c      MO^2 の計算
c
       do i=1,nm
          do j=1,nm
             amo2(i,j)=amo(i,j)*amo(i,j)
             bmo2(i,j)=bmo(i,j)*bmo(i,j)
```

```
         end do
      end do
c
c     amo, の初期化
c
      do i=1,nm
c
         amoall(i)=0.0
         bmoall(i)=0.0
         amospxpy(i)=0.0
         bmospxpy(i)=0.0
         amos(i)=0.0
         bmos(i)=0.0
         amopx(i)=0.0
         bmopx(i)=0.0
         amopy(i)=0.0
         bmopy(i)=0.0
         amopz(i)=0.0
         bmopz(i)=0.0
         abmoall(i)=0.0
         abmospxpy(i)=0.0
         abmos(i)=0.0
         abmopx(i)=0.0
         abmopy(i)=0.0
         abmopz(i)=0.0
c
      end do
c
      do i= 1,nm
          j= number
               amos(i)=amos(i)+amo2(i,(j-1)*4+1)
               amopx(i)=amopx(i)+amo2(i,(j-1)*4+2)
               amopy(i)=amopy(i)+amo2(i,(j-1)*4+3)
               amopz(i)=amopz(i)+amo2(i,(j-1)*4+4)
               bmos(i)=bmos(i)+bmo2(i,(j-1)*4+1)
               bmopx(i)=bmopx(i)+bmo2(i,(j-1)*4+2)
               bmopy(i)=bmopy(i)+bmo2(i,(j-1)*4+3)
               bmopz(i)=bmopz(i)+bmo2(i,(j-1)*4+4)
      end do
c
c    合計値の計算
c
      do i=1,nm
         amospxpy(i)=amos(i)+amopx(i)+amopy(i)
         bmospxpy(i)=bmos(i)+bmopx(i)+bmopy(i)
         amoall(i)=amospxpy(i)+amopz(i)
         bmoall(i)=bmospxpy(i)+bmopz(i)
      end do
c
c   α軌道, β軌道 の和
c
      do i=1,nm
         abmoall(i)=amoall(i)+bmoall(i)
         abmospxpy(i)=amospxpy(i)+bmospxpy(i)
         abmos(i)=amos(i)+bmos(i)
         abmopx(i)=amopx(i)+bmopx(i)
         abmopy(i)=amopy(i)+bmopy(i)
         abmopz(i)=amopz(i)+bmopz(i)
      end do
c
c     Data 出力
c
c
c     d95.f より引用
c--------------------------

      dx=(abs(bei(1)-bei(nm))+10.0)/float(n2-1)
      de=dx
      write(*,*) 'de = ',de
      de2=de*de
      dd=3.0*de

      do i=1,n2
c
         xc(i)=float(int(bei(1))-5.0)+dx*float(i-1)
c
         ys(i)= 0.0
         ypx(i)= 0.0
         ypy(i)= 0.0
         ypz(i)= 0.0
         yspxpy(i)=0.0
         yall(i)=0.0
c
      end do
c
      do i= 1,nm
      do j=1,n2
         if(abs(bei(i)-xc(j)).lt.dd) then
      ys(j)=ys(j)+exp(-(bei(i)-xc(j))**2/de2)*abmos(i)
```

```
      ypx(j)=ypx(j)+exp(-(bei(i)-xc(j))**2/de2)*abmopx(i)
      ypy(j)=ypy(j)+exp(-(bei(i)-xc(j))**2/de2)*abmopy(i)
      ypz(j)=ypz(j)+exp(-(bei(i)-xc(j))**2/de2)*abmopz(i)
      yspxpy(j)=yspxpy(j)+exp(-(bei(i)-xc(j))**2/de2)*abmospxpy(i)
      yall(j)=yall(j)+exp(-(bei(i)-xc(j))**2/de2)*abmoall(i)
         endif
      end do
      end do
c--------------------------- c sc: Scaling Factor
      sc=1.0/(de*sqrt(3.1415926535))
      write(*,*) 'scaling factor = ',sc
c
      do i=1,n2
         ys(i)=ys(i)*sc
         ypx(i)=ypx(i)*sc
         ypy(i)=ypy(i)*sc
         ypz(i)=ypz(i)*sc
         yspxpy(i)=yspxpy(i)*sc
         yall(i)=yall(i)*sc
c
         sumys=sumys+ys(i)
         sumypx=sumypx+ypx(i)
         sumypy=sumypy+ypy(i)
         sumypz=sumypz+ypz(i)
         sumyspxpy=sumyspxpy+yspxpy(i)
         sumyall=sumyall+yall(i)
      end do
         sumys=sumys*dx
         sumypx=sumypx*dx
         sumypy=sumypy*dx
         sumypz=sumypz*dx
         sumyspxpy=sumyspxpy*dx
         sumyall=sumyall*dx
c
      write(*,*) 'Sum yall=8 ',sumyall
      write(*,*) 'Sum ys=2',sumys
      write(*,*) 'Sum ypx=2',sumypx
      write(*,*) 'Sum ypy=2',sumypy
      write(*,*) 'Sum ypz=2',sumypz
      write(*,*) 'Sum yspxpy=6 ',sumyspxpy
c
      j=index(jobname,' ')-1
c
      open(62,file=jobname(:j)//'-all.dat2')
      do i=1,n2
         write(62,1000) xc(i), yall(i)
      end do
      write(62,*) '#',xc(n2)
      close(62)

      open(63,file=jobname(:j)//'-spxpy.dat2')
      do i=1,n2
         write(63,1000) xc(i), yspxpy(i)
      end do
      write(63,*) '#',xc(n2)
      close(63)

      open(64,file=jobname(:j)//'-s.dat2')
      do i=1,n2
         write(64,1000) xc(i), ys(i)
      end do
      write(64,*) '#',xc(n2)
      close(64)

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

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

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


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

A.4　ドープ原子情報取り出しコマンド

これは フッ素の場合のコマンドである。他の原子について使用する場合は ” F ”を書き換えればよい。

```
#!/bin/csh
#
foreach i(‘/bin/ls *$file‘)
echo "#"$i
grep "  F  " $i
grep TOTAL  $i
end
echo final
```

A.5　out ファイル処理プログラム

```
c
c     計算結果を読み込み、処理する為のプログラム。
c     out ファイルから ドープ原子の位置、電荷、
c     系の TOTAL ENERGY を読み込み、加工するプログラム。
c
c     使い方
c
c     % use out
c     % fa X TOTAL > ! data
c     % out-read.out
c
c     で実行。
c
c     読み込むデータ
c
c     x(i,j) , y(i,j) , z(i,j)  : それぞれの xyz 座標
c     charge1(i,j) : 形式電荷
c     charge2(i,j) : 電子密度
c     energy(i)    : TOTAL ENERGY
c
c     ここでの i と j の意味は、
c
c      i : i 番目の out ファイルからのデータ ( i = 1 , m1 )
c      j : i 番目の out ファイルの j 番目の原子 ( j = 1 , atomcount(i) )
c
c     である。
c
c     出力させることのできるデータは、
c
c     data-l-z            :  グラファイトの中心からの距離と高さ
c     data-z-chrge        :  グラファイトのからの高さと電荷
c     data-total-charge   :  X 原子のドープ数と総電荷
c     data-l-charge       :  グラファイトの中心からの距離と電荷
c     data-n-energy       :  ドープ数と吸着エネルギー
c     data-energy-charge  :  総電荷と吸着エネルギー
c
c     が出力可能である。
c
      implicit real*8(a-h,o-z)
      character*16 tmp
      character*32 jobname
      character tmp1*1, tmp3*3
      character tmp2*10, tmp4*5
      integer atomcount
      parameter(n=1000)
      dimension mcount(n),tmp(n),atomcount(n)
      dimension x(n,n),y(n,n),z(n,n),charge1(n,n),charge2(n,n)
      dimension zz(n,n),a(n,n),tcharge(n)
      dimension energy(n),energy2(n)
c
      call getarg(1,jobname)
      jo=index(jobname,' ')-1
c
      write(*,*) ' Program Start!'
c
c     Li 原子数のカウント 1
c      ( 一つのファイルからのデータの行数を mcount(n) に読み込む)
c     読み込みファイル数は m1 に。
c
      m1 = 0
      m2 = 0
      open(60,file=jobname(:jo))
      read(60,*)
      do i = 1,10000
         read(60,'(a)') tmp1
         if ( tmp1.eq."f") then
            m1 = m1 + 1
            mcount( m1 ) = m2
            GOTO 20
         else
            if ( tmp1.eq."#") then
               m1 = m1 + 1
               mcount( m1 ) = m2
               m2 = 0
```

```
            else
               m2 = m2 + 1
            end if
         end if
      end do
 20   close(60)
c
c     Li 原子数のカウント 2
c     (正しく終了したものと、途中で終了したものと、
c     行われなかったものの判別を行う。)
c
      open(60,file=jobname(:jo))
      do i = 1 , m1
         read(60,*)
c
         do j = 1 , mcount(i)
            read(60,'(a)') tmp2
            if ( tmp2.eq."          ") then
c
c     結果の判別 ( 途中終了のものも含む )
c
               do k = 1 , 1000
                  read(60,'(a)') tmp4
                  atomcount(i) = atomcount(i) + 1
                  if ( tmp4.ne."     ") then
                     do kk = 1 , atomcount(i) + 2
                        read(60,*)
                     end do
                     goto 40
                  endif
               end do
            end if
c
         end do
c
c     行われなかったもの
c
         atomcount(i) = 0
 40   end do
      close(60)
c
c     データの読み込み
c
      open(60,file=jobname(:jo))
c
      do 70 i = 1, m1
         read(60,'(a)') tmp(i)
         job=index(tmp(i),' ')-1
         tmp(i) = tmp(i)(:job)
c
c     無駄な行 ( nnn ) のとばし
c
         nnn = mcount(i) - atomcount(i) * 2  - 3
c
         nnn2 = atomcount(i) * 3
         if(atomcount(i).eq.0.or.nnn2.ne.nnn) then
 60         do j = 1 ,mcount(i)
               read(60,*)
            end do
            atomcount(i) = 0
            goto 70
         end if
c
         do j = 1 , nnn
            read(60,*)
         end do
c
         do j = 1, atomcount(i)
            read(60,*) aaa,tmp3,charge1(i,j),charge2(i,j)
         end do
c
         do j = 1, atomcount(i)
            read(60,*) aaa,tmp3,x(i,j),y(i,j),z(i,j)
         end do
c
         read(60,*) tmp1 , tmp1 , tmp1 , energy(i)
         read(60,*)
         read(60,*)
c
 70   continue
      close(60)
c
c     読み込みデータの処理
c
c     z 座標の絶対値化
c
      do i = 1, m1
      do j = 1, atomcount(i)
      zz(i,j) = abs(z(i,j))
      end do
      end do
c
c   中心からの距離 l の計算
c
      do i = 1,m1
      do j = 1, atomcount(i)
      a(i,j) = sqrt( (x(i,j) - 0.7091) **2 + ( y(i,j) - 1.2282) **2)
      end do
```

```
      end do
c
c   total charge の計算
c
      do i= 1,m1
         tcharge(i) = 0.0d0
         if(atomcount(i).ne.0) then
            do j = 1, atomcount(i)
               tcharge(i) = tcharge(i) + charge1(i,j)
            end do
         end if
      end do
c
c     データ書き込み
c
      open(61,file=jobname(:jo)//'-z-charge')
      do i = 1 , m1
c     write(61,*) tmp(i)
         do j = 1 , atomcount(i)
            write(61,1001) zz(i,j),charge1(i,j)
         end do
      end do
      close(61)
c
      open(62,file=jobname(:jo)//'-l-z')
      do i = 1 , m1
c     write(62,*) tmp(i)
         do j = 1 , atomcount(i)
            write(62,1002) a(i,j),zz(i,j)
         end do
      end do
      close(62)
c
      open(63,file=jobname(:jo)//'-total-charge')
      do i = 1 , m1
         if(atomcount(i).ne.0) then
c     write(63,*) tmp(i)
            write(63,1003) atomcount(i),tcharge(i)
         end if
      end do
      close(63)
c
      open(64,file=jobname(:jo)//'-l-charge')
      do i = 1 , m1
c     write(64,*) tmp(i)
         do j = 1 , atomcount(i)
            write(64,1004) a(i,j),charge1(i,j)
         end do
      end do
      close(64)
c
c     吸着エネルギーの計算、出力
c
 90   write(*,*) '吸着エネルギーの計算も行いますか？'
      write(*,*) '必要なら１を'
      write(*,*) '終了させるなら２を入力してください'
      read(*,*) what
c
      if(what.eq.1) then
         call kenergy (m1,atomcount,energy,energy2)
c
c     出力
c
         open(65,file=jobname(:jo)//'-energy-n')
         do i = 1 , m1
            k = atomcount(i)
            if (k.ne.0) then
               write(65,1007) energy2(i) , k , tmp(i)
            end if
         end do
         close(65)
c
         open(66,file=jobname(:jo)//'-energy-charge')
         do i = 1,m1
            k = atomcount(i)
            if (k.ne.0) then
               write(66,1006) energy2(i) ,tcharge(i)
            end if
         end do
         close(66)
         open(67,file=jobname(:jo)//'-test')
         do i = 1,m1
            k = atomcount(i)
            if (k.ne.0) then
               write(67,2000) z(i,1),energy2(i)
            end if
         end do
         close(67)
C

c
         goto 100
         end if
c
      if(what.eq.2) goto 100
```

```
      goto 90
c
 100  write(*,*) 'Program END!'
c
 1000 format(f17.4,f10.4,f10.4,f12.6)
 1001 format(f17.4,f12.6)
 1002 format(f17.4,f10.4)
 1003 format(I10,f12.6)
 1004 format(f17.4,f12.6)
 1005 format(f17.4,f10.4,f12.6)
 1006 format(f10.5,f13.6)
 1007 format(f20.5,I7,'      ',a10)
 1010 format(f5)
 2000 format(f10.4,f15.5)
      stop
      end
c
c
c
      subroutine kenergy (m1,atomcount,energy,energy2)
c
c     吸着エネルギーを求めるプログラム
c
c     Ech  : 吸着前の構造の total energy
c     1 : C24H12(no-SYM) 、 2 : C24H12(SYM)
c     3 : C54H18(no-SYM) 、 4 : C54H18(SYM)
c
c     Ex  : 吸着する原子の total energy
c     1 : F、 2: Cl、 3 : Br、 4 : I、 5 : Li
c
c     atomcount   : 吸着した原子数
c
      implicit real*8(a-h,o-z)
      integer atomcount
      dimension energy(m1),energy2(m1)
      dimension atomcount(m1)
c
c     使用したグラファイトの特定
c
      write(*,*) ' C24H12(no-SYM)→1、 C24H12(SYM)→2,'
      write(*,*) ' C54H18(no-SYM)→3、 C54H18(SYM)→4,'
      write(*,*) ' another → 0 '
      read(*,*) ng
c
c     ドープした原子の特定
c
      write(*,*) 'F→1,Cl→2,Br→3,I→4'
      write(*,*) 'Li→5 '
      write(*,*) ' another → 0 '
      read(*,*) nx
c
      Ech1 = -3027.95421
      Ech2 = -3027.13237
      Ech3 = -6675.72785
      Ech4 = -6675.43907
c
      Ex1 = -437.51717
      Ex2 = -315.19495
      Ex3 = -352.5399
      Ex4 = -291.45509
      Ex5 = -5.30000
      Ex6 = 0.00000
c
      if(ng.eq.0) then
         write(*,*) ' 母材の TOTAL ENERGY ='
         read(*,*) Ech
      else
         if(ng.eq.1) Ech = Ech1
         if(ng.eq.2) Ech = Ech2
         if(ng.eq.3) Ech = Ech3
         if(ng.eq.4) Ech = Ech4
      end if
c
      if(nx.eq.0) then
         write(*,*) ' ドープ原子の TOTAL ENERGY ='
         read(*,*) Ex
      else
         if(nx.eq.1) Ex=Ex1
         if(nx.eq.2) Ex=Ex2
         if(nx.eq.3) Ex=Ex3
         if(nx.eq.4) Ex=Ex4
         if(nx.eq.5) Ex=Ex5
         if(nx.eq.6) Ex=Ex6
      end if
c
      if(Ech.eq.0.0d0) stop
      if(Ex.eq.0.0d0) stop
c
c     計算
c
      do i = 1 , m1
         k = atomcount(i)
         if (k.ne.0) then
            energy2(i) = -(Ech+Ex*k-energy(i)) / k
         end if
```

```
      end do
c
      return
      end

   newpage
```

A.6 arc ファイル処理プログラム

```
c
c     mopac arc ファイルから、電荷、内部座表による結合距離
c     を読み出すプログラム。
c
c     炭素 24 個 用
c
c     実行方法 ( X 原子 の場合 )
c
c     % use arc
c     % fa-X > ! data
c     % arc-read.out data
c
c     で実行。
c
c     data-total-cherge
c     data-in-distance-charge
c     data-out-distance-charge
c
c     が出力。
c
      implicit real*8(a-h,o-z)
      character*32 jobname
      character tmp1*1, tmp3*3
      parameter(n=1000)
      dimension x(n,n),y(n,n),z(n,n),charge(n,n),tcharge(n)
      dimension kk(n,n)
      dimension mcount(n)
      integer check(n,n),in(n),out(n)
      real*8 lengthin , lengthout
c
      call getarg(1,jobname)
      jo=index(jobname,' ')-1
c
      write(*,*) 'Program Start!'
c
      m1 = 0
      m2 = 0
      m3 = 0
      m4 = 0
      open(60,file=jobname(:jo))
      read(60,*)
      do i = 1,10000
         read(60,'(a)') tmp1
         if ( tmp1.eq."f") then
            m1 = m1 + 1
            mcount( m1 ) = m2
            GOTO 20
         else
            if ( tmp1.eq."#") then
               m1 = m1 + 1
               mcount( m1 ) = m2
               m2 = 0
            else
               m2 = m2 + 1
            end if
         end if
      end do
 20   close(60)
c
c     データ読み込み。
c
      open(60,file=jobname(:jo))
c
      do i = 1, m1
      read(60,*) tmp3
      do j = 1, mcount(i)
      read(60,1000) tmp3,x(i,j),k,y(i,j),k,z(i,j),k,
     &kk(i,j),k,k,charge(i,j)
      end do
      end do
      close(60)
c
c     結合位置のチェック
c     check(i,j) = 0 内
c     check(i,j) = 1 端
c
      do i = 1 , m1
         do j = 1,mcount(i)
            if(kk(i,j).le.7) check(i,j) = 0
            if(kk(i,j).eq.8) check(i,j) = 1
            if(kk(i,j).eq.9) check(i,j) = 1
            if(kk(i,j).eq.10) check(i,j) = 0
            if(kk(i,j).eq.11) check(i,j) = 1
            if(kk(i,j).eq.12) check(i,j) = 1
```

```
            if(kk(i,j).eq.13) check(i,j) = 0
            if(kk(i,j).eq.14) check(i,j) = 1
            if(kk(i,j).eq.15) check(i,j) = 1
            if(kk(i,j).eq.16) check(i,j) = 0
            if(kk(i,j).eq.17) check(i,j) = 1
            if(kk(i,j).eq.18) check(i,j) = 1
            if(kk(i,j).eq.19) check(i,j) = 0
            if(kk(i,j).eq.20) check(i,j) = 1
            if(kk(i,j).eq.21) check(i,j) = 1
            if(kk(i,j).eq.22) check(i,j) = 0
            if(kk(i,j).eq.23) check(i,j) = 1
            if(kk(i,j).eq.24) check(i,j) = 1
            if(kk(i,j).ge.25) stop
         end do
      end do
c
      do i = 1 , m1
         out(i) = 0
         in(i)  = 0
         do j = 1,mcount(i)
            if (check(i,j).eq.0) in(i)  = in(i)  + 1
            if (check(i,j).eq.1) out(i) = out(i) + 1
            if (check(i,j).ne.1.and.check(i,j).ne.0) stop
         end do
      end do
c
c   total charge の計算
c
      do i = 1,m1
         tcharge(i)=0.0d0
         do j=1,mcount(i)
            tcharge(i)=tcharge(i)+charge(i,j)
         end do
      end do
c
c     m1           : データ数
c     charge(i,j) : i 番目のデータの j 番目の原子の電荷
c     tcharge(i)  : i 番目のデータの総電荷
c     mcount(i)   : i 番目のデータのドープ原子数
c     kk(i,j)     : i 番目のデータの j 番目の原子の結合原子の番号
c
c     in(i)  : i 番目のデータの内部に結合した数
c     out(i)  : i 番目のデータの端に結合した数
c
c
c     データ出力
c     (ドープ原子数)  (X 原子の総電荷)
c
      open(61,file=jobname(:jo)//'-total-charge')
      do i = 1 , m1
         write(61,1001) mcount(i),tcharge(i)
      end do
      close(61)
c
c     データ出力
c     (結合距離)   (電荷)   (結合原子番号)
c     in  :  内部の炭素に結合
c     out :  端の炭素に結合
c
      open(62,file=jobname(:jo)//'-in-distance-charge')
      open(63,file=jobname(:jo)//'-out-distance-charge')
      do i = 1 , m1
         do j = 1,mcount(i)
            if (check(i,j).eq.0) then
               write(62,1002) x(i,j),charge(i,j),kk(i,j)
            end if
            if (check(i,j).eq.1) then
               write(63,1002) x(i,j),charge(i,j),kk(i,j)
            end if
            if (check(i,j).ne.1.and.check(i,j).ne.0) then
               stop
            end if

         end do
      end do
      close(62)
      close(63)
c
c     結合数と結合距離の平均
c
      open(64,file=jobname(:jo)//'-in-length')
      open(65,file=jobname(:jo)//'-out-length')
      do i = 1 , m1
         lengthin  = 0.0d0
         lengthout = 0.0d0
         do j = 1 , mcount(i)
            if(check(i,j).eq.0) then
               lengthin = lengthin + x(i,j)
            else
               lengthout = lengthout + x(i,j)
```

```
            end if
         end do
c
         if(in(i).ne.0) then
            lengthin = lengthin / in(i)
            write(64,1003) mcount(i) , lengthin
         end if
         if(in(i).ne.0) then
            lengthout = lengthout / out(i)
            write(65,1003) mcount(i) , lengthout
         end if
      end do
      close(64)
      close(65)
c
      open(66,file=jobname(:jo)//'-in-charge')
      open(67,file=jobname(:jo)//'-out-charge')
      do i = 1 , m1
         chargein  = 0.0d0
         chargeout = 0.0d0
         do j = 1 , mcount(i)
            if(check(i,j).eq.0) then
               chargein = chargein + charge(i,j)
            else
               chargeout = chargeout + charge(i,j)
            end if
         end do
c
         if(in(i).ne.0) then
            chargein = chargein / in(i)
            write(66,1003) mcount(i) , chargein
         end if
         if(in(i).ne.0) then
            chargeout = chargeout / out(i)
            write(67,1003) mcount(i) , chargeout
         end if
      end do
      close(66)
      close(67)
c
       write(*,*) ' Program END!'
c
 1000 format(a5,f13.8,I3,f13.7,I3,f13.7,I3,I5,I5,I5,f12.4)
 1001 format(I17,f13.5)
 1002 format(f17.4,f12.4,I5)
 1003 format(I7,f17.4,f12.4,I5)
      stop
      end
```

付録 B

MOPAC の入力 DATA

MOPAC の入力ファイルの例として、活性化エネルギーを求めるための入力データを示す。 B.1では、 $C_{54}H_{18}$ のグラファイト上の Li の活性化エネルギーを求める入力データ、 B.2では、 $C_{24}H_{12}$ のグラファイト上の F の活性化エネルギーを求める入力データを示す。

B.1　リチウムの活性化エネルギー用入力データ

n= 0.0 ～ 8.0 、 m = 30.0 ～ 60.0 の中で任意の値を入力し計算する。

```
 SYMMETRY T=1.0D NOINTER GNORM=0.5 &
 PM3 GEO-OK UHF PULAY SHIFT=2
 Graphite Li symmetry adopted MOPAC coodrdinates
C      0.00000000   0    0.00000000   0    0.00000000   0   0   0   0
C      1.41815137   1    0.00000000   0    0.00000000   0   1   0   0
C      1.41815137   0  120.00000000   0    0.00000000   0   2   1   0
C      1.41815137   0  120.00000000   0    0.00000000   0   3   2   1
C      1.41815137   0  120.00000000   0    0.00000000   0   4   3   2
C      1.41815137   0  120.00000000   0    0.00000000   0   5   4   3
C      1.41815137   0  120.00000000   0  180.00000000   0   1   2   3
C      1.41815137   0  120.00000000   0    0.00000000   0   7   1   2
C      1.41815137   0  120.00000000   0    0.00000000   0   8   7   1
C      1.41815137   0  120.00000000   0  180.00000000   0   2   3   4
C      1.41815137   0  120.00000000   0    0.00000000   0  10   2   3
C      1.41815137   0  120.00000000   0    0.00000000   0  11  10   2
C      1.41815137   0  120.00000000   0  180.00000000   0   3   4   5
C      1.41815137   0  120.00000000   0    0.00000000   0  13   3   4
C      1.41815137   0  120.00000000   0    0.00000000   0  14  13   3
C      1.41815137   0  120.00000000   0  180.00000000   0   4   5   6
C      1.41815137   0  120.00000000   0    0.00000000   0  16   4   5
C      1.41815137   0  120.00000000   0    0.00000000   0  17  16   4
C      1.41815137   0  120.00000000   0  180.00000000   0   5   6   1
C      1.41815137   0  120.00000000   0    0.00000000   0  19   5   6
C      1.41815137   0  120.00000000   0    0.00000000   0  20  19   5
C      1.41815137   0  120.00000000   0  180.00000000   0   6   1   2
C      1.41815137   0  120.00000000   0    0.00000000   0  22   6   1
C      1.41815137   0  120.00000000   0    0.00000000   0  23  22   6
C      1.41815137   0  120.00000000   0  180.00000000   0   8   7   1
C      1.41815137   0  120.00000000   0    0.00000000   0  25   8   7
C      1.41815137   0  120.00000000   0    0.00000000   0  26  25   8
C      1.41815137   0  120.00000000   0  180.00000000   0   9   8   7
C      1.41815137   0  120.00000000   0    0.00000000   0  28   9   8
```

```
C      1.41815137   0  120.00000000   0    0.00000000   0  29  28   9
C      1.41815137   0  120.00000000   0    0.00000000   0  11  10   9
C      1.41815137   0  120.00000000   0    0.00000000   0  31  11  10
C      1.41815137   0  120.00000000   0  180.00000000   0  12  11  10
C      1.41815137   0  120.00000000   0    0.00000000   0  33  12  11
C      1.41815137   0  120.00000000   0    0.00000000   0  34  33  12
C      1.41815137   0  120.00000000   0    0.00000000   0  14  13  12
C      1.41815137   0  120.00000000   0    0.00000000   0  36  14  13
C      1.41815137   0  120.00000000   0  180.00000000   0  15  14  13
C      1.41815137   0  120.00000000   0    0.00000000   0  38  15  14
C      1.41815137   0  120.00000000   0    0.00000000   0  39  38  15
C      1.41815137   0  120.00000000   0    0.00000000   0  17  16  15
C      1.41815137   0  120.00000000   0    0.00000000   0  41  17  16
C      1.41815137   0  120.00000000   0  180.00000000   0  18  17  16
C      1.41815137   0  120.00000000   0    0.00000000   0  43  18  17
C      1.41815137   0  120.00000000   0    0.00000000   0  44  43  18
C      1.41815137   0  120.00000000   0    0.00000000   0  20  19  18
C      1.41815137   0  120.00000000   0    0.00000000   0  46  20  19
C      1.41815137   0  120.00000000   0  180.00000000   0  21  20  19
C      1.41815137   0  120.00000000   0    0.00000000   0  48  21  20
C      1.41815137   0  120.00000000   0    0.00000000   0  49  48  21
C      1.41815137   0  120.00000000   0    0.00000000   0  23  22  21
C      1.41815137   0  120.00000000   0    0.00000000   0  51  23  22
C      1.41815137   0  120.00000000   0  180.00000000   0  51  23  22
C      1.41815137   0  120.00000000   0    0.00000000   0  53  51  23
H      1.09441126   1  120.00000000   0  180.00000000   0  26  27  25
H      1.09441126   0  120.00000000   0  180.00000000   0  30  25  29
H      1.09441126   0  120.00000000   0  180.00000000   0  29  30  28
H      1.09441126   0  120.00000000   0  180.00000000   0  32  28  31
H      1.09441126   0  120.00000000   0  180.00000000   0  35  31  34
H      1.09441126   0  120.00000000   0  180.00000000   0  34  35  33
H      1.09441126   0  120.00000000   0  180.00000000   0  37  33  36
H      1.09441126   0  120.00000000   0  180.00000000   0  40  36  39
H      1.09441126   0  120.00000000   0  180.00000000   0  39  40  38
H      1.09441126   0  120.00000000   0  180.00000000   0  42  38  41
H      1.09441126   0  120.00000000   0  180.00000000   0  45  41  44
H      1.09441126   0  120.00000000   0  180.00000000   0  44  45  43
H      1.09441126   0  120.00000000   0  180.00000000   0  47  43  46
H      1.09441126   0  120.00000000   0  180.00000000   0  50  46  49
H      1.09441126   0  120.00000000   0  180.00000000   0  49  50  48
H      1.09441126   0  120.00000000   0  180.00000000   0  52  48  51
H      1.09441126   0  120.00000000   0  180.00000000   0  53  51  54
H      1.09441126   0  120.00000000   0  180.00000000   0  54  53  27
XX     1.41815137   0   60.00000000   0    0.00000000   0   1   2   3
XX     n            0    m            0    0.00000000   0  73   1   3
Li     2.00000000   1   90.00000000   0   90.00000000   0  74  73   4

 2 1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
 2 1 19 20 21 22 23 24 23 24 25 26 27 28 29 30 31 32
 2 1 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
 2 1 49 50 51 52 53 54 73
 55 1 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 55 1 72
```

B.2　ハロゲンの活性化エネルギー測定用入力データ

n = 1.5 ～ 4.0 で任意の値を入力し計算する。

```
T=1.OD NOINTER GNORM=0.1 PM3 GEO-OK PULAY SHIFT=2 UHF
Graphite + F active Energy
neutral
 XX    0.00000000  0    0.0000000  0    0.0000000  0    0    0    0
 XX    4.50000000  0    0.0000000  0    0.0000000  0    1    0    0
 XX    1.00000000  0   90.0000000  0    0.0000000  0    1    2    0
 XX    1.00000000  0    0.0000000  0   90.0000000  0    1    2    3
 XX    2.01907439  1    0.0000000  0   90.0000000  0    2    1    3
 C     1.79000000  1   42.8800000  1   90.0000000  0    2    1    3
 C     1.41000000  1  120.0000000  1    0.0000000  1    6    5    4
 C     1.41000000  1  120.0000000  1    0.0000000  1    7    6    5
 C     1.25600000  1   76.4100000  1  -90.0000000  0    1    2    3
 C     1.25600000  1   76.4100000  1   90.0000000  0    1    2    3
 C     1.41000000  1  120.0000000  1    0.0000000  1   10    4    5
 C     1.41000000  1  120.0000000  1    0.0000000  1   11   10    4
 C     1.79000000  1   42.8800000  1  -90.0000000  0    2    1    3
 C     1.41000000  1  120.0000000  1    0.0000000  1   13    5    6
 C     1.41000000  1  120.0000000  1    0.0000000  1   14   13    5
 C     1.41000000  1  120.0000000  1  180.0000000  1    6    7    8
 C     1.41000000  1  120.0000000  1    0.0000000  1   16    6    7
 C     1.41000000  1  120.0000000  1    0.0000000  1   17   16    6
 C     1.41000000  1  120.0000000  1  180.0000000  1    7    8    9
 C     1.41000000  1  120.0000000  1    0.0000000  1   19    7    8
 C     1.41000000  1  120.0000000  1    0.0000000  1   20   19    7
 C     1.41000000  1  120.0000000  1  180.0000000  1    8    9    4
 C     1.41000000  1  120.0000000  1    0.0000000  1   22    8    9
 C     1.41000000  1  120.0000000  1    0.0000000  1   23   22    8
 C     1.41000000  1  120.0000000  1  180.0000000  1    9    4    5
 C     1.41000000  1  120.0000000  1    0.0000000  1   25    9    4
 C     1.41000000  1  120.0000000  1    0.0000000  1   26   25    9
 H     1.09700000  1  120.0000000  1  180.0000000  1   11   10   12
 H     1.09700000  1  120.0000000  1  180.0000000  1   12   11   13
 H     1.09700000  1  120.0000000  1  180.0000000  1   14   13   15
 H     1.09700000  1  120.0000000  1  180.0000000  1   15   14   16
 H     1.09700000  1  120.0000000  1  180.0000000  1   17   16   18
 H     1.09700000  1  120.0000000  1  180.0000000  1   18   17   19
 H     1.09700000  1  120.0000000  1  180.0000000  1   20   19   21
 H     1.09700000  1  120.0000000  1  180.0000000  1   21   20   22
 H     1.09700000  1  120.0000000  1  180.0000000  1   23   22   24
 H     1.09700000  1  120.0000000  1  180.0000000  1   24   23   25
 H     1.09700000  1  120.0000000  1  180.0000000  1   26   25   27
 H     1.09700000  1  120.0000000  1  180.0000000  1   27   26   10
XX     n           0    0.0000000  0    0.0000000  0    2    1    3
 Br    2.00000000  1   90.0000000  0    0.0000000  0   40    2    3
 C     0.00000000  1   90.0000000  0    0.0000000  0    4    2    3
 C     0.00000000  1   90.0000000  0    0.0000000  0    5    2    3
```

付録 C

著者の学外における発表実績

- 論文発表
  - **Electronic structure of fluorine doped graphite nanoluster**
    R.Saito, M.Yagi, T.Kimura, G.Dresselhaus, M.S.Dresselhaus
    J. Phys. Chem. Solid (accepted) Dec. 14(1998)
  - **Electronic States in Heavily Li-dopes Graphite Nanoclusters**
    M.Yagi, R.Saito, T.Kimura, G.Dresselhaus, M.S.Dresselhaus
    Submitted J. Mater. Res. Feb. 15(1999)
- 学会発表
  - フッ素ドープしたナノグラファイトの電子構造
    齋藤 理一郎・八木 将志
    1997 年 12 月　愛媛大学　重点領域研究「カーボンアロイ」
  - フッ素ドープナノグラファイトクラスターの電子状態
    M.Yagi,R.Saito,T.Kimura
    1998 年 3 月　日本物理学会年会
  - ドナー、アクセプター型ドープ微小黒鉛クラスターの電子状態
    齋藤 理一郎・八木 将志
    1998 年 7 月　国際キノコ会館　重点領域研究「カーボンアロイ」
  - ドナー、アクセプター型ドープ微小黒鉛クラスターの電子状態
    M.Yagi, A.Tashiro, R.Saito, T.Kimura
    1998 年 9 月　日本物理学会秋の分科会

- Li and F Doped Graphite Nanoclusters
  R.Saito, M.Yagi, A.Tashiro, T.Kimura
  1998，炭素，Tokyo
- グラファイト微結晶での新機能探索
  齋藤 理一郎，八木 将志
  1999 年 2 月　千葉大学　重点領域研究「カーボンアロイ」
- ハロゲン原子とグラファイト微結晶の化学反応
  M.Yagi, R.Saito, T.Kimura
  1999 年 1 月　第 16 回フラーレン総合シンポジウム