J. Chem. Software, Vol. 6, No. 3, p. 85–104 (2000)

分散メモリ型並列計算機に適した新しい大規模フォック行列生成アルゴリズム – 積分カットオフとの関連 –

高島 一[a], 山田 想[b], 小原 繁[c], 北村 一泰[a], 稲畑 深二郎[b], 宮川 宣明[b], 田辺 和俊[d], 長嶋 雲兵[e]*

[a] 大正製薬創薬研究所, 〒 330-8530 埼玉県大宮市吉野町 1-403
[b] 富士ゼロックス総合研究所, 〒 243-0494　神奈川県海老名市本郷 2274 番地
[c] 北海道教育大学教育学部, 〒 085-8580　北海道釧路市城山 1 丁目 15-55
[d] 物質工学工業技術研究所, 〒 305-8565　茨城県つくば市東 1-1
[e] 産業技術融合領域研究所, 〒 305-8562　茨城県つくば市東 1-1-4
**e-mail: umpei@nair.go.jp*

(Received: December 10, 1999; Accepted for publication: January 7, 2000; Published on Web: April 10, 2000)

分散メモリ型並列計算機上での大規模分子軌道計算で、計算時間短縮に大きな効果をもたらす積分カットオフの問題に注目し、積分カットオフの閾値が計算結果に与える影響や、大規模分子における積分カットオフの割合について調べた。大規模分子では、２つの添字だけで行なうカットオフによって生き残る基底数の割合が10%以下となりほぼ一定値になること、これにより、積分数は事実上基底数 N の二乗に減少することが判明した。よって、ホストと各プロセッサの間で行列転送を行なう際には積分カットオフに対応した転送行列のカットオフを行なう事が通信量やメモリ量の軽減に必須であることが分かった。

キーワード : Distributed Memory Parallel Computer, Ab Initio Molecular Orbital Calculation, Fock Matrix Generation, Integral Cutoff

1　はじめに

非経験的分子軌道法は、その原理から考えてさまざまな機能性分子の設計や開発に対して最も基盤的でかつ重要な手法である。しかし、その計算量が膨大であるため、設計したい大きさの大規模分子ではなくそれを簡素化した小規模モデル分子への適用がせいぜいである。実際、数年前までは数百基底（数十原子）の分子軌道計算でさえ多大な労力を必要としていた。そのため、分子設計に必要とする情報をなかなか十分与える事ができないでいる。ここ数年の計算

機の進歩、特にスーパーコンピュータや高性能ワークステーションのような大規模並列計算機の進歩により、ようやく（生体）高分子を意識した 1,000 基底を超える分子軌道計算 [1, 2, 9, 10, 14] が行われつつあるが、これらの計算機は高価であり設備の維持管理も大変であるため、研究者が研究室レベルで大規模計算を実現することは容易なことではない。そこで、我々は、「現実を反映した大規模分子系」の分子軌道計算を、「低コスト＝パーソナルユース」で実現すべく、Figure 1 に示すような大規模分子をターゲットとした分散小メモリ型並列アーキテクチャである分子軌道計算専用並列計算機 MOE（Molecular Orbital calculation Engine）の開発 [7, 11, 12] に現在取り組んでいる。

非経験的分子軌道計算において最も時間を要するのが二電子積分を計算しフォック行列を生成する部分であるため、大規模計算を高速化する際にはこの部分の並列化が肝要である。これまでに、スーパーコンピュータやワークステーションクラスタを用いた多くの並列分子軌道計算が行われているが、その殆どは、（並列）各マシンに全ての密度行列及び全てのフォック行列を持たせて従来のフォック行列作成アルゴリズムをそのまま並列化した、いわゆる共有メモリ型アルゴリズムを採用している。しかし、このような方法だと、例えば 100 残基程度の蛋白質に相当する 10,000 次元の計算に必要な密度行列及びフォック行列全てを保持するために、各プロセッサ毎に最低 800MB ものメモリが必要になる。しかも、必要となるメモリ容量は系のサイズの二乗に比例して増大するため、莫大なメモリ量を必要とする。このように、従来のアルゴリズムでは、計算機も高価なものとなりスケーラビリティに乏しい方法である。

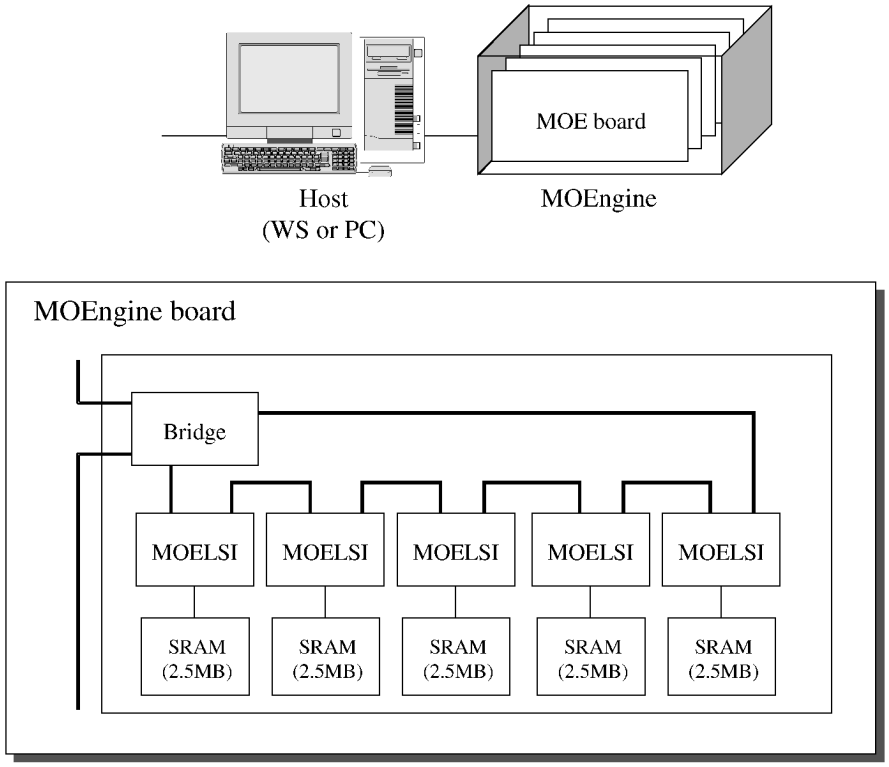

Figure 1. MOEngine, a special purpose machine for accelerating molecular orbital calculations.

現在開発中の専用計算機 MOEngine では、安価なコストで高速計算を実現するため搭載メモリ量（や設備の規模）を小さくしている。そのため各並列プロセッサに全ての行列要素を搭載できないので、MOEngine で大規模フォック行列を計算するためには、ホストと各プロセッサの間で行列要素の通信を行なう並列分散アルゴリズムが必要になる。その際、大規模分子軌道計算で重要な問題となる二電子積分のカットオフやそれに伴う行列要素転送のカットオフに関する考察が十分でなければ実用性に乏しくなる。これまでにフォック行列の並列分散アルゴリズムはいくつか [3, 4] 提唱されてはいるが、いずれもアーキテクチャを特定していたり通信面やカットオフの面の考察が十分でなかったりするため、蛋白質の様な大規模計算に適用するのは難しい。

そこで我々は、各専用計算機 MOEngine のような各プロセッサのメモリが小容量（数 MByte）しかない分散メモリ型並列アーキテクチャのもとで大規模フォック行列を計算するための新規アルゴリズムを考案した。本論文ではまず、大規模分子系で重要となる積分カットオフについて、以下の順序で説明する。最初に、分子軌道計算及び二電子積分とフォック行列の計算方法について概要を示す。次に、基底関数について、「縮約 (contracted)」や「原始 (primitive)」など以後の議論の理解に必要となるいくつかの言葉の定義を説明しておく。そして、二電子積分のカットオフの方法、並列フォック行列アルゴリズムにおける行列要素転送のカットオフとの関係、及び、大規模計算における積分カットオフの重要性、について詳細に述べる。なお、続く第二報 [17] では、我々が *RT* 並列アルゴリズムと呼んでいる新規フォック行列計算アルゴリズムの詳細について述べる。

2　非経験的分子軌道法について

本章では非経験的分子軌道法について概略する。計算の出発点は次の Schrödinger 方程式である。

$$\mathsf{H}\Psi = E\Psi \tag{2.1}$$

$$\mathsf{H} = -\sum_i \frac{1}{2}\nabla_i^2 - \sum_i \sum_A \frac{Z_A}{r_{Ai}} + \sum_{i\neq j} \frac{1}{r_{ij}} + \sum_{A\neq B} \frac{Z_A Z_B}{R_{AB}} \tag{2.2}$$

H はハミルトニアン、i, j は電子の番号、A, B は原子核の番号、Z_A は原子核 A の電荷（原子番号）を表す。式 (2.2) は第 1 項から順に電子の運動エネルギー、原子核による引力エネルギー、電子間クーロン反発エネルギー、原子核間の反発エネルギーを表す。また、式 (2.1) の E（エネルギー固有値）は分子の全エネルギーを、Ψ は分子内の全ての電子状態を表す全波動関数である。なお、単位は全て原子単位 (atomic unit) である。

Schrödinger 方程式は多体の線形方程式であり解析的に解くのは不可能なため、これを非線形の一体問題に近似して計算する。簡単のため、以下全て電子数 $2n$ の閉殻系を考える。まず、全波動関数 Ψ を次のようなスレーター行列式で表現する。

$$\Psi = |\phi(1)\phi(2)\Lambda\phi(2n)> \tag{2.3}$$

$\phi(1),\phi(2),\cdots,\phi(2n)$ は、1,2,$\cdots$,$2n$ 番目の電子の一電子分子軌道である。さらに、a 番目の分子軌道 ϕ_a を式 (2.3) のように原子軌道 χ の一次結合で表す。

$$\phi_a = \sum_{I=1}^{N} C_{aI}\chi_I \tag{2.4}$$

N は原子軌道の総数である。原子軌道の組は基底関数系（basis set）と呼ばれ、分子軌道計算を行なう際に入力データとして与えるものである。基底関数は、その選び方により計算の精度が左右される重要な因子であり、通常 $\exp(-\zeta r^2)$ で表される GTO（Gaussian Type Orbital）の線形結合で表現される。基底関数については次章で詳述する。式 (2.4) のような近似を、LCAO 近似（Linear Combination of Atomic Orbital）と呼ぶ。

式 (2.2) 〜式 (2.4) を式 (2.1) に代入し、式 (2.1) の全エネルギー E が最小（極小）となるような Ψ、すなわち式 (2.4) の係数行列**C** を求める手法の一つとして、Hartree-Fock-Roothaan の変分法 [13] が知られている。導出過程は成書 [16] にゆずり、変分法を用いた結果として次のような式が得られる。

$$\sum_{I=1}^{N} F_{IJ}C_{aI} = \sum_{I=1}^{N} S_{IJ}C_{aI}\varepsilon_a \tag{2.5}$$

ε_a は a 番目の軌道の軌道エネルギーである。このように、フォック行列**F** を対角化することにより、固有ベクトルである係数行列**C** と固有値であるエネルギー行列 ε を得る。フォック行列**F** 及び重なり行列**S** の各要素は、それぞれ以下の様に表される。

$$F_{IJ} = T_{IJ} + V_{IJ} + G_{IJ} \tag{2.6}$$ フォック行列

$$S_{IJ} = \int \chi_I(1)\chi_J(1)dr_1 \tag{2.7}$$ 重なり行列

$$T_{IJ} = -\frac{1}{2}\int \chi_I(1)\Delta\chi_J(2)dr_1 \tag{2.8}$$ 運動エネルギー積分

$$V_{IJ} = -\sum_A \int \chi_I(1)\frac{Z_A}{r_{1A}}\chi_J(1)dr_1 \tag{2.9}$$ 核引力積分

$$G_{IJ} = \sum_{K=1}^{N}\sum_{L=1}^{N} P_{KL}\{(IJ,KL) - \frac{1}{2}(IK,JL)\} \tag{2.10}$$ フォック行列（二電子項）

$$P_{IJ} = 2\sum_{a}^{occ} C_{aI}C_{aJ} \tag{2.11}$$ 密度行列

$$(IJ,KL) = \iint \chi_I(1)\chi_J(1)\frac{1}{r_{12}}\chi_K(2)\chi_L(2)dr_1dr_2 \tag{2.12}$$ 二電子積分

ここで、式 (2.10) の第一項はクーロン項、第二項は交換項と呼ばれている。式 (2.12) の添字 1 と 2 は、それぞれ電子 1 と電子 2 を表す。二電子積分は、その式の形から分かる通り、*I* と *J*、*K* と *L*、*IJ* と *KL* の入れ替えに対する対称性を有する。

(2.13) $(IJ,KL)=(IJ,LK)=(JI,KL)=(JI,LK)=(KL,IJ)=(KL,JI)=(LK,IJ)=(LK,JI)$

エネルギー *E* 及び係数行列**C** を求めるには、式 (2.6) ～式 (2.12) を用いてフォック行列を求めて式 (2.5) を解けばよい。しかし、フォック行列にはあらかじめ答えとなるべき係数行列**C** が入っている。そこで、実際の計算手順としては、Figure 2 に示すように、係数行列**C** の初期値を適当に与えて、式 (2.5) 及び式 (2.6) の計算を、係数行列**C**（正確には密度行列**P**）が収束するまで繰り返すことになる。収束後得られた状態は分子内部で矛盾のない場（Self Consistent Field）を作っていると考えられるので、この計算方法は SCF 法とも呼ばれる。

二電子積分は SCF 繰り返し計算のたびに用いられるが、その値は常に同じである。そこで、従来は最初に計算したものをディスクに保存しておき必要になるたびにディスクから読み出す方法（ディスクストレージ法）が主流であった。しかし、二電子積分の数は原理上基底数 *N* の 4 乗に比例して増加するためディスク容量が制限される事と、計算機の発達により演算時間とディスクアクセス時間が拮抗してきたため、現在では、SCF 計算で必要になるたびに二電子積分を計算し直し求まった二電子積分をすぐにフォック行列に足し込むダイレクト法 [5] が主流である。

いくつかのペプチド分子について実際の非経験的分子軌道計算を行った時の各ステップ別の計算時間結果を Table 1 に示す。計算したペプチド分子は G(Gly: 55 軌道), GA(Gly-Ala:110 軌道), GAQ(Gly-Ala-Gln: 207 軌道)、GAQM(Gly-Ala-Gln-Met: 316 軌道)、GAQMY(Gly-Ala-Gln-Met-Tyr:417 軌道) の 5 種類である。計算は SGI-CRAY 社製 R10000 を用いて行い、基底関数は 4-31G を用いた。全計算時間の実に 95 ～ 99%以上を占めるのが Figure 2 のステップ 3)、すなわち式 (2.12) で二電子積分を求めて式 (2.10) を計算するステップである。非経験的分子軌道計算の超高速化を実現するためには、いかに二電子積分を並列で高速に計算するかが重要であるかが分かる。また、基底数が多くなる（系のサイズが大きくなるか、もしくは精度の良い基底関数を使う）に従って二電子積分の計算時間の割合が大きくなる傾向が強まると言える。

3　基底関数

本章では、基底関数とそれに関連し後の議論を理解するために必要な事項について詳細に説明する。式 (2.4) で与えられている原子軌道 χ は基底関数とも呼ばれ、一般的には式 (3.1) で表すようなガウス関数の線形結合で表される。

$$\chi = (x-R_x)^{n_x}(y-R_y)^{n_y}(z-R_z)^{n_z}\sum_m d_m \exp[-\zeta_m(\mathbf{r}-\mathbf{R})^2] \qquad (3.1)$$

r、**R**、**n** はベクトルであり、それぞれ**r**=(*x*, *y*, *z*)、**R**=(R_x, R_y, R_z)、**n**=(n_x, n_y, n_z) で表される。**r** は電子の座標、**R** は原子核の座標、**n** は電子の角運動量である。なお、**n** の各成分はいずれもゼロ以上の整数値である。

$$\lambda = n_x + n_y + n_z \qquad (3.2)$$

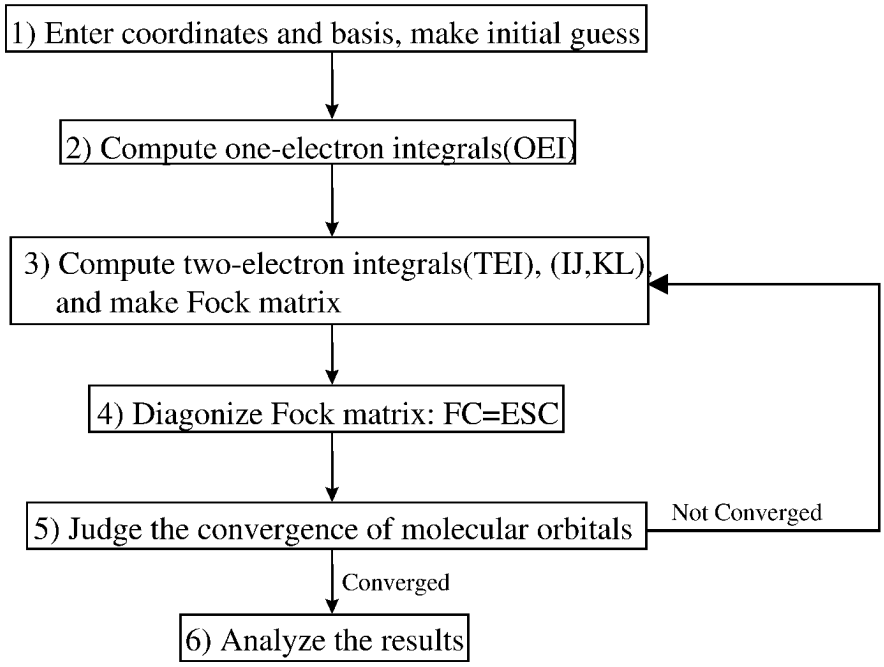

Figure 2. Typical flow charts for an ab initio molecular orbital calculation.

Table 1. Computing times in eeach SCF calculation step for some peptide molecules(sec).

	G	GA	GAQ	GAQM	GAQMY
Number of dimensions	55	110	207	316	427
Initial guess	0.1	0.7	4.4	15.0	43.7
One-Electron Integral	0.1	0.5	2.3	7.8	14.4
Two-Electron Integral and Fock Matrix	**21.9 (96.9%)**	**268.5 (98.8%)**	**2241.3 (99.1%)**	**10994.7 (99.4%)**	**32624.7 (99.4%)**
Diagonalization	0.1	1.0	6.9	28.4	103.3
Properties	0.1	0.5	2.8	9.7	24.2
Others	0.3	0.6	3.0	9.5	25.1
Total	**22.6**	**271.8**	**2260.7**	**11065.1**	**32835.4**

G: Gly, GA:Gly-Ala, GAQ: Gly-Ala-Gln, GAQM: Gly-Ala-Gln-Met, GAQMY: Gly-Ala-Gln-Met-Tyr

で表されるλは角運動量の大きさであり、「軌道量子数」とも呼ばれる。λが 0 の場合はその軌道を s 軌道、1 の場合は p 軌道、以下、d, f, g, h 軌道などと呼ぶ。ζ_m は軌道指数であり、軌道の空間的な広がりを示す。式 (3.1) で表すように軌道指数の異なる複数の軌道の線形結合で一つの基底関数を表す場合があり、そのようにして表された基底関数を「縮約基底関数（contracted basis）」と呼ぶ。その際の線形結合定数 d_m を「縮約係数（contracted coefficient）」と呼ぶ。軌道指数及び縮約係数はいずれも入力から与えられる。一方、縮約される前の、

$$\varphi = (x-R_x)^{n_x}(y-R_y)^{n_y}(z-R_z)^{n_z}\exp[-\zeta_m(\mathbf{r}-\mathbf{R})^2] \tag{3.3}$$

で表される関数 φ を「原始基底関数（primitive basis）」と呼ぶ。本論文では、縮約基底関数 χ を *I,J,K,L* のように番号付けし、また、原始基底関数 φ を *i,j,k,l* のように番号付けしている。

次に、「シェル (shell)」について説明する。軌道量子数が 1 の場合の縮約基底関数には、**n**=(1,0,0)、**n**=(0,1,0)、**n**=(0,0,1)、の 3 通りが存在する。同様に、軌道量子数が 2 の場合には 6 通りが存在する。そこで、式 (3.1) のうち、

$$\sum_m d_m \exp[-\zeta_m(\mathbf{r}-\mathbf{R})^2] \tag{3.4}$$

部分が共通で電子の角運動量 **n** だけが異なるものを「縮約シェル（contracted shell）」と呼ぶ。即ち、s 軌道の縮約シェルは 1 つの縮約基底関数で、p 軌道は 3 つの縮約基底関数で、それぞれ構成される。また、式 (3.3) のうちで

$$\exp[-\zeta_m(\mathbf{r}-\mathbf{R})^2] \tag{3.5}$$

部分が共通で電子の角運動量 **n** だけが異なるものを「原始シェル（primitive basis）」と呼ぶ。本論文では、縮約シェルを *R,S,T,U* のように、また、原始シェルを *r,s,t,u* のように番号付けする。

一般に、基底関数とは縮約基底関数の事をさし、式 (2.4) ～式 (2.12) やそこに登場する密度行列やフォック行列なども全て縮約基底関数をもとに記述されている。また、計算の次元数 *N* とは縮約基底関数の総数を指す。

基底関数の例を幾つか示しておく（[6] 等）。STO-*n*G 系は最小基底関数系ともよばれ、最も精度の荒い基底関数系であり、1 個の Slater Type Orbital（STO）を *n* 個の GTO の和である Contracted GTO（CGTO）で近似する方法である。STO-3G などが用いられる。4-31G や 6-31G などは二倍原子価殻基底関数系であり、内殻には 4 または 6 個の GTO からなる一個の CGTO を、原子価殻には、3 個の GTO からなる CGTO と 1 個の GTO からなる CGTO の二つを準備する方法で、最外殻軌道は分子環境の変化に応じた軌道の変化が可能である。さらに軌道の変化を取り込む方法として、電子雲の歪みを表すために一つ軌道量子数の高い分極関数を重原子の原子価殻に取り込んだ 6-31G*や水素原子にも取り込んだ 6-31G**、より精度の高い計算のため最外殻基底に対し三つ以上の CGTO を用いる方法など、多種多様の基底関数が存在する。基底関数を大きくとることによって計算精度は向上するが、計算時間が基底数の 4 乗に比例して増大する。そのため、対象分子の大きさ、得たい信頼度、計算時間などを考慮して最適の基底関数系を選択する必要がある。

4　積分カットオフ

4.1　概論

実際に非経験的分子軌道計算を行なう際に、特に大規模系の計算において計算時間を短縮するために日常的に使われている二電子積分のカットオフについて述べる。計算する二電子積分の個数を少なくする手段としては、式 (2.13) で示した二電子積分の対称性を利用しユニークな値を持つ二電子積分のみを計算する方法があるが、それ以外に、値が非常に小さいことをあらかじめ知る事ができる二電子積分の計算を積極的に排除する方法も広く用いられる。これが積分のカットオフと呼ばれる方法 [5] である。大規模系における積分カットオフの効果は非常に大きい。その手法を以下に説明する。

式 (2.12) のように縮約基底関数で記述された二電子積分は、原始基底関数を用いると、次式のように記述できる。

$$(IJ,KL) = \sum_{m1}\sum_{m2}\sum_{m3}\sum_{m4} d_{m1}d_{m2}d_{m3}d_{m4} \times g(i,j,k,l) \tag{4.1}$$

$$g(i,j,k,l) = \iint \varphi_i(1)\varphi_j(1)\frac{1}{r_{12}}\varphi_k(2)\varphi_l(2)d\tau_1 d\tau_2 \tag{4.2}$$

式 (4.2) の 1,2 は電子 1、電子 2 の添字を表す。
一般に、異なる中心をもつ二つのガウス関数

$$g_1 = \exp[-\zeta_1(r-\mathbf{R_1})^2] \tag{4.3}$$

$$g_2 = \exp[-\zeta_2(r-\mathbf{R_2})^2] \tag{4.4}$$

の積は、式 (4.5) に示すように一中心のガウス関数で表すことができる。

$$g = g_1 \times g_2 = K(\zeta_1,\zeta_2,\mathbf{R_1},\mathbf{R_2})\exp[-\zeta(r-\mathbf{R})^2] \tag{4.5}$$

$$k(\zeta_1,\zeta_2,\mathbf{R_1},\mathbf{R_2}) = \exp[-\frac{\zeta_1\zeta_2}{\zeta_1+\zeta_2}(\mathbf{R_1}-\mathbf{R_2})^2] \tag{4.6}$$

$$\zeta = \zeta_1 + \zeta_2 \tag{4.7}$$

$$\mathbf{R} = \frac{\zeta_1\mathbf{R_1}+\zeta_2\mathbf{R_2}}{\zeta_1+\zeta_2} \tag{4.8}$$

軌道指数 ζ_1、ζ_2 は正の実数値なので、式 (4.6) の関数 K の最小値は 0、最大値は 1 である。
式 (4.5) 〜 (4.8) の操作を式 (4.2) で表される二電子積分に対して行なうと、

$$g(i,j,k,l) = K(\zeta_i,\zeta_j,\mathbf{R_i},\mathbf{R_j}) \times K(\zeta_k,\zeta_l,\mathbf{R_k},\mathbf{R_l}) \times \iint \Lambda \tag{4.9}$$

となる。従って、$K(\zeta_i,\zeta_j,\mathbf{R}_i,\mathbf{R}_j)$ や $K(\zeta_k,\zeta_l,\mathbf{R}_k,\mathbf{R}_l)$ を計算すれば、原始基底関数で記述される二電子積分 $g(i,j,k,l)$ の値の大きさ（上限値）がどの程度であるか、あらかじめ知る事ができる。

関数 $K(\zeta_i,\zeta_j,\mathbf{R}_i,\mathbf{R}_j)$ は原始基底関数 i と j の情報のみで決定されるので、i と j がわかれば、**すべての** $g(i,j,*,*)$ が必要であるかどうか判断できる。同様に、関数 $K(\zeta_k,\zeta_l,\mathbf{R}_k,\mathbf{R}_l)$ は原始基底関数 k と l の情報のみで決定されるので、k と l が分かれば、**すべての**二電子積分 $g(*,*,k,l)$ が必要かどうか判断できる。このように、

「４つの添字を持つ二電子積分がフォック行列計算に必要かどうかを２つの添字だけから判断できる」、

という点は注目に値する。

また、縮約基底関数で記述された二電子積分 (IJ,KL) がカットオフできるかどうかは次の方法で判断する事ができる。$\zeta_1 > \zeta_1$' なるガウス関数

$$g_1' = \exp[-\zeta_1'(r-\mathbf{R_1})^2] \qquad (4.10)$$

を考えた時、関数 K は、

$$K(\zeta_1,\zeta_2,\mathbf{R_1},\mathbf{R_2}) < K(\zeta_1',\zeta_2,\mathbf{R_1},\mathbf{R_2}) \qquad (4.11)$$

なる大小関係を満たす。よって、縮約基底関数 I,J に含まれる原始基底関数 i,j の中でそれぞれ最も軌道指数が小さい i^{min},j^{min} で計算した関数 $K^{\max}(\zeta_i^{\min},\zeta_j^{\min},\mathbf{R}_i,\mathbf{R}_j)$、もしくは、縮約基底関数 K,L に含まれる原始基底関数 k,l の中でそれぞれ最も軌道指数が小さい k^{min},l^{min} で計算した関数 $K^{\max}(\zeta_k^{\min},\zeta_l^{\min},\mathbf{R}_k,\mathbf{R}_l)$、のいずれかがある与えられた閾値よりも小さければ、縮約基底二電子積分 (IJ,KL) に含まれる全ての原始基底二電子積分 $g(i,j,k,l)$ の値は十分に小さいと判断できるので、(IJ,KL) の計算をカットオフすることができる。

このように、関数 K の値がある与えられた閾値以下になる二電子積分の計算を排除する事で、計算精度を損ねる事なく計算時間を短縮することができる。通常、この閾値としては 10^{-15} または 10^{-20} 程度の値が用いられる。

なお、この積分カットオフの物理的な意味としては、軌道 I と軌道 J の重なり部分にある電子１の電子密度や軌道 K と軌道 L の重なり部分にある電子２の電子密度が十分に小さければ、これらで表現されるクーロン積分自体も十分小さいであろうからその計算を省略する事ができる、という風に解釈できる。

4.2　並列分散フォック行列アルゴリズムとの関係

簡単のため、Figure 3に示したような模式的なフォック行列生成アルゴリズムを例にあげて説明する。

これは、式 (2.10) に示したフォック行列二電子項をそのまま素直に展開した方法であり、添字駆動型アルゴリズムと呼ばれる。このアルゴリズムを並列分散化する場合について考える。必要に応じてホスト計算機との間で行列要素データの送受信を行なう事になるので、データの転送回数を減らし同じ行列要素を使いまわして利用するためには、行列要素の転送をできるだけ外側のループで行なうようにすればよい。一例を Figure 4 に示す。

Send はホスト計算機からプロセッサエレメントへのデータ送信を、receive はプロセッサエレメントからホスト計算機へのデータ受信をそれぞれ表す。プロセッサエレメントで必要な行列

```
for(I=1;I ≦ N;I++) {
for(J=1;J ≦ N;J++) {
for(K=1;K ≦ N;K++) {
for(L=1;L ≦ N;L++) {
   compute (IJ,KL), (IK,JL);
   F_IJ += P_KL*(IJ,KL);
   F_IJ -= 0.5*P_KL*(IK,JL);
}}}}
```

Figure 3. Index-driven Fock matrix algorithm

要素は F_{IJ} と P_{KL} だけである。よって、Figure 4 に示すように、四重ループの順序を *I,J,K,L* をから *I,L,J,K* に入れ替えることにより、行列転送を *J,K* の二重ループの外側に配置し、O(N^2) 回の積分計算に対し O(*N*) 個の行列転送量に抑える事ができる。

次に、Figure 4 のアルゴリズムを用いて計算時間の短縮化のために二電子積分のカットオフを行なうことを考える。計算すべき二電子積分は (*IJ,KL*) と (*IK,JL*) であり、行列情報を送受信する時点ではすでに *I* と *L* の値は定まっている事から

```
for(I=1;I ≦ N;I++) {
for(L=1;L ≦ N;L++) {

        send P_KL(1 ≦ K ≦ N)
        for(J=1;J ≦ N;J++) {
        for(K=1;K ≦ N;K++) {
            compute (IJ,KL), (IK,JL);
            F_IJ += P_KL*(IJ,KL);
            F_IJ -= 0.5*P_KL*(IK,JL);
        }}
        receive F_IJ(1 ≦ J ≦ I)

}}
```

Figure 4. Index-driven Fock matrix algorithm for distributed memories.

・(*IJ,KL*) からは、*I* に対するカットオフで生き残る *J* の組と、*L* に対するカットオフで生き残る *K* の組が、

・(*IK,JL*) からは、*I* に対するカットオフで生き残る *K* の組と、*L* に対するカットオフで生き残る *J* の組が

それぞれ分かる。この場合、上記二つの条件から、F_{IJ} の生き残る添字 *J* の組や、P_{KL} の生き残る添字 *K* の組はいずれも、*I* に対するカットオフで生き残るものと *L* に対するカットオフで生き残るものの論理和で与えられる。従って、*J* と *K* のループでは、1 ～ *N* の全てをではなく、

カットオフ操作で生き残る添字だけを実行すればよい。

カットオフ操作でどの添字が生き残るかについては、あらかじめ、全てのインデックスに対してカットオフでどの添字が生き残るかというリストを示した「カットオフテーブル」を作成しておくと便利である。この計算量は O(N^2) であり、また SCF 計算の間で変化する事はないので、式 (4.6) を用いて最初に一度ホスト計算機で作成しておけばよい。カットオフテーブルの具体例を Figure 5 に示す。この例では、基底数は 6 であり、基底番号 1 ～ 6 のそれぞれに対してカットオフ操作で生き残る添字のリストが与えられている。

カットオフ操作を行なう場合に Figure 4 のアルゴリズムに沿ったフォック行列作成方法を Figure 6 に示す。また、その手順を、Figure 5 のカットオフテーブルを用いながら具体的に説明する。

0) ホスト計算機でカットオフテーブルを作成しておく。

1) プロセッサエレメントに *I*,*L* を割りあてる。

例えば、*I*=3,*L*=2 とする。

I	Cutoff survival index
1:	1,4,5
2:	2,3,6
3:	2,3,5,6
4:	1,4,5
5:	1,3,4,5,6
6:	2,3,5,6

Figure 5. Cutoff survival index

2) 割り当てられた *I* と *L* に対して生き残る *J* と *K* の組をカットオフテーブルから読み出す。

カットオフテーブルから、

・*I*=3 に対して生き残る添字 *J* は {2,3,5,6}、

・*L*=2 に対して生き残る添字 *K* は {2,3,6}、

であることが分かる。よって、これらの論理和である {2,3,5,6} が、Figure 4 のアルゴリズムで実際に取りうる *J* と *K* の値になる。

3) 読み出したリストに従ってカットオフされた密度行列 P_{KL} の情報を作成する。

$P_{*2}(1 \leqq * \leqq 6)$ から、$P_{22}, P_{32}, P_{52}, P_{62}$ を取り出す。

4) 作成した P_{KL} の情報をプロセッサエレメントに転送する。

5) プロセッサで二電子積分を求め、フォック行列要素 F_{IJ} を計算する。

作成されるフォック行列要素は、$F_{32}, F_{33}, F_{35}, F_{36}$ である。

6) 計算されたフォック行列要素 F_{IJ} の情報をプロセッサエレメントから受け取る。

7) 全ての *I*,*L* ペアが終了していなければ、1) ～ 6) を繰り返す。

このように、積分カットオフの判断基準に従ったカットオフテーブルを参照することによって、カットオフを考慮した必要最小限の密度行列およびフォック行列をホスト計算機側で準備

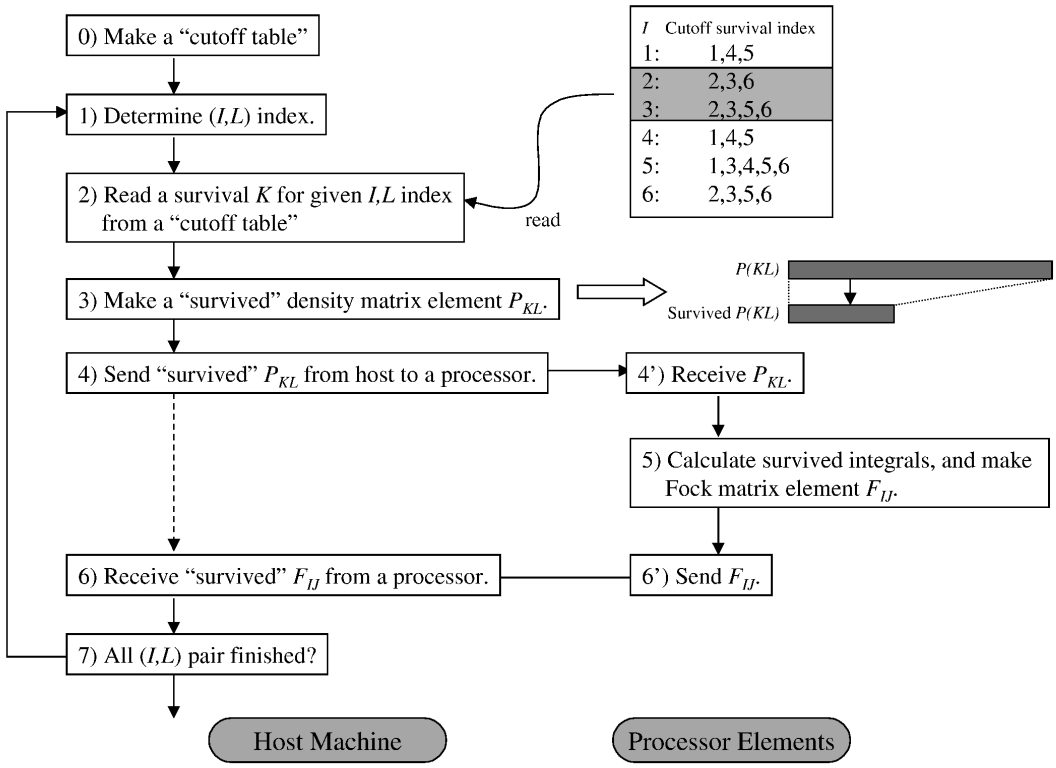

Figure 6. Flow charts of parallel Fock matrix algorithm for distributed memories.

してプロセッサエレメントと通信する事が可能になる。これにより、ホスト計算機とプロセッサエレメント間の転送量やプロセッサエレメントのメモリ量を大幅に減らす事ができる。

カットオフで生き残る K の添字は KL カットオフと IK カットオフの論理和で与えられるので、カットオフ生き残り率を α（0 以上 1 以下の実数）とすると、転送すべき密度行列 D_{KL} の要素数は $2\alpha N$ で与えられる。よって、本アルゴリズムを用いた場合の、二電子積分 1 個当りの行列要素の平均転送量（ここでは仮に [Word/ERI] とする）は、

$$\frac{\sum_{I=1}^{N}\sum_{L=1}^{N}(2\times 2\alpha N)}{\alpha^2 N^4} = \frac{4}{\alpha N}\ \text{[Word/ERI]} \qquad (4.12)$$

となる。これが、仮に行列転送のカットオフができないとすると、二電子積分一個当たりの行列転送量は $4/(\alpha^2 N)$ [Word/ERI] となり、カットオフ可能な場合に比べて $1/\alpha$ 倍の転送量（速度）とメモリ量が要求されることになる。4.3 節で明らかにするように、大規模系になるほど α の値はどんどん小さくなるので、大規模分子を計算する場合にはカットオフの効果を考慮する事が重要であることが分かる。

このように、分散小メモリ型アーキテクチャにおける大規模並列フォック行列計算アルゴリズムを考えるときには、従来のように積分の対称性や積分のカットオフを有効に利用して計算量の低減化を図る以外に、

・行列データの送受信は二重ループの外側で行ない、$O(N^2)$ 回の積分計算に $O(N)$ 個の行列要素

転送を行なう、
・転送する行列要素もカットオフを行なう、
ことが大事な点である。つまり、一度送信した行列要素は何回も計算に用いると同時に、計算に用いられない行列を転送するような無駄を省くことが、通信量の軽減とメモリ量の縮小の両方の点において重要である。次節では、蛋白質のような大規模分子において、積分カットオフ及びそれに伴う行列転送カットオフの割合が実際どの程度の大きさになるかついて検討する。

なお、Figure 4 に示すアルゴリズムの欠点は、式 (2.13) に示す積分の対称性が利用されていないため、重複した積分計算回数が多くなることである。この点を改善したアルゴリズムとして、我々は *RT* 並列アルゴリズムを開発した。その詳細は第二報 [17] で示す。

4.3 大規模計算におけるカットオフの実例

本節では、大規模計算においてどの程度積分カットオフを行なう事が可能かについて以下の手順で述べる。まず、SCF 収束過程における全エネルギー値の変化の様子を見る事で、全エネルギー値の数値的精度について考察する。次に、その数値的精度を満足するような結果を与える積分カットオフの閾値を調べる。そして、その積分カットオフの閾値を用いた時に、(*IJ*,*KL*) で表される縮約基底の二電子積分がどの程度カットオフされるのか、即ち、カットオフ操作を行うことによって行列要素の転送量はどの程度軽減できるのか、について、実際に蛋白質程度の大規模分子を用いて検討する。

まず、Table 1 で用いた分子を使って、計算された SCF 全エネルギー値がどの程度の数値的精度を持っているかについて調べた。計算に用いた非経験的分子軌道法のプログラムは GAMESS[15] である。DIRSCF =.TRUE. でダイレクト計算を行なった。GAMESS では、式 (4.6) で与えれられる関数 K の指数部の数値-log$K$ を、カットオフ閾値 ITOL として入力から指定できる。そこで、できるだけ精度の良い結果を得るために、ITOL=99 を用いて積分カットオフが殆どなされないようにした。また、NCONV=10 を用いて密度行列を十分収束させた（10^{-10}）。シュワルツの不等式による二電子積分のカットオフ [3] を用いないようにするため、SCHWRZ=.FALSE. を用いた。このようにして行なった SCF 計算の収束過程における密度行列の変化と全エネルギー値の変化（ΔE；kcal/mol）について調べたのが Figure 7(a) である。

SCF 収束過程では通常エネルギー値の変化は負の値になるが、図中の塗りつぶされた点はエネルギー変化が正の値だった事を示している。GAMESS プログラムのデフォルト値では、密度行列の変化が 10^{-5} 以下になれば収束したと判定している（NCONV=5）。しかし、Figure 7(a) の結果を見ると、確かに NCONV=5 の時点でどの分子も全エネルギーの変化が 10^{-5} kcal/mol 以下と十分小さくなっているものの全エネルギー値はまだ収束方向に向かっており、NCONV=5 では十分に収束しているとは言えない。密度行列の変化が 10^{-6} を下回る頃から、エネルギー値の変化が正負両方の値を取るようになり、10^{-7}kcal/mol 程度の幅を持って振動を始めているのが分かる。従って、今回計算に用いた程度の分子サイズの場合、密度行列の変化が 10^{-6} 以下 (NCONV=6) で SCF 計算が収束したとみなす事ができ、密度行列の収束に伴う全エネルギー値の数値的精度は 10^{-7}(kcal/mol) 程度であることが分かった。

これを踏まえた上で、先に述べた二電子積分カットオフの閾値をさまざまな値に変化させた場合の全エネルギー値に与える影響を調べた。ITOL= 99 の場合を積分カットオフがされな

い場合と見なして、この時の分子の全エネルギー値 Etot(99) を参照値とした。そして、ITOL の値を 8 から 20 の間で変化させてそれぞれの場合での全エネルギー値 Etot（ITOL）を求め、|Etot(ITOL)-Etot(99)| (kcal/mol) の値を調べたのが Figure 7(b) である。これにより、カットオフの閾値 ITOL を変化させる事によってどの程度全エネルギー値が変化するか、また、その変化が基底数（分子の大きさ）にどの程度依存しているか、が分かる。

Figure 7(b) より、カットオフ閾値 ITOL が 10^{-8} や 10^{-10} 程度だと、全ての積分を考慮した場合とみなせる ITOL=10^{-99} の場合に比べて明らかにエネルギー値の違いが見られるのが分かる。そして、その場合のエネルギー値の変化は、全般的に分子が大きくなるにつれて増大している。しかし、ITOL が 12 より大きい値、即ち、積分カットオフ閾値が 10^{-12} 以下だと、カットオフ閾値として 10^{-99} を用いた場合に比べて 10^{-7}kcal/mol 以下の違いでしかない。この大きさは、Figure 7(a) から求めた全エネルギー値の持っている数値的精度以下の範囲である。

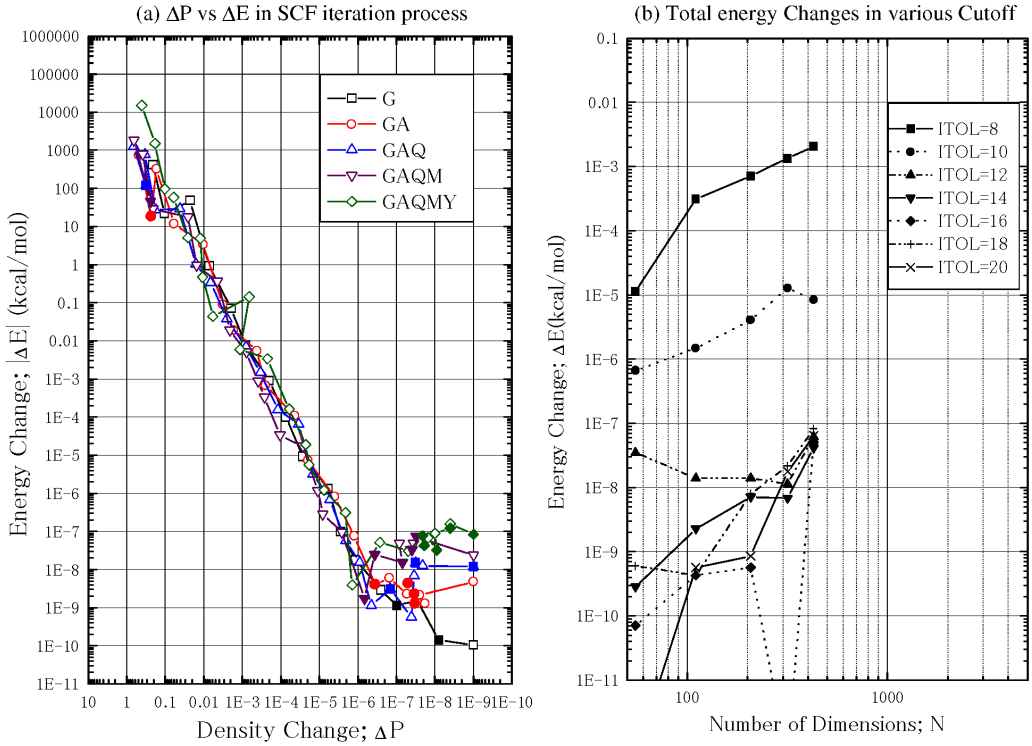

Figure 7. Energy Changes. (a) Relationship between density change (ΔP) and energy change (ΔE; kcal/mol) in SCF iteration process. Fill mark denotes that the energy change is positive.(b) Changes in total energies with various integral cutoff thresholds.

従って、10^{-15} 程度を閾値とした二電子積分のカットオフを行なうことにより、全エネルギー値は十分計算精度を保つ事ができるのがわかる。これは、分子軌道計算が有効桁数 16 桁の倍精度浮動小数点演算で行われており、フォック行列の値がおおよそ 0.01 から 100 程度の範囲であることを考え合わせると妥当な結果であるといえる。

次に、積分カットオフ閾値を 10^{-15} とした場合に、実際の大規模分子でどの程度カットオフが行われるのかについて調べた。縮約基底関数で表現される二電子積分 (*IJ*,*KL*) は N^4 個あるが、

カットオフを考慮した後に計算すべき二電子積分の数は、

$$(\alpha N \times N)^2 = (MN)^2 \qquad (4.13)$$

になる。そこで、小分子から蛋白質程度の大きさまでさまざまなサイズの分子で、カットオフ生き残り率 α および生き残った基底数 *M* を見積もった。結果を Figure 8 と Table 2 に示す。

縮約基底関数で表現される二電子積分 (*IJ*,*KL*) は、式 (4.1) で示す通り原始基底関数で表現される二電子積分 *g*(*i*,*j*,*k*,*l*) の和である。一方、積分がカットオフできるかどうかは原始基底関数（もしくは原始シェル）に与えられる軌道指数及び軌道中心で決定される。そこで、ある縮約基底二電子積分 (*IJ*,*KL*) を構成する**すべての**原始基底二電子積分 *g*(*i*,*j*,*k*,*l*) がカットオフされるなら、(*IJ*,*KL*) はカットオフされると判断できる。α はこの縮約基底二電子積分に対する生き残り率を表す。

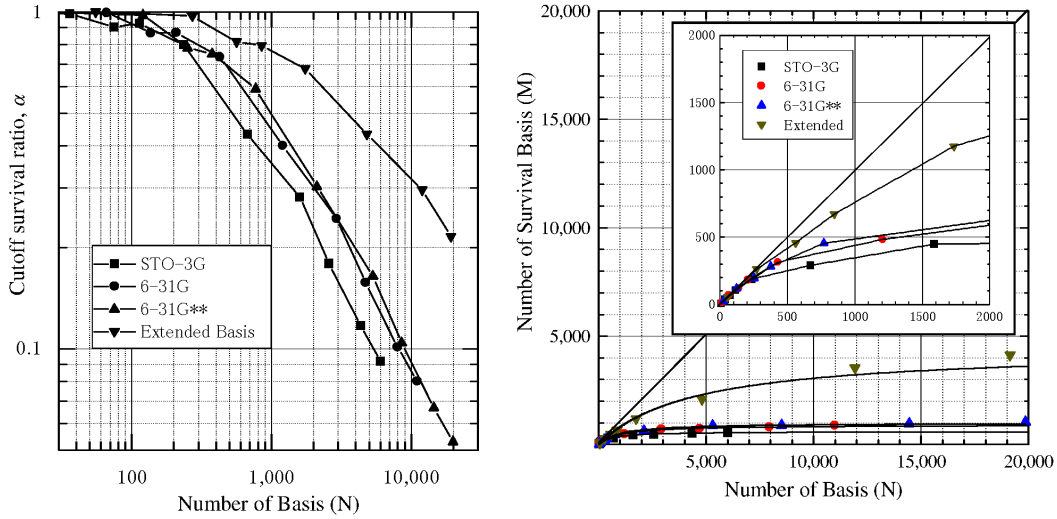

Figure 8. Effects of Cutoff. (a) Cutoff survival Ratio α. (b) Number of survival basis *M*.

計算に用いた分子は、H_2O、ベンゼン (C_6H_6)、アミノ酸である Arginine、GAQ(Gly-Ala-Gln)、GAQMY(Gly-Ala-Gln-Met-Tyr)、ヌクレオチド d(CGC)、Gramicidin A 二量体（30 残基）、BPTI（Bovine Pancreatic Trypsin Inhibitor、57 残基）、Human Immunoglobulin IGG1（99 残基）、Lysozyme（129 残基）、の 10 種類である。基底関数による違いを調べるため、STO-3G、6-31G、6-31G**、分極関数や diffuse function を取り込んだ拡張基底 (Extended Basis) として 6-311++G(3d,2p)、の 4 種類を用いた。積分カットオフの閾値 *K* は 10^{-15} を用いた。

Figure 8(a) は、基底数 *N* の分子におけるカットオフ生き残り率 α（全ての添字での平均値）を両対数プロットした図である。小分子で基底数 *N* も小さい時（100 以下）は、α はおおよそ 1 であり縮約基底関数で表現される積分のカットオフは殆どなされない*。しかしながら、分子サイズが大きくなり基底数が増えるにつれて急速に積分カットオフがなされるようになる。これは、より中心間距離が離れたガウス関数のペアが多くなるためである。10,000 次元程度にな

*一つの縮約基底の中にはさまざまな軌道指数を持つ原子基底が含まれているため、原子基底で表現される二電子積分 *g*(*i*,*j*,*k*,*l*) で見ると多くの積分がカットオフされている。

ると、STO-3G、6-31G、6-31G**クラスの基底関数では、カットオフ生き残り率αは 0.1 以下になっている。一方、拡張基底の場合、diffuse 関数や分極関数のような空間的に広がった基底関数が多数用いられているため、10,000 次元でもカットオフ生き残り率αが 0.3 程度と、他の基底関数に比べて少し大きな値である。このように、カットオフ生き残り率αは基底関数の種類によって異なり、より精度の高く空間的に広がった軌道をもつ基底関数ほどカットオフされにくく、最小基底である STO-3G のような空間的広がりを持たない基底関数系ではカットオフされる割合が大きい（カットオフ生き残り率αが小さい）事が分かる。

Table 2. Number of basis N and the number of cutoff survival basis M.

molecule	STO-3G		6-31G		6-31G**		Extended	
	N	M	N	M	N	M	N	M
H2O	7	7	13	13	25	25	55	55
C6H6	36	36	66	66	120	117	270	262
Arginine	74	67	136	119	250	196	560	457
GAQ	113	105	207	180	375	263	845	671
GAQMY	235	188	427	314	769	455	1733	1176
d(CGC)	668	288	1203	479	2106	635	4797	2069
Gramicidin	1587	448	2910	710	5295	869	11910	3525
BPTI	2572	462	4694	739	8504	891	19138	4121
IGG	4334	507	7940	803	14438	962	32476	4806
Lysozyme	6004	552	10967	881	19850	1044	44705	5578

Figure 8(b) は、基底数 N に対してカットオフで生存する平均基底数 M をプロットしたものである。図中の対角線は、カットオフが全くなされない場合を表している。Table 2 に数値を示す。基底数 N が小さい場合にはカットオフが殆どなくどの基底でも対角線上に近いが、基底数が大きくなると、精度の低い（広がった基底の少ない）基底関数系から順番に対角線から外れ出す。そして、5,000 基底程度以上になると、カットオフ後に生存する基底数 M は拡張基底の場合を除いて殆ど増加しなくなり、ほぼ一定値になっていることが分かる。このように、蛋白分子のように基底数 N が大きい場合では、式 (4.13) から判るように、計算しなければならない二電子積分の数は $O(N^4)$ から $O(N^2)$ になる。しかも、精度の低い基底関数ほど $O(N^2)$ になりやすい傾向が見える。先に述べた通り、精度の低い基底は電子雲の歪みを表現するための空間的に広がった関数が少ないので、他の中心を持つガウス関数との重なりが小さくなりカットオフされやすいのがその理由である。

なお、ここで示したカットオフ生き残り率αや生き残った基底数 M は分子全体での平均値であり、原子の種類や位置、縮約シェルの種類（例えば、内殻に相当する 1s 軌道か valence の p 軌道か）、等によって、同じ分子内でもカットオフされる割合がかなり異なることは注意すべきである。6-31G 基底における Lysozyme 分子の場合、カットオフで生き残る基底数 M の平均値は 881 であるが、最も少ない場合には M=91 であり、逆に最も多い場合には M=3409 であった。このように、原子の種類や位置、軌道の種類などによって、カットオフで生き残る基底数 M はばらつきがある。

では、ある軌道に対してカットオフ操作を行なった時に生き残る軌道の空間的配置はどのよ

うなものであろうか？一例として、6-31G 基底での Lysozyme 分子において、カットオフ操作で生き残る軌道の空間的広がりを示したのが Figure 9 である。

分子のほぼ中央に位置する 56 番目 Leucine の δ 炭素原子（大きな球で表している）の 2p 軌道に対してカットオフを考慮した場合に、生き残る軌道が空間的にどのように分布しているかを調べた。カットオフ閾値は 10^{-15} である。生き残る軌道を一つでももつ原子を小さな球で示しているが、これらの多くは半径 10 Å 以内に集まっている。しかし、原子の種類によっては、例えば 10 Å より離れていても生き残る軌道を持つものもあるし、逆に近くにあっても生き残る軌道を全く持たないものもある。さらに、小さな球で示されている原子であっても、その原子に含まれる全ての軌道が生き残るわけではなく「少なくとも一つの軌道が」生き残っているだけであり、多くの軌道はカットオフされている。

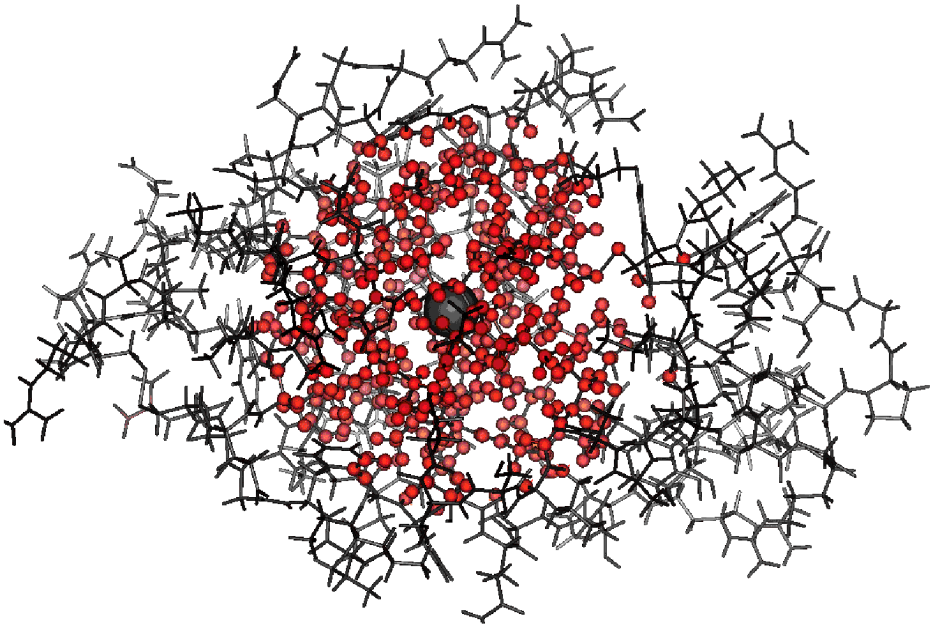

Figure 9. Cutoff scheme in Lysozyme molecule.

このように、ある添字 I の軌道に対してカットオフにより生き残る軌道 *J* は *I* の近傍の原子に集中しており、遠くの原子には殆ど存在しない。従って、蛋白質のような大規模系では、カットオフされる割合は非常に大きくなり二電子積分の数は実質上 $O(N^2)$ になることが理解できる。

大規模フォック行列を計算するための並列分散アルゴリズムでは、ホストープロセッサ間の転送量（速度）及びプロセッサエレメントのメモリ量の観点から、積分のカットオフ及びこれに伴う行列転送のカットオフを考慮する事が大事である。実際、大規模分子でのカットオフの割合を調べる事で、カットオフを考慮する場合としない場合においては 10 倍以上転送量（速度）やメモリ量が異なることが分かった。

そこで我々は、上記の点を取り入れて、しかも積分の対称性もある程度考慮した新規並列フォック行列アルゴリズムを考案した。このアルゴリズムは *I,K* ループが最外二重ループになり、ループ内を一つのジョブ単位として各プロセッサに振り分けている。但し、実際に用いるときには、縮約基底の単位ではなく縮約シェル *R,S,T,U* のインデックスを用いてループを回し、

縮約シェルの添字 *R,T* が最外ループに位置する。そこで、このアルゴリズムを我々は *RT* 並列アルゴリズムと呼んでいる。[17]

5　まとめ

我々は、低コストでの分子軌道法の高速計算を目指して、分散小メモリ型並列アーキテクチャである分子軌道専用計算機 MOEngine の開発を行なっている。一方、アルゴリズム面においても、従来の並列フォック行列計算アルゴリズムではなく、新規アルゴリズムの開発が必要となる。その際、必要に応じてホスト計算機と各プロセッサの間で行列要素データの送受信を行なうため、ホスト計算機と各プロセッサ間の通信量を減らすことが重要な課題である。また、プロセッサエレメントのメモリ量をできるだけ少なくし、しかもその量が計算サイズに依存しないようなアルゴリズムが望ましい。

そこで、本論文ではまず、大規模計算で大きな効果をもたらす積分カットオフの問題に注目し、積分カットオフの閾値が計算結果に与える影響や、大規模分子における積分カットオフの割合について調べた。大規模分子では、２つの添字だけで行なうカットオフによって生き残る基底数の割合が 10%以下となりほぼ一定値になること、これにより、積分数は事実上基底数 *N* の二乗に減少することが判明した。よって、ホストと各プロセッサの間で行列転送を行なう際には積分カットオフに対応した転送行列のカットオフを行なう事が通信量やメモリ量の軽減に必須であることが分かった。

本研究に当たりいつも有益な議論を頂いている島根大学助教授網崎孝志博士、九州大学助教授村上和彰博士に感謝する。本研究の一部は、産業技術情報基盤整備研究開発「ヘテロジニアスネットワークコンピューティング環境における分子シミュレーションシステムの構築」、科学技術振興調整費「第一原理計算手法に基づくシミュレーション技術に関する研究」、文部省科学研究費試験研究 B「非経験的分子軌道計算の超高速化に基づくハードウェア及びソフトウェアの開発」、科学技術振興事業団計算科学技術活用型特定研究開発推進事業（短期集中型）「大規模分子軌道計算を高速かつ低コストで実現するための研究開発」に基づくものである。

参考文献

[1] Alsenoy, C.V., Yu, C.H., Peeters, A., Martin, J.M.L., and Schäfer, L., *J. Phys. Chem.*, **102**, 2246-2251 (1998).

[2] Challacombe, M., and Schwegler, E., *J. Chem. Phys.*, **106**, 5526-5536 (1997).

[3] Foster, I. T., Tilson, J. L., Wagner, A. F., Shepard, R. L., Harrison., R. J., Kendall, R. A., and Littlefield. R. J., *J. Comp. Chem.*, **17**, 109-123 (1996).

[4] Furlani, T.R., King, H.F., Implementation of a parallel direct SCF algorithm on distributed memory computers, *J. Comp. Chem.*, **16**, 91-104 (1995).

[5] Häser, M., and Ahlrichs, R., *J. Comp. Chem.*, **10**, 104-111 (1989).

[6] Huzinaga. S., Andzelm, J., Klobukowski, M., Radzio-Andzelm, E., Sakai, Y., Tatewaki, H., *Gaussian Basis Sets for Molecular Calculations*, Elsevier, Amsterdam (1984).

[7] 稲畑深二郎, 山田　想, 大澤　拓, 沖野晃一, 冨田裕人, 橋本浩二, 早川　潔, 宮川宣明, 村上和彰, 電子情報通信学会技術報告, *ICD98-21*, 77-84 (1998).

[8] Lipkowitz, K. B., and Boyd, D. B., *Reviews in computational chemistry IV*, VCH Publishers, New York (1993), p.1-33.

[9] Mattson, T. G., *Parallel computing in computational chemistry*, ACS Symp. Series (1995).

[10] McWeeny. R, *Method of molecular quantum mechanics*, 2nd edition, Academics (1989).

[11] 村上和彰, 小原　繁, 長嶋雲兵, 網崎孝志, 田辺和俊, 北尾　修, 北村一泰, 高島　一, 宮川宣明, 稲畑深二郎, 山田　想, *CICSJ Bulletin*, **16**, 6-12 (1998).

[12] 小原　繁, 村上和彰, 長嶋雲兵, 網崎孝志, 田辺和俊, 北尾　修, 北村一泰, 高島　一, 宮川宣明, 稲畑深二郎, 山田　想, *CICSJ Bulletin*, **16**, 2-5 (1998).

[13] Roothaan, C.C.J., *Rev. Mod. Phys.*, **23**, 69 (1951).

[14] Sakuma, T., Kashiwagi, H., Takada, T., and Nakamura, H., *Int. J. Quantum Chem.*, **61**, 137-152 (1997).

[15] Schmidt, M. W., Baldridge, K. K., Boatz, J. A., Elbert, S. T., Gordon, M. S., Jensen, J. J., Koseki, S., Matsunaga, N., Nguyen, K. A., Su, S., Windus, T. L., Dupuis, M., Montgomery, J. A., *J. Comp. Chem.*, **14**, 1347-1363 (1993).

[16] Szabo, A., and Ostlund, N. S., *Modern quantum chemistry*, Macmillian (1982).
大野公男, 阪井建男, 望月裕志訳, 新しい量子化学 (上下), 東大出版会 (1987).

[17] 高島　一, 山田　想, 小原　繁, 北村一泰, 宮川宣明, 稲畑深二郎, 田辺和俊, 長嶋雲兵, 応用数理学会論文誌 (印刷中).

A Novel Algorithm for Large-Scale Fock Matrix Generation with Small Local Distributed Memory Parallel Architecture. – Relation to the Integral Cutoff –

Hajime TAKASHIMA[a], So YAMADA[b], Shigeru OBARA[c], Kunihiro KITAMURA[a], Shinjiro INABATA[b], Nobuaki MIYAKAWA[b], Kazutoshi TANABE[d] and Umpei NAGASHIMA[e]*

[a]Taisho Pharmaceutical Co. Ltd.
1-403 Yoshino, Ohmiya, Saitama 330-8530, Japan
[b]Fuji Xerox Co. Ltd.
2274 Hongo, Ebina, Kanagawa 243-0494, Japan
[c]Hokkaido University of Education
1-15-55 Shiroyama, Kushiro, Hokkaido 085-8580, Japan
[d]National Institute of Materials and Chemical Reactions
1-1 Higashi, Tsukuba, Ibaraki 305-8565, Japan
[e]National Institute for Advanced Interdisciplinary Research
1-1-4 Higashi, Tsukuba, Ibaraki 305-8562, Japan
**e-mail: umpei@nair.go.jp*

We are now developing a special purpose machine for accelerating ab initio molecular orbital calculations, MOEngine, a parallel architecture with small local distributed memories. This machine enables low-cost and high-performance molecular orbital calculations. MOEngine has such small memories, several megabytes, for each processor that all the matrix elements cannot be put on each memory. Conventional Fock matrix construction algorithms cannot be applied for MOEngine, and a new parallel distributed algorithm is required in which matrix element data are transferred between a host machine and each processor whenever it is necessary. Then, we developed a novel algorithm for large-scale Fock matrix generation with small local distributed memory parallel architecture. In this paper, we give a detailed explanation of "cutoff", and descrive the relationship between the "cutoff" and a Fock matrix generation algorithm for small distributed memory. In such an algorithm, "matrix element cutoff" which cooperates with the integral cutoff is indispensable for the decrease of the amount of data transfer. We have also investigated how much integral cutoff is done in large molecules. The ratio of the cutoff-survival basis number to the original basis number is less than 10%, and the survival basis number becomes nearly constant. This means that, in large molecules, the number of effective electron repulsion integrals decreases remarkably and is proportional to the order of N^2.

Keywords: Distributed Memory Parallel Computer, Ab Initio Molecular Orbital Calculation, Fock Matrix Generation, Integral Cutoff