

電子相関計算用積分変換モジュールの開発

望月 祐志^{a*}, 長嶋 雲兵^b

^a アドバンスソフト(株)・東京大学 生産技術研究所,

〒153-8904 東京都目黒区駒場 4-6-1, 東京大学 国際・産学共同研究センター

^b 産業技術総合研究所グリッド研究センター, 〒110-0015 東京都台東区東上野 6-9-3

*e-mail: fullmoon@fsis.iis.u-tokyo.ac.jp

(Received: August 15, 2003; Accepted for publication: October 10, 2003; Published on Web: December 17, 2003)

分子軌道計算において配置展開に基づいて電子相関を導入する場合、積分は基底関数の添字から分子軌道の添字に変換しておく必要がある。特に、基底関数の総数に対して4乗量である2電子積分の変換は計算コストが大きいため、演算数を可能な限り減らして効率的に処理することが重要となる。そこで、山本と長嶋によるアルゴリズム(*J. Comp. Chem.*, 9, 627 (1988))に基づいて閾値以上の積分のみを扱い、さらに基本線形代数ルーチン DAXPY と DDOT を導入して高速化を図った2電子積分変換モジュールを開発した。

キーワード: 分子軌道計算, 電子相関, 積分変換, 配置間相互作用

1 はじめに

分子軌道計算において単一電子配置の Hartree-Fock (HF) 近似を超えて電子相関を導入するには、単一配置の枠組みのままで密度汎関数理論 (DFT: density functional theory) によって相関エネルギー項を導入するか、配置間相互作用 (CI: configuration interaction) や摂動論 (PT: perturbation theory)、あるいは結合クラスター展開 (CC: coupled cluster) のように励起電子配置を含めた多配置の線形結合から相関エネルギーを求めるか、二つに大別される [1–4]。

分子軌道を展開する基底関数の総数を N とすれば計算量の形式的なオーダーは、よく知られているように、DFT では HF と同じく 4 乗、2 次の PT (MP2) では 5 乗、CI や CC では 6 乗であり、計算コスト的には DFT が最も有利である。しかし、電子構造が複雑な系や弱い結合状態の反応中間体などでは DFT の精度が低下する場合がある。PT は、弱い相互作用は記述出来るが、3d 遷移元素などの局在電子の相互作用が大きい系では十分ではないこともあり、汎用性と信頼性の観点からは CI の方が本質的には好ましい。CI

計算では、励起配置の生成が通常は 1,2 電子励起に限られるため、相関電子数が増えると記述が次第に悪化する“大きさ矛盾性”の欠陥があるが、結合電子対近似 (CEPA: coupled electron pair approximation) 的な考え方に基づき、元の CI コードの修正を最低限に留めつつ、多電子励起の寄与を取り込んで是正する近似法が提案されているので、中型以上の分子に対しても計算コスト的な問題を上手く解決していけるのであれば、CI の魅力は再評価されていくと期待される。

本稿では、続く 2 章で CI 法の概要をまとめた後、3 章で望月が進めている LCI プログラムの開発についてふれ、4 章では LCI の中で積分変換に関するモジュール群について個別に機能とアルゴリズムを記し、Linux 上での評価について報告していく。

2 CI 計算の概略

準備として、以下に CI 法の概略をまとめる。文献としては教科書の [1–4]、CI に特化したレビューとして [5] を参考にされたい。

CIでは、分子の基底状態を表わす HF 配置を参照して、占有軌道から仮想軌道へと電子を励起させた励起配置群を式 (1) のように線形結合

$$\Phi = \sum_I T_I \Psi_I \quad (1)$$

して多電子波動関数を記述する。係数のセット $\{ T_I \}$ は、ハミルトニアン行列 (2)

$$H_{IJ} = \langle \Psi_I | H | \Psi_J \rangle \quad (2)$$

を対角化する固有ベクトル、すなわち CI ベクトルとして得られ、固有値がエネルギーとなる。ハミルトニアン行列は、非零率が高々数%程度で非常に疎ではあるが、長さのオーダーは容易に 10^5 となり、 10^7 に達することもある。

式 (1) 中の個々の配置は、系のスピン状態を満たすように励起 Slater 行列式群を束ねて作られる配置関数 (CSF: configuration state function) によって与えられる。例えば、1 電子励起によって占有軌道と仮想軌道に 1 つずつ開殻が生じる配置で 1 重項状態を満たすには、開殻部分で $\alpha\beta$ と $\beta\alpha$ の 2 つのスピン並びを持つ Slater 行列式をまとめて 1 つの CSF とする。後述するハミルトニアンの行列要素の評価は、CSF を使う方が複雑になる。そのため、軌道と電子数を決めて可能なあらゆる励起とスピン結合を含める Full-CI 計算では、簡単さを最優先して Slater 行列式があらわに使われることが多い。

HF 配置からの 1,2 電子励起 CI (CISD: CI with singles and doubles) では、最低固有値のエネルギーと HF エネルギーとの差が電子相関のエネルギーに対応する。他方、1 電子励起 (CIS) に限って多数の根を解き、分子の励起エネルギースペクトルを見積ることも Gaussian[6] ではよくなされる。

目的とする状態の記述精度を上げたり、オゾン等のジラジカル記述や化学結合の解裂を扱う際には、参照配置を複数とする必要 (MRCISD: multi-reference CISD) があるが、この場合、入力する分子軌道には HF 法ではなく CASSCF (complete active space self-consistent-field) 法で求めたものを使う。CASSCF は、活性空間と称する小数の軌道群の中で Full-CI 展開を行いつつ、占有軌道全体を最適化する。なお、多参照の PT は近年、開発と応用が進んできているが、多参照 CC については、未だ開発途上にあるというのが現状である。

CI では展開長が 10^4 以上となること、またエネルギーの低い方の根だけを求めればよいことから、ハウ

スホルダー法等の直接対角化ではなく、試行的な CI ベクトルを逐次的に改良する反復解法が用いられる。反復法では、試行ベクトルとハミルトニアン行列の積である、いわゆる σ ベクトル

$$\sigma_I = \sum_J H_{IJ} T_J \quad (3)$$

が繰り返し計算される。改良は、残差を分子に、対角要素とエネルギーを分母に持つ補正ベクトル

$$R_I = (\sigma_I - E T_I) / (E - H_{II}) \quad (4)$$

を規格化を行いつつ CI ベクトルに繰り返し加えて行う。反復により、式 (4) の分母が閾値の範囲で零となれば、エネルギーと CI ベクトルが収束したと見なす。多根を解く際は、Davidson 法などが用いられる。

式 (2) のハミルトニアン行列の要素は、基底関数の添字から分子軌道の添字に変換

$$h_{ij} = \sum_{pq} c_{pi} c_{qj} h_{pq} \quad (5)$$

$$g_{ij,kl} = \sum_{pqrs} c_{pi} c_{qj} c_{rk} c_{sl} g_{pq,rs} \quad (6)$$

された 1 電子積分 h_{ij} と 2 電子積分 $g_{ij,kl}$ に対し、CSF 間の軌道占有のパターンとスピン結合の組み合わせによって決まる結合係数を掛けた積の和

$$H_{IJ} = \sum_{ij} A_{IJ}^{ij} h_{ij} + \sum_{ijkl} B_{IJ}^{ij,kl} g_{ij,kl} \quad (7)$$

で書ける。結合係数のセット $\{ A_{IJ}^{ij}, B_{IJ}^{ij,kl} \}$ は、ハミルトニアン行列のエネルギー表現とも呼ばれる。

式 (7) の和の添字は、ハミルトニアンが 2 電子の相互作用までしか含まないことから、配置の占有パターンの差が 3 電子以上では寄与が無くなるため、実際にはフルに走る必要はない。また、積分の添字に関する交換等値性

$$h_{ij} = h_{ji} \quad (8)$$

$$g_{ij,kl} = g_{ij,lk} = g_{ji,kl} = g_{ji,lk} = g_{kl,ij} = g_{kl,ji} = g_{lk,ij} = g_{lk,ji} \quad (9)$$

を用いれば、式 (7) の非零の和もコンパクトな正準形の添字の関係

$$ij = i(i-1)/2 + j, \quad i \geq j \quad (10)$$

$$kl = k(k-1)/2 + l, \quad k \geq l \quad (11)$$

$$ijkl = ij(ij-1)/2 + kl, \quad ij \geq kl \quad (12)$$

を使ってユニークな寄与だけを取れる。ここで、2 電子積分に関する節約は 1/8 である。

正準形の和としても、分子軌道の添字を4つ持つ2電子積分 $g_{ij,kl}$ の式(7)での処理はCI計算のコストを支配する。CISDの場合、CSF添字の I と J の各々を指定するのに最大で4つ(占有軌道で2、仮想軌道で2)づつ励起に係わる軌道の添字が現われるので、合わせて8つとなるが、 H_{IJ} で2電子積分が寄与を持つためには互いに2つは共通でなくてはならない。そのため2つ減り、結果としてコストのオーダーは基底数 N に対して6乗となる。

$g_{ij,kl}$ のリストを得る変換処理にも注意が要る。変換の式(6)には基底の添字が4つ、軌道の添字が4つ現われており、そのまま単純に8重のループで回すことはコスト的に現実的ではない。そこで、5乗オーダーのループ処理

$$g_{iq,rs} = \sum_p C_{pi} g_{pq,rs} \quad (13)$$

$$g_{ij,rs} = \sum_q C_{qj} g_{iq,rs} \quad (14)$$

$$g_{ij,ks} = \sum_r C_{rk} g_{ij,rs} \quad (15)$$

$$g_{ij,kl} = \sum_s C_{sl} g_{ij,ks} \quad (16)$$

で1/4づつの変換を続けて行う。基本的なアルゴリズムは幾つか知られてはいるが、プログラムとしての実装にあたっては改良する余地が残っている(本論文も、まさに2電子積分の変換に関するものである)。

ここで、式(3)の σ ベクトルの計算に話を戻す。式(7)から非零のハミルトニアン行列要素を分子積分と結合係数から一度あらわにつくってファイルに書き出し、 σ ベクトル計算の際に繰り返し読み出して処理するCI計算の仕方を Conventional-CI と呼ぶ。ここで、非零率を別とすれば、ハミルトニアン行列は本質的に N の6乗オーダーのデータ量であることに注意して欲しい。ファイルへの収納には要素の値と IJ の添字で合わせて1つあたり16バイトが必要なので、非零率1%の場合、展開長が 5×10^5 の場合で20GBもの容量になる。そのため Conventional-CI では、相関エネルギーに重要な寄与を与える励起CSF群をPT計算によって予め選択して限定的なCIを行う。実際に扱える対角化の規模としては、現在でも 10^6 程度が限界となっている。フルの対角化によって得るべき“真のエネルギー”を求めるのであれば、選択の閾値を変えながら計算を繰り返して外挿によって推定する他ない。また、選択によって相関電子対の扱いが均等でなくなり得るため、CPF (coupled pair functional) [7] や ACPF (averaged CPF) [8] のようにCEPA流に“大きさ矛盾性”を是正したCIがやり難いという弱みが Conventional-CI にはある。

Conventional-CI に対し、ハミルトニアン行列要素をあらわにはつくりしないで、個々の分子積分の寄与を直接 σ ベクトルの中に加算

$$\sigma_I \leftarrow \sigma_I + B_{IJ}^{ij,kl} \times g_{ij,kl} \times T_J \quad (17)$$

していく仕方が、Roosによって提案された Direct-CI である。Direct-CI でファイルに保存すべき主なデータは、 N の4乗オーダーの2電子積分、結合係数 $B_{IJ}^{ij,kl}$ 、ならびに式(4)の補正ベクトルの計算で分母に必要なハミルトニアン行列の対角要素であり、要求ファイルの容量は Conventional-CI に比してオーダー違いで少ない。実のところ、Direct-CI では 10^7 規模の大きなCI展開をそのまま扱うことが可能であり、今日では汎用CI計算の標準解法となっている。また、CPF/ACPF計算も容易である。

Direct-CI のプログラムは、2電子積分の処理に係る部分が Conventional-CI に比してずっと複雑になる。その複雑さは「2電子積分を占有軌道の添字と仮想軌道の添字の組合せと並び方に応じて分類し、各々の積分タイプに応じて随伴する I と J をグループ化してまとめ、 σ ベクトルへの寄与を効率よく処理するループ群を個別に場合分けしてつくりなくしてはいけない」ことにある。また、積分に乗じるべき結合係数の扱いにも工夫が要る。実装としては様々なやり方があり、MOLPRO や COLUMBUS などのプログラムとして利用出来るようになっている [9-14]。しかし残念ながら、著者らが知る限り、国産の汎用 Direct-CI のプログラムは未だ開発されてはいない。

3 LCIプログラム

LCI は、望月が提案した代表化エネルギー表現法 [15] に基づき、自身で開発している Direct-CI プログラムシステムの名称である。語頭の L には、{ large, liberal,... } などが込められている。

LCI の基本的な考え方は、

- Slater 行列式ではなく CSF を展開に使う：展開長の低減、高次スピン成分の混入問題の抜本的な回避
- 様々なタイプの CI 波動関数を組み立てられる：2電子励起を制限した分極型 CI、スピン結合様式の制限

- ハミルトニアンエネルギー表現、すなわち結合係数は粒子-空孔の定式化を用いて評価し、さらに代表化する：データ量の低減
- 閾値以上の値を持つ積分に対してのみ σ ベクトルの計算を行う (積分駆動型ループ): 無駄な演算の回避、将来の積分ファイル不使用型への拡張
- CI の抱える “大きさ矛盾性” は CPF/ACPF[7, 8] で是正する：最低限のプログラム修正
- 並列化を意識する：アルゴリズムレベルからの高効率の指向

であるが、「自在に手に入れられる国産の Direct-CI プログラムを確保する」ことも目的としては重要である。LCI の開発最終段階では、MOLPRO 等の他のプログラムが得手としない中大型の分子の様々な CI 問題を 6-31G** に代表される SVP (split-valence plus polarization) 程度の基底関数で扱えることを主に想定しており、具体的にはバイオテクノロジー関係で小ペプチドやポルフィリン等の生体機能分子、あるいはナノテクノロジー関係で遷移金属の複核錯体や表面吸着系を考えている。

基底関数による積分の生成と HF/CASSCF による軌道の最適化を GAMESS 等の外部の分子軌道コードに抛る LCI システムでは、式 (5) や (6) の積分変換の処理は入力インターフェース部分に相当する。入力データとして書けば $\{ h_{pq}, g_{pq,rs}, c_{pi} \}$ を受け取ることになる。積分を変換した後は、ハミルトニアン行列のエネルギー表現の生成、Direct-CI 法による CI ベクトルの求解、その後で得られる 1 電子密度行列

$$\gamma_{ij} = \sum_{IJ} A_{IJ}^{ij} T_I T_J \quad (18)$$

とそれを対角化して決まる自然軌道の生成といった一連の処理は LCI システムの内部で完結する。HF 軌道の占有数は $\{ 0, 1, 2 \}$ の整数だが、自然軌道では占有軌道空間からの励起によって仮想軌道空間に電子が若干量移るために 0-2 の間の小数值となる。自然軌道は系の電子状態の特徴を捕らえるのに便利であるし、自然軌道を入力とした別の CI 計算を行うことも可能である。従って、LCI の出力インターフェースとしては、自然軌道を他の分子軌道プログラム、あるいは可視化プログラムに渡すことが考えられる。

LCI の開発は現時点では第一段階にあり、モジュールとして開発が済んでいるのは、次章で述べる積分

変換に関するステップである。生体分子や錯体では、多くの場合に対称性は C_1 、すなわち対称性無であることから、コーディングに際して対称性は課していない。基底関数積分と軌道係数を提供してもらうインターフェース相手としては、オープンソースの代表的なコードである GAMESS[12] となっている。もちろん他のプログラムも可能であり、同じくオープンソースでは DALTON[14] が有望な相手であるし、国内では分割分子軌道 (FMO: fragmented MO) によりペプチド類の高速 HF 計算を可能としている ABINIT-MP[16] に対して電子相関計算の機能を提供する形での連携もあり得る。

4 積分変換に係わるモジュール

GAMESS から供給される基底関数による積分と分子軌道を使って行う積分変換は、実際には 3 つのモジュールが関与している。以下の 3 節で個々の機能とアルゴリズムについて述べ、最後の 1 節で評価を行う。

4.1 1 電子積分用 IFSORT

名称の IFSORT は、Integral-Fock-Sort から取ったものであり、モジュールの機能としては、Fock 型の 1 電子積分の生成、ならびに 2 電子積分の変換前にソーティングを行うようになっている。これらについて順に記す。

LCI では文献 [15] にあるように、仮想的な閉殻電子配置を “原点” として、そこからの差分によってあらゆる CSF セットを記述する粒子-空孔の定式化によってエネルギー表現を計算し、これによって閾値性に因らず 2 電子積分の処理量を根本的に減らす工夫をしている。このため、1 電子積分については、 h_{ij} に 2 電子積分の寄与を予め含めた閉殻型の Fock 演算子の分子軌道表現

$$f_{ij} = h_{ij} + \sum_{kl} (2g_{ij,kl} - g_{ik,jl}) \quad (19)$$

を使う必要がある。ここで、和の添字 kl は全ての閉殻軌道について走る。例えば CSF 間で占有軌道 m と仮想軌道 a が違う場合のハミルトニアン行列の要素は次式

$$\begin{aligned} \langle \dots a \dots | H | \dots m \dots \rangle &= \\ h_{am} + \sum_{kl} (2g_{am,kl} - g_{ak,ml}) &= f_{am} \end{aligned} \quad (20)$$

となるので、既に埋め込まれている 2 電子積分の和を扱う必要はない。

式 (19) で f_{ij} をつくる際も、分子軌道添字に変換された 2 電子積分をあらわに扱うことは得策ではない。そこで、普通の HF 計算と同様に、基底関数の添字で“原点”に関する Fock 行列

$$f_{pq} = h_{pq} + \sum_{rs} d_{rs} (2g_{pq,rs} - g_{pr,qs}) \quad (21)$$

$$d_{pq} = \sum_i c_{pi} c_{qi} \quad (22)$$

を先ず計算し、軌道添字に変換

$$f_{ij} = \sum_{pq} c_{pi} c_{qj} f_{pq} \quad (23)$$

を行う。

式 (21) 中で、密度行列 (式 (22) で定義されている) に掛け合わされる括弧内の 2 電子積分部分は、いわゆる Super Matrix に相当する。しかし、Fock 行列計算のためにだけ Super Matrix を改めてつくるのはコストが大きい。GAMESS では閾値以上の個々の積分を正準番地のラベルを付け、その寄与を直接扱っており、LCI でも同じアルゴリズムを使うことにした。つまり、 $g_{pq,rs}$ について次の 6 通りの寄与を計算

$$f_{pq} \leftarrow f_{pq} + 2 \times d_{rs} \times g_{pq,rs} \quad (24)$$

$$f_{rs} \leftarrow f_{rs} + 2 \times d_{pq} \times g_{pq,rs} \quad (25)$$

$$f_{pr} \leftarrow f_{pr} - d_{qs} \times g_{pq,rs} \quad (26)$$

$$f_{ps} \leftarrow f_{ps} - d_{qr} \times g_{pq,rs} \quad (27)$$

$$f_{qr} \leftarrow f_{qr} - d_{ps} \times g_{pq,rs} \quad (28)$$

$$f_{qs} \leftarrow f_{qs} - d_{pr} \times g_{pq,rs} \quad (29)$$

する。実装では、正準添字の同値性 (式 (10) ~ (12) で添字を $pqrs$ に読み替える) を重み因子で適宜考慮しつつ、読み込んだ積分バッファ長で回る単一ループで処理を行う。また、下述する積分のソーティングも同時に行うので、GAMESS の積分ファイルの読み込みは一度だけで済ませている。

“原点”の仮想閉殻配置の電子エネルギーは次式

$$E_F = \sum_i (h_{ii} + f_{ii}) \quad (30)$$

で書ける。閉殻 HF 配置を参照する CISD の場合、このエネルギーは HF の電子エネルギーそのものになる。また、入力された軌道が正準軌道の場合、Fock 行列の対角要素 f_{ii} は軌道エネルギー ε_i に対応する。開殻軌道が 1 つの 2 重項 HF を参照する場合、電子数が 1 つ少ない閉殻を“原点”とし、HF 配置の参照 CSF は「開殻軌道に関する α スピンの粒子生成演算子を作用させ

る」ものとして記述され、“原点”に対する追加のエネルギーを持つことになる。軌道 x と y の 2 つの開殻を持つ 3 重項 HF 配置を参照する場合は、エネルギーの低い方の軌道 x に閉殻を作って“原点”とすると、HF のエネルギーは次式

$$\langle x^\alpha y^\alpha | H | x^\alpha y^\alpha \rangle = E_F + f_{yy} - f_{xx} - g_{yy,xx} \quad (31)$$

となる。これは、「“原点”の配置に対し、軌道 x に β スピンの空孔を生成し、軌道 y に α スピンの粒子を生成する」ことで得られる。同様に、あらゆる励起 CSF 群の行列要素は粒子と空孔の生成演算子の積によって評価され、対応する分子積分 $\{ f_{ij}, g_{ij,kl} \}$ と結合係数 $\{ A_{IJ}^{ij}, B_{IJ}^{ij,kl} \}$ の積和で与えられる。こうしたエネルギー表現の詳細は、第二量子化演算子の代数について十分な準備が必要になるので、ここではこれ以上は踏み込まない。

式 (13) ~ (16) の 2 電子積分変換は、次節で述べる山本-長嶋のアルゴリズム [17] に基づいた ITRANS (Integral-Transformation の意) モジュールで行う。このアルゴリズムでは、前半 2 つの変換 $pq \rightarrow ij$ のために予め「ある 1 つの rs 対について閾値以上の $g_{pq,rs}$ を持つ全ての pq 対を集めておく」ソート処理が必要である。IFSORT では、前述の Fock 型 1 電子積分 f_{ij} の計算と同時にソートしている。ここで、 pq の並びについてはソートする必要がないのでロジックは直截に実装出来るのだが、注意が必要なのは正準添字で pq rs の場合だけでなく、 $pq < rs$ の場合も考慮、すなわち pq と rs を置き換えた処理も含めなくてはならないことである。これにより、正準性による節約が 1/8 から 1/4 に落ちるため、ソートされた 2 電子積分のファイルは元の GAMESS のファイルに比して 2 倍程の大きさになる。

メモリ上でソートする、すなわちインコアのソーティングの場合、第一次元をスタックしていく pq 、第二次元をソーティング変数の rs とする $W(N(N+1)/2, N(N+1)/2)$ を積分値のための配列、ならびに p と q の添字値を収納する配列 $IW(2, N(N+1)/2, N(N+1)/2)$ の 2 つを確保出来ればよく、 rs について順に順編成ファイルに書き出せばソーティング処理は完了する。インコアで処理できない場合は、配列 W の第一次元 (pq の添字値に関する配列 IW では第二次元) を適当な IO バッファ長 L で取り、GAMESS の 2 電子積分ファイルを読み込んで処理しながら、スタックが L になる毎

に作業用の直接編成ファイルにレコードとして書き出す。この時、吉嶺の連鎖 [18] に従って“1つ前のレコードの番地”を同時に書いておく。元の積分ファイルの読み込みが終わったら、スタックに残っているデータも書き出してしまおう(フラッシュ操作)。

順編成ファイルへのソート済み積分の書き出しでは、直接編成ファイルを「ある rs についての最後のレコードから逆向きに次々に読んでいく」ことになる。バッファ長 L は、IFSORT の既定値では 820 となっている。 L を大きくすれば IO 頻度は減るが、その分だけメモリ要求は大きくなる。セミアコ的な扱いとして、「ある rs まではインコアで、残りは直接ファイルを使ってソートする」ことも考えられるが、元の基底関数添字の 2 電子積分のファイルを複数回読まなくてはならないこと、またメモリ利用の最適化を含めたロジックが煩雑になることから、現時点ではシンプルさを優先して実装していない。基底関数の総数と利用出来るメモリ量に応じて、インコア式とファイル式をプログラムが自動的に選択する。CPU 時間はどちらの処理も大きくはないが、直接編成ファイルでのソーティングの場合、マシン環境に因るが経過時間はかかる。

4.2 2 電子積分用 ITRANS

LCI で採用した山本-長島の 2 電子積分変換アルゴリズム [17] には以下の特徴がある。

- 閾値以上の基底積分 $g_{pq,rs}$ だけを扱う。中間的に 2/4 だけ変換された $g_{ij,rs}$ についても閾値選別が適用出来る。
- 添字対 rs についてのソートは必要だが、 pq については不要である(前述)。
- ソートされた $g_{pq,rs}$ リストの読み込みは一度だけでよい。
- $g_{ij,rs}$ を吉嶺の連鎖 [18] に従って直接編成ファイルに書き出しつつ ij についてソートする。
- 後半の変換は ij についてソートされた $g_{ij,rs}$ を読み出して行い、閾値以上の $g_{ij,kl}$ を順編成ファイルに添字付きで書き出す。
- 最深ループはベクトル化可能である。

要は、積分の sparseness を活かして無駄な演算を避けられる点が一番の魅力で、中大型分子、特に分子形

状が広がった生体分子の扱いに有利である。実のところ、山本らが今から十五年も前に、基底総数 $N=232$ の鉄ポリフィリン錯体酵素モデルの CASSCF 計算を当時の最新鋭ベクトル機日立 S-810 上で実現出来た [19] のも、この変換アルゴリズムを JASON2 プログラムに実装したからこそであった。JASON2 では、当時の計算機環境の中で最適動作させるために、変換処理の核心部分に随伴して IO 処理やメモリ割当に相当の工夫がこらされていたが、今日では普及価格のパーソナルコンピュータでさえも数百 MB のメモリは容易に利用出来るので、LCI の変換モジュール ITRANS では、こうした付随処理を省き、元のアルゴリズムをシンプルにコーディングすることを先ず考えた。その際、最深部には、世界標準の基本線形代数ライブラリである BLAS [20] の中で、データ移動量が 1 次であるサブルーチン DAXPY と DDOT を導入することにした。BLAS はソースで使うことはもちろんだが、計算機毎に最適化されたバイナリライブラリとしてリンクすることも可能なので、パーソナルコンピュータからベクトル型スーパーコンピュータまでカバー出来る。また、JASON2 は直接編成の作業ファイルを常に使う仕様であったが、ITRANS では $g_{ij,rs}$ がメモリ上に展開出来れば、IFSORT と同様にインコアでの変換処理が自動的に行われるように書かれている。

これから、個々の 1/4 変換について順に見ていく。処理のコストが最も大きいのは、式 (13) の最初の 1/4 変換のステップである。変換すべき分子軌道の総数を $M(N)$ とすると、演算数のオーダーは $N^4 M/4$ となる。ここで、 α は基底積分の生存率で、因子 1/4 は添字正準性に因る。ITRANS では、この 1/4 変換の最深部分は以下のようにになっている。

```
DO IWPQ=1, ICOUNT
  IP=IBUFRS(1, IWPQ)
  IQ=IBUFRS(2, IWPQ)
  V=BUFRS(IWPQ)
  IF (IP.EQ.IQ) THEN
    CALL DAXPY(M, V, TACOF(1, IP), 1, GIQRS(1, IP), 1)
  ELSE
    CALL DAXPY(M, V, TACOF(1, IQ), 1, GIQRS(1, IP), 1)
    CALL DAXPY(M, V, TACOF(1, IP), 1, GIQRS(1, IQ), 1)
  END IF
END DO
```

この pq ループの外側には、IFSORT でソートされた積分バッファを読み込んで駆動される rs ループがあるので配列 GIQRS は rs の次元を持つ必要はない。言い換えると、上のループ処理の前に GIQRS 領域のクリア操作(零で初期化する)が要る。TACOF は変換に係る軌道係数 c_{pi} を転置し、第一次元を軌道、第二次元を基底とした配列 c'_{ip} である。これにより、配

列アクセスのストライドは 1 となり、DAXPY 内でのループアンロールによる加速が有効となる。IP と IQ が違う場合、正準性から二つの処理が並んでいる。

続く式 (14) の 2 段目の変換は次の通り。

```

IJ=0
DO I=1,M
  DO J=1,I
    IJ=IJ+1
    S=DDOT(N,ACOE(1,J),1,GIQRS(I,1),M)
    IF(DABS(S).GT.THIIJRS) THEN
      IZ=IZ+1
      GIJRS(IZ,IJ)=S
      LGIJRS(1,IZ,IJ)=IR
      LGIJRS(2,IZ,IJ)=IS
      IF(IZ.EQ.LBUFRS) THEN
        IREC=IREC+1
        WRITE(Dir.) IREC,IRECPR,GIJRS(IJ),LGIJRS(IJ)
        IZ=0
      END IF
    END IF
  END IF
END DO
END DO

```

IR と IS は外側で走っている rs ループの変数、ACOE は軌道係数 (c_{pi}) を基底-軌道の順に含む配列である。基底数 N に関する DDOT 処理の GIQRS のストライドは、軌道数 M となっている。

前半、2/4 変換の段階で $g_{ij,rs}$ となった時点で閾値判断があり、生き残ったものはスタック変数 IZ をインクリメントして直接編成ファイルへのバッファに入れる。IZ がバッファ長 LBUFRS に達すると、前のレコード番号 IRECPR と共にその IJ についてファイルに書き出す。これが、吉嶺の連鎖アルゴリズム [18] の要である。変換と書き込みの処理を最後の rs まで行い、バッファをファイルにフラッシュすれば、後半の $rs \rightarrow kl$ 変換に必要な ij に関するソート処理も済んでいる。

変換の後半は、直接編成ファイルから ij 毎に $g_{ij,rs}$ を吉嶺の連鎖により読み出して駆動されるループが外側になる。式 (15) の 3/4 段目の変換は、DAXPY を使って次のように書ける。

```

DO IWRS=1,ICOUNT
  X=GIJRS(IWRS)
  IR=LGIJRS(1,IWRS)
  IS=LGIJRS(2,IWRS)
  IF(IR.EQ.IS) THEN
    CALL DAXPY(M,X,ACOE(1,IR),N,GIJKS(IR,1),N)
  ELSE
    CALL DAXPY(M,X,ACOE(1,IS),N,GIJKS(IR,1),N)
    CALL DAXPY(M,X,ACOE(1,IR),N,GIJKS(IS,1),N)
  END IF
END DO

```

演算コストは、 $g_{ij,rs}$ の閾値生存率を λ として $\lambda N^2 M^3 / 2$ となる。式 (13) の最初の 1/4 変換の時と同様、上のループの前に配列 GIJKS 領域のクリアが必要である。

式 (16)、すなわち最終 4/4 段目の変換は、DDOT によって配列アクセスのストライドを 1 として処理出来る。

```

KL=0
DO K=1,M
  DO L=1,K
    KL=KL+1
    IF(IJ.GE.KL) THEN
      SS=DDOT(N,GIJKS(1,K),1,ACOE(1,L),1)
      IF(ABS(SS).GT.THIIJKL) THEN
        IZZ=IZZ+1
        GIJKL(IZZ)=SS
        LGIJKL(1,IZZ)=I
        LGIJKL(2,IZZ)=J
        LGIJKL(3,IZZ)=K
        LGIJKL(4,IZZ)=L
        IF(IZZ.EQ.LR2BUF) THEN
          WRITE(Seq.) LR2BUF,GIJKL,LGIJKL
          IZZ=0
        END IF
      END IF
    END IF
  END DO
END DO

```

閾値以上の $g_{ij,kl}$ は、添字を付けてバッファに入れてパッキングを行いつつ、順編成ファイルに書き出す。 ij のループを抜け、残っているバッファ領域をファイルにフラッシュすれば、ITRANS モジュールによる 2 電子積分の変換処理は終了する。

付記しておけば、 $g_{ij,kl}$ で添字の範囲を i と k を仮想軌道に関する a と b に、 j と l を占有軌道に関する m と n とすれば、MP2-PT による電子相関のエネルギー

$$E_{MP2} = \frac{\sum_{abmn} \{g_{am,bn}(2g_{am,bn} - g_{an,bm})\}}{\{\varepsilon_m + \varepsilon_n - \varepsilon_a - \varepsilon_b\}} \quad (32)$$

の計算に必要な積分は揃う (MP2 のコストのオーダーが N の 5 乗であるのは、積分変換に拠る) 言い換えると、ITRANS に僅かの修正を施すだけで MP2 計算のプログラムとして使える。

4.3 密度局在化用 PLOCAL

HF 計算で通常得られる正準分子軌道は、Fock 演算子の固有関数 (固有値が軌道エネルギー) となっているが、振幅は分子全体に広がっている。この状況は、小型分子だけでなく中大型の分子でも共通しており、生体分子も例外ではない。

閉殻の HF 配置を表わす Slater 行列式はユニタリー不変性 [1-5] を持っているので、占有軌道と仮想軌道のそれぞれの空間内で局在化、すなわち分子の特定領域に振幅が集中している局在化軌道に変換することがエネルギー不変のまま可能である。距離の離れた局在化軌道間の重なりは小さくなり、こうした組に関する 2 電子積分 $g_{ij,kl}$ は十分小さな値となるはずである (中間的な $g_{ij,rs}$ の段階でも小さくなると期待される) 従って、局在化軌道を使うことで、電子相関の記述に有効な積分値、さらには励起配置のタイプを電子対の

遠近に応じて低減出来る。これが、局在化相関計算の考え方である [21]。

幸い、CISD、それに CPF/ACPF[7, 8] もユニタリー不変性を持っているので、局在化軌道を使った計算は、積分駆動型の σ ベクトル計算を行う LCI では、処理すべき積分の数を減らして計算コストを下げる一つの方策となり得る。デメリットとしては、Fock 演算子が対角でなくなるため、結果としてハミルトニアン行列の対角主導性が低下し、 σ ベクトルの反復計算の回数が若干増えてしまうことである。

PLOCAL (Population-based localization の意) は、IFSORT と ITRANS に対する補助的なモジュールとして、Pipek と Mezey[22] の処方箋、すなわち Mulliken 電子密度を極大化して軌道局在化を行う機能を持っている。この方法での評価量は、軌道による電子密度の和

$$L = \sum_i \sum_A \langle i | P_A | i \rangle \quad (33)$$

$$\langle i | P_A | j \rangle = \sum_p^{(All)} \sum_q^{(A)} (c_{pi} s_{pq} c_{qj} + c_{qi} s_{qp} c_{pj}) / 2 \quad (34)$$

である。A は原子核について走る添字で、式 (34) 中で片方の和は A に限定されていることに注意して欲しい。実装にあっては、 2×2 の軌道対毎のユニタリー変換を繰り返していく。基底関数の重なり積分 s_{pq} は、1 電子積分 h_{pq} と同様に GAMESS からファイル経由で得ている。

PLOCAL では、局在化の変換効率を上げるために 2×2 変換の角度については、補角を含めた幾つかの可能性を一度に評価して、利得が最大となるものを選び出すようにしている。変換の範囲は、占有軌道群と仮想軌道群の各々で、エネルギーの低い内殻を正準軌道のまま保存する指定も出来る。変換された軌道群は、原子密度 $\langle i | P_A | i \rangle$ の大きさ順にソートされ、GAMESS からの正準軌道の係数と同一の形式で別のファイルに書き出される。

4.4 Linux 上での評価

LCI プログラムの開発は、Intel Pentium-III (800MHz) / 384MB-DRAM のパーソナルコンピュー

タ (Dell 製 Dimension XPS800R) に Linux RedHat-8[23] を乗せた環境で進めている。デバッグは Linux 標準の g77 で行い、動作確認後は Intel 製の Fortran コンパイラである ifc (Version7/Free 版) [24] でバイナリを作っている。

Table 1 に、テストした系と基底/変換軌道、それに CPU 時間 (秒) をまとめた。分子は、環状の水の多量体とアミノ酸で、Gaussian[6] を使い HF レベルで構造最適化を行っている。参考比較に用いた GAMESS は、配布時の標準指定に従った最適化のオプション

g77 -O2 -malign-double -fautomatic

でコンパイルしている。GAMESS の積分変換は、基底の数が増えるとファイルによる変換になり、パスが複数回になる (つまり、適当にブロック化して行われる) ので、パスの数を Table 1 中、GAMESS のタイミングに付随する括弧内に示している。LCI (ITRANS) は、GAMESS と同じ g77 オプションと ifc の両方でコンパイルしたものを並べている。DAXPY と DDOT も、サイト [20] からソースとしてダウンロードしたものを同時にコンパイルした。また、作業用の直接編成ファイルのバッファサイズ LBUFRS は 820、変換された積分を書き出すバッファのサイズ LR2BUF は 1148 に取った。積分の閾値は、基底積分 $g_{pq,rs}$ に対しては 10^{-10} 、2/4 変換の $g_{ij,rs}$ と 4/4 変換の $g_{ij,kl}$ の閾値 (THIJS と THIKL) は GAMESS に合わせて 10^{-9} としている。

g77 と ifc でタイミングを比較すると、やはり Intel-CPU に最適化された後者での実行は優位に高速である。水の 5 量体の 6-31G** の場合、ifc では g77 よりも 89 秒も速く、DAXPY/DDOT と ifc によるコンパイルの相性が良いことが伺える。グリシン 2 量体の 6-31G* の変換は、GAMESS ではテストマシン上ではメモリが不足して実行出来なかった。ITRANS は ifc コンパイルでは 699 秒で行えた (前処理の IFSORT での CPU 時間は 52 秒) が、g77 では 300 秒以上も遅く、水の 5 量体の場合より差が開いている。ただ、 $g_{ij,rs}$ を収納する直接編成ファイルと全変換された $g_{ij,kl}$ の書き出しの順編成ファイルが 1 つのドライブに集中したこと、そもそもマシンがやや旧型でディスクが低速であることから、ifc のコンパイルでも CPU 時間の倍近い 1375 秒も経過時間がかかっている。

Table 1. Timings of integral transformation for water polymers and amino acids on a Pentium-III (800MHz) personal computer (refer texts).

Molecule ^a	(H ₂ O) ₂	(H ₂ O) ₃	(H ₂ O) ₄	(H ₂ O) ₅	Gly	Ala	Gly ₂	Gly ₂
Basis set ^b	6-31G**	6-31G**	6-31G**	6-31G**	6-31G	6-31G	6-31G	6-31G*
No. Basis	50	75	100	125	55	68	97	151
No. Orbital	46	69	92	115	50	62	91	133
Timing ^c								
LCI / g77	4.5	32.0	107.2f	315.3f	8.1	22.2	95.0	1015.3f
LCI / ifc	3.4	23.5	78.1f	225.6f	6.0	16.7	69.7	698.9f
GAMESS ^d	3.8	28.3	90.7f(4)	240.6f(6)	6.5	18.5	81.6f(3)	unable

^a The structures of all molecules included here were optimized at HF level with the Gaussian program [6].

The shape of water polymers is of the cyclic type.

^b The s-contaminant of cartesian d-polarization functions was projected out from the orbital space.

^c CPU timing in seconds. The symbol “f” means a direct-access file-based transformation. The number of passes (or blocked-processing) is indicated in the parentheses for GAMESS.

^d GAMESS was compiled with g77 (see texts for options).

Table 2. Reduction with population-based localized orbitals [22] for water polymers and amino acids relative to the formal number of transformed integrals

Molecule ^a	(H ₂ O) ₄	((H ₂ O) ₄)	(H ₂ O) ₅	(H ₂ O) ₅	Gly ₂	Gly ₃	Gly ₄	Gly ₅	Trp
Basis set	STO-3G	6-31G**	STO-3G	6-31G**	STO-3G	STO-3G	STO-3G	STO-3G	STO-3G
No. Basis	28	100	35	125	53	76	99	122	87
No. Orbital	24	92	30	115	44	63	82	101	72
Reduction ^b									
Can. / 10 ⁻⁷	99.9	99.8	100.0	100.0	100.0	99.9	99.6	98.9	99.9
Loc. / 10 ⁻⁹	99.9	99.7	98.6	97.2	99.8	93.0	76.5	59.2	97.8
Loc. / 10 ⁻⁷	92.2	85.3	75.0	63.5	90.3	63.3	41.0	27.0	73.6

^a All the geometries of additional molecules to those in Table 1 were similarly optimized by the Gaussian program [6].

^b Reduction in percentage relative to the formal number of final transformed integrals with canonical indices. There is no reduction for canonical orbitals with threshold of 10⁻⁹.

GAMESS と ITRANS で CPU 時間を比べるのは、実装アルゴリズムが異なるのであくまでも参考ではあるが、g77 でコンパイルした場合には GAMESS が速いが、ifc を使えば GAMESS と同等か若干速いというところで、“及第レベル”に達していると言えるだろう。直接編成ファイルを使う場合、経過時間を向上させるには、IO のバッファサイズを最適化する必要があるが、マシン毎にパラメータは違ってくる。Pentium-4 でメモリが 1GB、7200/rpm といった高速回転ディスクを複数台 (RAID0 化するなどして) 持つような新しいパーソナルコンピュータなら、CPU 時間、経過時間共に、現版でも大幅に短縮されるはずであるし、ワーク

ステーションや大型計算機でも同様の改善が見込まれる。もちろん、直接編成ファイルを使う変換の経過時間をきちんと効率化するには、JASON2[17] と同様に IO 周りの最適化を行うべきで、今後の課題であろう。

次に、PLOCAL による密度局在化軌道を使った積分数の低減を水とアミノ酸類について調べてみた。占有軌道、仮想軌道共に局在変換を行っている。低減は、軌道範囲から決まる正準添字の積分数に対する比である。Table 2 に結果を示す。相関エネルギーを“化学的精度”の下限となる 0.1kcal/mol、つまり 10⁻⁴ の桁で見積もる場合、式 (4) の分子の収束判定は一桁下の 10⁻⁵ が安全である。式 (17) に示すように、積和係数と CI

ベクトルを乗じて σ ベクトルの要素に加算される $g_{ij,kl}$ の閾値としては、さらに二桁下の 10^{-7} は十分なマージンがある値である。この閾値では、6-31G**による水の 5 量体では局在化軌道では 63 %まで積分数が落ちる。水多量体の STO-3G 計算では、あまり効果はない。しかし、STO-3G によるグリシン類の計算では、鎖が伸びていくと局在化軌道を使うと積分数の減少は速く、5 量体では 27 %になっている。トリプトファンは、共役環を持つためか大きさの割に低減の効果は大きくはない。密度局在化の場合、直交性の要求から仮想軌道は占有軌道ほどコンパクトな局在性は得られないことが知られている。特に、6-31++G**のように SVP の組に広がった関数を補っていくと、その傾向は強くなる。仮想軌道については、別の局在化法を使うことも将来的には検討すべきかもしれない。

5 まとめ

本稿では、開発中の汎用 Direct-CI プログラムシステム LCI の中で、基底関数から分子軌道の添字へと 2 電子積分を変換する ITRANS モジュール、変換の前処理として基底関数の積分をソートしつつ、Fock 型の有効 1 電子演算子を生成する IFSORT モジュール、及び軌道局在化を行う PLOCAL モジュールについて報告した。IO 処理の最適化などに改良の余地は残されているが、積分変換の処理速度は、GAMESS[12] と比較して“及第レベル”にあると考えている。

現在、MRCISD エンジン開発前の“試金石”として、閉殻 HF 配置参照の CISD/CPF[7] を行う限定版を開発しているところである。今後も LCI の開発を鋭意進め、経過報告を行っていきたい。

本研究の一部は、文部科学省 IT プログラム「戦略的基盤ソフトウェアの開発 (FSIS)」において実施されたものであり、関係各位に深謝する [25]。LCI プログラムは、FSIS のプロジェクト成果物として公開される予定である。

JASON2 プログラムの 2 電子積分変換部分の参照を承諾いただいた中京大学の山本茂義助教授、GAMESS の積分周りについてメールで情報をいただいた GAMESS 管理者の M. Schmidt 博士に感謝する。また、CI 計算全般について議論や助言を継続していただいている北海道大学の田中皓教授にも謝意を表しておきたい。

参考文献

- [1] *Methods of Electronic Structure Theory*, H. F. Schaefer III ed., Plenum, New York (1977).
- [2] A. Szabo, N. S. Ostlund, *Modern Quantum Chemistry*, MacMillan, New York (1982).
- [3] *Lecture Notes in Quantum Chemistry I*, B. O. Roos ed., Springer-Verlag, Berlin (1992).
- [4] *Lecture Notes in Quantum Chemistry II*, B. O. Roos ed., Springer-Verlag, Berlin (1994).
- [5] T. Helgaker, P. Joergensen, J. Olsen, *Molecular Electronic-Structure Theory*, Wiley, Chichester (2000).
- [6] C. D. Sherrill, H. F. Schaefer III, *Advan. Quant. Chem.*, **34**, 143 (1999).
- [7] R. Ahlrichs, P. Scharf, C. Ehrhardt, *J. Chem. Phys.*, **82**, 890 (1985).
- [8] R. J. Gdanitz, R. Ahlrichs, *Chem. Phys. Lett.*, **143**, 413 (1988).
- [9] H. -J. Werner et al., *MOLPRO*.
<http://www.molpro.net/>
- [10] K. Andersson et al., *MOLCAS*.
<http://www.teokem.lu.se/molcas/>
- [11] H. Lischka et al., *COLUMBUS*.
<http://www.itc.univie.ac.at/~hans/Columbus/columbus.html>
- [12] M. Schmidt et al., *GAMESS*.
<http://www.msg.ameslab.gov/GAMESS/GAMESS.html>
- [13] M. F. Guest et al., *GAMESS-UK*.
<http://www.dl.ac.uk/CCP/CCP1/gamess.html>
- [14] T. Helgaker et al., *DALTON*.
<http://www.kjemi.uio.no/software/dalton/dalton.html>
- [15] Y. Mochizuki, N. Nishi, Y. Hirahara, T. Takada, *Theor. Chim. Acta*, **93**, 211 (1996).

- [16] T. Nakano et al., *ABINIT-MP*.
<http://molddb.nihs.go.jp/abinitmp/>
- [17] S. Yamamoto, U. Nagashima, T. Aoyama, H. Kashiwagi, *J. Comp. Chem.*, **9**, 627 (1988).
- [18] M. Yoshimine, *J. Comp. Phys.*, **11**, 449 (1973).
- [19] S. Yamamoto, J. Teraoka, H. Kashiwagi, *J. Chem. Phys.*, **88**, 303 (1988).
- [20] *BLAS*.
<http://www.netlib.org/blas/>
- [21] *Correlation and Localization*, P. R. Surjan ed., Springer-Verlag, Berlin (1999).
- [22] J. Pipek, P. G. Mezey, *J. Chem. Phys.*, **90**, 4916 (1989).
- [23] *RedHat Linux*.
<http://www.redhat.com>
- [24] *IFC*.
<http://www.intel.com/software/products/compilers/flin/>
- [25] *FSIS*.
<http://www.fsis.iis.u-tokyo.ac.jp/>

Development of Integral Transformation Modules for Correlated Calculations

Yuji MOCHIZUKI^{a*} and Umpei NAGASHIMA^b

^aAdvancesoft Inc. & Institute for Industrial Science,
Center for Collaborative Research, The University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 153-8904, Japan

^bGrid Technology Research Center, National Institute for Advanced Industrial Science and Technology
6-9-3 Higashi-ueno, Taito-ku, 110-0015, Japan

*e-mail: fullmoon@fsis.iis.u-tokyo.ac.jp

Integral transformation from basis functions to molecular orbitals is necessary in configuration-based correlation treatments of molecular orbital calculations. In particular, the processing of two-electron integrals having fourth power dependence of the number of basis functions can be costly, thus care should be taken to reduce the operation counts as far as possible. In this contribution, we report a development of integral transformation modules, based on Yamamoto-Nagashima's algorithm (*J. Comp. Chem.*, **9**, 627 (1988)) by which only non-zero integrals within threshold are efficiently processed. The BLAS routines, DAXPY and DDOT are used in the innermost part of the transformation step.

Keywords: Molecular orbital calculations, Electron correlation, Integral transformation, Configuration interaction