ハイパフォーマンス
コンピューティング　86－8
（２００１．５．２５）

相対論的分子軌道コードDIRACにおけるDHF計算のベクトル化

松村昌幸[1]　望月祐志[2]　与倉徹一[1]　平原幸男[1]　今村俊幸[2]

NEC情報システムズ[1]　日本原子力研究所[2]

DHF(Dirac-Hartree-Fock)は4成分スピノルを直截に扱う相対論的分子軌道法の基本近似である。代表的な計算コードの1つにDIRACがあり、アクチノイド等の重元素を含む分子系に適用されている。DHF計算では、膨大な数の2電子積分の生成と、それらの寄与をFock行列に加算する処理が繰り返され、求解のCPUコストを決めている。積分生成についてはベクトル化は容易だが、寄与の加算では行列の添字に衝突が起こるために単純なベクトル化は出来ない。今回は、作業配列を用いてこの問題を解決し、さらに前段階も修正することにより、トータルのベクトル化率をオリジナルDIRACの50%から90%にまで向上させた事例を報告する。

Vectorization of DHF Calculations in the DIRAC as a Relativistic Molecular-Orbital Code

Masayuki Matsumura[1]　Yuji Mochizuki[2]　Tetsu-ichi Yokura[1]
Yukio Hirahara[1]　Toshiyuki Imamura[2]

NEC Informatec Systems[1]　Japan Atomic Energy Research Institute[2]

DHF (Dirac-Hartree-Fock) is a fundamental relativistic molecular-orbital approximation in which four-component spinors are explicitly treated. DIRAC is a representative code for general DHF calculations, and has been used for such actinoid-containing molecular systems. Total cost for DHF is completely dominated by the iterative processing that a huge amount of two-electron integrals is generated and respective contribution to Fock matrix elements is accumulated. Vectorization for the integral generation is rather straightforward, but the accumulation is unable to simply vectorize since address conflicts of the elements take place. In this report, we show a tuning-up for the DIRAC-DHF calculations, based on vectorizing the integral accumulation step by introducing working array. Total vectorization efficiency is improved from 50% by the original code to 90% by the modified version.

1　はじめに

アクチノイド等の重元素では、核電荷が大きくなるために電子の運動速度が光速に近づき相対論効果が顕在化する[1]。従って、重元素を含む分子に対しては大成分(L:電子に対応) 2つに合わせて小成分(S:陽電子に対応) 2つの計4成分のスピノルから成るDirac方程式から出発して電子構造の計算をすることが必須となる。Dirac方程式の近似求解法には様々なものがあるが、Dirac-Hartree-Fock(DHF)は非相対論でのHartree-Fockの4成分への直截な拡張にあたり、質量-速度効果等のスカラー寄与と同時にスピン軌道相互作用を変分的に扱える利点がある。電子相関の導入に対しても、DHF波動関数を参照する摂動論や配置間相互作用を系統的に用いることが可能である。DIRAC[2]は、こうした4成分相対論的分子軌道計算を行える北欧のフリーウェアであり、後述のように2電子積分を直接計算するスキームを並列化して実行出来るために大規模DHFの求解が可能である。日本原子力研究所では、計算科学技術推進センターを拠点に並列処理の活用を進めている。DIRACの応用計算もこの推進活動の中で展開し、IBM-SP等を用いて核廃棄物中に存在する水和キュリウムイオンの電子構造などを扱っており、関数総数で2千超のDHF計算の実績がある[3]。しかし、DIRACはスカラー計算機上で開発

されてきたために、計算コストを決する積分処理部分をはじめとしてベクトル機での性能追求は未だなされていない。文部科学省で来年春の稼働を目指して開発中の地球シュミレータ (ES)[4] は、NEC-SX 系のベクトル CPU を 5 千以上も含む大規模並列機であり、大気・海洋・地殻のシュミレーションに威力を発揮すると期待されている。そして地球環境科学の観点からは、核廃棄物の処理と管理に関連するアクチノイド系分子の相対論的分子軌道計算も ES の応用分野の一角を占め得ると思われる。従って、ES を睨んだ上で DIRAC のベクトル機対応を図っていくことは意義ある事前準備と位置付けられる。今回は、第 1 段階として DIRAC-DHF 計算のベクトル化チューニングについて報告する。

2 DIRAC における DHF 計算

DHF では、系の分子軌道を

$$\phi_a = \sum_i^{(L\alpha)} c_{ia}^{(L\alpha)} \chi_i^{(L\alpha)} + \sum_i^{(L\beta)} c_{ia}^{(L\beta)} \chi_i^{(L\beta)} + i \sum_i^{(S\alpha)} c_{ia}^{(S\alpha)} \chi_i^{(S\alpha)} + i \sum_i^{(S\beta)} c_{ia}^{(S\beta)} \chi_i^{(S\beta)}$$

のように実数の大成分用基底 Gaussian スピノル 2 つ、虚数 i を因子に持つ小成分用基底 Gaussian スピノル 2 つの線形結合で記述し、非相対論の Hartree-Fock(HF) と同様に Slater 行列式を構成し、系の n 個の電子に関するエネルギー期待値

$$E = \sum_a^n < \phi_a | \hat{h_D} | \phi_a > + \frac{1}{2} \sum_{ab}^n [(\phi_a \phi_a | \frac{1}{r12} | \phi_b \phi_b) - (\phi_a \phi_b | \frac{1}{r12} | \phi_a \phi_b)]$$

の停留条件から、展開係数行列 $\{\mathbf{c}^L, \mathbf{c}^S\}$ を決定する。具体的には、Dirac の 1 電子エネルギー ($\hat{h_D}$:運動項と核引力項を含む)、それに 2 電子相互作用エネルギー (クーロン項と交換項) から成る Fock 行列を構築し、それを対角化してエネルギーと展開係数を得る : 対角化は全根が必要である。しかし、Fock 行列の 2 電子部分に含まれる電子密度が係数の内積から作られるという依存性を持っているため、適当な初期値から始めて収束まで反復して解く必要がある。小成分の基底 Gaussian 関数は、大成分基底 Gaussian の微分として定義される (動力学釣合) ことが普通で、例えば d 型の大成分関数からは p 型と f 型の小成分関数が生じる。従って、DHF の展開では大成分基底の 2 倍以上の数の小成分基底が合わせて加わることになる。

非相対論の直接 HF と同様、直接 DHF 求解でも反復の各回において、基底関数 4 つの組で定義される 2 電子積分

$$\int\int d\tau_1 d\tau_2 \chi_i(1) \chi_j(1) \frac{1}{r12} \chi_k(2) \chi_l(2)$$

を生成する処理、及び密度行列と掛けて 2 電子エネルギーとしての寄与を対応する Fock 行列要素に加算する処理の 2 つが CPU コストを支配している。ただし、大成分間の相互作用 (LL, LL) クラスだけではなく、大小成分間の (LL, SS)、及び小成分間の (SS, SS) の積分クラスも扱うことになり、小成分の原子核近傍での局在性によってスクリーニング操作が後の 2 クラスでは有効に働くものの、DHF では HF に比して労力が数十倍になる。一度生成した積分を仮にファイルに置くとすると、大小成分合わせて千基底の計算は容量・IO 的に現実的には不可能となり、結果として直接 DHF のアプローチが必要になる。

さて、DIRAC[2] における直接 DHF の実装では、4 元数

$$q = p_0 + p_1 \hat{s} + p_2 \hat{t} + p_3 \hat{u}$$
$$\mathbf{p} \in \Re,\ \hat{s}^2 = \hat{t}^2 = \hat{u}^2 = \hat{s}\hat{t}\hat{u} = -1$$

を使うことで Fock 行列をはじめとする複素行列群をブロック化し、メモリー要求量を減らすと共に、分子の持つ固有の対称性をスムーズに導入し、演算数の低減を達成している [5]。2 電子積分の加算は交換エネルギーに関するものが複雑で、Fock 行列のブロック $\{LL, LS, SS\}$ 毎に 4 元数に応じて 4 種のパターンがある。最終的なスクリーニングを経てバッファー内に蓄積された積分を交換項に加算するループの例は、

```
DO LL = 1,NBUF
  I = INDEX(1,LL)
  J = INDEX(2,LL)
  K = INDEX(3,LL)
  L = INDEX(4,LL)
  FI = -0.25D0 * BI(LL)
  F(K,I) = F(K,I) + FI * D(L,J)
  F(L,I) = F(L,I) + FI * D(K,J)
  F(K,J) = F(K,J) + FI * D(L,I)
  F(L,J) = F(L,J) + FI * D(K,I)
END DO
```

のようになる。**F** は Fock 行列、**D** は密度行列である。このループは、Fock 行列のアドレッシングに衝突が起こるために単純なベクトル化が出来ない。４つの交換項のアドレスは、例えば、

```
LL=1 / I=10, J=8, K=3, L=2
 --> (3,10) (2,10) (3,8) (2,8)
LL=2 / I=10, J=8, K=4, L=3
 --> (4,10) (3,10) (4,8) (3,8)
```

では、(3,10) 要素と (3,8) 要素が衝突していることが分かる。

サブルーチン FCKOU1 には、スクリーニング処理の後、Fock 行列要素への加算操作が並んでおり、$\{(LL,LL),(LL,SS),(SS,SS)\}$ の３つの積分クラスの全てがここで処理される。FCKOU1 に積分を供給する 2 電子積分の生成エンジン (HERMIT[6]) は、Gauss-Hermite 多項式表現に基づく定式化によりコンパクトでベクトル化し易い構成になっている。NEC-SX4 上 (1CPU) で指示行を適宜用いつつベクトル化して DIRAC のロードモジュールを作成し、DHF 計算を行うとジョブ全体のベクトル化率は 50%程度であり、非ベクトル化部分のほとんどの計算時間を FCKOU1 が占めていることが分かった。

DIRAC と主要開発者が共通する、いわば兄弟筋の非相対論分子軌道コードに DALTON[7] がある。DALTON の最大の特徴は、乱雑位相近似 (RPA) に基づく高次までの応答計算によって多彩な分子物性が求められる点である。さらに、積分添字のうち２つをタスクの分配単位 (IJ*) とする積分駆動型の並列化もなされている [8]。DIRAC も、収束した DHF 波動関数を基点とする線形応答計算により、電場存在下での分極率等の物性値を解析的に評価する機能 [9] を備えており、この RPA 計算でも積分処理の核心部分を FCKOU1 が担っている。３つの積分クラスのためにより複雑ではあるが、DHF/RPA の並列化制御も DALTON と同様に行われている。いずれにせよ、積分処理の最深部分にあたる FCKOU1 をベクトル化しない限り、ES に代表されるマルチベクトル機上での DIRAC-DHF 計算の性能向上は望めない。実は、DALTON でも”FCKOU1 相当”のルーチンがあり、開発元でも非ベクトル化部分として懸案となっていた。DIRAC を対象にベクトル化を行う今回の試みは、DALTON の改良にも反映され得るものである。

3　ベクトル化

3.1　積分加算部分

前章で記したように、FCKOU1 における 2 電子積分の加算がベクトル化の対象であることが分かった。こうしたアドレス衝突のあるループのベクトル化については、プラズマシュミレーションにおける粒子推進 (PP: Particle Pusher) の問題として作業配列を使うアルゴリズムが Nishiguchi らによって提案されている [10]。村井らは、ソーティングにおけるヒストグラムの生成に関して強制ベクトル化と再試行を組み合わせたアルゴリズムと PP の比較を行い、衝突頻度が少ない場合には性能とメモリー量の両面で再試行法が優れていることを示した [11]。私達も当初、両方のアルゴリズムを検討したのだが、2 電子積分の場合は既述のように衝突が頻繁に起きるために PP を用いることとした。

PP アルゴリズムの本質は、次のような作業用配列の導入にある。

```
F(K,I) = F(K,I) + FI * D(L,J)
              ↓
VF(IV,K,I) = VF(IV,K,I) + FI * D(L,J)
```

{I, J, K, L} は変数 IV によってインデックスリスト配列から取り出されるが、肝心の加算は IV により独立になされるために、ベクトル化しても衝突問題は起こらない。ベクトル化用の長さを NPP として前出の交換項ループを書き直してみると、

```
DO LL = 1,NBUF,NPP
  IL = MIN0((NBUF-LL+1),NPP)
  DO IV = 1,IL
    INT = IV + LL - 1
    I = INDEX(1,INT)
    J = INDEX(2,INT)
    K = INDEX(3,INT)
    L = INDEX(4,INT)
    FI = -0.25D0 * BI(INT)
    VF(IV,K,I) = VF(IV,K,I) + FI * D(L,J)
    VF(IV,L,I) = VF(IV,L,I) + FI * D(K,J)
```

```
    VF(IV,K,J) = VF(IV,K,J) + FI * D(L,I)
    VF(IV,L,J) = VF(IV,L,J) + FI * D(K,I)
  END DO
END DO
```

となり、衝突を避けてNPPの長さで加算のベクトル化を行うことが出来る。また、加算が結果的として分散されることで、長嶋らが指摘している並列化Fock行列構築における数値誤差の回避[12]と同様の効果があると期待され、ESのようにベクトル並列の場合にはその利が相乗的に活きてくると思われる。

PPのペナルティとして、Fock行列要素への全ての積分加算が終了した後で作業配列の寄与をまとめる処理

```
DO I = 1,NBASIS
  DO J = 1,NBASIS
    DO IV = 1,NPP
      F(J,I) = F(J,I) + VF(IV,J,I)
    END DO
  END DO
END DO
```

が新たに生じるが、ベクトル化により安価に処理出来る。一番の問題は、作業配列のメモリー要求に直結するNPPの大きさの設定で、ベクトル化の加速性能とのトレードオフを考える必要がある。ESもSXと同様にベクトルレジスタ長は256であるので、256までのPPの加速効果をサンプルコードを使って調べた。その結果を図1に示す。NPP=16では、PP化しない－すなわち、スカラーでの実行－時間を■で示しており、◆で示すPPでの時間がNPPの増加に応じて減っていることが分かる。理想的には、ベクトルレジスタ長のNPP=256、あるいは加速効果が鈍るNPP=128で作業配列を確保するのが望ましいが、システムで常時利用可能なメモリー量を考えてデフォルトとしてNPP=64を採用し、FCKOU1の積分加算部分の改造を行った：むろん、NPP値は実行時の入力データで変更可能である。なお、PPアルゴリズム導入に伴って修正された部分はFCKOU1、それに上位でのメモリー割当などを含めてステップ数で約1,500である。

3.2 スクリーニング部分

FCKOU1の動作をさらに解析していくと、加算前の積分スクリーニングの処理に加算処理本体に比しても無視し得ない時間がかかっていることが明らかになった。スクリーニングは、寄与の小さい積分値に関する無駄な加算を避けて演算の総数を減らすのに極めて有効であり、DIRACではかけ合わせるべき密度行列要素の値も考慮する工夫がなされている[3]。しかし、実装ではスカラー機での実行を想定しているために2重ループ内にIF文を多用した構造となっていることに加え、ベクトル実行時には配列要素のアドレッシングが不定になってしまい、SXの自動ベクトル化の対象外となってスカラー実行されていた。実のところ、積分加算部分をPPアルゴリズムによりベクトル化しただけの段階では、DHF計算トータルでの加速は期待よりも小さいものであった。そこで、スクリーニング部のベクトル化も行うことにした。

まず、配列の依存関係については作業用の配列を新たに導入することで回避した。IF文を多用している点については、以下のように対処した。SXシステムでは、IF文以下を実行するかどうかのビットマスクを生成するので基本的にはベクトル化可能である。ただし、IF文が真となる確率が低いとベクトル長が伸びずにメリットが出ない。そのため、なるべくループ内にIF文を置かない方が全体の効率が上がる。そこで、ループ内で変化しない変数をスクリーニングの判定に用いている処理は出来る限りループ外へ移動させた。ベクトル化の効率化をさらに図るため、2重ループ→1重ループへのループ融合を併せて行った。結局のところ、スクリーニングのベクトル化改造により、ステップ数は5倍以上(80 → 440)になった。

4 加速の結果

上述のように、サブルーチンFCKOU1におけるPPによる積分加算のベクトル化、前段スクリーニングのベクトル化、2つのチューニングを施した改造版DIRACと原版との比較を、表1の3つの例題について行った。計測はSX-4の1CPU上で、改造版での加算用の作業配列はNPP=64としている。結果を表2に示す。

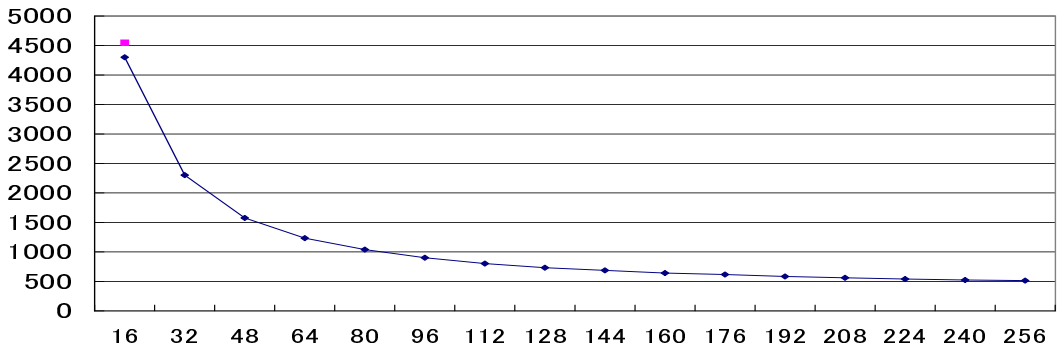

図 1: ベクトル長 NPP に対する実行時間の変化

表 2 を見ると、まずは良好な加速が得られていることが分かる。原版では 50%以上を占めていた FCKOU1 のジョブ全体での占有率は、今回の改造版では 30%以下にまで下げることが出来た。また、ベクトル化率についても 50%程度から 90%前後まで向上した。加速の効果では表中で H_2O の応答計算が 4 倍と顕著である。これは、DHF 収束後の RPA の求解でも FCKOU1 が繰り返し呼ばれるためで、物性値の評価ではベクトル化の威力が大きい。Cu と GaH 以外の他の DHF 計算でも、加速として 1.5〜2.0 倍が得られている。つまり、ベクトル CPU 1 つで 3 時間かかる DHF ジョブが 2 時間以下になることであり、様々な応用計算を展開していく上で大きな意味がある。むろん、並列実行でも今回のベクトル化による加速は活かされる。ES 稼働後は、さらにチューニングを進めて性能向上を図っていく。

最後に力の計算について触れる。DHF でも HF と同様に、各原子にかかる力を解析的に計算することにより構造最適化が可能で、DIRAC にも既に組込まれている [2]。力は、全エネルギーの一般化座標 Q に関する偏微分

$$F_Q = \frac{\partial E}{\partial Q}$$

で定義される。DIRAC では、微分された 2 電子積分の寄与を加算していく Fock 行列の構築と同様の処理が FCKOU1 で行われる。従って、ベクトル化の改造は構造最適化計算にも恩恵を与える。

5 今後

DIRAC の開発元では、密度汎関数法 (DFT) を現在組み込んでおり、これにより DHF とほぼ同様の計算コストで電子相関を実効的に取り込めるようになる [13]。DFT では格子による数値積分が現れるが、ベクトル化も並列化も共に容易である。ジョブ実行時に確保出来る CPU 数にもよるが、大小成分の基底関数の総数で数千程度の DHF/DFT 計算は ES 上では気軽に行えるのはもちろん、数万関数も視野に入るものと期待している : 対角化部分の並列化 [14] は必要となろう。また、DIRAC に実装されている摂動論等の他の相関計算法についても鋭意 ES を意識したチューニング作業を進めていく予定である。さらに、DIRAC の兄弟にあたる非相対論コード DALTON[7] についてもベクトル化を計画している。

6 謝辞

PP に基づくベクトル化について貴重なご議論を頂いた NEC 基礎研究所の高田主席研究員、NEC ソフトウェアの村瀬主任に深謝いたします [15]。また、

表 1: 例題

対象	計算	大成分 (総数)	小成分 (総数)
Cu	DHF	$15s10p6d$ (81)	$10s21p10d6f$ (193)
GaH	DHF	Ga: $16s12p8d$ (100)	$12s24p12d8f$ (236)
		H: $5s2p$ (11)	$2s5p2d$ (29)
H_2O	RPA	O: $9s4p1d$ (27)	$4s10p4d1f$ (68)
		H: $4s1p$ (7×2)	$1s4p1d$ (19×2)

表 2: 加速の結果

	Cu		GaH		H_2O	
	原版	改造版	原版	改造版	原版	改造版
全 CPU 時間 (秒)	256.39	106.16	1032.05	678.32	475.59	116.82
加速		2.4		1.5		4.1
FCKOU1 占有率 (%)	61.9	21.7	54.1	29.3	57.3	21.9
ベクトル化率 (%)	54.0	92.5	56.6	89.1	45.0	84.4

DIRAC の筆頭開発者で原版 FCKOU1 のコーディングをされたトロムソ大学 (ノルウェー) の Saue 博士から頂いたベクトル化に対する励まし、加えて DFT に関する情報 [13] について感謝いたします。

参考文献

[1] Pyykkö: Chem. Rev. **88** 563 (1988)

[2] http://dirac.chem.sdu.dk/

[3] 望月、館脇: 2000 年 分子構造総合討論会 3C8 (東京大学) / 2001 年 原子力学会春期年会 K2 (武蔵工業大学) / 2001 年 日本化学会春期年会 3D310 (甲南大学)

[4] http://www.gaia.jaeri.go.jp/main.html

[5] Saue et al.: Mole.Phys. **5** (1997) 937 / J. Chem. Phys. **111** (1999) 6211

[6] Helgaker et al.: "Molecular Electronic-Structure Theory", Wiley (2000) の積分生成に関する章を参照

[7] http://www.kjemi.uio.no/software/dalton/dalton.html

[8] Norman et al.: Chem. Phys. Lett. **253** (1996) 1

[9] Visscher et al.: Chem. Phys. Lett. **274** (1997) 181

[10] Nishiguchi et al.: J.Comp.Phys. **61** (1985) 519

[11] 村井ら: 並列処理シンポジウム JSPP'97、論文報告 p181

[12] Takashima et al.: J. Comp. Chem. **20** (1999) 443

[13] Saue, Helgaker.: Poster at DFT2000 Conference, http://ket.ch.cam.ac.uk/dft2000/

[14] 今村: 1999 年 応用数理学会 講演予稿集 p68 (愛媛大学)

[15] 高田、村瀬: 私信