

SIMD 演算の効率的利用に向けた アレイジョブ型 Hartree-Fock 法計算

本田 宏明^{1,4,a)} 稲富 雄一^{2,4} 眞木 淳³

受付日 2014年4月4日, 採録日 2014年9月24日

概要: 量子化学分野で頻繁に利用される制限 Hartree-Fock 分子軌道法計算では, これまで多くの効率的なアルゴリズムが提案され実装されている. しかしながら, その計算ボトルネックである 2 電子フォック行列 (G 行列) 計算において現状のアルゴリズムでは SIMD 演算を有効に利用することは難しい. そのため本研究では, 分子構造の異なる複数の入力データを同時に計算するアレイジョブ型の並列計算を SIMD 演算に割り当て効率的に利用する方法を提案した. また複数の同一入力によるテスト計算を行い, 本方法を利用した SIMD 演算による最大の性能向上値を測定した. その結果, 並列の G 行列計算に対し 2 ウェイ, 2 ウェイならびに 4 ウェイ, 8 ウェイの倍精度浮動小数点 SIMD 演算が可能な Intel Xeon X5650, Intel Xeon E5-2650, Intel Xeon Phi において, それぞれ最大 1.82 倍, 1.97 倍, 5.84 倍の速度向上が達成された. この結果から, 分子構造のほぼ同じ複数入力データ並列計算を SIMD 並列演算で計算する本方法は有効であると考えられる.

キーワード: Hartree-Fock 分子軌道法計算, アレイジョブ, 複数入力並列計算, SIMD 演算, Xeon プロセッサ

Array Job Type Hartree-Fock Calculation for Improving SIMD Operation Efficiency

HIROAKI HONDA^{1,4,a)} YUICH INADOMI^{2,4} JUN MAKI³

Received: April 4, 2014, Accepted: September 24, 2014

Abstract: Conventional Restricted Hartree-Fock (RHF) programs are often used in quantum chemistry studies and many efficient algorithms have been proposed and implemented. However, these algorithms have quite low SIMD operation level parallelism for the two-electron Fock-matrix (G-matrix) calculation which is the bottleneck of RHF program. We propose a array job type RHF calculation which calculates multiple-inputs of different molecular geometries in parallel, and those calculations are applied for SIMD parallel operations. We evaluated maximum performance for our method with test inputs whose molecular geometries are same by three types of desktop computers with 2-way SIMD Xeon X5650, 2 and 4-way SIMD E5-2650, and 8-way SIMD Phi processors. For parallel G-matrix computations, we achieved 1.82, 1.97 and 5.84 times speed-ups for Intel Xeon X5650, E5-2650, and Phi processor computers, respectively. It is expected that multiple-inputs parallel calculation with almost same molecular geometries, will improve the SIMD operation efficiency for practical calculations.

Keywords: Hartree-Fock molecular orbital calculation, array job, multiple-inputs parallel calculation, SIMD operation, Xeon processors

¹ 九州大学情報基盤研究開発センター
RIIT, Kyushu University, Fukuoka 812-8581, Japan
² 九州大学システム情報科学研究所
ISEE, Kyushu University, Fukuoka 819-0395, Japan
³ 九州先端科学技術研究所
ISIT, Fukuoka 814-0001, Japan
⁴ 独立行政法人科学技術振興機構, CREST

1. はじめに

理論化学分野, とりわけ *ab initio* 分子軌道法に基づく量

JST-CREST, Kawaguchi, Saitama 332-0012, Japan
a) honda.hiroaki.971@m.kyushu-u.ac.jp

子化学計算の分野は、近年の計算機環境の著しい発展とあいまって現在に至るまで着実に進歩しており、無機、有機、生化学や創薬等の広汎な分野へと応用が進んでいる。しかしながら、生体系に代表される大規模分子系への適用には種々の解決すべき問題が依然として残されている。その1つが、分子軌道法の Hartree-Fock 近似手法の際に実行時間ボトルネックとなる Fock 行列要素計算の効率的アルゴリズムの開発である。大規模系を対象とした Fock 行列の効率的な計算方法についての基本アルゴリズムとしては次章に説明する積分駆動型アルゴリズムがすでに確立されており [1], 多くの量子化学計算で実際に適用されている。しかしながら、計算対象の分子系がますます大きくなっている現在では、さらなる Fock 行列計算の高速化が必要となりつつある。

また大規模量子化学計算を実行するスーパーコンピュータについては、すでに 10 ベタスケールクラスが実現されており、100 倍の性能のエクサスケールに至るコンピュータシステムやソフトウェア技術、それに関わる周辺技術を扱う研究がさかんに行われている。その有名な技術報告としては、米国 DARPA による “Exascale Computing Study” [2] や、国際間連携プロジェクトによる “International Exascale Software Project” [3], 国内では「今後の HPCI に関する技術開発に関わる報告書」[4] 等が発表されている。それらの多くの研究では、アクセラレータを含むメニーコアプロセッサを搭載した数十万ノード以上の大規模並列コンピュータが開発されるのではないかと予想されており、このようなシステムを対象にしたハードウェアやソフトウェア両面における様々な議論が展開されている。

このような次世代の大規模並列コンピュータに対して、分子軌道法ではフラグメント分子軌道法等の適用により MPI や OpenMP によるハイブリッド並列を利用した 2 万並列に達するスケーラブルな計算が報告されている [5] が、メニーコアプロセッサへの対処については、GPGPU による研究が進んでいるものの [6], [7], [8], [9], [10], [11], 十分であるとはいえないのが実情である。

今後開発されるメニーコアプロセッサは、高性能低消費電力化の要請から個々のコアについては制御回路を簡素化する方向に、多数のコア間についてはネットワーク型通信方法の方向に向かうと考えられており、以下のような主要な特徴を持つと予想されている。

- (1) 100 コア以上のメニーコア構成
- (2) プロセッサごとの主記憶量の低減ならびに主記憶に対するコアごとの提供メモリバンド幅の低下
- (3) 命令処理におけるインオーダー実行
- (4) バス方式からパケットスイッチネットワーク通信型へのコア間通信方式の変更
- (5) 16 ウェイや 32 ウェイの SIMD 演算器の実装

これらの特徴を考慮した有効なプログラム実装が必要

になるが、本研究では特に多数のウェイ数を持つ並列度の高い SIMD 演算器の有効利用に着目した。2.5 節に後述するが、本研究が対象とする一般に広く利用されている Hatree-Fock 分子軌道法の計算ボトルネックとなる 2 電子 Fock 行列 (G 行列) 計算では全演算における SIMD 演算割合が 1%未満と低く、今後 16 ウェイや 32 ウェイの SIMD 演算器を持つプロセッサが開発された場合、最大でも 1/16 や 1/32 程度の実行性能となってしまうことが危惧されるためである。また、分子軌道法アプリケーションに対し SIMD 演算の使用効率に着目した研究はこれまでなされていない。

本研究ではこの問題に対し、複数の入力からなる同一の計算を SIMD 演算器に割り当てて同時に処理することで分子軌道法計算においても SIMD 演算資源を有効活用する方法を提案し、新規の実装ならびに性能評価を行った。量子化学計算では分子の解離過程や反応過程を対象とする場合に多数の分子構造の考慮を行う。この際には構造以外のすべての計算パラメータについて同じ計算を実行することが多く、複数入力データによる並列計算が実際の研究の場面で有効利用可能である。また、この手法は一般的にアレイジョブやバルクジョブと呼ばれる、入力の異なる複数の計算ジョブを並列に動作させる方法のみを利用している。しかしながら通常想定されているプロセスレベルでの並列計算ではなく、SIMD 演算レベルで複数入力データに対する並列計算を実行する点が異なる。

本研究では、複数入力並列計算を対象とした SIMD 演算の最大性能を調べるため、複数の同一入力を与えた場合の実行効率を測定した。倍精度浮動小数点数について異なる SIMD 演算ウェイ数を持つ 3 種類の Xeon プロセッサ、Xeon X5650 (2 ウェイ), Xeon E5-2650 (2, 4 ウェイ), Xeon Phi (8 ウェイ) を搭載したデスクトップ計算機環境を準備し、複数入力並列計算と通常の計算結果を比較した。また、実行時間測定とともに Xeon X5650 と E5-2650 プロセッサでの計算結果に対しては SIMD 演算効率や全演算中における浮動小数点演算数についてのハードウェアカウンタ測定値に基づく解析を行った。

2 章では分子軌道法プログラム全体構成ならびにボトルネックとなる G 行列計算構成、既存アルゴリズムにおける SIMD 演算が非効率となる原因について述べる。3 章では複数入力並列計算について、分子軌道法全体計算での取扱いや G 行列計算における SIMD 並列計算とするための実装の詳細、その有効性について述べる。4 章ではその性能評価について、5 章にまとめを述べる。

2. 既存分子軌道法計算と SIMD 演算効率

ここでは、分子軌道法計算手法の 1 つである偶数電子分子系における Restricted closed-shell Hartree-Fock 法 (RHF 法) を対象とし、その背景となる定式化、既存計算アルゴ

リズムならびに既存手法での SIMD 演算実行効率について説明する。

2.1 RHF 分子軌道法

RHF 法では以下の式 (1) に示す Hartree-Fock-Roothaan 方程式

$$\sum_{j=1}^N \{H_{ij}^{core} + G_{ij}\} C_{ja} = \sum_{j=1}^N S_{ij} C_{ja} \epsilon_a \quad (1)$$

を解くことにより, a 番目の分子軌道 $\phi(\mathbf{r})$

$$\phi_a(\mathbf{r}) = \sum_i^N C_{ia} \chi_i(\mathbf{r}) = \sum_i^N C_{ia} \sum_j^{M(i)} d_{ij} \xi_j(\mathbf{r}) \quad (2)$$

ならびに複数の分子軌道から構成される単一の Slater 行列式により全波動関数を得る [12]. 上式における N は基底関数数, H_{ij}^{core} は 1 電子 Fock 行列, G_{ij} は 2 電子 Fock 行列 (G 行列), S_{ij} は以下の式 (3) で定義される原子中心 $\mathbf{R}$ に複数配置された位置 $\mathbf{r}$ における角運動量 $\mathbf{n}$ の Cartesian Gauss 関数の縮約和

$$\begin{aligned} \chi_i(\mathbf{r}) &= \chi(\mathbf{r}, \mathbf{n}_i, \zeta_i, \mathbf{R}_i) \\ &= \sum_j^{M(i)} d_{ij} (r_x - R_{ix})^{n_{ix}} (r_y - R_{iy})^{n_{iy}} (r_z - R_{iz})^{n_{iz}} \\ &\quad \times \exp[-\zeta_j(\mathbf{r} - \mathbf{R}_i)^2] \end{aligned}$$

の重なり積分であり, 以下のようにそれぞれ定義される.

$$H_{ij}^{core} = \int \chi_i(\mathbf{r}_1) h(1) \chi_j(\mathbf{r}_1) d\mathbf{r}_1 \quad (3)$$

$$G_{ij} = \sum_{k=1}^N \sum_{l=1}^N D_{kl} \{2(ij, kl) - (ik, jl)\} \quad (4)$$

$$D_{ij} = 2 \sum_a^{N_{elec}/2} C_{ia} C_{ja} \quad (5)$$

$$S_{ij} = \int \chi_i(\mathbf{r}_1) \chi_j(\mathbf{r}_1) d\mathbf{r}_1. \quad (6)$$

ここで, $h(1)$ は 1 電子 Hamiltonian 演算子, N_{elec} は分子内電子数, (ij, kl) は以下の式 (7) で定義される電子反発積分である.

$$(ij, kl) = \int \int \chi_i(\mathbf{r}_1) \chi_j(\mathbf{r}_1) \frac{1}{|\mathbf{r}_1 - \mathbf{r}_2|} \chi_k(\mathbf{r}_2) \chi_l(\mathbf{r}_2) d\mathbf{r}_1 d\mathbf{r}_2. \quad (7)$$

この Hartree-Fock-Roothaan 方程式に基づく RHF 法では, 一般的には非線形の微積分方程式で記述される Hartree-Fock 方程式 [12] を, 無限遠方で 0 に漸近するあらかじめ定められた関数系 $\{\chi\}$ を用いることにより式 (1) の線形固有値問題に帰着させ, その解を式 (2) による $\{\chi\}$ の線形結合で与えられると近似している. その結果, 線形結合係数

$\mathbf{C}$ をコンピュータにより効率的に解くことを可能にしている. また, その計算においては, 方程式 (1) の対角化対象の一部の $\mathbf{G}$ 行列内に解となる線形結合係数 $\mathbf{C}$ を含むため, $\mathbf{C}$ から求められる密度行列 $\mathbf{D}$ が収束するまで繰返し計算を必要とする.

また計算式 (4) から, G_{ij} のすべての要素を求めるためには, $O(N^4)$ の計算コストが必要であるといえる. ここで, N は分子サイズと計算精度に比例した値となる. しかしながら, 重なり電荷条件による積分のカットオフや Schwarz の不等式の利用により, 実際の計算ではある閾値以上の項のみ和を計算するため, 対象とする系により $O(N^2)$ 近くまで計算量を削減することも可能である [13]. それでも巨大な分子系では N の 2 乗以上に比例した膨大な計算が必要である.

2.2 分子軌道法計算の全体実装アルゴリズム

図 1 に RHF 分子軌道法プログラムの主要構成を示す. 入力の後, 1 電子 Fock 行列計算ならびに初期密度行列計算を行う. 次に得られた密度行列を用いた G 行列計算を行い, Fock 行列対角化を通して新規の密度行列計算を求める. ここで, G 行列計算の入力としての密度行列と対角化後の新規の密度行列が収束し, ほぼ同じ結果が得られるまで繰返し計算を行う. 収束後に得られた密度行列計算から種々の物理量の計算を行う.

G 行列計算部分が全体計算におけるボトルネックであることが知られており, ペプチド分子等の基底関数数が 200 を超える分子ではその計算実行時間において 99% を占める. そのため分子軌道法計算の高速化のためには G 行列計算の高速計算が必須である [14].

2.3 G 行列計算実装

図 2 にボトルネックとなる G 行列計算のアルゴリズムを示す. 本研究では計算に積分駆動型アルゴリズム [1] を採用した. 全体として 8 重ループ構造をとっており, 大き

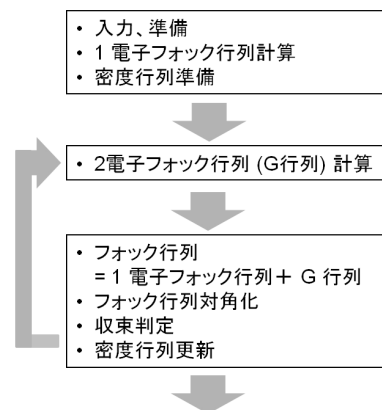


図 1 RHF 分子軌道法プログラムの主要構成

Fig. 1 Main structure of RHF molecular orbital program.

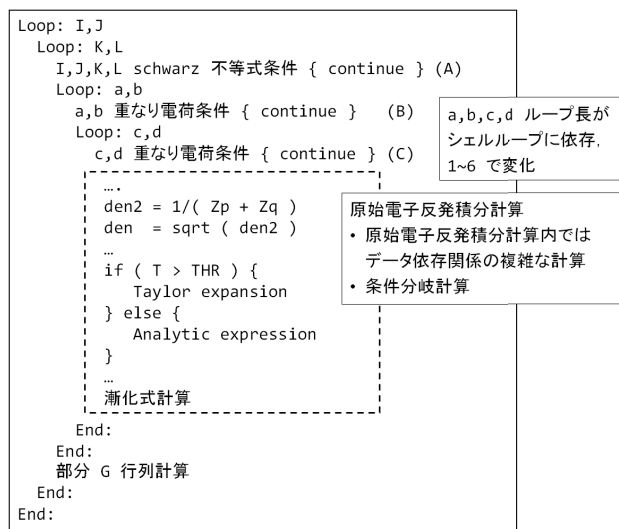


図 2 G 行列計算アルゴリズム

Fig. 2 G-matrix calculation algorithm.

な分子では式 (4) に対応する上位 4 重ループにて各ループ長が数百を超え、計算ボトルネックの原因となっている。式 (3) に対応した下位 4 重ループは上位ループの変数の値に応じて各ループ長が異なっており、1~6 の値をそれぞれとることが多い。8 重ループ最下層のループボディ部において原始電子反発積分と呼ばれる量が計算される。本研究では原始電子反発積分を小原積分法 [15] を利用し計算した。この小原積分法では積分を初期項と漸化項に分け計算する。初期項では 8 重ループの各ループ変数に依存して決定される値 T を計算し、この T 値により条件分岐による不完全ガンマ関数の計算を行う。 T がある閾値より小さい場合にはテーブル参照による Taylor 展開の方法で、 T が閾値より大きい場合には平方逆数計算を含む解析的な方法により計算する。漸化式部分については、上位 4 重ループ変数 I, J, K, L の組合せに依存し異なった種類の計算を行う。式 (3) に示した計算に使用する原子の Cartesian Gauss 型関数の組の最大角運動量により、漸化計算の種類（積分タイプ）数が異なり、 $L = n_x + n_y + n_z$ で定義される角運動量子数を、 $L = 0, 1, 2, 3 \dots (s, p, d, f \dots \text{関数})$ まで考慮する際、1, 6, 21, 55, $\dots$ 種類の候補より漸化計算式がそれぞれ選択される。

また、計 8 重ループにもなる大きな計算のため、不要な計算の削減のために図中の (A), (B), (C) の位置でループ内の計算寄与が閾値より小さいと判断される場合には計算がスキップされる。

分子軌道法に一般に用いられている静的負荷分散に基づくプロセス並列化計算では、図中の Loop: K, L 以下のループ計算回数による Cyclic 分割が用いられることが多く、本研究の実装においても同様の方法を採用した。

2.4 本研究での対象プログラム

本研究で利用した RHF 法プログラムは、九大の稲富らが中心に開発している超並列フラグメント分子軌道法プログラム (OpenFMO) のプロジェクト [16] の一部で開発された、すべて C 言語にて記述されたプログラムの一部である。本研究では、入力基底関数と原子順によって定まるシェルの順番を角運動量によって整列することで G 行列計算内で (ss, ss) , (ps, ss) , $\dots$ と、各積分タイプ順に G 行列を計算し加算する方法 (シェルソート法) を採用した。一方、一般に利用されている分子軌道法プログラム GAMESS [17] 等では、単一の G 行列プログラムで入力のシェル順に計算し複数の積分タイプを選択しながら計算する方法が採用されている。シェルソート法では、G 行列計算内の上位 4 重ループ内において分子積分の計算の際に積分タイプの条件分岐による選択の必要がない。また、図 2 の部分 G 行列計算内部において縮約基底分子積分から G 行列を計算する際に、実行時のループ長決定が不必要になり、固定の数値をあらかじめコード中に与えることが可能である。そのためコンパイラによる SIMD 計算向けコード生成が通常の方法に比較して容易であると予想される。

2.5 G 行列計算と SIMD 演算効率

上記に説明した通常の計算アルゴリズムは GAMESS 等の多くのプログラムで使用されているが、その SIMD 演算効率は低い。既存の GAMESS プログラムに対する評価のため、 $(\text{H}_2\text{O})_8$ 分子に対し G 行列計算部分のみのテスト計算を行った。ここで、積分計算アルゴリズムは GAMESS デフォルトである Pople の方法 [18] と Fletchar の方法 [19] との組合せを利用している。Intel Xeon X5650 プロセッサ、Intel Compiler 13.0 の -O3 最適化で PAPI ライブラリ [20] によるハードウェアカウンタの取得結果から、SIMD 化された浮動小数点演算数割合は 1% 未満であった。

一般に、コンパイラの自動並列化機能を利用しコード中の多重ループ箇所の SIMD 化オブジェクトコードの生成を行うには、最内ループが容易にループアンローリング可能であることならびにループボディ部にデータ依存性が少ないことが必要である。しかしながら、上記の一般的な G 行列計算コードでは、1. 下位 4 重ループ長が上位 4 重ループの変数 I, J, K, L に依存しすべて実行時に決定される、2. 原始積分計算からなるループボディ部に複雑なデータ依存性があり、条件分岐も存在する、といった性質があり、効率的な SIMD 計算が妨げられていると考えられる。

2.6 SIMD 演算が可能なアクセラレータによる既存並列計算

G 行列計算をアクセラレータにより加速実行するこれまでの試みとして、SIMD 演算器を持たない ERIC 2 電子積分計算専用プロセッサによる計算 [14] があるが、SIMD 演算

が可能なプロセッサでは、Cell プロセッサや GRAPE-DR によるテスト計算 [21], [22] のほか、GPGPU による計算が複数報告されている [6], [7], [8], [9], [10], [11]. 特に最近の GPGPU による研究結果では、1CPU に対し 20 倍以上の高速化が達成されている [11]. GPU の計算では上位 4 重ループが互いに計算依存性がないことに着目することで複数のスレッドからなるウォープ単位での SIMD 計算ならびに各スレッドレベルでの並列計算が行われている. しかしながらこれらの研究では、スレッドレベルでの動作を考慮した効率の良い実装について議論されているものの、各ウォープでどの程度の演算が実際に SIMD 実行されたかの定量的な調査までは行われていない.

3. 複数入力並列量子化学計算

まず、本研究で提案する RHF 計算に対する複数入力並列計算のためのプログラム実装について説明し、複数入力に関わる使用メモリ量の増加に対し RHF 計算では問題のないことを説明する. 最後に複数入力計算の具体的な使用方法ならびに有効性について説明する. なお、本分子軌道法計算において浮動小数点演算はすべて倍精度で行っている.

3.1 複数入力 RHF 分子軌道法全体プログラム

2.2 節で説明したが、プログラム全体においてのボトルネックは G 行列計算部分である. そのため、G 行列計算箇所のみで複数入力並列計算を SIMD 演算に割り当てて計算した. 図 1 に示したオリジナル RHF プログラムを変更した、複数入力 RHF 分子軌道法プログラムの主要構成を図 3 に示す.

1 電子 Fock 行列, G 行列, 密度行列等, プログラム全体を通して利用するデータは、すべて異なる入力ごとに離散

的な配置とした. このようにすることで SIMD 化計算を行う G 行列計算内部以外のコードでは既存のプログラムの再利用が可能である. G 行列計算内部では、計算の主要な入力である密度行列を、異なる入力ごとに離散的な配置から、行列要素の異なる入力に対応するデータを連続配置とするために並べ換えを行う. G 行列の計算後には、連続に並んだ異なる入力に対応する行列要素データを入力ごとの離散的な配置へ再び並べ戻しを行った. このように G 行列内部と外部とでデータ配置を変更することにより、G 行列計算以外では既存コードの大部分がそのまま利用可能である.

3.2 複数入力 G 行列計算プログラム

G 行列計算内部では、複数入力並列の計算に対しても分子構造パラメータ以外の入力は同一であるため、制御構造はおおまかにはオリジナルプログラムと同様である. しかしながら、図 2 に示した、(A), (B), (C) の 3 カ所の条件において、分子構造の異なる入力に依存して一般に真になる条件が異なる. そのため、複数の入力に対応する条件がすべて真になる場合にのみループボディ部の計算が省略可能である.

図 2 に示した G 行列計算内部での 8 重ループ内ループボディの原始電子反発積分部分について、プログラム変更点を含めた複数入力向けプログラムを図 4 に示す. 前節で説明したように、小原積分法を実装しているループボディ部は、8 重ループのすべてのループ添字に依存した条件分岐構造を含む初期項計算ならびに上位 4 重ループに依存した漸化計算部分がある. このループボディ部分の初期項および漸化計算に関する条件分岐箇所以外のすべてのコードについて、複数入力由来のループを作成して計算を行った. このループ記述に対しては Intel Compiler の言語拡張である配列表記を利用することで、複数の入力からなる計算を自動 SIMD 化対象とする指定を行った. ここでループ長に対応する数を即値で指定した. この際、データの適切なメモリ境界設定、ならびにメモリ境界を前提とした SIMD 演算生成のため、“`posix_memalign()`” 関数や、“`__attribute__((aligned()))`”, “`__assume_aligned()`”, “`#pragma vector aligned`” ディレクティブを使用した.

初期項計算の一部には、各入力に依存した $T[i]$ の値により真偽の定まる条件分岐箇所が存在する. この箇所については条件分岐箇所を内包するように各入力ごとに通常のループ計算コードを記述することとした.

この計算ボトルネック部分の G 行列計算のうち、制御構造自身の実行時間は無視できる程度であり、1 入力計算ではその 99.0% が各ループボディの実行時間の積算となっている. ループボディ部分の初期項計算に条件分岐を含むものの、漸化計算部分を含めるとループボディ部分のコードの大半が互いに依存性のない複数入力由来のデータから

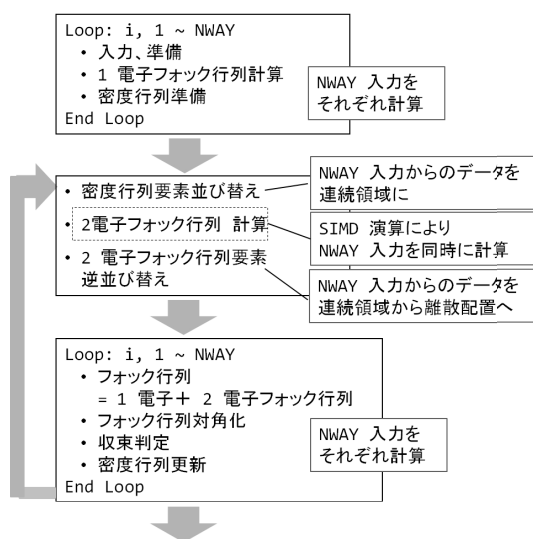


図 3 複数入力 RHF 分子軌道法プログラムの主要構成

Fig. 3 Main structure of multiple-inputs RHF calculation.

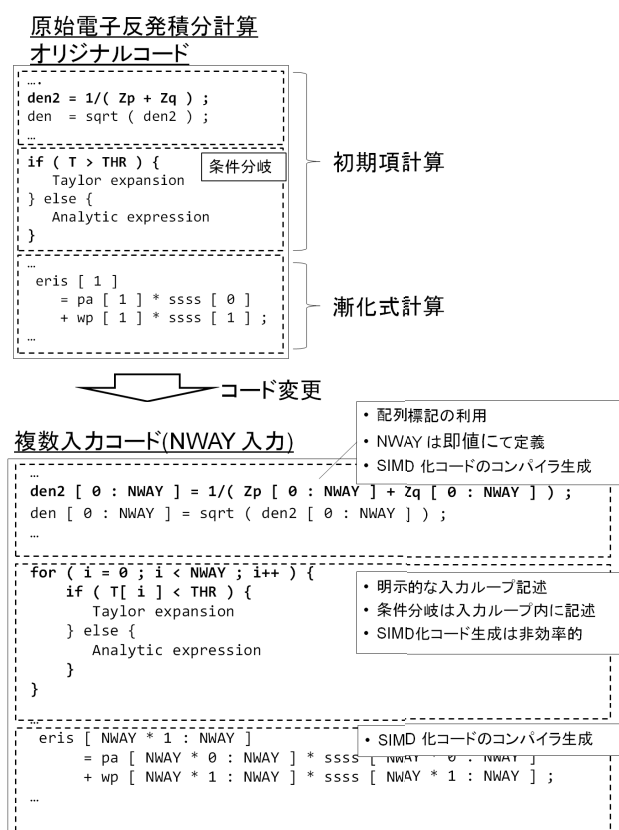


図 4 複数入力原始電子反発積分プログラム

Fig. 4 Multiple-inputs primitive electron repulsion integral program.

なる演算として、効率良く SIMD 化可能であると考えられる。

3.3 複数入力計算に関わる使用メモリ量の増加

複数入力からなるアレイ型ジョブを行う場合では、入力分に比例したメモリ量が必要である。一般の大規模 RHF 計算の基底関数数は実用的には 3,000 程度までであるが、3,000 基底では 1 入力の G 行列サイズは 34 MB 程度となり、計算に必要な密度行列 (D 行列) を合わせると 68 MB 程度となる。そのため、32 入力まで計算する場合では、G + D 行列サイズならびに 1 つ前の繰返し分を合わせて 4.4 GB 程度の記憶容量が必要になるが、これは将来のミニコアプロセッサ 1 CPU が利用可能な主記憶量に収まると予想され、計算が十分可能であるといえる。

3.4 複数入力計算の有用性

本研究で考慮している複数入力計算のユースケースとして、分子の振動モードの振動数を計算により求める場合があげられる。ここで、 H_2O 分子の 3 種類の振動モードを例にすると、その振動モードの計算には、まず H_2O 分子の 2 種類の $R(\text{O-H})$ 核間距離ならびに $\angle\text{H-O-H}$ の角度を独立に変化させた複数の分子構造に対する分子のエネルギーを求め、次にこれらのエネルギー値を利用したフィッティング

操作により、すべての分子構造自由度に対するエネルギー曲面に対する 2 次曲面近似を行うことで、その曲率から各振動モードの振動数を求める。この際、各自由度あたり平衡核配置付近の 5 点以上の点を用いることも多く、3 自由度を独立に計算する場合では $5^3 = 125$ 点分の分子構造の計算が必要となる。

本提案の複数入力計算の有効性は、計算に与える複数の入力のループボディ計算がどれだけ重複しているかに依存しており、ほぼ同じ分子構造であれば、図 2 の条件分岐 (A), (B), (C) の振舞いが似るため、ループボディの計算が重複する度合いが大きい。また、同一分子の場合には計算量が最小となる。上記の H_2O の 125 構造点分の計算において、逐次的に計算する場合と本研究の複数入力の同時計算を行う場合では、それぞれ 9,082,125 回ならびに 9,097,875 回のループボディの計算となり、ループボディ計算量の増加は 1.002 倍とわずかであった。このため、同一入力を使用した場合に SIMD 演算による速度向上比がこの値を大きく上回る場合には異なる分子構造入力での計算においても有用であると判断される。

4. 性能評価実験

4.1 性能測定方法

本研究では、2 ウェイの倍精度浮動小数点演算である SSE4.2 命令をサポートする Intel Xeon X5650 プロセッサ (2.67 GHz, 6 コア)、2 ウェイの SSE4.2 命令ならびに 4 ウェイの倍精度浮動小数点演算である AVX 命令をサポートする Xeon E5-2650 プロセッサ (2.00 GHz, 8 コア)、8 ウェイの倍精度浮動小数点演算である SIMD 命令をサポートする Xeon Phi プロセッサ (2.00 GHz, 60 コア) をそれぞれ搭載した、3 種類のデスクトップパソコンの計算環境を利用した。コンパイラとして、Intel Compiler 13.0.1 を利用し、すべてのプロセッサについての -O3 最適化のほか、Xeon E5-2650 環境では -AVX 最適化オプションを追加した。Xeon Phi については native 実行を行った。

計算の複数の入力データについてはすべて同一とし、 $(\text{H}_2\text{O})_8$ 分子、6-31G 基底関数系を使用した。この計算における 1 入力の基底関数数は 182 であり、G 行列のサイズはおおよそ 16.8 KB、32 入力計算でも 520 KB 程度である。この入力では図 2 の上位 4 重の各ループ長が最大 104 となり、下位 4 重ループ長は 1~6 で実行時に変動する。

各プロセッサに対し、コア分の MPI プロセスを立ち上げ、MPI のみのプロセス並列計算を行った。性能測定対象は 1 回の G 行列計算とし、MPI_Wtime 関数により実行時間を取得し、有効 GFLOPS 値ならびに実行効率を求めた。GFLOPS 数を算出する際の演算数については、今回の G 行列計算では内部に複雑な分子積分計算を含むため式から得ることが煩雑すぎるために SIMD 演算が発生しない条件で最適化されたプログラムの実行浮動小数点演算数を利用

した．具体的には Xeon X5650 環境下における -O3 最適化プログラムの 1 入力 1 並列の計算におけるハードウェアカウンタ値を利用した．実行効率を求める際のピーク性能値は Xeon X5650, E5-2650, Phi それぞれに対し, 64.08 (6 コア), 128.00 (8 コア), 1047.36 (60 コア) GFLOPS の値を使用した．この際, 実行時間にばらつきがあったため, 各プロセッサにおける 200 回ずつの測定で得られた最小の実行時間を採用した．また, X5650 と E5-2650 環境において, 各プロセッサに対し Hyper-threading 機能をオフにしたうえで計算のハードウェアカウンタ値を PAPI ライブラリ [20] を利用し測定した．ただし現在の PAPI 5.3.0 では, E5-2650 における倍精度 2 ウェイと 4 ウェイ SIMD 浮動小数点演算数が個別に出力されないため, 個別のカウンタ数を出力するための修正を施した．

4.2 複数入力並列計算の速度向上比

逐次実行計算における N 個の同一入力を処理する計算では, 1 個の入力の通常計算と比較して N 倍の実行時間が必要であるが, SIMD 演算資源を利用した並列計算が可能である場合には実行時間が減少する．これに対し, 本研究では以下で定義する速度向上比 (*Speedup.ratio*) を利用し評価した．

$$\text{Speedup.ratio} = \frac{\text{Exec.time}(1) \times N}{\text{Exec.time}(N)} \quad (8)$$

ここで $\text{Exec.time}(1)$, $\text{Exec.time}(N)$ は 1 入力ならびに N 入力における実行時間を示す．例として, 2 ウェイの倍精度 SIMD 演算器が利用可能なプロセッサで 2 入力の計算を行った場合, もしも効率的な SIMD 演算により 1 入力計算と同じ実行時間であったならば, 速度向上比 2 が得られる．このように上式 (8) で定義された速度向上比は, SIMD ウェイ数 (*SIMD.ways*) に対し

$$1 \leq \text{Speedup.ratio} \leq \text{SIMD.ways} \quad (9)$$

の性質を持ち, 最大で SIMD 演算器のウェイ数の値となる．

4.3 G 行列計算の性能測定結果

倍精度 2 ウェイ SIMD 演算器が利用可能な Xeon X5650 と 2, 4 ウェイ SIMD 演算器が利用可能な Xeon E5-2650, 8 ウェイ SIMD 演算器が利用可能な Xeon Phi のプロセッサからそれぞれ構成される 3 種類の計算機における 1~32 の計算入力データ数に対する G 行列の計算速度向上比を図 5 に示す．

Xeon X5650 における計算では, 入力数が増加するに従い速度向上比が増加し, 16 並列で 1.8 倍程度に達する．Xeon E5-2650 では, X5650 と同様に, 入力数が増加するに従い増加し, 16 並列では 2.0 倍程度に達した．Xeon Phi については, 16 並列で 5.8 倍の性能向上が得られた．その結果, すべてのプロセッサで高速化が達成された．特に,

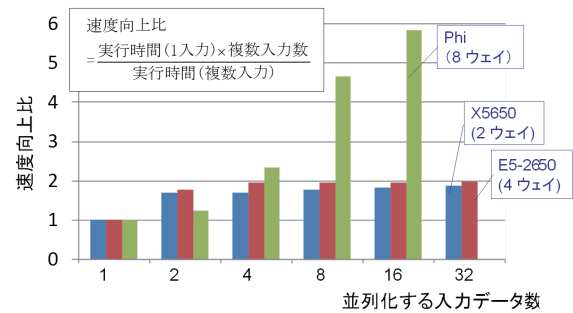


図 5 計算入力データ数に対する G 行列計算速度向上比

Fig. 5 Speedup ratios for multiple-inputs G-matrix calculation.

Xeon X5650 と Phi については, SIMD 演算のウェイ数に対する速度向上比が, 85%と 73%と, 効率的な結果が得られており, 本研究の方法は同一の複数入力の条件では有効であるといえる．また, E5-2650 における性能向上比は最大 SIMD 演算ウェイ数 4 に対して 50%と, 性能向上は果たしているものの X5650 や Phi に比較すると低い結果となった．

表 1 に, 上記計算において得られた有効 GFLOPS 値ならびに実行効率を示す．有効 GFLOPS 値については, X5650, E5-2650, Phi のピーク性能値が低い順番に良い結果が得られた．また実行効率については, 16 入力計算において 12.61%, 7.99%, 1.35%の結果が得られた．1 入力計算と比較した実行効率比は, すでに示した図 5 の速度向上比の結果に対応して 1.82, 1.97, 5.84 倍とそれぞれ向上しており, 複数入力を同時に扱うことで実行効率の向上を得ることが可能であった．

3.4 節において, 実際の H_2O 分子の 125 構造点計算の複数入力計算を行う場合では, 異なる構造を逐次的に計算する場合に比較しループボディ計算回数が 1.002 倍だけ増加することを示した．これに対し, 同一入力の速度向上比は Xeon Phi で 5.84 倍と大幅に大きく, 少なくとも上記の平衡核配置付近のほぼ同じ構造を持つ複数の H_2O 分子では有用な計算であると判断可能であり, 他の分子系に対しても同様に適用可能であると期待される．

4.4 G 行列計算のハードウェアカウンタ測定結果

G 行列計算の全倍精度浮動小数点演算数における SIMD 演算数比についてのハードウェアカウンタ測定結果を図 6 ならびに図 7 に示す．図 6 から, Xeon X5650 では複数入力並列度が 1 で SIMD 演算数比がほぼ 0 であるのに対し, 入力並列度が 2 以降ではほぼ 0.99 と, 100% 近くの浮動小数点演算が SIMD 化されている．一方, 図 7 に示した, 2 ウェイ, 4 ウェイの 2 種類の倍精度浮動小数点 SIMD 演算が可能な Xeon E5-2650 の結果から, 複数入力並列度が 1 の場合には SIMD 演算数比が 0.1 以下であったのに対し, 入力並列度が 2 の場合では 2 ウェイ SIMD 演算比が 0.98 と

表 1 有効 GFLOPS 数と実行効率
Table 1 Effective GFLOPS and execution efficiencies.

プロセッサ	有効 GFLOPS 値 ¹		実行効率 ² (%)		実行効率比
	1 入力	16 入力	1 入力	16 入力	
X5650 (2 ウェイ)	4.42	8.08	6.90	12.61	1.82
E5-2650 (2, 4 ウェイ)	3.88	7.67	4.04	7.99	1.97
Phi (8 ウェイ)	1.23	7.16	0.23	1.35	5.84

¹ 測定値による実行浮動小数点演算数/実行時間 (sec.)

² 有効 GFLOPS 値/ピーク性能値

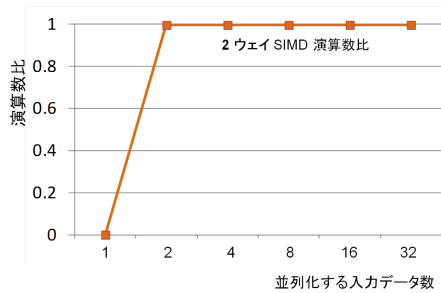


図 6 G 行列計算の全浮動小数点演算数における SIMD 演算数比 (Xeon X5650 を使用)

Fig. 6 Ratios of floating point SIMD operations to total floating point operations for G-matrix calculation (Xeon X5650).

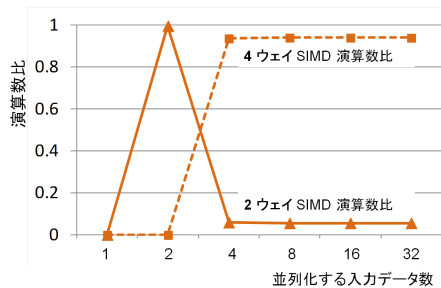


図 7 G 行列計算の全浮動小数点演算数における SIMD 演算数比 (Xeon E5-2650 を使用)

Fig. 7 Ratios of floating point SIMD operations to total floating point operations for G-matrix calculation (Xeon E5-2650).

なった。入力並列度が 4 の場合では 2 ウェイ SIMD 演算が最大 0.04, 4 ウェイ SIMD 演算が最大 0.94 と、ほぼ 4 ウェイ SIMD 演算での計算がなされている。X5658 と E5-2650 との比較から、並列度 4 以上では、X5658 の 2 ウェイ演算数比がほぼ 99%であったのに対し E5-2650 における 4 ウェイ演算数比は最大 94%程度であり、2 ウェイの SIMD 演算も若干ながら実行されていることが分かった。

4.5 G 行列計算のプロセス並列計算の並列化効率

図 8, 図 9, 図 10 に G 行列計算に対するプロセス並列数と速度向上比についての 1 入力計算と 16 入力計算の比較結果を示す。1 入力と 16 入力計算に対し、Xeon X5650 の 6 並列実行時では 89.5%と 85.9%の並列化効率となり、

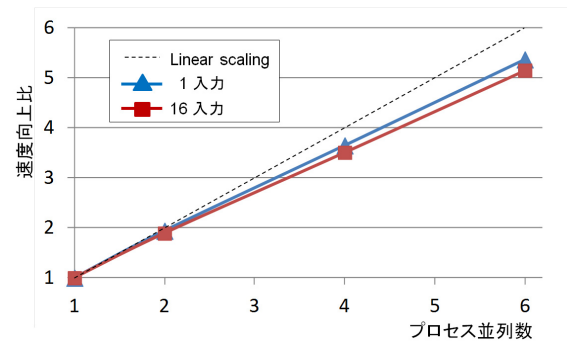


図 8 1 入力と 16 入力 G 行列計算のプロセス並列数に対する速度向上比 (Xeon X5650 を使用)

Fig. 8 Speedup ratios for 1 and 16 inputs G-matrix parallel calculation (Xeon X5650).

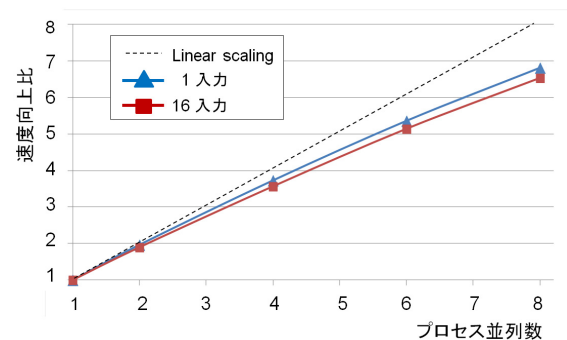


図 9 1 入力と 16 入力 G 行列計算のプロセス並列数に対する速度向上比 (Xeon E5-2650 を使用)

Fig. 9 Speedup ratios for 1 and 16 inputs G-matrix parallel calculation (Xeon E5-2650).

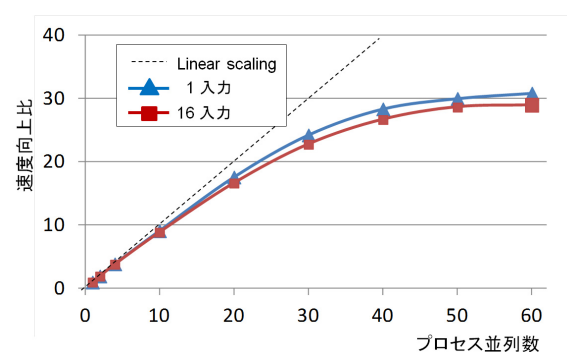


図 10 1 入力と 16 入力 G 行列計算のプロセス並列数に対する速度向上比 (Xeon Phi を使用)

Fig. 10 Speedup ratios for 1 and 16 inputs G-matrix parallel calculation (Xeon Phi).

Xeon E5-2650 の 8 並列実行時では 85.1% と 81.6%, Xeon Phi の 60 並列実行時においては 54.6% と 48.3% の結果を得た。その結果 6, 8, 60 プロセス並列において, 1 入力に対し 16 入力の並列化性能は 96.0%, 95.8%, 88.4% となった。これらの並列化性能の低下分は, 1 入力に対する 16 入力の各速度向上比の結果の 1.82 倍, 1.97 倍, 5.84 倍に比較して十分に少なく, 本方法は本測定のプロセス並列範囲ではスケーラブルであるといえる。

なお, 各結果における linear scaling に対する並列化効率の低下の原因は主に本並列計算において静的負荷分散を採用したためと考えられる。

5. まとめ

本論文では, 分子軌道法の Restricted closed-shell Hartree-Fock 計算における効率的な SIMD 演算の実行を目的とし, 複数の入力データによる並列計算を SIMD 演算に割り当てて計算する試みを行った。その際に, 同一入力を用いることで SIMD 演算が最も効率良く実行可能な条件で調査した。その結果, 倍精度浮動小数点数に対し 2 ウェイの SIMD 演算が可能な Xeon X5650, 2 ならびに 4 ウェイの SIMD 演算が可能な Xeon E5-2650, 8 ウェイの SIMD 演算が可能な Xeon Phi プロセッサ計算機環境において, 1.82 倍, 1.97 倍, 5.84 倍の速度向上結果がそれぞれ得られた。

通常の 1 入力計算ならびに 16 入力の複数入力計算において, Xeon Phi を利用した 60 プロセス並列計算では, それぞれ 54.6% と 48.3% の並列化効率が得られており, 16 入力計算は 1 入力に比較して 88.4% であった。この並列化性能の低下分は, 1 入力に対する 16 入力の各速度向上比の 5.84 倍に比較して十分に少なく, 本提案方法は本測定のプロセス並列範囲ではスケーラブルであるといえる。

また, 実際の計算では複数の異なる入力を実行するために, 今回得られた複数の同一入力計算における速度向上結果からは性能が低下する。しかしながら, H₂O 分子の 3 種類の振動モードを計算する際の平衡核配置近傍の 125 分子構造点を計算する例では, 逐次的にすべての分子構造を計算する場合に比較し複数入力計算の計算量の増加は 1.002 倍とわずかであるため, 本提案方法は有効に適用可能である。同様に, 本提案方法は分子構造がほぼ同じ複数の計算に適用する場合では, 十分に効率的な計算が可能であることが期待される。

謝辞 本研究の一部は, JST-CREST の研究領域「ポストパタスケール高性能計算に資するシステムソフトウェア技術の創出」における研究課題「省メモリ技術と動的最適化技術によるスケーラブル通信ライブラリの開発」の支援を受けている。

参考文献

- [1] Almlof, J., Faegri, K.J. and Korsell, K.: Principles for a Direct SCF Approach to LCAO-MO Ab Initio Calculations, *J. Comput. Phys.*, Vol.3, pp.383–399 (1982).
- [2] DARPA IPTO: ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems, DARPA IPTO (online), available from <http://www.cse.nd.edu/Reports/2008/TR-2008-13.pdf> (accessed 2014-03-31).
- [3] IESP: IESP:Documents, (online), available from <http://www.exascale.org/mediawiki/images/2/20/IESP-roadmap.pdf> (accessed 2014-03-31).
- [4] The T2K Open Supercomputer Alliance : SDHPC : 今後の HPCI 技術開発に関する報告書, (オンライン), 入手先 <http://www.open-supercomputer.org/workshop/sdhpc/> (参照 2014-03-31).
- [5] 稲富雄一, 眞木 淳, 本田宏明, 高見利也, 小林泰三, 青柳睦, 南 一生: 京コンピュータでの効率的な動作を目指した並列 FMO プログラム OpenFMO の高性能化, *J. Comp. Chem. Japan*, Vol.12, pp.145–155 (2013).
- [6] Yasuda, K.: Two-electron integral evaluation on the graphics processor unit, *J. Comput. Chem.*, Vol.29, No.3, pp.334–342 (2008).
- [7] Ufimtsev, I. and Martinez, T.: Quantum Chemistry on Graphical Processing Units. 1. Strategies for Two-Electron Integral Evaluation, *J. Chem. Theory Comput.*, Vol.4, No.2, pp.222–231 (2008).
- [8] Asadchev, A., Allada, V., Felder, J., Bode, B., Gordon, M. and Windus, T.: Uncontracted Rys Quadrature Implementation of up to G Functions on Graphical Processing Units, *J. Chem. Theo. Comp.*, Vol.6, No.3, pp.696–704 (2010).
- [9] 成瀬 彰: 分子軌道法プログラムの GPGPU 化, サイエンスフィックスシステム研究会アクセラレータ技術 WG 成果報告 (2012).
- [10] 梅田宏明, 塙 敏博, 庄司光男, 朴 泰祐: 分子軌道法の GPGPU 化に向けた行列加算手法の提案, 情報処理学会研究報告 HPC, Vol.138, No.19, pp.1–7 (2013).
- [11] 梅田宏明, 塙 敏博, 庄司光男, 朴 泰祐, 稲富雄一: フラグメント分子軌道法に現れる Fock 行列計算の GPGPU 化, 情報処理学会論文誌 コンピューティングシステム, Vol.6, No.4, pp.26–37 (2013).
- [12] Szabo, A. and Ostlund, N.: 新しい量子化学, 東京大学出版会 (1987).
- [13] Takashima, H., Yamada, S., Obara, S., Kitaura, K., Inaba, S., Miyakawa, N., Tanabe, K. and Nagashima, U.: A Novel Parallel Algorithm for Large-Scale Fock Matrix Construction with Small Locally Distributed Memory Architectures: RT Parallel Algorithm, *J. Comput. Chem.*, Vol.23, pp.1337–1346 (2002).
- [14] Nakamura, K., Honda, H., Inoue, K., Sato, H., Uehara, M., Komatsu, H., Umeda, H., Inadomi, Y., Araki, K., Sasaki, T., Obara, S., Nagashima, U. and Murakami, K.: A HighPerformance, LowPower Chip Multiprocessor for Large Scale Molecular Orbital Calculation, *Workshop on Unique Chip and Systems* (2005).
- [15] Obara, S. and Saika, A.: Efficient recursive computation of molecular integrals over Cartesian Gaussian functions, *J. Chem. Phys.*, Vol.84, pp.3963–3974 (1986).
- [16] Inadomi, Y.: OpenFMO, Kyushu University (online), available from <http://www.openfmo.org/OpenFMO/> (accessed 2014-03-31).
- [17] Gordon Research Group: The General Atomic and Molecular Electronic Structure System (GAMESS), Iowa State University (online), available from <http://www.msg.chem.iastate.edu/gamess/> (accessed 2014-03-31).

- [18] Pople, J. and Hehre, W.: Computation of Electron Repulsion Integrals Involving Contracted Gaussian Basis Functions, *J. Comp. Phys.*, Vol.27, pp.161-168 (1978).
- [19] Fletcher, G.: Recursion Formula for Electron Repulsion Integrals Over Hermite Polynomials, *Int. J. Quantum. Chem.*, Vol.102, pp.355-360 (2006).
- [20] PAPI: (online), available from (<http://icl.cs.utk.edu/papi/>) (accessed 2014-03-31).
- [21] 林 徹生, 本田宏明, 稲富雄一, 井上弘士, 村上和彰: Cell プロセッサへの分子軌道法プログラムの実装と評価, 情報処理学会研究報告 HPC, Vol.87, pp.103-108 (2006).
- [22] Makino, J., Hiraki, K. and Inaba, M.: GRAPE-DR: 2-Pflops massively-parallel computer with 512-core, 512-Gflops processor chips for scientific computing, *Supercomputing, 2007 (SC'07)*, pp.1-11 (2007).



眞木 淳

1967 年生. 1999 年北海道大学大学院理学研究科化学第二専攻博士後期課程修了. 九州大学情報基盤研究開発センター産学官連携研究員等を経て, 現在, 九州先端科学技術研究所研究員. 量子化学, 計算化学, ハイパフォーマンス・コンピューティングの研究に従事. 日本化学会会員.



本田 宏明 (正会員)

1970 年生. 2000 年北海道大学大学院理学研究科化学第二専攻博士後期課程修了. みずほ情報総研研究員, 九州大学システム情報科学研究院学術研究員, 九州先端科学技術研究所研究員を経て, 現在, 九州大学情報基盤研究開発センター学術研究員. 量子化学計算やアクセラレータ向け量子化学計算実装, 超伝導プロセッサ向け計算アルゴリズム開発, ハイパフォーマンス・コンピューティングに従事. 日本化学会, 分子科学会, コンピューター化学会, ACM 各会員.



稲富 雄一 (正会員)

1969 年生. 1992 年筑波大学第一学群自然科学類卒業. 1998 年同大学大学院博士課程化学研究科化学専攻修了. 筑波大学化学系技官, 産業技術総合研究所グリッドセンター研究員等を経て, 現在, 九州大学大学院システム情報科学研究院学術研究員. 量子化学計算プログラムの超並列化等に従事. 日本化学会, 分子科学会, 日本コンピュータ化学会, 情報計算化学生物学会各会員.