C-009

GPU を用いた McMurchie-Davidson アルゴリズムの高速化
A GPU Implementation of McMurchie-Davidson Algorithm

藤井 晴斗[1)] 伊藤 靖朗[1)] 横川 宜矢[1)] 鈴木 寛太[1)] 辻 聡樹[1)2)] 中野 浩嗣[1)] 笠置 明彦[2)]
Haruto Fujii Yasuaki Ito Nobuya Yokogawa Kanta Suzuki Satoki Tsuji Koji Nakano Akihiko Kasagi

1 はじめに

量子化学計算とは，分子のエネルギー等の特性を実際に実験を行うことなく計算のみで導き出すことを目的とした，計算化学の一分野である．その中でも，二電子反発積分 (ERI) と呼ばれる計算のコストが特に高く，分子を M 個の基底関数で表す場合は $O(M^4)$ の計算量が必要となる．また，ERI の計算手法に関しては，McMurchie-Davidson 法 [1] や Obara-Saika 法 [2] など，様々な特性を持つアルゴリズムが提案されている．本研究では，McMurchie-Davidson 法を用いた計算に着目し，高速化を行った．

2 McMurchie-Davidson 法

本項では，ERI の計算手法の一つである McMurchie-Davidson(MD) 法について説明する．まず，ガウス型軌道の線形結合として表される基底関数を以下のように定義する．

$$\chi_\mu(r) = \sum_{e=1}^{K_\mu} d_{\mu_e} G_{\mu_e}(r) \tag{1}$$

ここで，K_μ と d_{μ_e} はそれぞれ線形結合の項数と係数であり，ガウス型軌道は以下のように定義される．

$$G_{\mu_e}(r) = c_e (x-A_x)^{\mu_x} (y-A_y)^{\mu_y} (x-A_z)^{\mu_z} \exp\left(-\alpha_e |r-A|^2\right)$$

ここで，$A = (A_x, A_y, A_z)$ はガウス軌道の中心，c_e は正規化係数，指数部 μ_x, μ_y, μ_z は軌道の形状を決める非負整数，α_e は軌道の大きさを決定する指数である．ERI は，4 つの基底関数 $\chi_\mu, \chi_\nu, \chi_\lambda, \chi_\sigma$ に対して以下のように定義される計算である．

$$(\mu\nu|\lambda\sigma) = \sum_{e=1}^{K_\mu}\sum_{f=1}^{K_\nu}\sum_{g=1}^{K_\lambda}\sum_{h=1}^{K_\sigma} d_{\mu_e} d_{\nu_f} d_{\lambda_g} d_{\sigma_h} \left[\mu_e \nu_f | \lambda_g \sigma_h\right] \tag{2}$$

MD 法では以下の式 (3)–(5) を用いて ERI を評価する．

$$[ab|cd] = \frac{2\pi^{5/2}}{pp'\sqrt{p+p'}} \times \sum_{t=0}^{a_x+b_x}\sum_{u=0}^{a_y+b_y}\sum_{v=0}^{a_z+b_z}\sum_{t'=0}^{c_x+d_x}\sum_{u'=0}^{c_y+d_y}\sum_{v'=0}^{c_z+d_z} E_t^{a_x,b_x} E_u^{a_y,b_y} E_v^{a_z,b_z} E_{t'}^{c_x,d_x} E_{u'}^{c_y,d_y} E_{v'}^{c_z,d_z} R^0_{t+t',u+u',v+v'} \tag{3}$$

ここで，$p = \alpha+\beta, p' = \gamma+\delta$ である．$E_t^{a_x,b_x}$ の値は以下の漸化式で計算可能である．

$$\begin{cases} E_t^{i,j} = \dfrac{1}{2p} E_{t-1}^{i-1,j} - (A_x - B_x)\dfrac{q}{\alpha} E_t^{i-1,j} + (t+1) E_{t+1}^{i-1,j} \\ E_t^{i,j} = \dfrac{1}{2p} E_{t-1}^{i,j-1} + (A_x - B_x)\dfrac{q}{\beta} E_t^{i,j-1} + (t+1) E_{t+1}^{i,j-1} \\ E_t^{0,0} = \exp\left(-q\,(A_x - B_x)^2\right) \\ E_t^{i,j} = 0 \text{ if } t < 0 \text{ or } i + j < t \end{cases} \tag{4}$$

ここで，$q = \frac{\alpha\beta}{\alpha+\beta}$ であり，$E_u^{a_y,b_y} E_v^{a_z,b_z}$ の計算にはそれぞれ y, z 座標の値を使い，$E_{t'}^{c_x,d_x} E_{u'}^{c_y,d_y} E_{v'}^{c_z,d_z}$ の計算には G_c, G_d の 2 つのガウス型軌道を用いる．次に，$R^0_{t+t',u+u',v+v'}$ の値は以下の漸化式で計算可能である．

$$\begin{cases} R^n_{t,u,v} = (t-1) R^{n+1}_{t-2,u,v} + (P_x - P'_x) R^{n+1}_{t-1,u,v} \\ R^n_{t,u,v} = (u-1) R^{n+1}_{t,u-2,v} + (P_y - P'_y) R^{n+1}_{t,u-1,v} \\ R^n_{t,u,v} = (v-1) R^{n+1}_{t,u,v-2} + (P_z - P'_z) R^{n+1}_{t,u,v-1} \\ R^n_{0,0,0} = (-2\omega)^n F_n(\omega |P - P'|^2) \\ R^n_{t,u,v} = 0 \text{ if } t < 0 \text{ or } u < 0 \text{ or } v < 0 \end{cases} \tag{5}$$

ここで，$P = (P_x, P_y, P_z) = \frac{\alpha A + \beta B}{\alpha+\beta}, P' = (P'_x, P'_y, P'_z) = \frac{\gamma C + \delta D}{\gamma+\delta}, \omega = \frac{pp'}{p+p'}$ であり，$R^n_{0,0,0}$ の値は Boys 関数

$$F_n(x) = \int_0^1 t^{2n} e^{-xt^2} dt$$

で計算される．式 (5) の計算の中では, Boys 関数の計算コストが支配的である. 提案手法では, Boys 関数の評価に先行研究 [3] の手法を利用した.

3 提案アルゴリズム

本項では，MD 法を用いた ERI 計算に関する並列計算アルゴリズムを提案する．まず, $s_x = a_x + b_x + c_x + d_x, s_y = a_y + b_y + c_y + d_y, s_z = a_z + b_z + c_z + d_z, K = s_x + s_y + s_z$ を定義する．

ここから，式 (3) の評価に必要な漸化式 R の値を全て事前計算する手法を説明する．式 (3) の評価には，$0 \le t \le s_x, 0 \le u \le s_y, 0 \le v \le s_z$ を満たす $R^0_{t,u,v}$ の組が必要である．それらの各要素の値を得るために式 (5) を再帰的に計算すると，同じ項が多く現れ，冗長な計算が行われる. そこで，本手法では冗長な計算を回避することで無駄な計算を省略する. 具体的には，再帰式 (5) に登場する各要素を，互いに依存関係がなく同時に計算可能な $K+1$ 個の計算単位 (以下, *batch* と呼ぶ) 毎に分割し，それらの値を順に評価することで再帰式 R を効率よく評価する．各 batch $k (0 \le k \le K)$ には, それぞれ

$k = t + u + v, 0 \le n \le K - k$ を満たす全ての $R^n_{t,u,v}$ の組

が含まれるとする. batch k に含まれる R の値の個数は $\frac{(k+1)(k+2)(K-k+1)}{2}$ 個, R の値の総数は $\frac{(K+1)(K+2)(K+3)(K+4)}{24}$ 個となる．

ここで，式 (5) から，batch k に含まれるすべての値は batch $k-1, k-2$ のみに依存するため，batch k の各要素はそれぞれ独立に計算が可能である．図 1 に，$K = 3$ についての計算の流れを図示する．batch 0 の値は式 (5) の値を用いて計算され，batch k の値は batch $k-1, k-2$ の値を用いて計算可能であることが示されている．

ここからは，提案アルゴリズムを疑似コードとして説明する．

1） 広島大学大学院 先進理工系科学研究科
2） 富士通株式会社 コンピューティング研究所

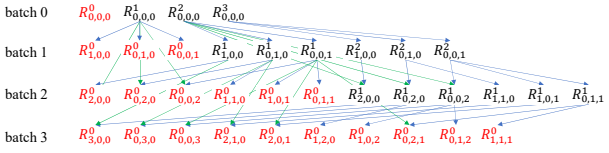

図 1　$K = 3$ の場合の例，式 (3) に必要な値を赤で強調

4 つのガウス型軌道についての ERI を計算する手法を Algorithm 1 に示す．このアルゴリズムの 1–3 行目では，提案した事前計算を行う．その後，得られた R の値を用いて，5–8 行目では式 (3) に相当する計算を行う．$E_t^{a_x,b_x}$ のように，式 (3) 内には E の計算も存在しているが，これは R と比較して計算量が小さいため，式 (4) の定義通りに再帰計算を行う．

Algorithm 1 ガウス型軌道についての並列 ERI 計算

Input: Four Gauss-type orbitals G_a, G_b, G_c, G_d
Output: ERI $[ab|cd]$
1: **for** $k = 0$ to K **do**
2: Compute R values in batch k **in parallel**
3: **end for**
4: $s \leftarrow 0$ // s is a shared variable, and parallel addition is performed on it
5: **for all** t, u, v, t', u', v' **do in parallel**
6: $s \leftarrow s + E_t^{a_x b_x} E_u^{a_y b_y} E_v^{a_z b_z} E_{t'}^{c_x d_x} E_{u'}^{c_y d_y} E_{v'}^{c_z d_z}$
7: $(-1)^{t'+u'+v'} R^0_{t+t',u+u',v+v'}$
8: **end for**
9: **return** $\frac{2\pi^{5/2}}{pp'\sqrt{p+p'}} \cdot s$

4　提案 GPU 実装

本項では，3 章で提案したアルゴリズムを，NVIDIA 社が提供する CUDA[4] を用いて GPU アーキテクチャ上に実装する手法を提案する．本実装では，1 つの CUDA Block に 1 組の基底関数に関する ERI 計算 (式 (2) に相当) を割り当てる．Algorithm 1 については，Block 内の各 Thread を用いた並列計算を行う．本実装では，3 つの Shared Memory(Block 内の各 Thread が読み書き可能である高速なメモリ) の配列を繰り返し使用し，batch 0 から順に計算を行うことで R の各要素を効率よく評価することが可能となっている．3 つの配列をそれぞれ sm_0, sm_1, sm_2 とすると，batch k の各要素を $sm_{(k \bmod 3)}$ に書き込む．これにより，各 batch の計算に必要な値を保持しつつ，不要となった値を上書きすることでメモリ量を削減することが可能となる．また，各 batch 計算の過程では，ERI の評価に必要な $R^0_{t,u,v}$ の各要素を保管するために別の Shared Memory に値を書き込む．

5　性能評価

本項では，提案した GPU 実装の性能評価を行う．本研究では，GPU として NVIDIA A100 Tensor Core GPU を，CPU として AMD EPYC 7702 CPU をそれぞれ使用し，GPU/CPU のそれぞれに batch ベースの実装と再帰関数ベースの実装を行った．ここで，batch ベースの実装は式 (3) の漸化式 R の計算を提案アルゴリズムを用いて計算するものであり，対して再帰関数ベースの実装は同様の計算を式 (5) を用いて計算するものである．

事前実験として，2 種類の実装の実行時間を後述の対象分子について計測したところ，GPU/CPU 共にすべてのケースで batch ベースの実装の方が高速であるという結果が得られ，batch ベースの実装が R の冗長な計算を削減できていることが示された．よって，本項では batch ベースの GPU/CPU 実装について性能評価を行う．

表 1 に，単原子分子である水素原子に対して correlation consistent 基底関数セット [5] を適用した場合の ERI 計算時間を示す．結果より，ERI 計算総数の少ない cc-PVDZ を除いたすべての基底関数セットについて CPU に対して高速化を達成しており，最大で 101 倍高速となった．また，ERI 計算の総数や K の最大値が増加するにつれ全体的な実行時間は増加している．

表 1　提案手法を用いた単原子分子の ERI 計算時間 [sec]

	monatonic Hydrogen (H)				
基底関数	cc-pVDZ	cc-pVTZ	cc-pVQZ	cc-pV5Z	cc-pV6Z
M	5	15	35	70	126
$\max(K)$	4	8	12	16	20
CPU	0.001	0.135	9.680	766.249	48,479.200
GPU	0.002	0.008	0.183	8.713	476.624
高速化率	0.60	16.67	52.90	87.94	101.71

表 2 に，様々な多原子分子での ERI 計算時間を示す．多原子分子としてベンゼン，ナフタレン，酸化銅を使用し，基底関数として STO-3G と 6-31G**をそれぞれ適用した．結果として，CPU と比較し最大で 24 倍高速となった．ベンゼン及びナフタレンは，分子数が異なるが，原子配置が似たような構造を持っており，K の最大値は同じとなっている．よって，実行時間は異なるものの，高速化率は近い値となっている．また，酸化銅はベンゼンやナフタレンと比較し，K の値が大きい基底関数の割合が高い．その上で，K の最大値が同じであっても高速化率が高くなっている為，batch ベースの実装は K の値が大きいほど高い効果を発揮する実装だといえる．

表 2　提案手法を用いた多原子分子の ERI 計算時間 [sec]

	Benzene (C_6H_6)		Naphthalene ($C_{10}H_8$)		Copper oxide (CuO)	
基底関数	STO-3G	6-31G**	STO-3G	6-31G**	STO-3G	6-31G**
M	36	120	58	190	24	54
$\max(K)$	4	8	4	8	8	12
CPU	17.198	584.714	113.575	3972.26	14.847	274.386
GPU	2.385	44.736	15.722	290.210	0.841	11.141
高速化率	7.21	13.07	7.22	13.69	17.65	24.63

6　結論

本論文では，ERI 計算に用いられる MD アルゴリズムの為の効率的な GPU 実装を提案した．提案手法では特に GPU に適した並列処理単位の batch を導入し，限られたサイズの Shared Memory を用いて ERI 計算を行うことが可能となった．NVIDIA A100 Tensor Core GPU 上での実装は，AMD EPYC 7702 CPU 上での実装と比較し，単原子分子では最大 101 倍，多原子分子では最大 24 倍の高速化を達成した．

参考文献

[1] McMurchie, L. E., Davidson, E. R. (1978). One- and two-electron integrals over Cartesian Gaussian functions. Journal of Computational Physics, 26(2), 218-231.
[2] Obara, S., Saika, A. (1986). Efficient recursive computation of molecular integrals over Cartesian Gaussian functions. The Journal of Chemical Physics, 84(7), 3963-3974.
[3] Tsuji, S., Ito, Y., Nakano, K., Kasagi, A. (2023). Efficient GPU-Accelerated Bulk Evaluation of the Boys Function for Quantum Chemistry. In 2023 Eleventh International Symposium on Computing and Networking (CANDAR) (pp. 49-58).
[4] NVIDIA Corporation. (2024). CUDA Programming Guide
[5] Dunning, T. H. Jr. (1989). Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen. The Journal of Chemical Physics, 90(2), 1007-1023.