Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186-5140
Volume 13, Number 1, pages 54–56, January 2024

A GPU Implementation of the Two-Electron Repulsion Integral Evaluation

Haruto Fujii*, Satoki Tsuji*†, Yasuaki Ito*, and Koji Nakano*
*Graduate School of Advanced Science and Technology, Hiroshima University
Kagamiyama 1-4-1, Higashi-Hiroshima, 739-8527, JAPAN
†Computing Laboratory, Fujitsu Limited
Kamikodanaka 4-1-1, Nakahara-ku, Kawasaki, 211-8588, JAPAN

***Abstract*—Computational quantum chemistry aims to derive properties of a molecule, such as its energy, without the need for actual experimental work. This is done by taking the types and coordinates of atoms within the molecule as input. In this study, we accelerate the computation of the two-electron repulsion integrals that mainly dominate the computing time. Specifically, we focus on the McMurchie-Davidson algorithm, where the two-electron repulsion integrals are obtained through recurrence computation. The idea of the proposed GPU implementation is to expand the recurrence relations and efficiently compute them using multiple threads of the GPU. Our proposed method resulted in an acceleration of up to 3.5 times compared to GPU implementations using recursive functions, and up to 17 times compared to CPU implementations similar to those using the GPU.**

***Index Terms*—molecular integrals, two-electron repulsion integrals, quantum chemistry, GPU, CUDA**

I. はじめに

量子化学計算とは，実際に実験や計測などを行うことなく分子系に含まれる原子の種類と座標に対して計算のみで分子の全エネルギーなどを求める手法である．その計算時間の大半を占める代表的なものとして **2 電子反発積分 (ERI, two-electron repulsion integrals)** がある．具体的な計算量は，N 個の基底関数を用いて系を表現する場合，$O(N^4)$ となる．ERI の計算コストを削減できれば量子化学計算全体の時間を大きく削減できるため，様々な先行研究が行われている．Johnson らの研究 [1] では，シュワルツの不等式 [2] を用いたスクリーニングや対称性の利用により $O(N^2)$ に近づき，さらに疎行列の性質を用いることで $O(N)$ に近づくことが報告されている．また，ERI の計算手法に関して，McMurchie-Davidson 法 [3] や Obara-Saika 法 [4] など，特性の異なる様々な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，高速化を行う．

II. MCMURCHIE-DAVIDSON 法

本項では，ERI の計算手法の 1 つである McMurchie-Davidson 法について説明する．まず，原始 GTO(p,q,r,s) についての ERI$\langle pq|rs\rangle$ は，以下のように表される．

$$\langle pq|rs\rangle = \iint \phi_p(r_1)\,\phi_q(r_1)\,\frac{1}{r_1-r_2}\phi_r(r_2)\,\phi_s(r_2)\,dr_1dr_2 \tag{1}$$

$$\phi_k(r) = N_k x_k^n y_k^l z_k^m \exp(-\alpha_k |r-R_k|) \tag{2}$$

(2) 式について，N_k はガウス軌道の規格化定数，R_k は軌道の中心 (=原子核の座標)，(x_k,y_k,z_k) はベクトル $r-R_k$ の各要素，(n,l,m) はガウス軌道の x,y,z 方向の方位量子数，α_k はガウス軌道の指数部である．

4 つのガウス軌道 $(\phi_A,\phi_B,\phi_C,\phi_D)$ についての McMurchie-Davidson 法では，まずは 2 種類のガウス軌道 ϕ_A,ϕ_B を合成し，新たなガウス軌道 Ω_{AB} とする．

$$\Omega_{AB} = \phi_A\phi_B = x_A^n x_B^{\bar{n}} y_A^l y_B^{\bar{l}} z_A^n z_B^{\bar{m}} \exp\left(-\left(\alpha_A r_A^2 + \alpha_B r_B^2\right)\right) \tag{3}$$

$$= x_A^n x_B^{\bar{n}} y_A^l y_B^{\bar{l}} z_A^n z_B^{\bar{m}} E_{AB} \exp\left(-\left(\alpha_P r_P^2\right)\right) \tag{4}$$

ここで，P,α_P,r_P,E_{AB} はそれぞれ

$$P = \frac{\alpha_A A + \alpha_B B}{\alpha_A+\alpha_B} \tag{5}$$

$$\alpha_P = \alpha_A + \alpha_B \tag{6}$$

$$r_P = r - P \tag{7}$$

$$E_{AB} = \exp\left(-\frac{\alpha_A\alpha_B}{\alpha_A+\alpha_B}|A-B|^2\right) \tag{8}$$

として表される．ϕ_C,ϕ_D についても同様に計算して

$$\Omega_{CD} = x_C^{n'} x_D^{\bar{n'}} y_C^{l'} y_D^{\bar{l'}} z_C^{m'} z_D^{\bar{m'}} E_{CD} \exp\left(-\alpha_Q r_Q^2\right) \tag{9}$$

を得る．次に，$x_A^n x_B^{\bar{n}}$ などの 2 軌道間のペアについて，エルミートガウス関数を用いた展開を行う．$x_P = x-P, x_A = x-A$ を利用すると

$$x_A = x_P + P - A \tag{10}$$

$$= x_P + \bar{P}A \tag{11}$$

として表すことが可能である．x_A, x_B を上記のように x_p で置換することで，

$$x_a^n x_B^{\bar{n}} = \sum_{N=0}^{n+\bar{n}} d_N^{n,\bar{n}} \Lambda_N(x_P;\alpha_P) \tag{12}$$

として表すことが可能となる．ここで，$d_N^{n,\bar{n}}$ は

$$d_N^{n+1,\bar{n}} = \frac{1}{2\alpha_P} d_{N-1}^{n,\bar{n}} + \bar{P}A d_N^{n,\bar{n}} + (N+1) d_{N+1}^{n,\bar{n}} \tag{13}$$

$$d_N^{n,\bar{n+1}} = \frac{1}{2\alpha_P} d_{N-1}^{n,\bar{n}} + \bar{P}B d_N^{n,\bar{n}} + (N+1) d_{N+1}^{n,\bar{n}} \tag{14}$$

$$d_0^{0,0} = 1 \tag{15}$$

として表され，$\Lambda_N(x_P;\alpha_P)$ はエルミート多項式の関係から

$$x_P\Lambda_N = N\Lambda_{N-1} + P\bar{A}_x\Lambda_N + \frac{1}{2}\frac{\Lambda_N}{\alpha_P} \tag{16}$$

という漸化式で表される．同様の操作を y_A, z_A に対しても行うことで，

$$x_a^n x_B^{\bar{n}} y_A^l y_B^{\bar{l}} z_A^m z_B^{\bar{m}} \tag{17}$$

$$= \sum_{NLM} d_N^{n,\bar{n}} d_L^{l,\bar{l}} d_M^{m,\bar{m}} \Lambda_N(x_P;\alpha_P)\Lambda_L(y_P;\alpha_P)\Lambda_M(y_P;\alpha_P) \tag{18}$$

として和の形式で表現可能となる．

$\Lambda_N(x_P;\alpha_P)\Lambda_L(y_P;\alpha_P)\Lambda_M(y_P;\alpha_P)$ を $[NLM|1]$ という記法で表すと，ERI 全体は

$$\langle AB|CD\rangle = \sum_{NLM}\sum_{N'L'M'} d_{NLM} d_{N'L'M'} \left[NLM|r_{12}^{1}|N'L'M'\right] \tag{19}$$

と表すことが可能となり，この計算は

$$\left[NLM|r_{12}^{-1}|N'L'M'\right] = \lambda(-1)^{N'+L'+M'} R^0_{N+N',L+L',M+M'} \tag{20}$$

$$\lambda = \frac{2\pi^{5/2}}{\alpha_P\alpha_Q\sqrt{\alpha_P+\alpha_Q}} \tag{21}$$

と表される．この漸化式 R は，

$$R^n_{t+1,u,v} = tR^{n+1}_{t-1,u,v} + X_{PQ}R^{n+1}_{t,u,v} \tag{22}$$

$$R^n_{t,u+1,v} = uR^{n+1}_{t,u-1,v} + Y_{PQ}R^{n+1}_{t,u,v} \tag{23}$$

$$R^n_{t,u,v+1} = vR^{n+1}_{t,u,v-1} + Z_{PQ}R^{n+1}_{t,u,v} \tag{24}$$

$$R^n_{0,0,0} = (-2\xi)^n F_n(\xi R^2_{PQ}) \tag{25}$$

$$\xi = \frac{\alpha_P\alpha_Q}{\alpha_P+\alpha_Q} \tag{26}$$

$$R^n_{t,u,v} = 0 \ (t<0 \text{ or } u<0 \text{ or } v<0) \tag{27}$$

として計算可能である．ここで，$F_n(\xi R^2_{PQ})$ は Boys 関数と呼ばれる関数で，

$$F_j(T) = \int_0^1 u^{2j}\exp\left(-Tu^2\right)du \tag{28}$$

として計算可能な関数である．

III. 提案 GPU 実装

本節では ERI 計算のための提案 GPU 実装を紹介する．実装には，CUDA [5](NVIDIA 社が提供する，NVIDIA 社 GPU のための統合開発環境) を使用する．まず，ERI の計算手法，及び漸化式の展開アルゴリズムを導入し，それらを用いた GPU 並列計算手法を説明する．Johnson らの研究 [1] では，McMurchie-Davidson 法の漸化式を展開したアセンブリコードをメタプログラミングにより生成することで，漸化式を効率よく計算している．彼らの手法では再帰関数が無くなるため計算効率が向上するが，アセンブリコードは逐次計算となっており，この点において更なる高速化を見込める．そこで，本研究では計算する必要のある漸化式を計算の依存関係に基づいて分類し，適切に並列計算を行うことで計算の高速化を行った．提案手法は 2 つの計算から構成される．まず，(22)〜(24) 式で計算される漸化式 $R^i_{a,b,c}$ について展開し，計算の依存性のない部分をまとめて並列計算する．$a+b+c=k$ となる $R^i_{a,b,c}$ すべてについての計算を Step k とすると，Step k の各要素は互いに依存関係を持たず，独立に計算が可能である．また，Step k の各要素の計算においては，(22)〜(24) 式から，Step $k-2$, Step $k-1$ の要素が必要である．この考えを元に，(19) 式から決定される計算範囲 ($K=\max(N+L+M+N'+L'+M')$) までの Step 0, Step 1, ..., Step K の合計 $K+1$ Steps を順に計算することで，1 つの ERI 計算に必要な漸化式 $R^i_{a,b,c}$ をすべて事前計算することが可能になる．

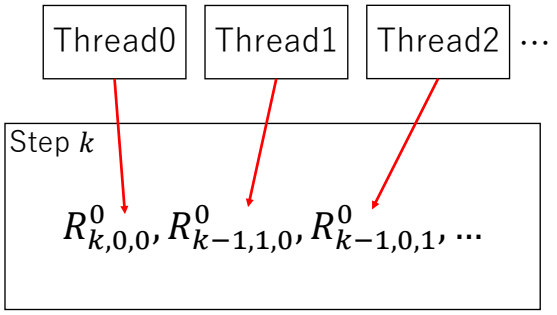

Fig. 1. Step 内計算における CUDA Thread の割り当て

次に，ERI についての並列計算を行う．(19) 式の通り，ERI の計算では 6 重ループについての計算が必要となるため，それらを先程の方法で計算した値を用いて計算する．その後，それぞれ求めた値を集約することで，実際の ERI 計算を行うことが可能となる．

ERI を GPU デバイス上で並列計算する手法については，様々な先行研究が存在する．Ivan らの研究 [6] においては，

1) 1B1CI: 1 つの CUDA Block が 1 組の縮約 GTO(CGTO) に関する ERI を計算
2) 1T1CI: 1 つの CUDA Thread が 1 組の縮約 GTO に関する ERI を計算
3) 1T1PI: 1 つの CUDA Thread が 1 組の原始 GTO(PGTO) に関する ERI を計算

の 3 種類の実装が提案されている．本研究では，新たに 1B1PI:1 つの CUDA Block が 1 組の PGTO に関する ERI を計算という手法を用いて実装を行った．この実装により，1 組の PGTO に関する ERI で共有可能な $R^n_{t,u,v}$ を Shared Memory 上で扱うことが可能となる．

先程の手法を CUDA で実装する場合，まずは Step 内で計算すべき $R^i_{a,b,c}$ をそれぞれ各 Thread が計算する．図 1 に，一般の Step k での割り当ての様子を示す．このようにして各 Step を順に計算することで，提案手法の前半の計算が完了する．また，提案手法では，求めた $R^i_{a,b,c}$ の値を CUDA の Shared Memory に書き込むことにより Block 内で共有し，各 Thread からの高速なアクセスを可能としている．

次に，実際の ERI 計算を行う．本実装では，(19) 式における (N,L,M,N',L',M') の組ごとに 1Thread を割り当てる．そして，各 Thread は，計算を行った値を atomicAdd 関数を用いて Block 内の Shared Memory に書きこむ．これにより，1 つの ERI 計算が完了する．

TABLE I
各種分子においての実行時間

		H_2O	C_2H_5OH	H_2SO_4	$C_{14}H_{10}$	C_6H_5-I
基底関数 CGTO の総数		7	21	31	80	64
実行時間 [ms]	CPU	6.47E+1	1.79E+3	1.26E+4	4.80E+5	7.19E+5
	GPU(再帰関数)	1.63E+1	1.65E+3	1.06E+4	1.24E+5	1.38E+5
	GPU(提案手法)	7.04E+0	6.30E+2	3.04E+3	7.20E+4	4.22E+4
高速化率	CPU/提案手法	9.19	2.84	4.14	6.67	17.03
	GPU/提案手法	3.26	1.90	3.49	1.72	3.27

IV. 性能評価

本節では，第 III 節で提案した GPU 実装の性能評価をおこなう．実行時間の計測では，Intel Xeon Gold 6338 CPU と NVIDIA A100 GPU を用いた．また，CPU 実行についてはシングルスレッドの実行とし，GPU 実行については Block を PGTO に関する ERI の組の分だけ起動し，各 Block 内の Thread 数は全て 256 で固定して実行した．この実行環境で次の 2 つの実験を行った．

実験 1: 実験 1 では，実在する様々な分子系に対して 3 手法 (CPU, GPU(再帰関数)，GPU(提案手法)) で計測を行った．この実験は，実際の計算において全体的にどの程度の高速化率が得られるかを検証することを目的としている．ここでは H_2O(水分子), C_2H_5OH(エタノール), H_2SO_4(硫酸), $C_{14}H_{10}$(アントラセン), C_6H_5-I(ヨードベンゼン) を対象の系，基底関数を STO-3G としたときの実行時間を計測した．

実験 2: 実験 2 では，$H_{50}, CH_{45}, ..., C_{10}$ のように，基底関数の総数が 50 個で，炭素と水素のみで構成された仮想的な系について，炭素数ごとにそれぞれ 2 手法 (GPU(再帰関数)，GPU(提案手法)) で計測を行った．これらの系では，炭素数が増加するにつれて基底関数中の p 軌道の割合も増加する．この実験は，再帰関数での実装と比較して提案手法が高角運動量を扱う ERI にどの程度有効であるかを検証することを目的としている．

表 I に，実験 1 に関する実験結果のデータを示す．結果として，GPU(提案手法) は，CPU 実装と比較して最大 17.03 倍，GPU(再帰関数) と比較して最大 3.49 倍高速化された．GPU を用いた 2 種類の実装に着目すると，実行したすべての実在系について提案手法の方が高速という結果となった．今回の実験においては，最低でも 1.72 倍，最高で 3.49 倍の高速化率が得られたため，提案手法で実行可能な系においては提案手法を用いる方が良い結果が期待できると考えられる．

実験 2 に関する実験結果のデータを Fig.2 に示す．実験 2 においては，炭素原子が 0,1 個の系においては GPU(再帰関数) の方が高速となり，2 個以上含まれる系においては GPU(提案手法) が高速となった．実行時間について，GPU(再帰関数) では炭素数が 0 から 10 に増加すると実行時間は 4.519 倍に増加したが，GPU(提案手法) では 1.206 倍の増加に抑えられている．これは，炭素の割合が増加するにつれて p 軌道を含む ERI の割合が増加するが，提案手法では高角運動量に関する ERI を再帰関数を用いずに高速に行えるため，GPU(再帰関数) よりも実行時間の増加率が低くなったと推察される．

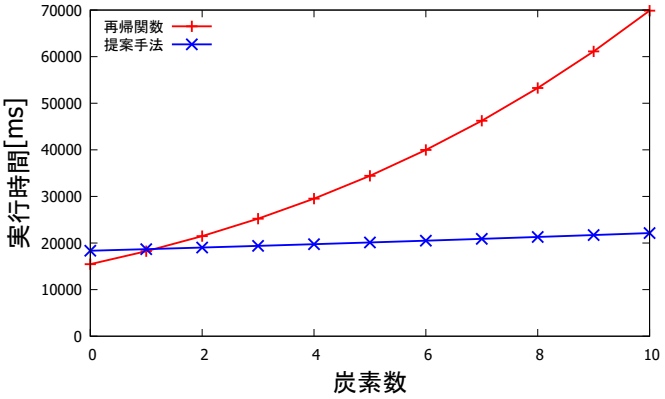

Fig. 2. 炭素と水素から構成される基底関数の総数が 50 となる系における，漸化式での実装と提案手法での実装の実行時間

V. まとめ

本研究では，量子化学計算における 2 電子反発積分 (ERI) 計算手法の 1 つである McMurchie-Davidson 法上で特に計算コストの大きい漸化式である $R^n_{t,u,v}$ を，再帰式を展開して適切に分類し，A100 GPU 上で実装して並列計算を行った．結果として，CPU での実装と比較して最大 17.03 倍，再帰関数を用いた実装と比較して最大 3.49 倍の高速化を達成した．そして，基底関数中の p 軌道の割合を変化させるスケーリングを行い，その結果から提案手法は再帰関数を用いた評価よりも高角運動量についての ERI を比較的高速に計算でき，角運動量の大きな系での実行に適している手法であることが示された．

References

[1] K. G. Johnson, S. Mirchandaney, E. Hoag, A. Heirich, A. Aiken, and T. J. Martínez, "Multinode multi-GPU two-electron integrals: Code generation using the Regent language," *Journal of Chemical Theory and Computation*, vol. 18, no. 11, pp. 6522–6536, 2022, pMID: 36200649.

[2] J. L. Whitten, "Coulombic potential energy integrals and approximations," *The Journal of Chemical Physics*, vol. 58, no. 10, pp. 4496–4501, 08 2003.

[3] L. E. McMurchie and E. R. Davidson, "One- and two-electron integrals over cartesian gaussian functions," *Journal of Computational Physics*, vol. 26, no. 2, pp. 218–231, 1978.

[4] S. Obara and A. Saika, "Efficient recursive computation of molecular integrals over Cartesian Gaussian functions," *The Journal of Chemical Physics*, vol. 84, no. 7, pp. 3963–3974, 04 1986.

[5] NVIDIA, "CUDA C++ Programming Guide v12.0," https://docs.nvidia.com/cuda/cuda-c-programming-guide/, 2022.

[6] I. S. Ufimtsev and T. J. Martínez, "Quantum chemistry on graphical processing units. 1. strategies for two-electron integral evaluation," *Journal of Chemical Theory and Computation*, vol. 4, no. 2, pp. 222–231, 2008, pMID: 26620654.