การเขียนโปรแกรมปริพันธ์หนึ่งและสองอิเล็กตรอนในโปรแกรมมอคคา

นายนพคุณ คำศรี

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาเคมี ภาควิชาเคมี
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2559
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

PROGRAMMING OF ONE- AND TWO-ELECTRON INTEGRALS IN MOCCA PROGRAM

Mr. Noppakoon Kharmsri

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Chemistry
Department of Chemistry
Faculty of Science
Chulalongkorn University
Academic Year 2016

Thesis Title	PROGRAMMING OF ONE- AND TWO-ELECTRON INTEGRALS IN MOCCA PROGRAM
By	Mr. Noppakoon Kharmsri
Field of Study	Chemistry
Thesis Advisor	Associate Professor Viwat Vchirawongkwin, Dr. rer. nat.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

นพคุณ คำศรี : การเขียนโปรแกรมปริพันธ์หนึ่งและสองอิเล็กตรอนในโปรแกรมมอคคา. (PROGRAMMING OF ONE- AND TWO-ELECTRON INTEGRALS IN MOCCA PROGRAM) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ.ดร.วิวัฒน์ วชิรวงศ์กวิน, 39 หน้า.

การนำทฤษฎีควอนตัมมาประยุกต์ใช้กับปัญหาทางเคมีได้รับการยอมรับอย่างกว้างขวางจากนักเคมีทั่วโลก ระเบียบวิธีการคำนวณทางเคมีควอนตัมหลายๆระเบียบวิธีต้องทำการประมวลผลปริพันธ์หนึ่งและสองอิเล็กตรอน การประมวลผลปริพันธ์หนึ่งและสองอิเล็กตรอนเป็นขั้นตอนที่ต้องใช้เวลาในการประมวลผลมากที่สุดและเป็นอุปสรรคต่อการประยุกต์ใช้ทฤษฎีควอนตัมกับระบบทางเคมีขนาดใหญ่ การคำนวณทางเคมีควอนตัมบนหน่วยประมวลผลกราฟิกส์เป็นทางเลือกหนึ่งในการเร่งความเร็วในการประมวลผล งานวิจัยชิ้นนี้ได้ทำการเขียนโปรแกรมการคำนวณปริพันธ์หนึ่งและสองอิเล็กตรอนในโปรแกรมมอคคาซึ่งคำนวณบนหน่วยประมวลผลกราฟิกส์ โปรแกรมมอคคาสามารถคำนวณปริพันธ์หนึ่งและสองอิเล็กตรอนได้ถูกต้องเมื่อเทียบกับโปรแกรมมาตรฐาน โดยโปรแกรมมอคคาสามารถคำนวณปริพันธ์หนึ่งและสองอิเล็กตรอนได้ถูกต้องมากกว่าทศนิยมตำแหน่งที่ 11 เมื่อเทียบกับโปรแกรมมาตรฐานและโปรแกรมมอคคาสามารถคำนวณได้เร็วกว่าโปรแกรมมาตรฐานประมาณ 150-7000 เท่า

ภาควิชา.............เคมี.................. ลายมือชื่อนิสิต...

สาขาวิชา.............เคมี.................. ลายมือชื่อ อ.ที่ปรึกษาหลัก...

ปีการศึกษา........2559.......................

5671992123 : MAJOR CHEMISTRY
KEYWORDS : GPU/CUDA

NOPPAKOON KHARMSRI : PROGRAMMING OF ONE- AND TWO-ELECTRON INTEGRALS IN MOCCA PROGRAM. ADVISOR : ASSOC. PROF. VIWAT VCHIRAWONGKWIN, Dr. rer. nat., 39 pp.

Application of quantum theory in chemical problem is acceptable by many chemists around the world. One- and two-electron integrals are the most time-consuming part limiting quantum theory application for large scale chemical system. Operation on graphic processing units (GPUs) for quantum chemistry calculation is one choice for accelerating the calculations. In this work, the calculations of one- and two-electron integrals were implemented for MOCCA program on GPU. According to the results, the one- and two-electron integrals calculated from the MOCCA program are realiable compared with GAMESS (US) program. The accuracy of one- and two-electron integrals calculated by MOCCA program is better than the level of 1×10^{-11} compared with GAMESS (US) program. MOCCA program can calculate faster than GAMESS (US) program about 150-7000 times.

Department : ....Chemistry....... Student's Signature..

Field of Study : ....Chemistry........ Advisor's Signature...

Academic Year : ...2016.................

ACKNOWLEDGEMENTS

This thesis was success by my endeavor and good support from my adviser, Assoc. Prof. Dr. Viwat Vchirawongkwin, and members in my research groups. I impress for their suggestions and friendship. The knowledge and experience from master's degree study will be the basis for my future.

I would like to specially appreciate and acknowledge Assoc. Prof. Dr. Vudhichai Parasuk, Dr. Nattapong Paiboonvorachat and Dr. Wikorn Punyain who acted as the thesis exam committee members and took the time to inspect and comment on this thesis.

Furthermore, I would like to appreciate Departmetn of Chemistry Faculty of Science Chulalongkorn University for the support of the infrastructure. I would like to thank National Research University Project of Thailand's Office of Higher Education Commission (WCU-58-017-FW). I also thanks science achievement scholarship of thailand (SAST) for giving me an oppotunity for study in master's degree.

Finally, I would like to acknowledge my family for their belief in me.

CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Literature Review

Quantum theory has been developed for interpreting the properties of the system containing very small particles. Phenomena of those systems can be described by solving their Schrödinger equations. Development of the solution of those equation have been ongoing [1], and it can be applied in chemistry such as interpreting chemical phenomena and predicting properties of molecules. To solve the Schrödinger equation of many cases, it requires one- and two-electron integrals. Like other scientific calculations, the problem for the applications of quantum theory in chemistry is the execution of a massive amount of data [2, 3], thus requiring an efficient computation.

Development of microprocessors based on a single CPU for sequential programs by increasing clock speed had been focused by many computer developers, resulting in a rapid performance increase and cost reduction. Energy-consumption and heat-dissipation are the main problems for increasing clock speed [4,5]. The development of microprocessors based on a multiple processing units or multi-core processors designed for sequential and parallel applications are the solution of those problems. Multi-core processors development maintains the execution speed of sequential program. The cost of microprocessor based on a multi-core CPU is so expensive, limiting the parallel software development to a small number of software developers. The graphics processing units (GPUs) initially have been developed for a demand of video game players for lighting the graphic operation of CPU [2]. The execution model of GPUs is the single instruction multiple threads (SIMT) [6]. To compare theoretically between multi-core CPU and many-core GPUs, the performance of G-PUs are greater although CPU has a higher clock rate speed, however, there are many of much smaller cores in GPUs. In addition to graphic applications, scientific and other calculations can be implemented on GPUs [7]. For the initial time of using GPUs for non-graphical applications, scientific calculation implementation was not popular for many developers due to the difficulty in programing. However, nVidia has developed a general-purpose GPUs (GPGPUs) and Compute Unified Device Architecture (CUDA) to facilitate developer for GPGPU programing [8]. CUDA is an application programing interface (API), based on C/C++ and Fortran programming language. For accleration of massive-data calculation, the computation of scientific works on GPUs is one of the good choices at the present time. Researchers implement many computational methods on GPU [4, 7–9].

Our research group also has developed a computational chemistry program, namely Molecular Orbital Calculation with CUDA (MOCCA) [10]. Hatree-Fock method has been implemented in MOCCA program, already. Unfortunately, MOCCA program did not have its own one- and two-electron integrals calculation codes. This work is the implementation of one- and two-electron integrals for MOCCA program. One- and two-electron integrals parts are the most time-consuming step in quantum chemistry calculations. For one-electron integrals, there are overlap integral, kinetic energy integral and potential integral. Electron repulsion integral is two-electron integrals. Many research works have implemented efficient two-electron integrals calculation code on GPU [2, 11–15]. Yasuda implemented the Rys quadrature method, one of the two-electron integral schemes, on nVidia GeForce 8800 GTX GPU for single precision [11]. It was applied to Taxol and Valinomycin molecules, simulated with 3-21G and 6-31G basis set for calculation testing. The preliminary timing results show the considerable speedup over the calculation on CPU for s and p functions. Subsequently, he and his co-worker further developed the algorithm based on McMurchie and Davidson (MD) method for efficient calculation of d function on GPU [12]. Ufimtsev and Marinez implemented the Rys quadrature method on nVidia GeForce 8800 GTX GPU for single precision [2]. The program was applied to a system of 64 hydrogen atoms simulated with STO-6G and 6-311G basis sets compared with GAMESS (US) program running on AMD Opteron 175 CPU. The timing results showed that implemented program on GPU can calculate faster than GAMESS (US) for more than 100 times in some cases. Asadchev and his co-workers developed two-electron integrals calculation based on Rys quadrature method for GAMESS (US) program on GPU [13]. The algorithm was designed for efficient calculation extended to the g function. The performance of the algorithm was evaluated on the nVidia GeForce GTX 275 and the nVidia Tesla T10 GPUs. The calculation on nVidia Tesla T10 GPU showed the speedups are around 25 times for double precision and 50 times for single presion compared with CPU. Miao and his co-worker implemented two-electron integrals calculation based on recurrence relation methods (VRR and HRR) for s, p and d functions on GPU [14]. It was applied to self-consistent field (SCF) calculations. From calculation results, speedups of calculation on GPUs reach 10 to 100 times compared to CPU and the accuracies are better than 1×10^{-7} for systems with more than 4,000 basis functions. Afterward, Miao and his-coworker developed the two-electron integrals calculation algorithm for g function. [15]. It was applied to SCF calculation and the calculation result show speedups of 10 to 18 times compared to CPU.

In this research, the calculation accuracy and speed of implemented one- and two-electron integrals calculation on MOCCA program were tested comparing with GAMESS (US) program [16]. For accuracy examination, one- and two-electron integrals calculation of MOCCA program performed on three molecule, namely, methane, carbonmonoxide and water with STO-3G, 3-21G and 6-31G basis sets. In addition, the groups of (sp,df), (sd,pf) and (sf,pd) two-electron integrals of ammonia molecule with aug-cc-pVDZ-RI basis set were calculated on MOCCA program compared with GAMESS (US) program for accuracy testing. One- and two-electron integrals were applied to Hartree-Fock calculation for methane, carbonmonoxide and water molecules with 6-31G basis set for accuracy examination. The speed examination of two-electron integrals calculation of MOCCA program performed on three theoretical models consisting of H_{30}, H_{40} and H_{50}. All theoretical models contain s-type atomic orbitals only.

CHAPTER II

THEORY AND METHOD

2.1 Schrödinger equation

Quantum chemistry is a branch of chemistry that it applies quantum mechanics to solve the problem. For quantum mechanics, the state of a system can be described by the wave function or state function. Quantum theory can be applied correctly for very small particles such as electrons. The fundamental equation of quantum mechanics is the time-independent Schrödinger equation.

$$\hat{H}\Psi = E\Psi. \tag{2.1}$$

From Eq. (2.1), taking the Harmiltonian operator, $\hat{H}$, on the wave function, Ψ, will get the total energy, E, of the system. For quantum chemistry, the system is atomic or molecular system. Total energy consists of a kinetic energy of nuclei, K_n, a kinetic energy of electron, K_e, a nuclear-nuclear repulsion, V_{nn}, an electron-electron repulsion, V_{ee}, and a nuclear-electron attraction, V_{ne},

$$E = K_n + K_e + V_{nn} + V_{ee} + V_{ne}.$$

A combination of all nuclei and electrons's wave functions is the total wave function of the system or molecule. It is hard to solve the equation that the nuclei have uncertain coordinates. Born-Oppenheimer approximation has been utilized to solve this problem by assuming the nuclei are immobile compared with electrons [17]. All nuclei's kinetic energy is vanished. From this approximation, the total wave function can be separated and written as:

$$\Psi_{\text{total}} = \Psi_{\text{electrons}}\Psi_{\text{nucleus}}.$$

Electronic wavefunction can be approximated by a linear combination of atomic orbitals or atomic wave function (LCAO). The electronic Hamiltonian, focused on electrons, of the molecule can be written as:

$$\hat{H}_{\text{electronic}} = -\frac{1}{2}\sum_{i=1}^{2n}\nabla_i^2 - \sum_{i=1}^{2n}\sum_{\text{all }\mu}\frac{Z_\mu}{r_{\mu i}} + \sum_{i=1}^{2n}\sum_{j>i}^{2n}\frac{1}{r_{ij}}.$$

where $2n$ is the number of electrons.

2.2 Gaussian function and one- and two-electron integrals

The set of mathematical functions used to approximate an atomic wave function are called basis function. In practice, there are complicated evaluation such as one- and two-electron integrals. It requires an appropiate function such as Gaussian-type functions (Eq. (2.2)) to facilitate such evaluation.

$$\chi(A, \alpha, l, m, n) = x_A^l y_A^m z_A^n \exp(-\alpha r_A^2), \tag{2.2}$$

where l, m and n are angular momenta of the system. This function is the most popular function for calculation in many commercial quantum chemistry programs, because the product of two Gaussians on two centers is a Gaussian with a third new center. Slater-type function is one of the most appropriate function for approximating Hydrogen atomic wave function. Slater-type functions are not popular for simulation in many works. It require excessive calculation times. On the other hand, Gaussian type-function can be used to simulate the wave function of Hydrogen atom. A gaussian function (STO-1G) is not sufficient to simulate Hydrogen atomic wave function as approximate as Slater-type function. Combination of several Gaussian-type functions (STO-3G) give a good approximation of Hydrogen atomic wave function nearby Slater-type function. To solve the molecular electronic wave function, all of the calculation methods for electronic energy evaluation require one- and two-electron integrals results. There are many research works related to the solution or calculation method of one- and two-electron integral such as recurrence relation, Rys quadrature method and others [18–22]. In this research, one- and two-electron integrals calculation methods developed by Taketa's group [18] were implemented into MOCCA program on GPU. For this method, the expression of one- and two-electron integrals are completely general, so the electron integrals can be calculated over the basis functions of arbitrarily high angular momentum.

2.3 Compute Unified Device Architecture (CUDA)

CUDA is a general-purpose parallel computing platform and programming model developed by NVIDIA to facilitate developers for programming on GPU. CUDA is designed for supporting cooperation of CPU and GPUs. For software developed by CUDA, there are host and device. The former phase operates sequentially on CPU and the latter operates parallel on GPU. For calculation of GPU, the CPU is the

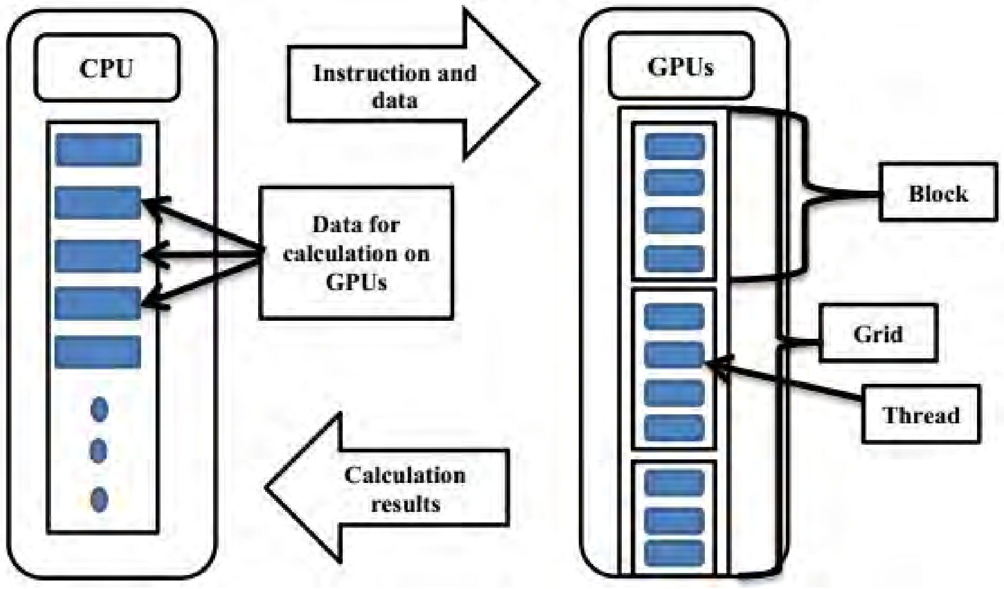
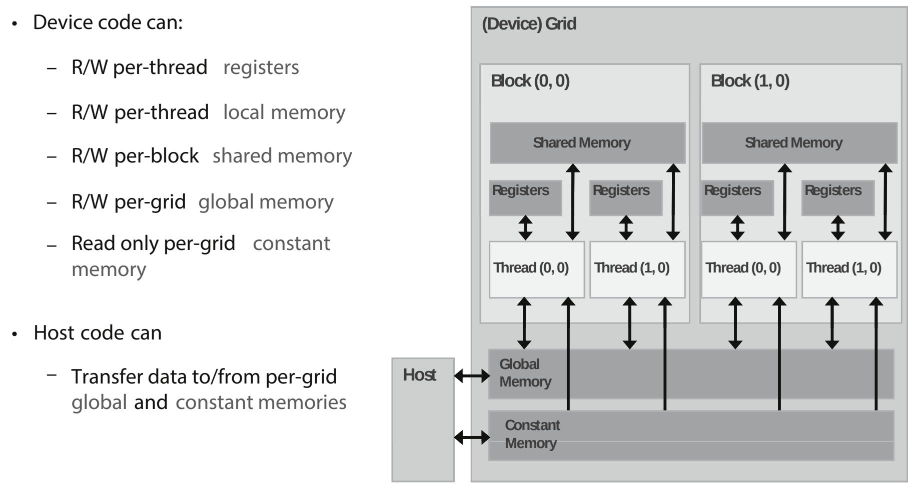


Figure **2.1**: CPU-GPU corporation.

main processor while the GPU is a coprocessor for parallel computation. Instructions and data are sent from the host phase to the device phase. The calculation model of CPU-GPU coporation is master-slave model (Fig. **2.1**) [23]. The slave node parallel operates the same instruction with different input data from the master node. After GPU execution, calculated data are transfered from GPU to CPU. For data management of GPU calculation, CUDA memory model is supported by application programming interface function. Programmers need to allocate memory and transfer data from CPU to GPU. There are several type of GPU memory (Fig. **2.2**) [24].



• Device code can:
– R/W per-thread registers
– R/W per-thread local memory
– R/W per-block shared memory
– R/W per-grid global memory
– Read only per-grid constant memory
• Host code can
– Transfer data to/from per-grid global and constant memories
(Device) Grid
Block (0, 0)
Block (1, 0)
Shared Memory
Registers
Thread (0, 0)
Thread (1, 0)
Host
Global Memory
Constant Memory

Figure **2.2**: The device memory model is managed by device and host code.

For global and constant memory, they are managed by CPU. Constant variables are stored in constant memory for using by kernel computation. Global and constant memory can be utilized by all thread for computing. There are shared memory in every streaming multiprocessor (SM). Data transfer rate of shared memory is higher than global memory. Shared memory have been used for storing data, which are often used for computation. The register memory are used by every therad privately.

The model for parallel execution provided by CUDA is blocks and threads in a grid (Fig.**2.3**) [24]. A grid has many blocks having a unique address (blockIdx) and each block consists of many thread having unique address (threadIdx). Address of block and thread can be used to identify each thread in block. There are multiple SM per GPU. Hundreds of threads can be executed on each SM. So thousands of threads can be executed concurrently on a GPU. The architechture of GPU is a Single Instruction Multiple Thread (SIMT) [6]. Groups of 32 threads called warps execute the same instruction at the same time [6].

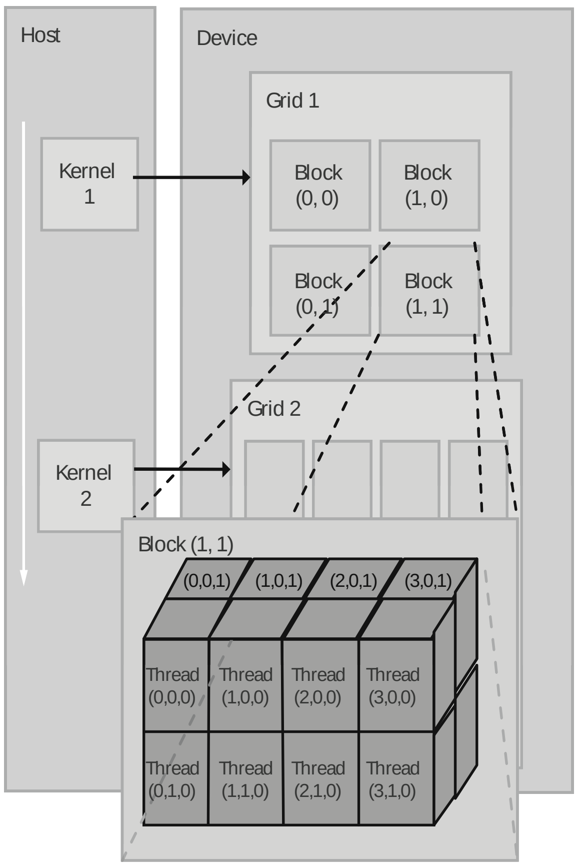


Figure **2.3**: Kernel assignment of host to a grid of blocks for same execution of all threads consisting in the grid.

CHAPTER III

CALCULATION DETAIL

3.1 Molecular Orbital Calculation with CUDA program (MOC-CA)

MOCCA program is an object-oriented program (OOP) implemented via CUDA C/C++ language [10]. This program has been designed with polymorphism and inheritance, allowing developers for convenient further development. There are t-wo main objects consisting of Molecule object and methodological objects. The Molecule object operating on CPU, link and controls other objects, while the method-ological objects operate on GPU. The main process is controlled by Molecule object operation consists of reading file, data rearrangement, selecting method and writing results. For the first step, an input file is read data for all contents to register in the object memories. For the data rearrangement step, the Molecule object transfers all data from the Read object to store in other objects. For selecting method step, the Molecule object interacts to a kind of calculation method's object to compute the results. All requirement of data are loaded to the calculation method object. After calculation is success, all results are copied back from the method object and the results are written to a file. For calculation of MOCCA program, atomic orbital-s are simulated by Gaussian-type function or linear combination of Gaussian-type functions.

3.2 One- and two-electron integrals calculation implementation

One-electron integrals including overlap, kinetic and potential integrals and two-electron integrals calculation method developed by Taketa group [18] were imple-mented into MOCCA program. Organizing threads of the implemented program, grid and block are organized as a 1-dimension array of blocks and a 1-dimension array of threads, respectively. Calculation accuracy and speed of implemented one- and two-electron integrals calculations were compared to the standard program (GAMESS (US) program). Implemented one- and two-electron integrals calculations were applied to Hartree-Fock calculation for electronic energies (a.u.) evaluation compared with GAMESS (US) program. The Hartree-Fock calculation part of MOCCA program was developed by my co-worker [10]. GAMESS (US) program is free and acceptable program. It is widely used by many researchers. GAMESS (US) program was implemented on the Intel Core i7-980x CPU while MOCCA program was implemented on the NVIDIA GeForce GTX 970 Ti GPUs.

Fig. **3.1** shows the flowchart of electron integral evaluation. Firstly, the input file is read to CPU, and data of input file is prepared for calculation on GPU. Then, the device memory is allocated on GPU for data of input file. The data are transfered from CPU to the allocated memory on GPU. Variables for calculation procedure are defined and initialized. After that, electron integrals are calculated. All equations and pseudo codes for the implementation of overlap integral, kinetic energy integral, potential integral and two-electron integral are shown below. The calculation results are stored on GPU memory and then calculation results have been transfered from GPU to CPU. Finally, output file is written on CPU.

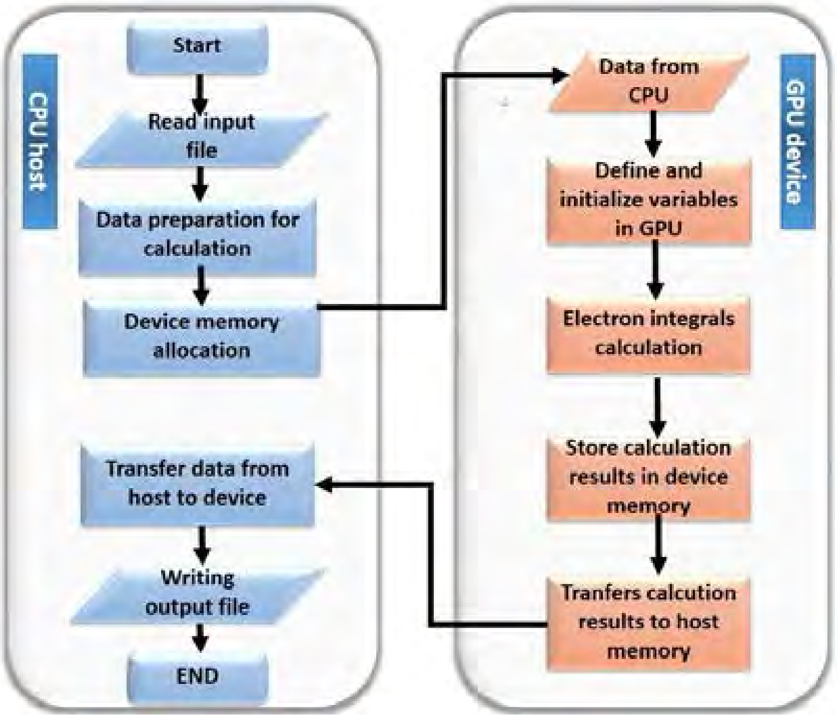

CPU host
Start
Read input file
Data preparation for calculation
Device memory allocation
Transfer data from host to device
Writing output file
END
GPU device
Data from CPU
Define and initialize variables in GPU
Electron integrals calculation
Store calculation results in device memory
Tranfers calcution results to host memory

Figure **3.1**: Flowchart of electron integral evaluation.

3.2.1 Overlap integral

Define $\chi(A, \alpha_1, l_1, m_1, n_1) = x_A^{l_1} y_A^{m_1} z_A^{n_1} \exp(-\alpha_1 r_A^2)$ as a function of atomic orbital centered at A and $\chi(B, \alpha_2, l_2, m_2, n_2) = x_B^{l_2} y_B^{m_2} z_B^{n_2} \exp(-\alpha_2 r_B^2)$ as a function of atomic orbital centered at B. The new center of product of two function is (P_x, P_y, P_z) ;

$$P_x = \frac{\alpha_1 A_x + \alpha_2 B_x}{\alpha_1 + \alpha_2} \ , \ P_y = \frac{\alpha_1 A_y + \alpha_2 B_y}{\alpha_1 + \alpha_2} \ , \ P_z = \frac{\alpha_1 A_z + \alpha_2 B_z}{\alpha_1 + \alpha_2}$$

Overlap integral is the overlap volume between two atomic orbitals of moleucle. Overlap integral equation [18] is written as:

$$S = \int \chi(A, \alpha_1, l_1, m_1, n_1) \chi(B, \alpha_2, l_2, m_2, n_2) \, dx \, dy \, dz$$

$$= \left(\frac{\Pi}{\gamma}\right)^{3/2} \exp\left(-\frac{\alpha_1\alpha_2\overline{AB}^2}{\gamma}\right) \sum_{i=0}^{[(l_1+l_2)/2]} f_{2i}(l_1, l_2, \overline{PA}_x, \overline{PB}_x)\frac{(2i-1)!!}{(2\gamma)^i}$$
$$\sum_{j=0}^{[(m_1+m_2)/2]} f_{2j}(m_1, m_2, \overline{PA}_y, \overline{PB}_y)\frac{(2j-1)!!}{(2\gamma)^j} \sum_{k=0}^{[(n_1+n_2)/2]} f_{2k}(n_1, n_2, \overline{PA}_z, \overline{PB}_z)\frac{(2k-1)!!}{(2\gamma)^k}$$

where $\overline{AB}$ is the distance between A and B atoms.

$$\begin{aligned}
\Pi &= 3.14159265358979323846 \\
\overline{PA}_x &= P_x - A_x \\
\gamma &= \alpha_1 + \alpha_2 \\
f_{2i}(l_1, l_2, \overline{PA}_x, \overline{PB}_x) &= \sum_{i=0,l_1}^{i+j=2k} \sum_{j=0,l_2} \overline{PA}_x^{\,l_1-1}\binom{l_1}{i}\overline{PB}_x^{\,l_2-1}\binom{l_2}{j} \\
(2i-1)!! &= 1 \times 3 \times 5 \times ..... \times (2i-1)
\end{aligned}$$

The pseudo code for overlap integral implementation is shown below.

The pseudo code for overlap integral implementation.

```
function Overlap_integral
START
assign pi constant = 3.14159265358979323846
assign threehalf constant = 1.5
assign integer i = 0
assign integer j = 0
assign overlap_result = 0

declare variable gamma
declare vector P, AB, PA, PB

declare orbital A, B;
assign coordinate to A atomic orbital
assign coordinate to B atomic orbital

calculate square of distance between A and B orbital (AB2)

for(assign i=0; i < number of premitive gaussian function of A atomic orbital; i++)
   for(assign j=0; j < number of premitive gaussian function of B atomic orbital; j++)
      gamma = premitive gaussian function of A[i].alpha + premitive gaussian function of B[j].alpha
      calculate new center of multiplication of premitive gaussian function of A[i] and B[i] and  assign it to P variable.
             PA = vector P - vector A
             PB = vector P - vector B

             overlap_result += premitive gaussian function of A[i].constant
                                  * premitive gaussian function of B[j].constant
                                  * premitive gaussian function of A[j].normalization constant
                                  * premitive gaussian function of B[j].normalization constant
                                  * pow( (pi/gamma) , (threehalf) )
                                  * exp(-(premitive gaussian function of A[i].alpha
                                  * premitive gaussian function of B[j].alpha*AB2)/gamma)
                                  * f_2i(A atomic orbital.lx, B atomic orbital.lx, PA.x, PB.x, gamma)
                                  * f_2j(A atomic orbital.ly, B atomic orbital.ly, PA.y, PB.y, gamma)
                                  * f_2k(A atomic orbital.lz, B atomic orbital.lz, PA.z, PB.z, gamma)
end for
end for
    return overlap_result
END
```

3.2.2 Kinetic energy integral

Defined $\chi(A,\alpha_1,l_1,m_1,n_1) = x_A^{l_1} y_A^{m_1} z_A^{n_1} \exp(-\alpha_1 r_A^2)$ is a function of atomic orbital centered at A and $\chi(B,\alpha_2,l_2,m_2,n_2) = x_B^{l_2} y_B^{m_2} z_B^{n_2} \exp(-\alpha_2 r_B^2)$ is a function of atomic orbital centered at B. Kinetic energy integral equation [18] is written as:

$$\begin{aligned}
KE = &\int \chi(A,\alpha_1,l_1,m_1,n_1)(\frac{-\nabla^2}{2})\chi(B,\alpha_2,l_2,m_2,n_2)\,dr \\
&= \alpha_2\{2(l_2+m_2+n_2)+3\}S(l_2,m_2,n_2) - 2\alpha_2^2\{S(l_2+2,m_2,n_2) \\
&+ S(l_2,m_2+2,n_2) + S(l_2,m_2,n_2+2)\} \\
&- \frac{1}{2}\{l_2(l_2-1)S(l_2-2,m_2,n_2) + m_2(m_2-1)S(l_2,m_2-2,n_2) \\
&+ n_2(n_2-1)S(l_2,m_2,n_2-2)\}
\end{aligned}$$

where

$$S(l_2,m_2,n_2) = \int \chi(A,\alpha_1,l_1,m_1,n_1)\chi(B,\alpha_2,l_2,m_2,n_2)\,dx\,dy\,dz$$

The pseudo code for kinetic energy integral implementation is shown below.

The pseudo code for kinetic energy integral implementation.

```
function Kinetic_integral
START
assign pi constant = 3.14159265358979323846
assign threehalf constant = 1.5
assign integer i = 0
assign integer j = 0
assign kinetic_result = 0
declare variable gamma
declare vector P, AB, PA, PB
declare orbital A, B
assign coordinate to A atomic orbital
assign coordinate to B atomic orbital

calculate square of distance between A and B orbital (AB2)
for(assign i=0; i < number of premitive gaussian function of A atomic orbital; i++)
   for(assign j=0; j < number of premitive gaussian function of B atomic orbital; j++)
       gamma = premitive gaussian function of A[i].alpha + premitive gaussian function of B[j].alpha
       calculate new center of multiplication of premitive gaussian function of A[i] and B[i] and assign it to P variable.
       PA = vector P - vector A
       PB = vector P - vector B

assign u constant = B atomic orbital.lx - 2
if(u<0) u = 0
assign v constant = B atomic orbital.ly - 2
end if
if(v<0) v = 0
assign w constant = B atomic orbital.lz - 2
end if
if(w<0) w = 0
end if
    kinetic_result += premitive gaussian function of A[i].constant
                      * premitive gaussian function of B[j].constant
                      * premitive gaussian function of A[j].normalization constant
                      * premitive gaussian function of B[j].normalization constant
                      * (( ( premitive gaussian function of B[j].alpha )
                      * ( (2 * ( B atomic orbital.lx + B atomic orbital.ly + B atomic orbital.lz  ) ) + 3 )
                      * Overlap_integral(l2,m2,n2)) - ( 2*pow(premitive gaussian function of B[j].alpha,2) )
                   * ( Overlap_integral(l2+2,m2,n2) + Overlap_integral(l2,m2+2,n2) + Overlap_integral(l2,m2,n2+2) ) -0.5
                    * ( (  (B atomic orbital.lx)*(B atomic orbital.lx - 1) * Overlap_integral(l2-2,m2,n2) + ((B atomic orbital.ly)
                   * (B atomic orbital.ly - 1)*Overlap_integral(l2,m2-2,n2) + ((B atomic orbital.lz)*(B atomic orbital.lz - 1)
                      * Overlap_integral(l2,m2,n2-2))))
end for
end for
    return kinetic_result
END
```

3.2.3 Potential integral

Defined $\chi(A, \alpha_1, l_1, m_1, n_1) = x_A^{l_1} y_A^{m_1} z_A^{n_1} \exp(-\alpha_1 r_A^2)$ is a function of atomic orbital centered at A and $\chi(B, \alpha_2, l_2, m_2, n_2) = x_B^{l_2} y_B^{m_2} z_B^{n_2} \exp(-\alpha_2 r_B^2)$ is a function of atomic orbital centered at B. The position of nucleus is (C_x, C_y, C_z). The new center of product of two function is (P_x, P_y, P_z) ;

$$P_x = \frac{\alpha_1 A_x + \alpha_2 B_x}{\alpha_1 + \alpha_2} \ , \ P_y = \frac{\alpha_1 A_y + \alpha_2 B_y}{\alpha_1 + \alpha_2} \ , \ P_z = \frac{\alpha_1 A_z + \alpha_2 B_z}{\alpha_1 + \alpha_2}$$

Potential integral or nuclear attraction integral is the potential energy of attraction between electrons in atomic orbital and nucleus of molecule. Potential integral equation [18] is written as:

$$\begin{aligned} NAI &= \int \chi(A, \alpha_1, l_1, m_1, n_1) \frac{1}{r_C} \chi(B, \alpha_2, l_2, m_2, n_2)\, dr \\ &= \frac{2\Pi}{\gamma} \exp\left(-\frac{\alpha_1 \alpha_2 \overline{AB}^2}{\gamma}\right) \sum_{i,r,u} A_{i,r,u}(l_1, l_2, A_x, B_x, C_x, \gamma) \\ &\sum_{j,s,v} A_{j,s,v}(m_1, m_2, A_y, B_y, C_y, \gamma) \sum_{k,t,w} A_{k,t,w}(n_1, n_2, A_z, B_z, C_z, \gamma) F_V\left(\frac{\overline{CP}^2}{4 \in}\right) \end{aligned} \tag{3.1}$$

where

$$V = i + j + k - 2(r + s + t) - (u + v + w)$$

$$A_{i,r,u}(l_1, l_2, A_x, B_x, C_x, \gamma) = (-1)^i f_i(l_1, l_2, \overline{PA}_x, \overline{PB}_x) \frac{(-1)^u i! \overline{PC}_x^{\,2} (0.25\gamma)^{r+u}}{r! u! (i - 2r - 2u)!}$$

$$\overline{PC}_x = P_x - C_x \ , \ \in = \frac{0.25}{\gamma} \ , \ \gamma = \alpha_1 + \alpha_2$$

$$\Pi = 3.14159265358979323846$$

$$F_V(t) = \int_0^1 u^{2V} \exp(-tu^2) du$$

In Eq. (3.1), the summation with respect ot indexes i, r and u extend from 0 to $l_1 + l_2$, $[i/2]$ and $[(i - 2r)/2]$, respectively. The range of (j, s, v) or (k, t, w) can be found in the same way.

The pseudo code for potential integral implementation is shown below.

The pseudo code for potential integral implementation.

```
function Potential_integral
START
assign pi constant = 3.14159265358979323846
assign integer rr = 0
assign integer vv = 0
assign potential_result = 0

declare variable gamma
declare vector P, AB, PA, PB, PC

declare orbital A, B
declare nucleus C
assign coordinate to A atomic orbital
assign coordinate to B atomic orbital
assign coordinate to C nucleus

calculate square of distance between A and B orbital (AB2)

for(rr=0; rr < number of premitive gaussian function of A atomic orbital; rr++)
   for(vv=0; vv < number of premitive gaussian function of B atomic orbital; vv++)
        gamma = premitive gaussian function of A[rr].alpha + premitive gaussian function of B[vv].alpha

        calculate new center of multiplication of premitive gaussian function of A[rr]
        and B[vv] and assign it to P variable.

        PA = vector P - vector A
        PB = vector P - vector B
        PC = vector P - vector C

        assign t = pow(distance between P and C,2) * gamma
        assign  F array
        allocate memory for F array
        calculate auxiliary function results and put them to F array
        assign mocca_type half_NAI = 0

         for(assign integer i = 0; i <= (A atomic orbital.lx + B atomic orbital.lx); i++)
             for(assign integer r = 0; r <= i/2; r++)
                 for(assign integer u = 0; u <= (i - 2*r)/2 ; u++)

         for(assign integer j = 0; j <= ( A atomic orbital.ly + B atomic orbital.ly); j++)
              for(assign integer s = 0; s <= j/2; s++)
                  for(assign integer v = 0; v <= (j - 2*s)/2; v++)

         for(assign integer k = 0; k <= ( A atomic orbital.lz + B atomic orbital.lz); k++)
              for(assign integer t = 0; t <= k/2; t++)
                  for(assign integer w = 0; w <= (k - 2*t)/2; w++)

                   assign unsigned integer cof = i + j + k - 2*(r + s + t) - (u + v + w)
             half_NAI = half_NAI + (A_function( i, r, u, A atomic orbital.lx, B atomic orbital.lx, PA.x, PB.x, PC.x, gamma))
                             * (A_function( j, s, v, A atomic orbital.ly, B atomic orbital.ly, PA.y, PB.y, PC.y, gamma))
                           * (A_function( k, t, w, A atomic orbital.lz, B atomic orbital.lz, PA.z, PB.z, PC.z, gamma))*F[cof]
```

```
end for   end for   end for
end for   end for   end for
end for   end for   end for
        potential_result +=  premitive gassian function of A[rr].constant
                             * premitive gaussian function of B[vv].constant
                             * premitive gaussian function of A[rr].normalization constant
                             * premitive gaussian function of B[vv].normalization constant
                             * ((2*pi)/gamma) * exp(-(premitive gaussian function of A[rr].alpha
                             * premitive gaussian function of B[vv].alpha*AB2)/gamma)
                             * half_NAI

     free memory of  F array

end for
end for
END
```

3.2.4 Two-electron integral

Defined $\chi(A,\alpha_1,l_1,m_1,n_1) = x_A^{l_1} y_A^{m_1} z_A^{n_1} \exp(-\alpha_1 r_A^2)$ is a function of atomic orbital centered at A and $\chi(B,\alpha_2,l_2,m_2,n_2) = x_B^{l_2} y_B^{m_2} z_B^{n_2} \exp(-\alpha_2 r_B^2)$ is a function of atomic orbital centered at B. The new center of product of two function is (P_x, P_y, P_z) :

$$P_x = \frac{\alpha_1 A_x + \alpha_2 B_x}{\alpha_1+\alpha_2} \ , \ P_y = \frac{\alpha_1 A_y + \alpha_2 B_y}{\alpha_1+\alpha_2} \ , \ P_z = \frac{\alpha_1 A_z + \alpha_2 B_z}{\alpha_1+\alpha_2}$$

Defined $\chi(C,\alpha_3,l_3,m_3,n_3) = x_C^{l_3} y_C^{m_3} z_C^{n_3} exp(-\alpha_3 r_C^2)$ is an function of atomic orbital centered at C and $\chi(D,\alpha_4,l_4,m_4,n_4) = x_D^{l_4} y_D^{m_4} z_D^{n_4} exp(-\alpha_4 r_D^2)$ is an function of atomic orbital centered at D. The new center of product of two function is (Q_x, Q_y, Q_z) :

$$Q_x = \frac{\alpha_3 C_x + \alpha_4 D_x}{\alpha_3+\alpha_4} \ , \ Q_y = \frac{\alpha_3 C_y + \alpha_4 D_y}{\alpha_3+\alpha_4} \ , \ Q_z = \frac{\alpha_3 C_z + \alpha_4 D_z}{\alpha_3+\alpha_4}$$

Electron repulsion integral is the potential energy of repulsion between electrons in atomic orbitals of molecule. Electron repulsion integral equation [18] is written as:

$$\begin{aligned} ERI = &\iint \chi(A,\alpha_1,l_1,m_1,n_1)\chi(B,\alpha_2,l_2,m_2,n_2)\frac{1}{r_{12}}\chi(C,\alpha_3,l_3,m_3,n_3)\chi(D,\alpha_4,l_4,m_4,n_4) \\ & dr_1\, dr_2 \\ & = \frac{2\Pi^2}{\gamma_1\gamma_2}(\frac{\Pi}{\gamma_1+\gamma_2})^{1/2} \exp\left(-\frac{\alpha_1\alpha_2\overline{AB}^2}{\gamma_1} - \frac{\alpha_3\alpha_4\overline{CD}^2}{\gamma_2}\right) \\ & \times \sum_{i_1,i_2,r_1,r_2,u} B_{i_1,i_2,r_1,r_2,u}(l_1,l_2,A_x,B_x,P_x,\gamma_1|l_3,l_4,C_x,D_x,Q_x,\gamma_2) \\ & \times \sum_{j_1,j_2,s_1,s_2,v} B_{j_1,j_2,s_1,s_2,v}(m_1,m_2,A_y,B_y,P_y,\gamma_1|m_3,m_4,C_y,D_y,Q_y,\gamma_2) \\ & \times \sum_{k_1,k_2,t_1,t_2,w} B_{k_1,k_2,t_1,t_2,w}(n_1,n_2,A_z,B_z,P_z,\gamma_1|n_3,n_4,C_z,D_z,Q_z,\gamma_2) F_V\left(\frac{\overline{PQ}^2}{4\delta}\right) \end{aligned} \tag{3.2}$$

where

$$
\begin{aligned}
V &= i_1 + i_2 + j_1 + j_2 + k_1 + k_2 - 2(r_1 + r_2 + s_1 + s_2 + t_1 + t_2) - (u + v + w) \\
B_{i_1,i_2,r_1,r_2,u}&(l_1, l_2, A_x, B_x, P_x, \gamma_1 | l_3, l_4, C_x, D_x, Q_x, \gamma_2) \\
&= (-1)^{i_2} f_{i_1}(l_1, l_2, \overline{PA}_x, \overline{PB}_x) f_{i_2}(l_3, l_4, \overline{QC}_x, \overline{QD}_x) \\
&\times \frac{i_1! i_2!}{(4\gamma_1)^i_1 (4\gamma_2)^i_2 \delta^{i_1+i_2}} \frac{(4\gamma_1)^{r_1} (4\gamma_2)^{r_2} \delta^{r_1+r_2}}{r_1! r_2! (i_1 - 2r_1)! (i_2 - 2r_2)!} \\
&[i_1 + i_2 - 2(r_1 - r_2)]! \frac{(-1)^u p_x^{i_1+i_2-2(r_1-r_2)-2u\delta^u}}{u! [i_1 + i_2 - 2(r_1 + r_2) - 2u]!} \\
\gamma_1 &= \alpha_1 + \alpha_2 \\
\gamma_2 &= \alpha_3 + \alpha_4 \\
p_x &= Q_x - P_x \\
\delta &= 0.25(\frac{1}{\gamma_1} + \frac{1}{\gamma_2}) \\
\Pi &= 3.14159265358979323846 \\
F_V(t) &= \int_0^1 u^{2V} \exp(-tu^2) du
\end{aligned}
$$

In Eq. (3.2), the summation with respect to the indices i_1, i_2 , r_1 , r_2 and u extend from 0 to l_1+l_2, l_3+l_4 , $[i_1/2]$, $[i_2/2]$ and $[(i_1+i_2)/2-r_1-r_2]$, respectively. The range of (j_1, j_2, s_1, s_2, v) or (k_1, k_2, t_1, t_2, w) can be found in the same way.

The pseudo code for two-electron integral implementation is shown below.

The pseudo code for two-electron integral implementation.

```
function Electron reulsion integral
START
assign two constant = 2
assign mocca_type mione constant = -1
assign  mocca_type pi constant = 3.14159265358979323846
assign  mocca_type twohalf constant = 0.5
assign  mocca_type threehalf constant = 1.5
declare integer variable rr = 0
declare integer vv = 0
declare integer ss = 0
declare integer tt = 0
declare mocca_type ERI_result = 0

declare mocca_type gamma_1, gamma_2
declare vecter P, Q, PA, PB, QC, QD, QP

declare orbital A, B, C, D
assign coordinate to A atomic orbital
assign coordinate to B atomic orbital
assign coordinate to C atomic orbital
assign coordinate to D atomic orbital

for(rr=0; rr < number of premitive gaussian function of A atomic orbital; rr++)
   for(vv=0; vv < number of premitive gaussian function of B atomic orbital; vv++)
        for(ss=0; ss< number of premitive gaussian function of C atomic orbital; ss++)
           for(tt=0; tt< number of premitive gaussian function of D atomic orbital; tt++)

     gamma_1 = premitive gaussian function of A[rr].alpha + premitive gaussian function of B[vv].alpha
     gamma_2 = premitive gaussian function of C[ss].alpha + premitive gaussian function of D[tt].alpha

          calculate new center of multiplication of premitive gaussian function of A[rr]
          and B[vv] and assign it to P variable.

          calculate new center of multiplication of premitive gaussian function of C[ss]
          and D[tt] and assign it to P variable.

            PA = vector P - vector A
            PB = vector P - vector B
            QC = vector Q - vector C
            QD = vector Q - vector D
            QP = vector Q - vector P

        assign  F array
        allocate memory for F array
        assign mocca_type delta = 4*gamma_1*gamma_2/(gamma_1 + gamma_2)
        assign mocca_type t = 0.25 * (P.distance(P, Q)*P.distance(P, Q)) * delta
        calculate auxiliary function results and put them to F array

        assign half_ERI = 0

for(assign unsigned integer i1 = 0; i1 <= (A atomic orbital.lx  + B atomic orbital.lx); i1++)
for(assign unsigned integer i2 = 0; i2 <= (C atomic orbital.lx  + D atomic orbital.lx); i2++)
         for(assign unsigned integer r1 = 0; r1 <= i1/2 ; r1++)
         for(assign unsigned integer r2 = 0; r2 <= i2/2 ; r2++)
                for(unsigned int u = 0; u <=( (i1 + i2)/2 - r1 - r2) ; u++)

for(assign unsigned int j1 = 0; j1 <= (A atomic orbital.ly + B atomic orbital.ly); j1++)
for(assign unsigned int j2 = 0; j2 <= (C atomic orbital.ly + D atomic orbital.ly); j2++)
        for(assign unsigned int s1 = 0; s1 <= j1/2 ; s1++)
        for(assign unsigned int s2 = 0; s2 <= j2/2 ; s2++)
                 for(unsigned int v = 0; v <= ( (j1 + j2)/2 - s1 - s2) ; v++)

for(unsigned int k1 = 0; k1 <= (A atomic orbital.lz + B atomic orbital.lz); k1++)
for(unsigned int k2 = 0; k2 <= (C atomic orbital.lz + D atomic orbital.lz); k2++)
        for(unsigned int t1 = 0; t1 <= k1/2 ; t1++)
        for(unsigned int t2 = 0; t2 <= k2/2 ; t2++)
                 for(unsigned int w = 0; w <= ( (k1 + k2)/2 - t1 - t2)  ; w++)

unsigned int cof = i1 + i2 + j1 + j2 + k1 + k2 - 2*(r1 + r2 + s1 + s2 + t1 + t2 )  - u -v -w

  half_ERI = half_ERI + (A.BFA_function(i1, i2, r1, r2, u, A atomic orbital.lx, B atomic
             orbital.lx, C atomic orbital.lx, D atomic orbital.lx, PA.x, PB.x, QC.x, QD.x
             ,QP.x, gamma_1, gamma_2))*(A.BFA_function(j1, j2, s1, s2, v, A atomic
             orbital.ly, B atomic orbital.ly, C atomic orbital.ly,D atomic orbital.ly
             , PA.y, PB.y, QC.y, QD.y, QP.y, gamma_1, gamma_2))*(A.BFA_function(k1, k2, t1
             , t2, w, A atomic orbital.lz, B atomic orbital.lz, C atomic orbital.lz,
             D atomic orbital.lz, PA.z, PB.z, QC.z, QD.z, QP.z, gamma_1, gamma_2)) *F[cof]
 end for  end for  end for
 end for  end for  end for
 end for  end for  end for

    ERI_result += premitive gaussian function of A[rr].constant
                   * premitive gaussian function of B[vv].constant
                   * premitive gaussian function of C[ss].constant
                   * premitive gaussian function of D[tt].constant
                   * premitive gaussian function of A[rr].normalization constant
                   * premitive gaussian function of B[vv].normalization constant
                   * premitive gaussian function of C[ss].normalization constant
                   * premitive gaussian function of D[tt].normalization constant
                   * pow(  pi,two)*2*pow( (pi/( gamma_1 + gamma_2 ) ),twohalf)
                   * pow(gamma_1,mione) * pow(gamma_2,mione)
                   * exp(-(premitive gaussian function of A[rr].alpha*premitive gaussian function of B[vv].alpha
                   * pow(distance between A and B,2))/gamma_1)
                   * exp(-(premitive gaussian function of C[ss].alpha
                   * premitive gaussian function of D[tt].alpha
                   * pow(distance between C and D,2))/gamma_2) * half_ERI
        free memory of  F array
 end for
 end for
 end for
 end for
         return ERI_result
 END
```

3.3 Calculation accuracy testing

Calculation accuracy testing of one- and two-electron integral of MOCCA program is seperated into two parts, including direct accuracy testing and Hartree-Fock calculation. For direct accuracy testing, one- and two-electron integrals results of MOCCA program were compared directly with GAMESS (US) program. For Hartree-Fock calculation, implemented one- and two-electron integrals were also applied to Hartree-Fock calculation on MOCCA program compared with GAMESS (US) program.

3.3.1 Direct accuracy testing

Direct accuracy testing of one- and two-electron integral of MOCCA program is seperated into two parts, including full accuracy testing and special testing. For full accuracy testing, one-electron integrals including overlap, kinetic and potential integrals and two-electron integrals of methane, carbon monoxide and water molecules were calculated by MOCCA program compared with GAMESS (US) program. Atomic orbitals of methane, carbon monoxide and water molecules were simulated using STO-3G, 3-21G and 6-31G basis sets. The number of two-electron integrals calculated by MOCCA program is not equal to GAMESS (US) program (Table **3.1**). Cauchy-Schwarz screening is available in GAMESS (US) program for screening very little value of two-electron integrals. MOCCA program calculated all of integrals of the system.

Table **3.1**: The number of two-electron integrals calculated by MOCCA and GAMESS (US) program of each calculation testing cases.

Basis set	CH_4		H_2O		CO	
	GAMESS (US)	MOCCA	GAMESS (US)	MOCCA	GAMESS (US)	MOCCA
STO-3G	921	6,561	228	2,401	512	10,000
3-21G	10,695	83,521	2,260	28,561	4,350	104,976
6-31G	10,695	83,521	2,260	28,561	4,350	104,976

All one- and two-electron integrals calculated by GAMESS (US) program of each testing systems were compared to MOCCA program. The maximum and average error of MOCCA program were investigated compared to GAMESS (US) program using python language implemented program. The verification perform on Intel Core i5 CPU. Maximum error and Average error were calculated by

$$Maximum\ error = \max_{1\leq i\leq num} \left| R_{GAMESS\ (US)}[i] - R_{MOCCA}[i] \right|$$

$$Average\ error = \frac{\sum_{i=1}^{num} \left| R_{GAMESS\ (US)}[i] - R_{MOCCA}[i] \right|}{num},$$

where $R_{MOCCA}[i]$ is the i order integral calculated by MOCCA program, $R_{GAMESS\ (US)}[i]$ is the i order integral calculated by GAMESS (US) program, and num is the number of integrals calculated by GAMESS (US) program.

For special testing, all two-electron integrals of one of (sp,df), (sd,pf) and (sf,pd) groups of ammonia molecule with aug-cc-pVDZ-RI basis set were calculated by MOCCA program compared with GAMESS (US) program. Table **3.2** show two-electron integral of (sp,df), (sd,pf) and (sf,pd) groups.

Table **3.2**: Two-electron integrals of (sp,df), (sd,pf) and (sf,pd) groups.

(sp,df)	(sd,pf)	(sf,pd)
(sp,df), (sp,fd), (ps,df), (ps,fd), (df,ps), (df,sp), (fd,ps), (fd,sp)	(sd,pf), (sd,fd), (ds,pf), (ds,fd), (pf,sd), (pf,ds), (fp,sd), (fp,ds)	(sf,dp), (sf,pd), (fs,dp), (fs,pd), (dp,sf), (dp,fs), (pd,sf), (pd,fs)

Coordinates of s, p, d and f atomic orbital for calculation testing are shown in Table **3.3**. Angular momentum, exponent, and contraction coefficient of gaussian-type function for atomic orbital simulation are shown in Table **3.4**.

Table **3.3**: Coordinate of atomic orbitals for (sp,df), (sd,pf) and (sf,pd)-type two-electron integrals.

Atom (atomic orbital-type)	Coordinate (Bohr)		
	x	y	z
H (s)	1.0139900233640282	-0.6966744989081943	-0.7736725789922515
H (p)	1.0139353418889756	-0.3216725580090439	0.9901962696397845
H (d)	1.0139046391276065	1.0184033187174335	-0.2164947672177483
N (f)	-0.2189257301552365	-0.0000040492583971	-0.0000020816689214

Table **3.4**: Angular momentums, exponents, and contraction coefficients of gaussian-type functions for atomic orbitals simulation.

Atom (atomic orbital-type)	Angular momentum			Exponent	Contraction coefficient
	l	m	n		
H (s)	0	0	0	0.11272018383	1.0000000
H (p)	0	1	0	0.28586021512	1.0000000
H (d)	1	1	0	0.30382076276	1.0000000
N (f)	1	1	1	0.41976031664	1.0000000

MOCCA program calculated all integrals in the group but GAMESS (US) program calculated only one of integral in the group. The maximum error of each testing groups were investigated compared to GAMESS (US) program. The vertification perform on Intel Core i5 CPU. Maximum error was calculated by

$$Maximum\ error = \max_{1 \leq i \leq num} \left| R_{GAMESS\ (US)} - R_{MOCCA}[i] \right|,$$

where $R_{MOCCA}[i]$ is the i order integral in the group calculated by MOCCA program, $R_{GAMESS\ (US)}$ is one of integral in the group calculated by GAMESS (US) program, and num is the number of integrals of the group calculated by MOCCA program.

3.3.2 Hartree-Fock calculation

Implemented one- and two-electron integrals calculations were applied to Hartree-Fock calculation on MOCCA program for three molecules including methane, carbon monoxide and water molecules simulated with 6-31G basis set compared with GAMESS (US) program. Before apply implemented one- and two-electron integrals calculation to Hartree-Fock calculation of MOCCA program, the accuracy of Hartree-Fock calculation of MOCCA program were tested by calculated electronic energy (a.u.) compared with GAMESS (US) program using one- and two-electron integrals results from GAMESS (US) program. For Hartree-Fock calculation, the number of two-electron integrals calculated by MOCCA program is equal to the number of two-electron integrals calculated by GAMESS (US) program for every testing molecules. The absolute errors of calculation results of MOCCA program compared with GAMESS (US) program were calculated by

$$Absolute\ error = \left|E_{MOCCA} - E_{GAMESS\ (US)}\right|,$$

where E_{MOCCA} is electronic energy (a.u.) calculated by MOCCA program, and $E_{GAMESS\ (US)}$ is electronic energy (a.u.) calculated by GAMESS (US) program.

3.4 Calculation speed testing

To avoid wrap divergence problem, (ss|ss)-type two-electron integral is selected for the calculation speed comparision between MOCCA and GAMESS (US) program. (ss|ss)-type two-electron integral was also implemented into MOCCA program. The (ss|ss)-type two-electron integrals of linear-hydrogen models including H_{30}, H_{40} and H_{50} were calculated by MOCCA program. All hydrogen atoms in each models have 1 s-type atomic orbital. Calculation testing is seperated into 3 cases including A, B and C (Table **3.5**). Each case have the different exponent for gaussian-type function used to simulate s-type atomic orbital. Exponent and contraction coefficient for gaussian-type function are extracted from coemd-2 basis set for hydrogen atom [25]. From Table **3.5**, first, third and fifth in column shell label the exponent and contraction coefficient using to simulate first, third and fifth shell s-type atomic orbital of hydrogen atom with coemd-2 basis set, respectively. There are 3 comparisons between MOCCA and GAMESS (US) program in 1 case with the same exponent and contraction coefficient including comparision for H_{30}, H_{40} and H_{50} models. Calculation speed in integrals per second of MOCCA program were compared with GAMESS (US) program.

Table **3.5**: Exponent and contraction coefficient of gaussain-type function used to simulate s-type atomic orbitals of hydrogen atom for each calculation testing cases (A, B and C).

Case	Shell	Exponent	Contraction coefficient
A	first	150.2760700	1.000000000000
B	third	9.6283875	1.000000000000
C	fifth	0.5325897	1.000000000000

CHAPTER IV

RESULTS AND DISCUSSION

The discussion was seperated into two parts: the computational accuracy and the speed test for both one- and two-electron integrals calculation in the MOCCA program compared with GAMESS (US) program. Both programs used the same data, which obtained from the GAMESS (US) program to be an initial parameter.

4.1 Calculation accuracy

Calculation accuracy part include direct accuracy testing and Hartree-Fock calculation. For direct accuracy testing part, one- and two-electron integrals results of MOCCA program were compared directly to the results of GAMESS (US) program. For Hartree-Fock calculation part, implemented one- and two-electron integrals calculations were applied to Hartree-Fock calculation of MOCCA program compared to GAMESS (US) program. The Hartree-Fock calculation of MOCCA program was developed by my co-worker [10].

4.1.1 Direct accuracy testing

The tests utilized the same information of coordinates, contraction coefficients and exponents to be the initial parameter. MOCCA and GAMESS (US) program were implemented on different processors. For two-electron integrals calculation, Taketa method and Rys quadrature method were applied for calculation in MOCCA and GAMESS (US) program, respectively. Calculaion results of both programs are slightly different. From one- and two-electron integrals results of MOCCA program, the results of MOCCA program are realiable compared with GAMESS (US) program.

For full accuracy testing, there are nine testing cases consisting of methane (STO-3G, 3-21G and 6-31G), carbon monoxide (STO-3G, 3-21G and 6-31G) and water (STO-3G, 3-21G and 6-31G). Maximum error and average error were evaluated. Table **4.1** shows maximum error and average error of one and two-electron integrals results of methane, water and carbon monoxide molecules calculated by MOCCA program compared to GAMESS (US) program. For one-electron integrals, the largest maximum errors are 9.99×10^{-16}, 107.00×10^{-16} and $15{,}200.00\times10^{-16}$ for overlap integrals, kinetic energy integrals and potential integrals, respectively. Increasing of maximum error between the overlap integral and kinetic energy integrals due to round-off error. Kinetic energy integral is complicated than overlap integral calculation, because the calculation is essential to evaluate overlap integral function for 7 times. The largest maximum error of potential integral is larger than overlap and kinetic energy due to round-off error. The complicated binomial coefficients and auxiliary function are essential in the calculation of potential integral. To compare between maximum error and average error of potential integrals of methane, carbon monoxide and water molecules, the maximum error and average error of carbon monoxide molecule are larger than methane and water molecules, because of polarization effect of the electron density between p-type orbitals. For methane and water molecules, there are only carbon and oxygen atoms having p-type orbitals, respectively. There are p-type orbitals in both of carbon and oxygen atoms in carbon monoxide molecule. Thus, polarization effect in carbon monoxide molecule is influential than methane and water molecules.

For two-electron integrals or electron respulstion integrals of full accuracy testing, the largest maximum error is 710.00×10^{-16} in the case of carbon monoxide molecule simulate with 3-21G basis set. For comparison between 3 molecules (methane, water and carbon monoxide molecules) with the same basis set, maximum error and average error increase when incresing of polarization effect influence of the electron density between p-type orbitals.

Table **4.1**: Maximum error and average error ($\times 10^{-16}$) of one- and two-electron integrals calculation results of methane, water and carbon monoxide molecules.

Integral type	Basis set	CH_4		H_2O		CO	
		Max.	Avg.	Max.	Avg.	Max.	Avg.
Overlap	STO-3G	9.99	1.50	9.99	1.60	9.99	1.04
	3-21G	9.99	1.78	9.99	1.42	9.99	1.03
	6-31G	9.99	0.77	9.99	1.29	9.99	0.85
Kinetic	STO-3G	24.00	3.90	20.00	2.60	24.00	2.08
	3-21G	9.90	0.42	102.00	2.80	97.70	1.98
	6-31G	97.90	2.02	107.00	3.48	107.00	3.37
Potential	STO-3G	4,970.00	1,020.00	8,030.00	1,290.00	11,100.00	1,860.00
	3-21G	5,970.00	821.00	8,030.00	945.00	10,000.00	142.00
	6-31G	4,970.00	944.00	8,030.00	961.00	15,200.00	1,440.00
Electron repulsion	STO-3G	302.00	24.80	400.00	43.00	490.00	60.70
	3-21G	350.00	17.80	566.00	30.70	710.00	48.00
	6-31G	299.00	19.00	370.00	30.10	650.00	64.80

For special testing, all two-electron integrals of one of (sp,df), (sd,pf) and (s-f,pd) groups of ammonia molecule with aug-cc-pVDZ-RI basis set were calculated by MOCCA program compared with GAMESS (US) program. Table **4.2** shows maximum error of all two-electron integrals in (sp,df), (sd,pf) and (sf,pd) groups calculated by MOCCA program compared with GAMESS (US) program. From Table **4.2**, the largest maximum error is 1.00×10^{-16}. Calculation results of MOCCA program are slightly different with GAMESS (US) program. For MOCCA program, two-electron integrals in the same groups are not different significantly with each others.

Table **4.2**: Maximum error of all two-electron integrals in (sp,df), (sd,pf) and (sf,pd) groups calculated by MOCCA program compared with GAMESS (US) program.

Two-electron integral group	Maximum error ($\times 10^{-16}$)
(sp,df)	1
(sd,pf)	1
(sf,pd)	1

4.1.2 Hatree-Fock calculation

Hartree-Fock calculations of MOCCA program were tested by three molecules including methane, carbon monoxide and water molecules simulated with 6-31G basis set. The tests utilized the same information of coordinates, contraction coefficients and exponents to be the initial parameter. Electronic energies (a.u.) of each testing molecules were evaluated compared with GAMESS (US) program by absolute error. Before applied implemented one- and two-electron integrals calculation to Hartree-Fock calculation of MOCCA program, the accuracy of Hartree-Fock calculation of MOCCA program were tested by calculated electronic energy (a.u.) compared with GAMESS (US) program using one- and two-electron integrals results from GAMESS (US) program. From Table **4.3**, column A1 and A2 are electronic energy (a.u.) calculated by MOCCA program using one- and two-electron integrals results from GAMESS (US) program and absolute error compared with GAMESS (US) program. From column A2, the largest absolute error is 2×10^{-8}. Hartree-Fock calculation of MOCCA program is corrected compared with GAMESS (US) program. Column B1 and B2 are electronic energy (a.u.) calculated by MOCCA program using implemented one- and two-electron integrals calculations and absolute error compared with GAMESS (US) program. From column B2, the accuracy of Hartree-Fock calculation of MOCCA program is better than 1×10^{-7} compared with GAMESS (US) program. From Hartree-Fock calculation results, implemented one- and two-electron integrals calculations can be applied to Hartree-Fock calculation correctly compared with GAMESS (US) program.

Table **4.3**: Electronic energies (a.u.) calculated by Hartree-Fock method of three molecules including methane, carbon monoxide and water molecules simulated with 6-31G basis set. Num is the number of calculated two-electron integrals for Hartree-Fock calculation.

Molecule	Num	E_{MOCCA}		E_{GAMESS}	Absolute error ($\times 10^{-8}$)	
		A1	B1		A2	B2
CH_4	10695	-40.18014138	-40.18014138	-40.18014138	0	0
H_2O	2260	-75.98507830	-75.98507830	-75.98507832	2	2
CO	4350	-106.75183300	-106.75183299	-106.75183300	0	1

4.2 Calculation speed testing

The (ss|ss)-type two-electron integrals of linear-hydrogen models including H_{30} , H_{40} and H_{50} were calculated by MOCCA program compared with GAMESS (US) program for calculation speed in integrals per second (ints/s). The tests utilized the same information of coordinates, contraction coefficients and exponents to be the initial parameters. Calculation testing is seperated into 3 cases including A, B and C. Each case have the different exponent for gaussian-type function used to simulate s-type atomic orbital of hydrogen atoms. Exponent and contraction coefficient are extracted from coemd-2 basis set for hydrogen atom [25]. The number of integrals evaluated by MOCCA program is not equal to GAMESS (US) program with the same model in each case. Cauchy–Schwarz screening is avaliable in GAMESS (US) program for culling a tiny value of two-electron integrals. Cauchy–Schwarz screening is not implemented into MOCCA program because, it is not in the scope of this research. The goal of this research is only making MOCCA program to calculate one- and two-electron integrals reliably. For MOCCA program, calculation times of the same model with different cases is not different significantly because the number of integrals evaluated by MOCCA program with different case is not different. The number of integrals evaluated by MOCCA program are 810,000, 2,560,000 and 6,250,000 for H_{30} , H_{40} and H_{50}, respectively. Calculation speed of

MOCCA program were calculated by

$$Calculation\ speed = \frac{Num}{Avg.\ time},$$

where Num is the number of integrals evaluated by MOCCA program for each model, $Avg.\ time$ is the average calculation times of the model with different cases. Calculation speed of MOCCA program for H_{30} , H_{40} and H_{50} are 36103228.49, 37108978.63 and 37463061.42 integrals per second, respectively. The number of integrals evaluated by GAMESS (US) program, calculation speed of GAMESS (US) program and calculation speed of MOCCA program compared with GAMESS (US) program are shown in Table **4.4**.

Table **4.4**: Calculation speed testing results of GAMESS (US) program. Num is the number of integrals evaluated by GAMESS (US) program. Shell label the exponent and contraction coefficient using to simulate s-type atomic orbital of hydrogen atom with coemd-2 basis set.

Case (Shell)	Model	Num	Calculation speed (ints/s)	MOCCA/GAMESS (times)
	H_{30}	465	4650.00	7764.14
A (first)	H_{40}	820	8200.00	4525.49
	H_{50}	1275	12750.00	2938.23
	H_{30}	2610	26100.00	1383.27
B (third)	H_{40}	4674	46740.00	793.94
	H_{50}	7194	71940.00	520.75
	H_{30}	25019	250190.00	144.30
C (fifth)	H_{40}	47042	235210.00	157.77
	H_{50}	75592	251973.33	148.68

Table **4.4** and Fig. **4.1** show calculation speed and performance of MOCCA program compared with GAMESS (US) program. From Table **4.4** and Fig. **4.1**, calculation speed of MOCCA program is higher than GAMESS (US) program for every cases. MOCCA program can calculate faster than GAMSS (US) program about 150 to 7000 times.

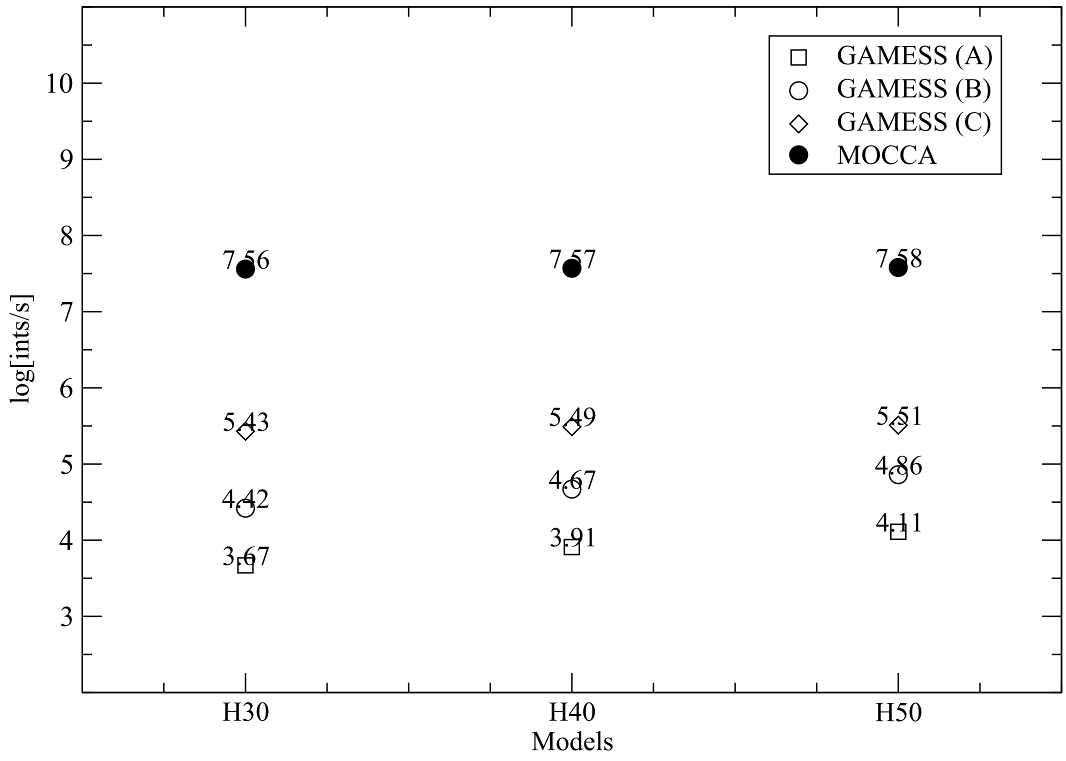


Figure **4.1**: The performance of MOCCA and GAMESS (US) program, which presented by comparing integrals/second (horizon axis) in the base 10 logalithm function and theoretical models including H_{30} (H30), H_{40} (H40) and H_{50} (H50).

CHAPTER V

CONCLUSION

This research implemented one- and two-electron integrals developed by Taketa group into MOCCA program. The calculation results and time were compared with the standard program. One- and two-electron integrals calculation results of MOCCA program show its accuracy for the level of double precision. From calculation testing, the largest maximum errors are 9.99×10^{-16}, 107.00×10^{-16}, $15{,}200.00\times10^{-16}$ and 710.00×10^{-16} for overlap integrals, kinetic energy integrals, potential integrals and two-electron integrals, respectively. From calculation speed testing, MOCCA program can calculates faster than standard program about 150 to 7000 times. Implemented one- and two-electron integrals were applied to Hartree-Fock calculation in MOCCA program compared with the standard program. From Hartree-Fock results of MOCCA program, the accuracy of Hartree-Fock calculation is better than 1×10^{-7} compared with the standard program. For the practical use of MOCCA program, calculation speed of electron integrals will be improved for the next version of MOCCA program. To improve electron integrals calculation part of MOCCA program, Cauchy–Schwarz screening and electron integrals calculation methods including Rys quadrature, recurrence relation method or/and other calculation methods will be implemented for MOCCA program in the future.

REFERENCES

[1] X. Ge., S. Binnie, D. Rocca, R. Gebauer, S. Baroni *Turbo TDDFT 2.0 Hybrid Functionals and New Algorithms within Time-Dependent Density-Functional Perturbation Theory* Comput. Phys. Commun., **2014**, *185*, 2080–2089.

[2] I. Ufimtsev, T. Martinez *Quantum Chemistry on Graphical Processing Units. 1. Strategies for Two-Electron Integral Evaluation* J. Chem. Theory Comput., **2008**, *4*, 222–231.

[3] J. Stone, D. Hardy, I. Ufimtsev, K. Schulten *GPU-Accelerated Molecular Modeling Coming of Age* J. Mol. Graph. Model., **2010**, *29*, 116–125.

[4] Y. Maruyama, F. Hirata *Modified Andeson Method for Accelerating 3D-RISM Calculations Using Graphics Processing Unit* J. Chem. Theory Comput., **2012**, *8*, 3015–3021.

[5] S. Cook *CUDA Programming;* Elsevier: USA, 2012.

[6] J. Cheng, M. Grossman, T. McKercher; John Wiley and Sons, Inc.: USA, 2014.

[7] M. Nitsche, M. Ferreria, E. Mocskos, M. Lebrero *GPU Accelerated Implementation of Density Functional Theory for Hybrid QM/MM Simulations* J. Chem. Theory Comput., **2014**, *10*, 959–967.

[8] I. Ufimtsev, T. Martinez *Quantum Chemistry on Graphical Processing Units. 2. Direct Self-Consistent-Field Implementation* J. Chem. Theory Comput., **2009**, *5*, 1004–1015.

[9] C. M. Isborn, N. Luehr, I. Ufimtsev, T. Martinez *An overview of the Amber biomolecular simulation package* Wires. Comput. Mol. Sci., **2013**, *3*, 198–210.

[10] C. Pornpiganon; Chulalongkorn University: Thailand, 2014.

[11] K. Yasuda *Two-Electron Integral Evaluation on the Graphic Processing Unit* J. Comput. Chem., **2007**, *29*, 334–342.

[12] H. M. K. Yasuda *Efficient Calculation of Two-Electron Integrals for High Angular Basis Functions* Int. J. Quantum Chem., **2014**, *114*, 543–552.

[13] A. Asadchev, V. Allada, J. Felder, B. Bode, M. Gordon, T. Windus *Uncontracted Rys Quadrature Implementation of up to G Functions on Graphical Processing Units* J. Chem. Theory Comput., **2010**, *6*, 696–704.

[14] Y. Miao, J. K.M Merz *Accleration of Electron Repulsion Integral Evaluation on Graphics Processing Units via Use of Recurrence Relations* J. Chem. Theory Comput., **2013**, *9*, 965–976.

[15] Y. Miao, J. K.M Merz *Acceleration of High Angular Momentum Electron Repulsion Integrals and Integral Derivatives on Graphics Processing Units* J. Chem. Theory Comput., **2015**, *11*, 1449–1462.

[16] M. Schmidt, K. Baldridge, J. Boatz, S. Elbert, M. Gordon, J. Jensen, S. Koseki, N. Matsunaga, K. Nguyen, S. Su, T. Windus, M. Dupuis, J. Montgomery *General Atomic and Molecular Electronic Structure System* J. Comput. Chem., **1993**, *14*, 1347–1363.

[17] M. Born, J. Oppenheimer *Zur Quantentheorie der Molekeln* Ann. Phys., **1927**, *84*, 457–484.

[18] K. O-ohata, H. Taketa, S. Huzinaga *Gaussian-Expansion Method for Molecular Integral* J. Phys. Soc. Jpn., **1966**, *21*, 2313–2324.

[19] L. Mcmurchie, E. Davidson *One- and Two- Electron Integral over Cartesian Gaussian Function* J. Comput. Phys., **1978**, *26*, 218–231.

[20] M. Dupuis, J. Rys, H. King *Computation of Electron Repulsion Integral Using the Rys Quadrature Method* J. Chem. Phys., **1983**, *4*, 154–157.

[21] S. Obara, A. Saika *Efficient Recursive Computation of Molecular Integrals over Cartesian Gaussian Functions* J. Chem. Phys., **1986**, *7*, 3963–3974.

[22] M. Head-Gordon, J. Pople *A Method for Two-Electron Gaussian Integral and Integral Derivative Evaluation Using Recurrence Relations* J. Chem. Phys., **1988**, *9*, 5777–5786.

[23] N. Kharmsri, V. Vchirawongkwin *One- and two-electron integrals calculations of MOCC program on graphic processing units (GPUs)* ANSCSE20, **2016**, pages 80–88.

[24] D. B. Kirk, W. mei W. Hwu; Elsevier: USA, 2010.

[25] J. Lehtola, P. Manninen, M. Hakala, K. Hamalainen *Completeness-optimized basis sets. Application to ground-state electron momentum densities* J. Chem. Phys., **2012**, *137*, 104105–1–104105–8.

VITAE

Personal Details

Name	Mr.Noppakoon Kharmsri
Date of Birth	April 5, 1991
Place of Birth	Uttaradit, Thailand
Address	155/1, Chaijumpon, Laplae, Uttaradit 53130, Thailand
Telephone	088-282-3290
E-mail address	milkbasisset@gmail.com

Education

2013-2016	M.Sc. in Chemistry, Chulalongkorn University, Thailand
2009-2012	B.Sc. in Chemistry (second honors), Naresuan University, Thailand
2002-2008	Uttaradit School, Uttaradit, Thailand
1995-2002	Anuban Uttaradit School, Uttaradit, Thailand

Presentation

Noppakoon Kharmsri, Viwat Vchirawongkwin One- and two-electron integral calculation of MOCCA program on graphic processing units (GPUs) *The* 20th *International Annual Symposium on Computational Science and Engineering*, Kasetsart University, Bangkhen Campus, Bangkok, Thailand, July 27–29, **2016**, pp 121–122.