Development and Validation of the Computational Code GEM for Simulating Quantum Few-Body Systems through Optimization on the Supercomputer Fugaku

Shigeyoshi Aoyama[1,5(✉)], Yusuke Tsunoda[2,5], Noritaka Shimizu[2,5], Hideo Matsufuru[1,5], Issaku Kanamori[3,5], and Emiko Hiyama[4,5]

[1] Computing Research Center, High Energy Accelerator Research Organization (KEK), Tsukuba, Ibaraki 305-0801, Japan
shigeyoshi.aoyama@kek.jp
[2] Center for Computational Sciences, University of Tsukuba, 1-1-1 Tennodai, Tsukuba 305-8577, Japan
[3] RIKEN Center for Computational Science, Kobe 650-0047, Japan
[4] Department of Physics, Tohoku University, Sendai 980-8578, Japan
[5] RIKEN Nishina Center, Wako 351-0198, Japan

Abstract. The Gaussian expansion method (GEM) is a prominent approach widely used for precision calculations in few-body systems. While the calculation of its matrix elements is considerably complex, to alleviate these computational complexities, the Gauss-Lobe technique was proposed by one of the authors (Hiyama). This approach has been implemented in a well-established computational code (GEM code) that has been applied to the analysis of numerous few-body systems over the years. Furthermore, it is renowned for its high precision and versatility in few-body system calculations. However, the core of this code was developed more than thirty years ago and, while it has undergone numerous updates over the decades, it is not architected to fully exploit the performance of modern massively parallel computers, such as the Supercomputer Fugaku as well as recent GPU-based systems.

In this paper, we report on and discuss the modernization of GEM code, which has long been based on a design for single-processor systems. We present its implementation for high-performance hybrid parallel computing on the Supercomputer Fugaku and evaluate its computational performance. We demonstrate its performance using the ^{4}He pentamer, a five-body system for which detailed analysis is currently highly anticipated.

Through the modernization of the GEM code in this study, the hybrid parallelization achieved a 122.5-fold speed-up using 120 nodes, demonstrating a so-called "super-linear speed-up" that exceeds the ideal value. This result clearly highlights the significant utility and effectiveness of the parallelization strategy implemented in the modern GEM code. Furthermore, this paper describes a practical technique for adjusting the ratio

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16760, pp. 484–499, 2027.
https://doi.org/10.1007/978-3-032-30521-3_31

between MPI parallelism and thread parallelism without modifying the original GEM source code. This technique enables efficient parallelization for six-body or even larger few-body systems.

Keywords: few-body system · massive parallel computing · HPC

1 Introduction

Numerical approaches to quantum few-body systems have emerged as a pivotal methodology in the analysis of quantum systems. Given the recent breakthroughs in generative AI and machine learning, there is growing anticipation for realizing quantum computing to accelerate these technologies. Consequently, the precise simulation of quantum few-body systems not only represents a challenge to fundamental issues at the frontiers of conventional science but is also expected to serve as a vital theoretical foundation for the practical realization of quantum computing.

Furthermore, many intriguing yet unresolved issues remain in the field of few-body physics, such as the excited states of the ^{4}He pentamer (five-body system) investigated in this paper. The ^{4}He trimer (three-body system) and ^{4}He tetramer (four-body system) are characterized by a compact ground state located near the van der Waals potential pocket of several angstroms; on the other hand, their first excited states are S-wave halo-like states that extend to several hundred angstroms, far beyond the potential pocket. Such states are also considered to be related to Efimov states [1] and Bose—Einstein condensation [2–4], which continue to be of profound physical interest. Similarly, systems of five or more ^{4}He atoms are expected to exhibit such a state with an S-wave halo tail extending to several hundred angstroms near the ^{4}He one-atom separation threshold.

To solve quantum few-body systems, various sophisticated methods have been developed, including the Gaussian Expansion Method (GEM) [5–7], an extension of the Faddeev theory to N-body systems [8,9], Hyperspherical Harmonics (HH) [10], the No-Core Shell Model (NCSM) [11,12], and Green's Function Monte Carlo (GFMC) [13], etc. Among these, GEM is widely recognized for its exceptional capability to rigorously solve Hamiltonians in which complex parti- cle rearrangements occur, and where ground and excited states must be treated simultaneously despite their vastly different spatial scales. This makes GEM particularly effective for systems such as those addressed in this study, where a compact ground state and a significantly extended halo-like excited state must be treated within a single unified framework. While various equivalent approaches to the Gaussian Expansion Method (GEM) have been developed by numerous research groups [14–18] due to its high utility, this paper employs the Gauss-Lobe basis framework. In this framework, as proposed by one of the present authors (Hiyama), the angular momentum integration is formulated as a limit of Gaussian integrations, making the hand-calculation of complex few-body integrals much more manageable.

Beyond the ^{4}He pentamer addressed in this study, various five-body systems remain at the forefront of contemporary few-body physics. Notable examples include nuclear systems such as ^{8}He (^{4}He+n+n+n+n) and ^{5}H (p+n+n+n+n), as well as hypernuclei like ${}^{5}_{\Lambda}$H ($p+n+n+n+\Lambda$). Furthermore, kaonic nuclei such as $K^- + p + n + n + n$ and quark cluster systems like the Θ^+ pentaquark ($uudds$) continue to be subjects of intense theoretical and experimental interest.

Furthermore, by increasing the number of particles in few-body calculations to ten or more and comparing the results with those of many-body systems, it is expected that the fundamental principles underlying many-body physics can be more firmly established. From this perspective, the authors have been making efforts to increase the number of particles that can be treated within the GEM framework. Figure 1 schematically illustrates the ongoing and future expansions of our methodology. The blue text represents the entire project, while the red (this work) and blue texts indicate the specific approaches taken toward it.

At CSHPC 2025, some of authors (S. Aoyama and T. Myo) reported an extension of the stochastic variational method (SVM) [21–23], referred to as SVM$^+$, as a machine learning-inspired approach [24] (the upper part of Fig. 1), specifically aimed at reducing memory consumption when increasing the number of particles. As a benchmark, we have already achieved an approximately 99% reduction in memory usage at the three-body system stage. Similarly, the primary bottleneck when increasing the number of particles is the computational cost of calculating matrix elements. In this paper, we report on the development of a newly optimized GEM code focused on this challenge and the resulting performance improvements.

At HPC Asia 2026, some of the present authors (E. Hiyama, I. Kanamori, and H. Matsufuru) reported on solving large-scale eigenvalue problems in quantum few-body systems (as shown in the left part of Fig. 1) and specialized numerical algorithms of diagonalization (as shown in the right part of Fig. 1) [25]. In the Gaussian Expansion Method (GEM), the strong linear dependence of the basis functions results in dense matrices with large off-diagonal elements. In such cases, it is well known that even minor numerical inaccuracies can be amplified during the diagonalization of large-dimensional matrices, making it difficult to obtain high- precision solutions. Consequently, solving the eigenvalue problem itself becomes a challenging task. In Ref. [25], a systematic analysis of the numerical errors occurring during actual diagonalization was performed using EigenExa [25], an MPI-parallelized eigensolver had developed on the supercomputer Fugaku.

The original Gaussian Expansion Method (GEM) code was developed over 30 years ago for three-body systems and has since been extended and applied to four- and five-body systems. Having been utilized for decades, this code is not only remarkably stable but also capable of maintaining a high precision of over ten decimal places, making it a highly versatile tool applicable to a wide range of physical systems.

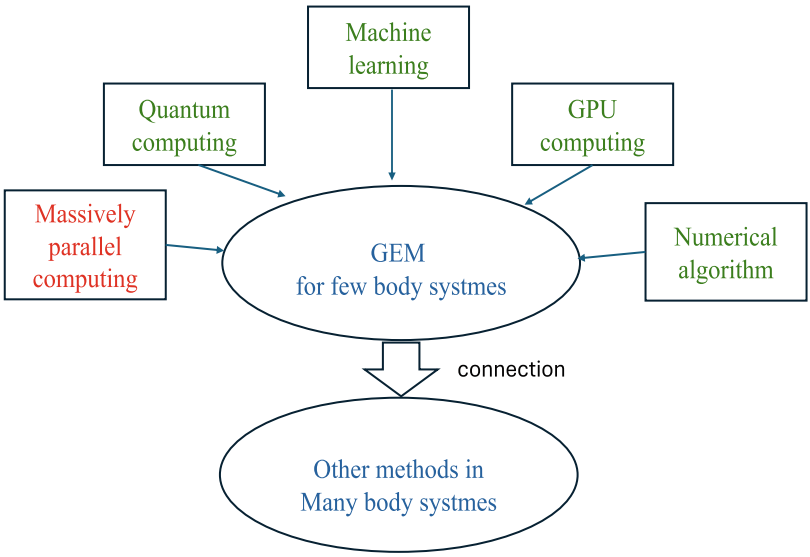


Fig. 1. Schematic diagram of GEM's development and expansion.

However, this legacy code was not designed for modern hybrid parallelization or GPU acceleration. Furthermore, the original GEM code does not account for the efficient utilization of memory and registers, such as SIMD (Single Instruction, Multiple Data) operations. Consequently, the code typically achieves only a few percent, specifically around 3–4% (in present computation), of the peak FLOPS specifications, and its memory bandwidth utilization remains below 1%. Thus, there remains substantial potential for further acceleration—on the order of 10^7 at least—through code-level optimizations (such as SIMD), intra-node GPU parallelization, and large-scale MPI parallelization ($10\times1{,}000\times1{,}000$). Such advancements could bring currently inaccessible six-, seven-, and eight-body systems within reach of precision calculations.

Although the supercomputer Fugaku, Japan's current flagship supercomputer, is not equipped with GPUs, its massive scale of over 150,000 nodes. If approximately 10% of the total node count is utilized, there is the potential for tens of thousands of times faster computation without GPUs, provided that the parallel efficiency relative to the node count can be brought close to 100%. Therefore, in this paper, we aim as a first step to maximize this speed-up ratio during the parallelization of the GEM code (the left part of Fig. 1). Furthermore, optimization of memory bandwidth and register utilization, which are potential avenues for further acceleration on Fugaku, is beyond the scope of this study and will be addressed in future work, such as during the extension of the code to six-body systems or larger.

The remainder of this paper is organized as follows. In Sect. 2, the framework relating GEM used in the present calculation is briefly explained. In Sect. 3,

we present the calculation results and provide a detailed discussion. Finally, a summary and our conclusions are given.

2 Method

2.1 Gaussian Basis Function

Here, we briefly explain GEM for a five-body bosonic system with Jacobi coordinate. The detail of GEM is given in review papers [7,26]. In GEM, the wave function Ψ_{JM} (J is the total angular momentum and M is its z-component) of the Schrödinger equation,

$$(H - E)\Psi_{JM} = 0, \tag{1}$$

is obtained by diagonalizing the Hamiltonian in a space spanned by Gaussian basis functions which are constructed on Jacobi coordinates. ^{4}He–^{4}He potential in the Hamiltonian used in this research, $V(r) = V_1 \exp(-\alpha_1 r^2) + V_2 \exp(-\alpha_2 r^2)$, with parameters, $V_1 = 2000\,\mathrm{K}$, $\alpha_1 = 1.0\ \mathrm{Å}^{-2}$, $V_2 = -8.2823\ \mathrm{K}$, $\alpha_2 = 0.082189\ \mathrm{Å}^{-2}$. This two range (attractive and repulsive) Gaussian potential was made to reproduce the observed properties of ^{4}He dimer [27].

In Fig. 2, the Jacobi coordinates for a five-body system is displayed $\boldsymbol{R_i}$ is i-th Jacobi coordinate, L_i is angular momentum, b_i is a width parameter of Gaussian type basis functions. (a) is 4(K)+1 type or in order type which describes the K type for the four-body system plus a ^{4}He atom. (b) is 4(H)+1 type which describes the H type for the four-body system plus a ^{4}He atom. (c) is 3+2 type which describes trimer and dimer which first appear in the five-body system. If a trimer plus dimer like state is important, (c) is more appropriate than (a) or (b).

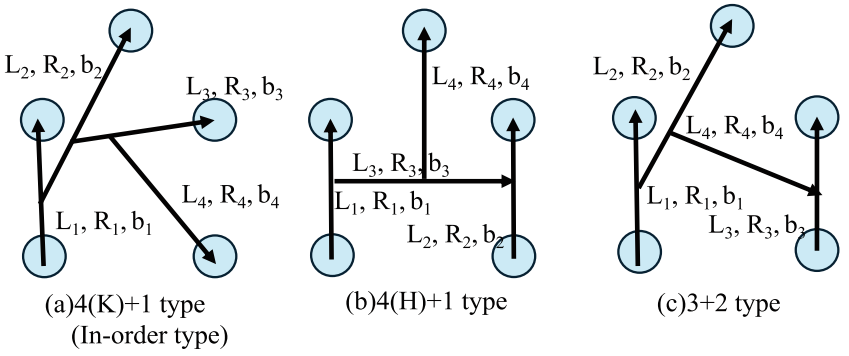


Fig. 2. Jacobi coordinates for five-body system.

The total wave function Ψ_{JM} is written as

$$\Psi_{JM} = \sum_{type=1}^{3} \sum_{i=1}^{i_{\max}} c_i \mathcal{S} \Phi_i^{type}(\boldsymbol{R_1}, \boldsymbol{R_2}, \boldsymbol{R_3}, \boldsymbol{R_4}). \tag{2}$$

where $\mathcal{S}$ is the symmetrization operator for bosonic systems, Φ_i is the five-body basis function, and *type* in Eq. (2) shows the type of Jacobi coordinate as shown in Fig. 2 (type=1: 4(K)+1, type=2: 4(H)+1, type=3: 3+2). In a five-body bosonic system with the same mass, the symmetrization for the wave function is equivalent to rearranging the Jacobi coordinate [7,26]. For example, in a 4(K)+1 type, the number of permutations is $5!/2 = 60$.

The Schrödinger equation is expressed in the matrix-element form as follows:

$$\sum_{j=1}^{N}(H_{ij} - ES_{ij})c_j = 0 \quad (i = 1, ..., N), \tag{3}$$

where the hamiltonian matrix-elements H_{ij} are given by

$$H_{ij} = T_{ij} + V_{ij}. \tag{4}$$

Here, T_{ij}, V_{ij}, and S_{ij} represent the kinetic energy, potential energy, and overlap matrix-elements, respectively. Here, H_{ij} represents the hamiltonian matrix-elements, which consist of the kinetic energy matrix-elements T_{ij} and the potential energy matrix-elements V_{ij}. Additionally, S_{ij} are referred to as the overlap matrix-elements.

The five-body basis function in Eq. (2) is represented as

$$\begin{aligned}\Phi_i(\boldsymbol{R_1}, \boldsymbol{R_2}, \boldsymbol{R_3}, \boldsymbol{R_4}) &= \phi_{n_1 L_1}(R_1)\,\phi_{n_2 L_2}(R_2)\,\phi_{n_3 L_3}(R_3)\,\phi_{n_4 L_4}(R_4) \\ &\times \left[\left[\left[Y_{L_1}(\widehat{R}_1) Y_{L_2}(\widehat{R}_2)\right]_{L_{12}} Y_{L_3}(\widehat{R}_3)\right]_{L_{123}} Y_{L_4}(\widehat{R}_4)\right]_{JM}, \quad (5) \\ i &\equiv \{n_1, n_2, n_3, n_4, L_1, L_2, L_{12}, L_3, L_{123}, L_4\}. \quad (6)\end{aligned}$$

where the R_i is i-th Jacobi coordinate, L_i is the angular momentum of the spherical harmonic function Y_L, and n_i is the basis number. *type* is omitted for simplicity. Radial dependence of the basis functions $\phi_{nL}(R)$ is taken as Gaussians multiplied by R^L with ranges in geometric progression [7,26],

$$\phi_{nL}(R) = N_{nL} R^L \mathrm{e}^{-a_n R^2}, \tag{7}$$

$$a_n = 1/b_n^2, \tag{8}$$

$$b_n = b_{\min}\gamma^{n-1} \quad (n = 1, \ldots, n_{\max}), \tag{9}$$

b_n is a Gaussian width parameter, γ is the common ratio given by $(b_{\max}/b_{\min})^{\frac{1}{n_{\max}-1}}$ and N_{nL} is a normalization constant. The integration of the matrix elements for the angular momentum part is very complicated due to many of the spherical harmonic functions, $Y_{LM}(\hat{R})$, with the permutations of Jacobi coordinates. To avoid this difficulty, we introduce a technique using the following basis function (called Gauss-Lobe):

$$R^L e^{-\nu R^2} Y_{LM}(\hat{R}) = \lim_{\varepsilon \to 0} \frac{1}{(\nu\varepsilon)^L} \sum_{k=1}^{k_{\max}} C_{L,M,k} \exp\left(-\nu\left(\mathbf{R} - \varepsilon \mathbf{D}_{L,M,k}\right)^2\right), \tag{10}$$

where vectors $D_{L,M,k}$ and the order of expansion $k_{\max}$ are determined for L and M. When ε is set to a near zero value, the integration of the angular momentum part can be carried out by a straightforward Gaussian integration [7]. While the left and right sides of Eq. (10) are mathematically equivalent through the limiting procedure, it has been confirmed that the matrix elements calculated using the Gauss-Lobe basis numerically agree with those obtained using the conventional Gaussian basis to high precision.

2.2 Channels

Table 1. Quantum numbers (orbital angular momenta) of the channels, which are considered to be significant for low-energy states of ^{4}He pentamer with $J = 0^+$.

type	No.	L_1	L_2	L_{12}	L_3	L_{123}	L_4
$4(K)+1$	1	0	0	0	0	0	0
	2	0	1	1	1	0	0
	3	0	2	2	2	0	0
	4	2	0	2	2	0	0
	5	2	2	0	0	0	0
$4(H)+1$	6	0	0	0	0	0	0
	7	0	2	2	2	0	0
	8	2	0	2	2	0	0
	9	2	2	0	0	0	0
$3+2$	10	0	0	0	0	0	0
	11	0	2	2	2	0	0
	12	2	0	2	2	0	0
	13	2	2	0	0	0	0

In Table 1, the quantum numbers (orbital angular momenta) of the channels are listed, which are considered to be significant for low-energy states of ^{4}He pentamer with $J = 0^+$. The first column corresponds to the Jacobi coordinates illustrated in Fig. 2. The second column provides channel numbers to make it easy to identify which channels are included in the subsequent calculations. The third and following columns list the orbital angular momenta defined for each set of Jacobi coordinates as described in Eq. (5) and Fig. 2. For example, in the case of the 4(K)+1 type, the total angular momentum J is coupled as $[[[[L_1L_2]_{L_{12}}L_3]_{L_{123}}]L_4]_J$. One of the significant features of the GEM basis is that it is a non-orthogonal basis, where each basis function possesses strong overlaps with others. Furthermore, within each channel, permutations between particles (symmetrization in the case of ^{4}He atoms) are performed. For instance, in the 4(K)+1 type, each state is a superposition of 60 (= 5!/2) basis functions after

boson permutation, which leads to the linear dependence. Consequently, even if Channel 1 and Channel 2 appear at first glance to be orthogonal in their angular momentum and have no overlap, they do, in fact, overlap. If a sufficient number of Gaussian basis functions are superimposed in Channel 1, a large portion of Channel 2's space is effectively covered; thus, Channel 2 serves a complementary role to fill in what remains missing from Channel 1. For this reason, the number of basis functions in Channel 2 can fundamentally be reduced compared to Channel 1 without loss of accuracy. This principle applies to other channels as well; the basis space that would normally be required for a later-added channel is already partially incorporated by the preceding basis functions. Therefore, convergence can be achieved by introducing only a small number of additional basis functions. From a numerical methodology perspective, a key advantage of GEM is the ability to achieve rapid convergence with a relatively small number of basis functions by utilizing this rearrangement of Jacobi coordinates and the strong linear dependence of the Gaussian basis.

While the channels listed in Table 1 do not exhaust all possibilities even from the viewpoint of including subsystem angular momenta up to $L = 2$. For the case of the ^{4}He pentamer in this study, they represent a sufficient selection to demonstrate the practical feasibility of the numerical calculations. This selection provides sufficiently converged results for the objectives of the present paper.

3 Results

3.1 Parallelization

In the development of the present code, we adopted a standard "hybrid parallelization" strategy, utilizing MPI for inter-node parallelism and thread-level parallelism for intra-node operations. The supercomputer Fugaku has relatively small memory capacity per node compared to other major supercomputers. To utilize the limited memory resources efficiently, our code avoids intra-node MPI parallelization. While one might consider a method that saves memory by writing out matrix-elements as they are calculated, such an approach could cause storage I/O delays when scaled to tens of thousands of nodes in the future, placing an excessive load on the shared file system. For this reason, we designed the system to batch writing at the memory capacity limit. Furthermore, as will be shown later, the speed-up ratio for MPI is parallel efficiency of 98.3%. This indicates that no significant delays caused by aggregating data to specific nodes via MPI have occurred at the scale of the five-body system in this study.

While we initially planned to use OpenMP for intra-node parallelization, the matrix-element calculations consist of a simple single-loop structure, which is appropriate for applying the OpenMP to that loop. Consequently, we found that the standard compiler's automatic parallelization provided approximately a 10-fold speed-up. It means that significant speed-up is not expected even if OpenMP is adopted. We mention that a 48-fold speed-up would be achieved using 48 cores on a single node in the ideal case. However, as will be demonstrated later, the current version is expected to reduce the computation time

from approximately one week to just a few tens of minutes, representing less than 1% of the resources on Fugaku. The computational speed achieved with MPI+"automatic parallelization" is sufficient for solving the five-body problem. Therefore, to avoid unnecessary bugs that may arise from manual OpenMP implementation, for this initial version of the five-body system code, we have decided to rely on automatic parallelization for the time being. When computational time and resources become more demanding for extensions to six-body systems or larger, we may revisit the implementation of the remaining potential for the speed-up.

The specific points of parallelization are described below. First, for MPI parallelization, we targeted the double loops over the channels (bra and ket) as the primary parallelization points. For instance, in the case of the 13 channels shown in Table 1, $13 \times (13+1)/2 = 91$ MPI processes can be utilized. However, if the number of channels is small, the degree of parallelism is limited; for a single channel, parallelization is not possible at all.

To circumvent this, we decomposed each channel into several segments using different Gaussian bases. For example, if a single channel is divided into N segments, $N \times (N+1)/2$ parallel processes become possible. We refer to this approach as "virtual channel decomposition". Furthermore, while the calculation time for matrix elements within the same channel is almost identical, it generally varies between different channels. In actual calculations, we improved efficiency by adjusting these division numbers to equalize the computation time across each process, facilitating load balancing.

In GEM, a matrix-element expression contains the Gaussian width parameters of the basis functions for both bra and ket states $(b'_1, b'_2, b'_3, b'_4, b_1, b_2, b_3, b_4)$. For example, if we consider a spatial superposition of 10 basis functions for each of the degrees of freedom, the number of independent matrix elements reaches 10^8. In the actual code, the calculation follows this structure:

```
 do i = 1, 100000000
     ham(i) = matrix(bp1(i), bp2(i), bp3(i), bp4(i),
&                    b1(i), b2(i), b3(i), b4(i))
 enddo
```

While the matrix elements depend on various factors such as the angular momentum specified by each channel, for the sake of simplicity, we have expressed them in the form of a function `matrix()`, highlighting only their dependence on the parameters b. The explicit expressions for the matrix elements in three- and four-body systems have been provided in previous review papers [7,26]. Although we do not show the explicit form of matrix elements for the five-body system, the formulation for the five-body system is a straightforward extension of those cases.

As there are no data dependencies across the iterations of i, the parallelization efficiency for this part was expected to be extremely high. Initially, we planned to implement OpenMP for this loop. However, due to its remarkably simple structure, we found that the compiler's standard auto-parallelization feature alone could achieve approximately a tenfold speed-up.

Here, we describe the "virtual channel decomposition" used to increase the number of MPI processes, as its implementation relies on the loop structure discussed above. In our code, the channel types, angular momenta, and Gaussian range parameters (as shown in Table 1) are provided via an input file to determine the matrix elements to be calculated. For each Jacobi coordinate, these Gaussian range parameters are defined by the initial value $b_i^{(1)}$, the final value $b_i^{(N)}$, and the number of basis functions N_i $(i = 1, 2, 3, 4)$, which consequently determines the common ratio of the geometric progression.

The fundamental idea is to virtually treat a single channel as multiple channels by partitioning these parameters (initial value, final value, and number of bases). For instance, in the case of a $10 \times 10 \times 10 \times 10$ basis set, the last parameter b_4 can be divided into 10 segments, which are then described as 10 distinct virtual channels in the input file. Consequently, the 10^8 iterations per channel mentioned in the previous example are reduced by a factor of 10, to 10^7 iterations per virtual channel.

This approach allows us to shift a portion of the loops typically handled by thread-level parallelism (b_4 and b_4') toward MPI-level parallelism without any modification to the source code itself. While a degree of parallelism in the hundreds is sufficient for intra-node thread parallelism (matching the number of cores per node), the scale required for the full five-body calculation is significantly larger.

For the final calculations of the five-body system, a few hundred nodes are sufficient, for which the current decomposition of (b_4 and b_4') is adequate. However, for massive supercomputers like Fugaku, which features over 150,000 nodes, one can easily extend this logic to the loops corresponding to (b_3 and b_3'). This enables efficient calculations across tens of thousands of nodes by shifting the parallel workload to the MPI level without requiring any code revisions. For systems with six-body or larger, it is anticipated that the required MPI parallelism will reach the scale of thousands or even tens of thousands of processes. In such cases, the technique described above—virtually decomposing channels to balance the workload across the massive MPI resources—becomes increasingly essential.

Since the actual memory per node on Fugaku is 32 GB, it is recommended to limit memory usage to a maximum of around 25 GB, accounting for the memory reserved by the system. Based on our evaluation of the matrix elements in Fortran arrays and the MPI work areas used in this program, the maximum matrix dimension that can be processed directly on a single node is slightly over 30,000. During the development phase, we considered dynamic memory allocation. However, we found that when memory usage approached its limits, it occasionally led to segmentation faults or unexpected numerical results, posing a risk of providing incorrect data to users. To ensure computational reliability, we opted for a fixed-dimension design, capping total memory usage at under 20 GB as a safety margin. This allocation includes approximately 4 GB each for the overlap, kinetic energy, and potential matrices, as well as the MPI work area, with an additional 3 to 4 GB reserved for intermediate work arrays.

Although the final five-body calculations in this study exceed this 20 GB limit, we addressed this by partitioning the calculations and storing the intermediate results in storage. The final diagonalization was then performed on a server equipped with a large-capacity memory for simplicity. Although an exact time cannot be specified as the post-processing server is shared by multiple users, the calculations in Table 2 typically take several hours. However, for actual calculations involving larger dimensions and repeated diagonalizations, we plan to perform diagonalizations using the parallelized subroutine EigenExa on Fugaku as investigated [25].

3.2 Parallel Speed-up Ratio

In this subsection, we demonstrate the parallel performance of the current code using the speed-up ratio achieved with 120 nodes as an example. Even for the advanced calculations of the ^{4}He pentamer discussed in this paper, all necessary computations for the research can be completed within one to several days (a single calculation takes from several to several tens of minutes), provided that high parallel efficiency is achieved (using 120 nodes as in this subsection).

As previously mentioned, the calculations of these matrix elements are entirely independent of one another, and the computation time for all elements within the same channel is practically identical. This represents an ideal case for parallelization, where the execution time is expected to decrease in inverse proportion to the number of nodes. For instance, in the 120-node calculation shown below, if the negligible delays caused by the MPI communication overhead are ignored, the execution time should ideally be reduced to 1/120 of the time required for a single node (representing a 120-fold speed-up).

In Fig. 3, a speed up ratio of MPI parallelization is displayed. The solid line is the computed speed-up ratio and the dashed line is the ideal line. The number of computed matrix elements is 720600 with channel 1. The value is chosen appropriately so that the computation would finish in approximately 10 min on a single node, thereby avoiding any unnecessary load on the system and causing computational delays for other users. The numerical calculations in this study were performed using the `lang/tcsds-1.2.42` compiler environment on the supercomputer Fugaku. Loading this environment specifically defines the execution paths and versions for the development tools, including Fujitsu Compiler 4.12.2 (Fortran/C/C++/Tool) and its associated mathematical libraries.

The source code was compiled with the Fujitsu Fortran compiler using the following command:

```
mpifrtpx -Kfast -SSL2 -mcmodel=large
```

In this command, `-Kfast` specifies maximum optimization to leverage the performance of the A64FX processor, `-SSL2` enables the use of the mathematical library for efficient numerical calculations, and `-mcmodel=large` sets the memory model to accommodate the large-scale memory requirements of few-body systems.

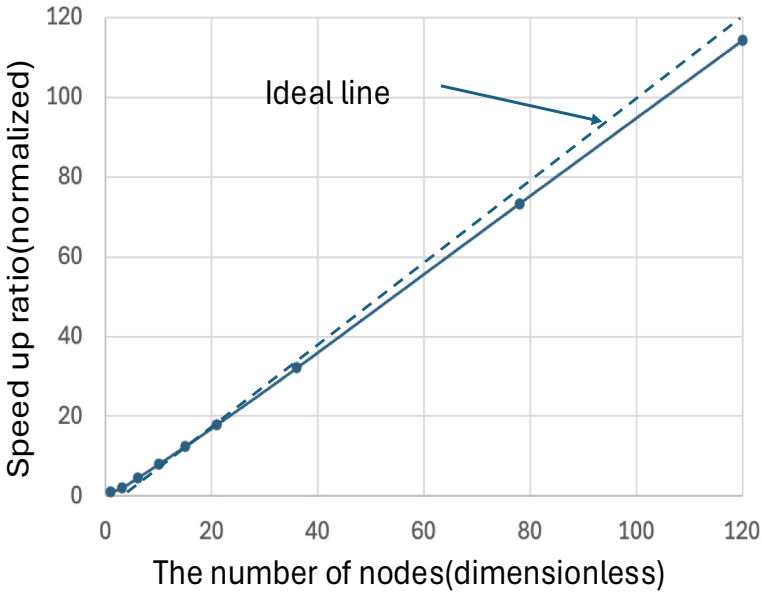


Fig. 3. Parallel speed-up ratio of the matrix-element calculation with MPI parallelization. The code achieves a 118-fold speed-up on 120 nodes (98.3% efficiency).

It is readily apparent that the calculated values are extremely close to the ideal line. The actual measurement yielded a 118-fold speed-up using 120 nodes, which corresponds to an exceptionally high parallel efficiency of approximately 98%. Given that an MPI operation is performed at the final stage to gather all matrix elements into the rank 0, the slight deviation from the ideal 120-fold speed-up is to be expected, considering the inter-node communication involved.

Figure 4 illustrates the speed-up ratio achieved through hybrid parallelization. The solid line represents the computed speed-up ratio, while the dashed line indicates the ideal linear scaling. In this benchmark, a total of 6,481,800 matrix-elements were calculated for Channel 1 for the same reasons as in the MPI implementation. The source code was compiled using the following command:

```
mpifrtpx -Kfast -Kparallel -SSL2 -mcmodel=large
```

In this command, `-Kparallel` is the automatic parallelization option.

The speed-up on 120 nodes reached 122.5-fold, slightly exceeding the ideal linear scaling. While this may appear counterintuitive at first glance, it indicates a so-called "super-linear speed-up" [28]. This phenomenon is likely attributable to improved cache efficiency and optimized memory bandwidth utilization during thread parallelization, as the problem size per node decreased. Specifically, at 120 nodes, the number of matrix-elements to be processed per node is reduced to 1/120 of the original workload, leading to a reduction in the memory occupancy

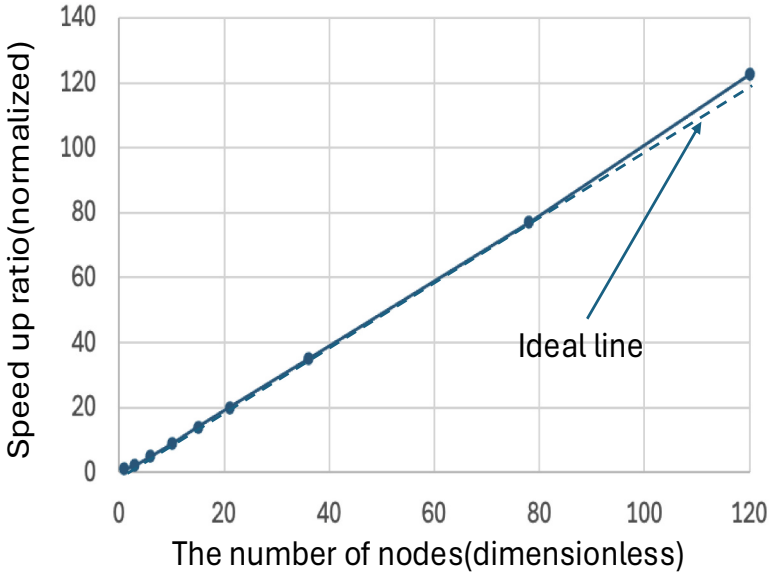


Fig. 4. Parallel speed-up ratio of the matrix-element calculation with Hybrid parallelization. The code achieves a 122.5-fold speed-up on 120 nodes (102% efficiency).

per node. Considering the cache specifications of the A64FX processor (8 MB L2 cache) [29], it is highly probable that the observed super-linear speed-up was driven by an improved cache hit rate. This resulted from partitioning the problem into smaller datasets that fit more efficiently within the cache hierarchy, as expected.

3.3 Computed Energies

In Table 2, the calculated energies and Gaussian range parameters for each channel are shown. The first column shows the channel numbers defined in Table 1. The second column lists the ground-state energies obtained from the calculation in units of Kelvin. The third column shows the energies of the first excited state. While the first excited state is of significant interest in relation to Efimov physics and Bose—Einstein condensation, this paper, as a computational science study, only presents the numerical results for three important channels; physical analysis along with the converged numerical results including other channels will be discussed elsewhere. The subsequent columns list the number of basis functions for each Jacobi coordinate, along with the minimum and maximum values of the Gaussian range parameters. The final column shows the dimension of the matrix that was actually diagonalized. It should be noted that the energy values are not the result of diagonalizing a single channel alone, but the values obtained when the basis functions of the new channel are added to all the basis functions from the preceding rows.

Table 2. Calculated energies and Gaussian range parameters for each channel.

No.	E_0	E_1	N_1	b_1^{min}	b_1^{max}	N_2	b_2^{min}	b_2^{max}	N_3	b_3^{min}	b_3^{max}	N_4	b_4^{min}	b_4^{max}	N	N_{tot}
1	-0.808	-0.440	12	0.5	20	12	0.5	20	12	0.5	20	18	0.7	200	31104	31104
6	-1.205	-0.555	10	0.5	20	10	0.5	20	10	0.5	20	12	0.5	100	12000	43104
11	-1.303	-0.556	8	0.5	20	8	0.5	20	8	0.5	20	10	0.5	40	5120	48224

The large number of basis functions in the first channel is due to its role as the primary channel for reproducing the tetramer system. Furthermore, as previously explained, since a large portion of the other channels is already covered due to the linear dependence of the Gaussian basis, a common practice in GEM is to selectively allocate basis functions to the early channels that are considered physically important. The second channel, Channel 6, represents ^{4}He atoms coupled in an H-type configuration. In the ^{4}He tetramer, this is known as one of the major channels that accelerates convergence when added to the K-type; thus, it is expected to be important for the five-body system as well. The following 3+2 channel physically represents a configuration of a trimer and a dimer, which is a channel that appears for the first time in a five-body system. Therefore, this channel is expected to be important for making the five-body system more deeply bound than the four-body system. Regarding the ground state, other five-body calculations using different interactions have been reported, notably $E = -1.295(14)$ K [30] and $E = -1.300(2)$ K [31]. Since these potentials are tuned to the two-body system, the energies similar to the present calculation are expected for the pentamer. Our result is $E = -1.303$ K, which is consistent with these previous studies.

4 Summary and Conclusions

In this study, we extended a computational code GEM based on the Gauss-Lobe technique—a well-established approach for high-precision few-body calculations—to enable its efficient utilization on modern supercomputers. In particular, our development focused on optimization specifically tailored for hybrid parallel computing environments of the supercomputer Fugaku. Performance benchmarks using the ^{4}He pentamer demonstrated that the modernized GEM code achieves excellent scalability. On 120 nodes, we observed a 122.5-fold speed-up, representing a "super-linear speed-up" that exceeds ideal linear scaling. This result underscores the effectiveness of our hybrid parallelization strategy.

Furthermore, we introduced a practical technique for adjusting the balance between MPI and thread parallelism without necessitating extensive modifications to the original source code. This approach not only ensures the portability and maintainability of the GEM code but also paves the way for efficient precision calculations of the six-body systems or larger. To facilitate a deeper understanding of quantum few-body systems, we plan to release the GEM code to the

public following the preparation of comprehensive documentation and manuals. We anticipate that this modernized GEM framework will serve as a powerful tool for exploring the intricate dynamics of complex few-body systems.

The ^{4}He few-body system is a fundamental quantum-mechanical many-body system that has attracted significant attention, particularly for its relevance to Efimov states and Bose—Einstein condensation. The modernized GEM code developed in this work was implemented using the ^{4}He pentamer as a benchmark, which is at the forefront of few-body physics and for which detailed analysis and understanding are highly anticipated. Consequently, this code is far more than a mere benchmark tool; it is a fully production-ready framework capable of meeting the rigorous demands of frontier research. A comprehensive discussion of the physical properties and detailed analytical results will be reported in a separate, dedicated publication in the near future. Furthermore, we will apply and analyze other five-body systems, such as ^{4}He$+n+n+n+n$ or ^{4}He$+p+n+n+n$ in nulei, which will lead to the future expansion of our methodology to the six-body systems or larger.

Acknowledgements. This work was supported by Japan Society for the Promotion of Science (JSPS) KAKENHI with Grant Number 25H01270 and by the ERATO TOMOE project with Grant Number JPMJER 2304. We use Supercomputer Fugaku at RIKEN Center for Computational Science (HPCI project ID: hp250328). A part of the development of the GEM code is carried out on a supercomputer Miyabi, provided by the Multidisciplinary Cooperative Research Program in the Center for Computational Sciences, University of Tsukuba.

References

1. Efimov, V.: Phys. Lett. B **33**, 563 (1970)
2. Einstein, A.: Sitzungsber. Preuss. Akad. Wiss, 261 (1924)
3. Anderson, M.H., Ensher, J.R., Matthews, M.R., Wieman, C.E., Cornell, E.A.: Science **269**, 198 (1995)
4. Funaki, Y., Tohsaki, A., Horiuchi, H., Schuck, P., Röpke, G.: Phys. Rev. Lett. **91**, 262502 (2003)
5. Kamimura, M.: Phys. Rev. A **38**, 621 (1988)
6. Kameyama, H., Kamimura, M., Fukushima, Y.: Phys. Rev. C **40**, 974 (1989)
7. Hiyama, E., Kino, Y., Kamimura, M.: Prog. Part. Nucl. Phys. **51**, 223 (2003)
8. O. A. Yakubovsky, Sov. J. Nucl. Phys. **5**, 937 (1967) [Yad. Fiz. **5**, pp.1312 (1967)]
9. Kamada, H., Nogga, A., Glöckle, W., Vary, J.P., Barrett, B.R.: Phys. Rev. C **64**, 044001 (2001)
10. Kievsky, A., Rosati, S., Viviani, M., Marcucci, L.E., Girlanda, L.: J. Phys. G: Nucl. Part. Phys. **35**, 063101 (2008)
11. Navrátil, P., Vary, J.P., Barrett, B.R.: Phys. Rev. Lett. **84**, 5728 (2000)
12. Barrett, B.R., Navrátil, P., Vary, J.P.: Prog. Part. Nucl. Phys. **69**, 131 (2013)
13. Carlson, J., et al.: Rev. Mod. Phys. **87**, 1067 (2015)
14. Hiyama, E.: Few-Body Syst. **53**, 189 (2012)
15. Hiyama, E.: Prog. Theor. Exp. Phys. 01A204 (2012)
16. Varga, K., Suzuki, Y., Lovas, R.G.: Nucl. Phys. A **571**, 447 (1994)

17. Mitroy, J.: Rev. Mod. Phys. **85**, 693 (2013)
18. Aoyama, S., Arai, K., Suzuki, Y., Descouvemont, P., Baye, D.: Few-Body Syst. **52**, 97 (2012)
19. Aoyama, S., Myo, T., Katō, K., Ikeda, K.: Prog. Theor. Phys. **116**, 1 (2006)
20. Aoyama, S., Katō, K., Ikeda, K.: Prog. Theor. Phys. Suppl. **142**, 35 (2001)
21. Suzuki, Y., Varga, K., Usukura, J.: Nucl. Phys. A **631**, 91 (1998)
22. Suzuki, Y., Varga, K.: Stochastic variational approach to quantum-mechanical few-body problems (Lecture notes in physics, Vol. 54), Springer, Berlin (1998)
23. Kukulin, V.I., Krasnopol'sky, V.M.: J. Phys. G: Nucl. Phys. **3**, 79 (1977)
24. Aoyama, S., Myo, T., Yoshida, D.: Computational Science and Its Applications – CSHPC2025 (ICCSA 2025 Workshops), Proceedings, Lecture Notes in Computer Science, Vol. 15890, Springer, pp. 42(2026)
25. Yoshida, D., Hayama, E., Imamura, T., Kanamori, I., Matsufuru, H.: SCA/HPCAsiaWS '26 : Proceedings of the Supercomputing Asia and International Conference on High Performance Computing in Asia Pacific Region Workshops, pp. 133 (2026)
26. Hiyama, E., Kamimura, M.: Front. Phys. **13**(6), 132106 (2018)
27. Naidon, P.: (RIKEN Nishina Centre). private communication
28. Ristov, S., Prodan, R., Gusev, M., Skala, K.: Proceedings of the Federated Conference on Computer Science and Information Systems, p. 889 (2016)
29. Sato, M.: Proceedings of the SC20, No.47, p. 1 (2020)
30. Blume, D., Greene, C.H.: J. Chem. Phys. **112**, 8053 (2000)
31. Bazak, B., Valiente, M., Barnea, N.: Phys. Rev. A **101**, 010501(R) (2020)