16.447 TFlops and 159-Billion-dimensional
Exact-diagonalization for Trapped Fermion-Hubbard Model
on the Earth Simulator

Susumu Yamada
Japan Atomic Energy
Research Institute
6-9-3 Higashi-Ueno, Taito-ku
Tokyo, 110-0015, Japan
yamada@koma.jaeri.go.jp

Toshiyuki Imamura
The University of
Electro-Communications
1-5-1 Chofugaoka, Chofu-shi
Tokyo, 182-8585, Japan
imamura@im.uec.ac.jp

Masahiko Machida∗
Japan Atomic Energy
Research Institute
6-9-3 Higashi-Ueno, Taito-ku
Tokyo, 110-0015, Japan
mac@koma.jaeri.go.jp

ABSTRACT

In order to study a possibility of superfluidity in trapped atomic Fermi gases loaded on optical lattices, we implement an exact diagonalization code for the trapped Hubbard model on the Earth Simulator. Comparing two diagonalization algorithms, we find that the performance of the preconditioned conjugate gradient (PCG) method is 1.5 times superior to the conventional Lanczos one since the PCG method can conceal the communication overhead much more efficiently. Consequently, the PCG method shows 16.447 TFlops (50.2% of the peak) on 512 nodes. On the other hand, we succeed in solving a 159-billion-dimensional matrix by using the conventional Lanczos method. To our knowledge, this dimension is a world-record. Numerical results reveal that an unconventional type of superfluidity specific to the confined system develops under repulsive interaction.

1. INTRODUCTION

Since the experimental success [1, 2, 3] of the Bose-Einstein condensation in the trapped atomic Bose gas honored by the Nobel Prize in 2001, the research streamline in atomic physics has been directed toward another difficult condensation, that is, superfluidity of the atomic Fermi gas [4, 5]. The condensation in fermion system is one of the most universal issues in fundamental physics, since particles which form matter, i.e., electron, proton, neutron, quark, and so on, are all fermions. Motivated by interests based on such a wide background, we numerically explore a possibility of superfluidity in the atomic Fermi gas [6]. Our undertaking model is the fermion-Hubbard model [7] with trapping potential. The Hubbard model describes a many-body fermion system on a discrete lattice, which can be realized by a standing wave created due to two laser interference [8] in the atomic Fermi gas.

The Hubbard model is one of the most intensively-studied models by computers because it owns very rich physics although the model expression is quite simple [7]. The Hamiltonian of the Hubbard model with a trap potential [6, 9] is given as

\[
H = -t \sum_{i,j,\alpha} (a_{j,\alpha}^\dagger a_{i,\alpha} + H.C.) + U \sum_i n_{i,\uparrow} n_{i,\downarrow} + \left( \frac{2}{N} \right)^2 V \sum_{i,\sigma} n_{i,\sigma} \left( i - \frac{N}{2} \right)^2 ,
\]

where \(t\), \(U\), and \(V\) are the hopping parameter from \(i\)-th to \(j\)-th sites (normally \(j\) is the nearest neighbor site of \(i\)), the repulsive energy for on-site double occupation of two fermions, and the parameter characterizing the strength of the trapping potential, respectively, as schematically shown in Fig. 1, and \(a_{i,\sigma}, a_{i,\sigma}^\dagger\) and \(n_{i,\sigma}\) are the annihilation, the creation, and the number operator of a fermion with pseudospin \(\sigma (=\uparrow\text{ (up)}\text{ or } \downarrow\text{ (down)})\) on the \(i\)-th site, respectively.

The computational finite-size approaches on the Hubbard model are roughly classified into two types. The first one is the exact diagonalization using the Lanczos method [10], and the second one is the quantum Monte Carlo [7]. The former directly calculates the ground and the low lying excited states of the model, and moreover, obtains various physical quantities with considerably high accuracy. However, the numbers of fermions and sites are severely limited because the matrix size of the Hubbard Hamiltonian approximately grows exponentially with increasing these numbers. See Appendix A for how to make a matrix and Table 3 for an example about how the matrix size depends on the numbers. On the other hand, the latter has an advantage in these numbers, but confronts a fatal problem as the negative sign in the probability calculation [7]. Therefore, its reliability has been always argued. From these contexts, as long as computational resources are infinitely permitted, the exact diagonalization is clearly found to be the best way. Thus, one can raise a challenging theme for supercomputing, that is, to implement the exact diagonalization code on the present top-class supercomputer, i.e., the Earth Simulator [11], and to examine how large matrices can be solved and how excellent performance can be obtained.

∗Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SC'05 November 12-18, 2005, Seattle, Washington, USA
(c) 2005 ACM 1-59593-061-2/05/0011...$5.00.
The Earth Simulator developed by NASDA (presently JAXA), JAERI, and JAMSTEC, is situated on the flag-ship class of highly parallel vector supercomputer of the distributed-memory type. The theoretical peak performance is 40.96 TFlops, and the total memory size is 10 TB. The architecture of the Earth Simulator is quite suitable for scientific and technological computation [11], due to well-balance of the processing speed of the floating point operation and the memory bandwidth as well as the network throughput. Therefore, several applications achieved excellent performance, and some of them won the Gordon Bell Prize at the Supercomputing conferences [12, 13, 14, 15, 16].

In this paper, we develop a new type of high performance application which solves the eigenvalue problem of the Hubbard Hamiltonian matrix (1) on the Earth Simulator and present progress in numerical algorithm and parallelization technique to obtain the best performance and solve the world-record class of large matrices. In the algorithmic issue, we suggest a new profitable algorithm, i.e., the preconditioned conjugate gradient (PCG) method\(^\text{1}\) as an alternative one for the huge-scale matrix diagonalization. Comparing between the PCG method and the conventional Lanczos method, we find that the PCG method is much more excellent than the Lanczos method except for the memory usage. On the other hand, in the parallelization technique issue, we investigate a hybrid parallelization which combines the inter-node parallelization using MPI for distributed memory with the intra-node parallelization using the automatic parallelization for shared memory in addition to the vectorization. Consequently, we find a practical technique to save the memory. Such a technique is crucial when efficient memory use is severely required.

The contents of this paper are as follows. In Section 2, we introduce two eigenvalue solvers to diagonalize the Hamiltonian matrix of the Hubbard model and compare their convergence properties. Section 3 presents the parallelization technique of two solvers on the Earth Simulator, and especially focus on a specific technique to save the memory. Section 4 shows actual performance in large-scale matrix diagonalizations on the Earth Simulator. Finally, we present typical simulation results which show a possibility of an unconventional superfluidity in the Hubbard model with the trap potential in Section 5.

2. NUMERICAL ALGORITHMS

The core of our program calculates the smallest eigenvalue and the corresponding eigenvector for \(Hv = \lambda v\), where Hamiltonian matrix \(H\) is sparse, real, and symmetric. Thus, several iterative numerical algorithms, i.e., the power method, the Lanczos method, the conjugate gradient (CG) method, and so on, are applicable. As a first step of this research, we explore the most effective algorithm on the Earth Simulator before actual large-scale simulations, since the Earth Simulator is a public computer and a trial use of hundreds of nodes is limited. Moreover, the regulation of the Earth Simulator requests any users to tune their applications up to at least 95% and 50% in term of the vectorization ratio and the parallelization efficiency, respectively, and to precisely estimate resource demand as memory usage, disk storage, and CPU time in advance of the submission of large-scale job. In this section, we concentrate on two numerical algorithms, the Lanczos method and the CG method, in terms of the memory use and the performance on the Earth Simulator.

2.1 Lanczos Method

The Lanczos method is one of the subspace projection methods that creates a Krylov sequence and expands invariant subspace based on the procedure of the Lanczos principle [17] (see Fig. 2(a)). Eigenvalues of the projected invariant subspace well approximate those of the original matrix, and the subspace can be represented by a compact tridiagonal matrix.

The main recurrence part of this algorithm repeats to update the Lanczos vector \(v_{i+1}\) from \(v_{i-1}\) and \(v_{i}\) as seen in Fig. 2(a). Therefore, the memory requirement of the recursion is \(2N\) words. In addition, an \(N\)-word buffer is required for storing an eigenvector. Consequently, the total memory requirement of the Lanczos method is \(3N\) words.

As the algorithm is shown in Fig. 2(a), the main loop iterates until \(m\) or until the coefficient \(\beta_i < \epsilon\). Thus, the upper limit of the iterations is formally \(m\). Some reports point out that \(m\) should be set \(2\sqrt{N}\) ordinarily. However, according to several preliminary tests, \(2\sqrt{N}\) is too large to approximate the maximum eigenvalue in our case. In the following, we choose much smaller empirical number than \(2\sqrt{N}\), i.e., 200 or 300, as an iteration count\(^2\).

2.2 Conjugate Gradient Method

As another projection method exploring invariant subspace, the conjugate gradient (CG) method is a popular algorithm, which is frequently used for solving linear systems in large-scale simulations. It is also applicable to the\(^3\)

---

\(^1\)Calculating the smallest eigenvalue of the matrix \(H\) is equivalent to the minimization of the Rayleigh quotient, and the minimization is executed by using the PCG method.

\(^2\)In order to obtain an eigenvalue with its eigenvector, we need to execute the Lanczos recursion twice (see Appendix B). Therefore, in the following results, the number of iterations \(m\) is defined as that of the first Lanczos recursion.
Figure 2: Algorithms of two eigenvalue solvers.

Table 1: A performance test of two algorithms on 10 nodes of the Earth Simulator for a problem of 12 fermions on 20 sites.

3The receive-buffers for rowwisely partitioned $V^T$ and columnwisely partitioned $W_2^T$.
trix due to the hopping between neighboring sites, and the diagonal matrix originated from the presence of the on-site repulsion, respectively. We note that in large-scale problems the non-zero elements of large-matrices $I \otimes A$ and $A \otimes I$ cannot be expand on the whole memory, while $A$ and $D$ can be stored on the memory of each node.

In the core operation $Hv$, the matrix-vector multiplications are transformed into the matrix-matrix multiplications as

$$Hv \mapsto \begin{cases} \bar{D}v & \iff (I \otimes A)v \mapsto AV \\ (A \otimes I)v & \iff VA^T \end{cases} \quad (3)$$

where the matrix $V$ is obtained from the vector $v$ by the following procedure. First, decompose the vector $v$ into $n$ blocks, and order in the two-dimensional manner as follows,

$$v = \begin{pmatrix} v_{1,1}, & v_{2,1}, & \ldots, & v_{n,1}, & v_{1,2}, & v_{2,2}, & \ldots, & v_{n,2}, & \ldots, & v_{1,n}, & v_{2,n}, & \ldots, & v_{n,n} \end{pmatrix}^T.$$ 

Here, a pair of subscripts of each element $v$ formally indicates a position of row and column of the matrix $V$. The $k$-th element of the matrix $D, d_k$, is also mapped onto the matrix $\bar{D}$ in the same manner, and the operator $\circ$ means an elementwise multiplication.

### 3.2 Data Distribution, Parallel Calculation, and Communication

The matrix $A$, which represents the site hopping of up (or down) pseudo-spin fermions, is a sparse matrix, and its sparsity is high. In contrast, the matrices $V$ and $\bar{D}$ must be treated as dense matrices, and their dimension is large enough to overfull the memory capacity of a single node. Therefore, while all non-zero elements of the matrix $A$ are stored on all the nodes in the compressed row storage format, the matrices $V$ and $\bar{D}$ are columnwisely partitioned. Moreover, the rowwisely partitioned $V$ must be also stored on each node for parallel computing of $VA^T$.

The core operation $Hv$ including the data communication can be written down as follows:

- **CAL1:** $E^c = \bar{D}^c \circ V^c$,
- **CAL2:** $W_1^c = E^c + AV^c$,
- **COM1:** communication to transpose $V^c$ into $V^r$,
- **CAL3:** $W_2^r = V^r A^T$,
- **COM2:** communication to transpose $W_2^r$ into $W_2^T$,
- **CAL4:** $W^c = W_1^T + W_2^T$,

where the superscripts $c$ and $r$ denote columnwise and rowwise partitioning, respectively. The above operation procedure twice includes the matrix transpose which normally requires all-to-all data communication. In the MPI standards, the all-to-all data communication is realized by a collective communication MPI_Alltoallv. However, due to irregular and incontiguous structure of the transferring data, the data-communication should be executed by a point-to-point or a one-side communication function. On the Earth Simulator, since the one-side communication function MPI_Put more excellently runs than the point-to-point communication, MPI_Put is recommended by the developers [19].

In the procedure of the matrix-matrix multiplication, the calculations CAL1 and CAL2 and the communication COM1 is clearly found to be independently executed. Moreover, although the relation between CAL3 and COM2 is not so simple, the overlap can be realized in a pipelining fashion as shown in Fig. 3. Thus, the two communication processes (COM1 and COM2) are principally hidden behind the calculations.

### 3.3 Communication Overlap on the Earth Simulator

The MPI standard formally guarantees simultaneous execution of computation and communication when using the non-blocking point-to-point communications and the one-side communications. This principally enables to hide the communication time behind the computation time, and improve the performance. However, the overlap between communication and computation practically depends on an implementation of the MPI library. In fact, the MPI library installed on the Earth Simulator had not provided any functions of the overlap until the end of March 2005, and the non-blocking MPI_Put had worked as a blocking communication like MPI_SEND, in which, the completion of the data transfer is verified without any synchronization calls. Since April 2005, the MPI library on the Earth Simulator has begun to support the overlap function, and moreover, serve a compiler option to activate the overlap function or not in MPI_Put*. In Section 3.4, we present a technique to save the memory by using the blocking character in MPI_Put on the Earth Simulator.

*Presently (in July 2005), the overlap function is not supported.
3.4 A Technique to Save Memory

As mentioned in Section 3.3, MPI_Put installed on the Earth Simulator can work as the blocking communication as well as the non-blocking one. Using MPI_Put in non-blocking mode, call of MPI_Win_Fence to synchronize all processes is required in each pipeline stage. Otherwise, two N-word communication buffers (send- and receive-buffers) should be retained until the completion of all the stages. On the other hand, the completion of each stage is assured by return of the MPI_Put in the blocking mode, and send-buffer can be repeatedly re-used. Consequently, one N-word communication buffer (the send-buffer W^s) becomes free in the blocking mode. Thus, we can adopt the blocking MPI_Put to extend the maximum limit of the matrix size. At a glance, this choice seems to sacrifice the overlap functionality of the MPI library. However, one can manage to overlap computation with communication even in the use of the blocking MPI_Put on the Earth Simulator. The way is as follows. The blocking MPI_Put can be assigned to a single PE per node by the intra-node parallelization technique. Then, the assigned processor dedicates only the communication (see Fig. 4). As a result, the calculation load is divided into seven PE’s. The intra-node parallelism is described by the automatic parallelization using CDIR compiler directives, and the eight tasks run in parallel. This parallelization method, which we call task assignment (TA) method, imitates a non-blocking communication operation, and enables to overlap between the blocking communication and the calculation on the Earth Simulator (see Appendix D for more details). Here, we note that although the TA method just loses the floating-operation potential of a single processor, the total elapsed time does not almost change compared to a case using the non-blocking communication if the communication time is larger than 8/7 times of the calculation time. Fortunately, our program just satisfies such a condition. A performance comparison between the blocking MPI_Put with the TA method and the non-blocking MPI_Put is shown in Table 2. This result clearly shows that the elapsed times of both methods are almost comparable as we expect.

3.5 Effective Usage of Vector Pipelines

The theoretical Flops rate in the single processor of the Earth Simulator is calculated by

$$F = \frac{4(ADD + MUL)}{\max(ADD, MUL, VLD, VST)} \text{ GFlops},$$  \hspace{1cm} (4)

where ADD, MUL, VLD, VST denote the number of additions, multiplications, and vector load and store operations, respectively. According to the formula (4), the performance of the matrix multiplications AV and VA^T shown in Fig. 5 is normally 2.67 GFlops. However, DO-loop unrolling decreases the number of VLD and VST instructions and improves the performance. In fact, when the loop unrolling factor is 12 in the multiplication, the performance is estimated to be 6.86 GFlops (see Fig. 5 (c)). Moreover, the

![Figure 4: A schematic figure for the task division.](image)

![Figure 5: (a), (b) Programs and (c) Flops estimation for the multiplications AV and VA^T. Here, n is the dimension size of the matrix A. See text for VLD, VST, MUL, and ADD.](image)

![Table 2: A comparison of total elapsed time between the blocking MPI_Put with the TA method and the non-blocking MPI_Put. The calculation problem is the same as Model 1 in Table 3.](table)
The optical lattice [8] is a standing wave created by the optical trapping potential. We diagonalize the Hubbard Hamiltonian \( H \) including the trapping potential (1) and calculate the binding energy, which is a probe for superfluidity, with an optical lattice [8]. We find a condition

\[
5 \times ndim + 16 \times itr \times ndim + 2 \times nnz \times ndim \quad \text{for the Lanczos method},
\]

\[
35 \times ndim + 46 \times itr \times ndim + (itr + 2) \times (2 \times nnz - ndim) \quad \text{for the PCG method},
\]

where \( ndim \), \( itr \) and \( nnz \) are the dimension of the Hamiltonian matrix \( H \), the number of iterations, and the number of the non-zero elements of \( H \), respectively.

As shown in Table 4, the PCG method shows better convergence property and solves the eigenvalue problems about 3 times faster than the Lanczos method. Moreover, the PCG method overlaps communication tasks with calculations more than the Lanczos method since the communication is efficiently hidden in a routine calculating inner products intrinsic to the PCG method. The best performance of the PCG method is 16.447 TFlops on 512 nodes which is 50.2\% of the theoretical peak. On the other hand, Table 3 and 4 show that the Lanczos method can solve up to the 159-billion-dimensional Hamiltonian matrix on 512 nodes. To our knowledge, this size is the largest in history of the exact diagonalization method of Hamiltonian matrices.

### 5. NUMERICAL RESULTS

In this section, we numerically study the repulsive Hubbard model with the trapping potential [6, 9] to examine a possibility of superfluidity of an atomic Fermi gas loaded on an optical lattice [8]. We diagonalize the Hubbard Hamiltonian \( H \) including the trapping potential (1) and calculate an binding energy, which is a probe for superfluidity, with varying the repulsive \( U(> 0) \) and \( V \). We find a condition in which Cooper pair (superfluid) function develops based on the binding energy [6], and discuss the significance of the numerical results.

The optical lattice [8] is a standing wave created by the
interference effect of two laser beams. It gives a periodical lattice potential for atoms and enables to experimentally study the Hubbard model [7] showing strongly correlated lattice potential for atoms and enables to experimentally study the mechanism of superconductivity originating from the Coulomb repulsion. However, it is still unclear whether or not the repulsive interaction really leads to a pairing interaction which can bring about high-T_c superconductivity exceeding T_c ∼ 150K. On the other hand, in the atomic Fermi gas [4, 5] loaded on the optical lattice, various physical parameters, such as the strength of hopping, the interaction, and the particle density, are systematically tunable. Thus, the research for the Cooper pairing via a repulsive interaction in the presence of the optical lattice will be a next big experimental challenge in the field of the trapped atomic gas [22].

Since the atomic Fermi gas is trapped inside harmonic-well type of potential as schematically seen in Fig. 1, one should actually include the effect of the trap potential. In the presence of the trap, atoms tends to form a cluster in the center of the trap, which naturally leads to a dome-like density profile around the trap center [9] in relatively small U/t regime as seen in Fig. 6(a). On the other hand, since the presence of the on-site repulsion excludes the double occupancy of atoms, and consequently, a flat density profile with one atom per one site becomes favorable in a large on-site-repulsion regime as seen in Fig. 6(b). The flat region is called Mott core below. Thus, since the trap potential and the repulsive interaction have opposite effects as explained above, their interplay is an interesting problem. See Fig. 6(c) for the entire shape change of the particle density profile as a function of U/t. In this paper, we show a possibility of fermion superfluidity associated with the opposite effects. We expect that a superfluid phase emerges when the Mott core is formed as seen in Fig. 6(b). In such a situation, the compressibility in the atom density is almost zero in the center of trap [9], whereas it shows a finite value around the edges of the core [9]. This makes it possible to fluctuate the atom number only around the Mott core edges and leads to a singlet Cooper pairing via the Mott core [6] at the core edges.

In order to confirm the above scenario about the superfluidity, we perform the exact diagonalization on the repulsive Hubbard model with the trapping potential. First, we fix the trapping strength, i.e., V/t = 29.7 and examine how the binding energy of two Fermions E_b [10] changes with varying U/t. The binding energy of two Fermi atoms is given by

$$E_b \equiv E_g(n + 1, n + 1) + E_g(n, n) - 2E_g(n, n + 1).$$

Here, E_g(n, n) is the ground state energy in the case of N_F = n \uparrow + n \downarrow fermions, which is evaluated by the exact diagonalization scheme using both the conventional Lanczos

Figure 6: Particle density profile (a) U/t = 4, (b) U/t = 12, and (c) 0 ≤ U/t ≤ 15 for 12 fermions (6 \uparrow, 6 \downarrow) systems in 24-site Hubbard model with the trapped potential (V/t = 29.7).

Figure 7: U/t dependences of the binding energy E_b of two fermions for 4 fermions (2 \uparrow, 2 \downarrow), 6 fermions (3 \uparrow, 3 \downarrow), 8 fermions (4 \uparrow, 4 \downarrow), 10 fermions (5 \uparrow, 5 \downarrow), and 12 fermions (6 \uparrow, 6 \downarrow) in 24-site Hubbard model with the trapped potential (V/t = 29.7).
method [10] or the new algorithm (the PCG method). We mainly use the PCG method for systematic calculations because of its fast turn-round. If $E_b$ is negative, it then means that an attractive interaction works between two atoms and a Cooper pairing correlation develops. Therefore, the negative $E_b$ is a probe for superfluidity. In the numerical experiments, the number of site $N$ is taken 24, the range of $i$ is from 1 to 24, and the number of fermions $N_F$ varies from $5(2↑,2↓)$ to $14(7↑,7↓)$. In the case of $N_F=14$, we use the Lanczos method since the case is executable only by using the Lanczos method with the technique for the memory save described in Section 3.4. Fig. 7 shows $U/t$ dependences of $E_b$ from $N_F=4$ to 12. Here, we note that $E_b$ of $N_F=12$ requires $E_b$ of $N_F=14$ as expressed in (5). It is found that $E_b$ goes to negative in a large $U/t$ region and the amplitude of the negative $E_b$ slightly increases with increasing $N_F$ up to $N_F=10$. Here, we note that the negative $E_b$ emerges only above a certain $V$. If $V$ is very small, then the system just behaves as the bulk one-dimensional system which shows not superfluidity but anomalous metallic features [9]. Thus, it is found that both the trapping potential and the repulsive force contribute to the negative $E_b$.

Finally, let us discuss the significance of the numerical results. The amplitude value of negative $E_b$ depends on the trapping potential. The maximum amplitude is found to be $|E_b|\sim 0.05t$ at $U/t \sim 8$ as seen in Fig. 7. Depending on the trapping potential, this value exceeds the binding energy scale in High-$T_c$ cuprate superconductors, i.e., the superconductivity with $T_c \sim 150K$. This result clearly indicates that it is possible to lift up or down the superfluid transition temperature by controlling the confinement effect. These surprising results can be easily confirmed by loading the atomic Fermi gas on the optical lattice, while it means that one can freely control the superconducting transition temperature $T_c$ if we can realize a potential like the present harmonic trap inside real solid-state matters.

6. CONCLUSIONS

The best performance (16.447TFlops) of the present application is comparable to those of other applications on the Earth Simulator as reported in SC2002[12, 13, 14], SC2003[15] and SC2004[16]. However, we would like to point out that our application requires a huge amount of communications in contrast to the previous ones. Therefore, we made much effort to hide the communication by paying an attention to the architecture of the Earth Simulator. As a result, we clarified that the new algorithm (PCG) shows the best performance and drastically shortens the total elapsed time. This is quite useful for systematic calculations like the present problem which explores a condition that an unconventional type of superfluidity appears. Furthermore, we found the technique combining inter- and intra-node parallelisms to extend the maximum dimension of the matrix without any significant performance damages. In conclusion, we obtain the best performance by the new algorithm and the world record of the large matrix operation by the technical advance. We believe that these results are new outstanding achievements in high performance computing.

Acknowledgements

The authors in CCSE JAERI thank G.Yagawa, T.Hirayama, C.Arikawa, N.Inoue and T.Kano for their supports and acknowledges K.Itakura and all staff members in the Earth Simulator for their supports in the present calculations. One of the authors (M.M.) acknowledges T.Egami and P.Piekarcz for illuminating discussion about diagonalization for d-p model, H.Matsumoto and Y.Ohashi for their collaboration on the optical-lattice fermion systems, and T.Ishida and Y.Morita for their financial support. Two of authors (S.Y and M.M.) thank Y.Oyanagi and T.Hotta for their illuminating discussion in the diagonalization technique.

7. REFERENCES

APPENDIX

Appendix A: How to Make a Matrix

Let us briefly show how to make a matrix in the exact diagonalization method. The state depicted in Fig. 1 is described as $A \equiv |001011 \rangle \langle 001011| \geq 1 |11000 \rangle \langle 11000|$. All possible configurations are taken into consideration, and the matrix elements are created as the expectation values of the Hamiltonian, e.g., $<A|H|A>$ is a diagonal component, and $<A|H|B>$, where $|B>$ is a different state, is a non-diagonal one. Here, we note that since the Hamiltonian (1) breaks translational symmetry due to the presence of the trap potential, we can not do any matrix size reductions using the symmetry. In terms of the fermion configuration, there are following rules. The numbers of fermions and sites are fixed. Two fermions with the same pseudo-spin can not occupy a site. This means that the occupation by fermions with different pseudo-spins on a site becomes the maximum occupation per site. The Pauli principle gives these rules.

Table 5: Comparison among three preconditioners.

<table>
<thead>
<tr>
<th># Iteration</th>
<th>Residual Error</th>
<th>Elapsed Time (sec)</th>
<th>FLOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0) NP</td>
<td>268</td>
<td>1.44E-09</td>
<td>78.904</td>
</tr>
<tr>
<td>1) PJ</td>
<td>133</td>
<td>1.404E-09</td>
<td>40.785</td>
</tr>
<tr>
<td>2) ZS-PJ</td>
<td>91</td>
<td>1.255E-09</td>
<td>28.205</td>
</tr>
</tbody>
</table>

The test eigenvalue problem is the same as the one in Section 2.3.

Appendix B: Eigenvector of the Lanczos Method

In the Lanczos method, the eigenvector $z$ is computed by

$$z \leftarrow z + y_{k+1}v_k \quad (k = 0, 1, \ldots, m - 1),$$

where $y = (y_1, y_2, \ldots, y_m)^T$ is the eigenvector of the tridiagonal matrix and $v_k$ is the $k$-th Lanczos vector. Due to the memory limitation, only two Lanczos vectors are stored and updated by each Lanczos recursion. Thus, the past Lanczos vectors $v_k$ are lost. Therefore, we must execute the Lanczos recursion twice to obtain the eigenvector.

Appendix C: Choice of Preconditioner

It is well-known that the preconditioning improves convergence of the CG method. However, it is generally hard to predict an effective preconditioning before actual test calculation. In this section, we focus on the following five preconditioners:

1. Point Jacobi,
2. Zero-shift point Jacobi,
3. Block Jacobi,
4. Neumann-polynomial expansion,
5. SSOR-type iteration.

Here, 1, 3, 4, and 5 are very popular preconditioners for the CG method and 2 (zero-shift point Jacobi) is a modified version of 1 (point Jacobi). The zero-shift point Jacobi is a diagonal scaling preconditioner shifted by $\mu_k$ to amplify the ground-state eigenvector, i.e., the preconditioning matrix is given by $T = (D - \mu_k I)^{-1}$, where $D$, $I$, and $\mu_k$ are the diagonal part of the matrix $H$, the identity matrix, and an approximate of the smallest eigenvalue which appears in the PCG iteration, respectively. In order to solve a huge matrix and achieve higher performance, we select 1 and 2, since they do not require any data communication and any extra storage.

Now, let us show a result of the preconditioner test, in which we solve the same eigenvalue problem as Section 2.3 to compare 1 and 2. Table 5 summarizes a performance test of three cases, 0) without preconditioner (NP), 1) point Jacobi (PJ), and 2) zero-shift point Jacobi (ZS-PJ) on the Earth Simulator, and Fig. 8 illustrates these convergence properties. These results clearly reveal that the zero-shift point Jacobi is the best preconditioner.

Appendix D: Overlap between Blocking Communication and Calculation

In order to examine the possibility of overlap between blocking communication and calculation when utilizing TA method, we perform the multiplication $Hv$ on 10 nodes (80...
Figure 8: Convergence properties of three preconditioners.

Figure 9: A comparison of the process timechart on node No.0 for the multiplication $Hv$ between the blocking communication with the TA method and non-assignment method (NA).

PE’s) of the Earth Simulator. A test matrix $H$ is a 1.5-billion-dimensional matrix derived from the one-dimensional 20-site Hubbard model with 12 fermions ($6 \uparrow, 6 \downarrow$). We measure the elapsed time of the four calculation steps $\text{CAL1-4}$ and the two communications $\text{COM1-2}$ shown in Section 3.2. We show the timecharts of TA and non-assignment (NA) methods in Fig. 9. As shown in the figure, the calculations and the communications are executed serially in NA method, while the calculation and the communication in TA method are executed simultaneously.