Numerical Methods for Large-scale Quantum Material Simulations

Hiroto Imachi
Tottori University

January 2017

Contents

Chapter 1

Introduction

Nowadays, quantum material simulations, electronic state calculations, form a rigorous theoretical foundation of material science and engineering [1]. These simulations treat each electron as a quantum 'wave'. Several simulation softwares, like Gaussian [2] and VASP [3], are the de facto standards and are used widely among experimental and industrial researchers, as well as theoretical ones. Small system, such as benzene molecule (C_6H_6), can be calculated even by a laptop computer. These simulations, however, are not applicable to large-scale systems, for example, one-hundred-nanometer-scale systems, owing to heavy computational costs and a strong need appears for numerical methods in such large-system calculations. On the other hand, the current and next-generation supercomputers work as massively parallel computing and the numerical methods for these machines are crucial issues in computational science.

The present thesis is devoted to numerical methods for large-scale quantum material simulation, so as to satisfy the need, in particular, for industrial application [4, 5, 6]. We focused on fundamental numerical problems in quantum material simulations and these problems appear the Schrödinger equations in the time-independent or time-dependent forms. The thesis consists of three main issues; hybrid parallel generalized eigenvalue problem solver, quantum wavepacket dynamics simulation solver, and extreme parallelism on the full system of the K computer for an order-N electronic structure calculation code (N is the number of atoms in a system). The three parts are relevant as follows; an order-N electronic structure calculation code ELSES(=Extra Large Scale Electronic Structure calculations) [7] enables one-hundred-nm-scale quantum molecular dynamics (MD) simulations. Electronic structure can be calculated via generalized shifted linear equations or generalized eigenvalue equations. The order-N calculation is achieved when generalized shifted linear equations are solved. Generalized eigenvalue problem solver gives more reliable numerical result with larger ($O(N^3)$) computational costs. The order-N solver and the hybrid parallel general-

ized eigenvalue problem solver are complement to each other. They are massively parallel numerical solvers and show high scalability up to the full system of the K computer, one of the fastest supercomputers in the world. Based on the quantum MD results, quantum wavepacket dynamics simulations are executed to calculate transport properties.

At first the background of the thesis such as overview of parallel computing and organic materials is explained. Organic materials play a crucial role among next-generation IoT (Internet of Things) products, such as display, battery and sensor, since they form flexible atomic structures and enable ultra-thin, light, flexible (wearable) devices with a low fabrication cost. Because disorder (or randomness) in atomic structures is important for properties of organic materials, large-scale (100nanometer or 10^8 atom scale) quantum material simulation methods are required. Efficient parallel computation is a key technique for realization of such large-scale quantum material simulations. Algorithms and implementations must be carefully chosen depending on a required physical quantity and parallel computer architecture.

Hybrid parallel generalized eigenvalue problem solver Optimal hybrid numerical solvers are constructed for massively parallel generalized eigenvalue problem (GEP). The strong scaling benchmark was carried out on the K computer and other supercomputers for electronic structure calculation problems in the matrix sizes of $M = 10^4 - 10^6$ with up to 10^5 processor cores. The procedure of GEP is decomposed into the two subprocedures of the reducer from the GEP to the standard eigenvalue problem (SEP) and the solver of SEP. A hybrid solver is constructed, when a routine is chosen for each subprocedure from the three parallel solver libraries of ScaLAPACK, ELPA and EigenExa. The hybrid solvers with the two newer libraries, ELPA and EigenExa, give better benchmark results than the conventional ScaLAPACK library. The detailed analysis on the results implies that the reducer can be a bottleneck in next-generation (exa-scale) supercomputers, which indicates the guidance for future research of parallel generalized eigenvalue problem solvers.

Quantum wavepacket dynamics simulation solver This part focuses on transport calculations for condensed organic polymers. One-hundred-nm-scale electronic structure calculations were carried out by ELSES on the K supercomputer. The transport calculations were carried out as a theoretical extension for the quantum (hole) wavepacket dynamics simulation. The calculation is based on a time-dependent Schrödinger type equation $i\frac{d}{dt}\psi = H_{WP}\psi$ for a hole wavefunction $\psi(t)$ with a modelled Hamiltonian H_{WP}. Analysis of time evolution of $\psi(t)$ gives mobility, which is an important parameter for electronic device performance. The method was applied to a single polymer chain and condensed polymers. The result of mobility calculation is consistent to the experimental trend.

Order-N electronic structure calculation code A novel parallel linear-algebraic

algorithm was introduced to electronic state calculations. The benchmark shows an extreme strong scaling (75% in parallel efficiency) and a qualified time-to-solution (less than 10^2 sec in elapsed time) on the full system of the K computer. Their mathematical foundation is generalized shifted linear equations ($(zS-H)x = b$), instead of conventional generalized eigenvalue equations. The foundation has a highly parallelizable mathematical structure and applicable to many scientific areas. The simulation of organic polymer devices was carried out in academic-industrial collaboration. Using the electronic state calculations, a network analysis on connected polymer networks was carried out. Small networks of several polymers that electronically connected are extracted. The quantum wavepacket dynamics simulation was employed for transport calculations for the extracted networks. The simulation and data analysis reveal that electronic waves propagate on connected polymer networks and contribute the electrical current. The present simulation method will give the insights of next-generation electronic devices and their fabrication process.

Chapter 2

Background

2.1 Basics of parallel computing

Parallel computing is a form of computational methods which exploits concurrency in a program to execute efficiently. Concurrency is a character of a program that its partial tasks can be executed simultaneously at overlapping time intervals. Here efficiency means mainly shortening of the elapsed time for a program. In another context, efficiency also means reduction of required memory on a machine, which leads to enable larger computation on whole machines. Transforming a program to assign processors for the simultaneous partial tasks so that the elapsed time is shorten is called parallelization. Parallelism, chance of parallelization, appears in various levels of a program. Parallelism is called coarse-grain when each of the partial tasks is relatively large and consumes long elapsed time. Conversely, it is called fine-grain when they are relatively small. Generally, coarse-grain parallelism requires smaller additional computational costs on parallelization. Parallelization needs support from both sides of hardware and software.

2.1.1 Hardware

A parallel computer is a computer which has multiple processors and parallel programs can run on it. Most of recent computers are kind of parallel computer because they ordinarily have multiple cores on a CPU chip. A supercomputer is a parallel computer which has very high computational capacity and mainly used for scientific computing. In general, computational capacity of supercomputers is compared by FLOPS (value of how many floating-point operations can be executed per second). For example, the K computer, one of the fastest supercomputers in the world, has theoretical peak performance of 11.28 Peta FLOPS.

Parallel computers are classified into two classes, shared memory machines or

distributed memory machines, by their memory architecture. On a shared memory machine all the CPU cores are able to access to the single memory unit. On a distributed memory machine, there are multiple memory units and a CPU core is able to access to a part of them. One should use communication between the memory units to share data on a distributed memory machine. Although it is easier to implement a parallel program on a shared memory machine than on a distributed memory machine, there is a limitation on the number of computational cores on a shared memory machine due to costs for preserving memory coherence. Therefore almost all of modern supercomputers are distributed memory machines that connect many shared memory computer nodes by communication network.

As stated above, a parallel computer consists of processors, memory units, and network. Among these, costs of memory access and network access are relatively large in modern supercomputers. Therefore it is important to reduce memory and network access as much as possible for implementation of an efficient parallel program.

2.1.2 Software

Programming models for parallel computation are classified by their memory model. On the shared memory model which a single memory space are shared by all processors, APIs (application programming interface) like pthreads (POSIX threads) or OpenMP [8] are commonly used. On the other hand, on the distributed memory model which each processor refers local memory space, required data are shared by an explicit send and receive. This style of sharing data is called message passing. De facto standard API for the message passing programming model is MPI [9]. Almost all of modern supercomputers provide an implementation of OpenMP and MPI. An example of famous MPI implementation is MPICH [10].

It should be noted that the classification of programming models is independent from the classification of memory architecture of hardware (shared memory machine or distributed memory machine). One can run MPI parallel program on a shared memory machine. Also, one can run a shared memory parallel program written in such as a PGAS (Partitioned Global Address Space) language on a distributed memory machine [11].

2.1.3 Performance evaluation of parallel program

Performance of a parallel program is often evaluated by how the elapsed time T decreases with increase of the number of processors n. It is called scaling property. There are two measures of scaling property, weak scaling and strong scaling. In weak scaling, the problem size (computational complexity) is increased in proportion to the number of processors. In strong scaling, the problem size is fixed

and only the number of processors is changed. Ideally, T is independent of n in weak scaling, and T is inversely proportional to n in strong scaling. However, in reality, T is larger than them because there are costs for parallelization itself. Generally, it is more difficult to improve strong scaling property than weak one because the fraction of costs for parallelization itself increases as n increases.

Strong scaling property can be predicted by Amdahl's argument (or Amdahl's law) [12]. Suppose that there are a program which is divided into two parts and an amount of a computational resource s (for example, the number of processors). The elapsed time of one part decreases in proportion to s. That of the other part does not change while the amount of the resource changes. p is the percentage of the elapsed time for the former part when $s = 1$. T_s is the total elapsed time as s changes. Under the above condition, speedup ratio $r(s) \equiv T_1/T_s$ is determined by

$$r(s) = \frac{1}{(1-p) + \frac{p}{s}}. \tag{2.1}$$

In context of parallelization, it indicates that benefit from parallelization is small when the fraction of the non-parallelized part $1 - p$ is large. Also the upper limit of the speed up ratio is determined by

$$\lim_{s\to\infty} r(s) = \frac{1}{1-p}. \tag{2.2}$$

For example, when $p = 0.9$, the maximum speedup is ten times as fast as the baseline.

2.1.4 Trends of large-scale parallel computing

Past research of parallel computing had a tendency to increase solvable problem size, in other words, a tendency to improve weak scaling property, not strong scaling property. Because recent supercomputers have very high computational capacity, the maximum solvable problem size for them is sometimes out of the range of interest from a viewpoint of real application. Therefore strong scaling property is becoming more and more important.

In recent supercomputers, although performance increase of computational units is rapid, that of data transfer functions such as memory access and communication is relatively slow. The separation between them will expand and existing parallel programs may suffer from low scalability on next generation supercomputers.

Those problems on modern large-scale parallel computing cannot be solved by effort in a single field. Related fields of hardware architecture, algorithm, implementation and application must be simultaneously considered and adjusted for each other. Such a style of design with collaboration of related fields is often called co-design.

2.2 Basics of numerical linear algebraic routines

Numerical linear algebraic solvers are fundamental components of scientific computing. Examples of typical problems in numerical linear algebra are linear equation and eigenvalue problem. Elementary methods for the problems such as calculating inverse matrix for linear equation or solving characteristic polynomial for eigenvalue problem are often not suitable to calculation on computers due to large computational cost and numerical error. Instead of them, algorithms and implementations suitable to computers has been extensively studied [13, 14, 15]. Especially, numerical linear algebraic solvers for large matrices have strong needs among various applications with the current and next-generation supercomputers.

Numerical linear algebraic solvers are classified into two categories. One is dense-matrix solver and the other is sparse-matrix solver. In the dense-matrix solvers, all the matrix elements are treated as non-zero values and a routine requires $O(M^2)$ memory cost and $O(M^3)$ operation cost, typically, where M is the matrix dimension. In the sparse-matrix solvers, on the other hand, only non-zero elements are treated explicitly and a routine typically requires $O(N)$ memory cost and cheaper operation cost, where N is the number of non-zero elements. Sparse-matrix solvers are again classified into two categories. One is sparse direct solver and the other is sparse iterative solver. Sparse direct solvers sequentially transform a matrix into some standard form to compute a solution. Sparse iterative solvers do not transform a matrix and use it only for matrix-vector multiplications to compute a solution. The thesis does not treat sparse direct solvers. Sparse matrices emerge from various areas of scientific computing. For example, discretization of differential equation often generates a sparse matrix, reflecting spatial locality of differential operators. In the thesis, the hybrid generalized eigenvalue solver in Chap. 3 is classified into dense eigenvalue problem solvers, and the electronic state calculation solver in Chap. 4 is classified into sparse linear equation solvers.

Design of dense-matrix solvers does not relatively depend on property of input matrices. Therefore, there are *de facto* standard libraries for dense numerical linear algebra calculation, such as BLAS [1], LAPACK [2] and ScaLAPACK [3]. BLAS [16] is an API set of basic linear algebra operations like summation and multiplication of matrices and vectors. LAPACK [17] is a solver collection of numerical linear algebra problems like linear equation, eigenvalue decomposition and singular value decomposition. LAPACK is constructed based on BLAS. This separation of implementation level realizes both of high performance and portability. ScaLAPACK [18, 19] is the distributed parallel version of LAPACK. These libraries are available on almost all of modern supercomputers and highly opti-

[1] BLAS = Basic Linear Algebra Subprograms
[2] LAPACK = Linear Algebra PACKage
[3] ScaLAPACK = Scalable Linear Algebra PACKage

mized for each architecture. But several routines give severe bottlenecks in the computational speed with current massively parallel architectures.

On the other hand, design of sparse-matrix solvers highly depends on property of input matrices. Moreover, parallelization and optimization are generally more difficult than dense-matrix solvers due to lack of regularity in calculation. Therefore, it is important to consider specific structures in a problem when designing sparse-matrix solvers. In sparse iterative solvers, the most time-consuming procedure is often matrix-vector multiplication. CRS (Compressed Row Storage) [15] is a common sparse matrix format. In CRS format, a matrix data is stored by two integer arrays colind, rowptr and a floating-point value array values. values stores the values of the non-zero elements of the matrix in row-wise order. colind has the same length with values and contains the column indices of the corresponding non-zero elements. rowptr has the length of the number of rows in the matrix and contains indices in values and colind where the data of each row starts. Pseudo-code of matrix-vector multiplication $y = Ax$ in CRS format is as follows:

```
1: procedure MATRIX-VECTOR MULTIPLICATION IN CRS FORMAT(A, x)
2:     for i = 1 ... m do
3:         for k = A.rowptr[i] ... A.rowptr[i + 1] − 1 do
4:             j = A.colind[k]
5:             y[i]+ = A.values[k] ∗ x[j]
6:         end for
7:     end for
8: end procedure
```

2.3 Basics of electronic state calculations

This section is devoted to a brief review of the theoretical foundation of electronic state calculation. The explanation focuses on the mathematical structure of the theory and ignores several physical aspects, such as spin freedom of electron, for simplicity.

2.3.1 Theoretical foundation of electronic state calculation

The fundamental theory of quantum material simulation or electronic state calculation stems from Schrödinger's equation for many electron wavefuntions

$$\Psi \equiv \Psi(\boldsymbol{r}_1, \boldsymbol{r}_2, ..., \boldsymbol{r}_{N_{\mathrm{elec}}}), \tag{2.3}$$

a complex scalar function, where N_{elec} is the number of electrons and $\boldsymbol{r}_j$ is the position of the j-th electron. It is very difficult, however, to solve the equation

with materials directly, because the computational cost increases exponentially as the function of $N_{\rm elec}$. Walter Kohn, a Nobel Prize winner in Chemistry on 1998, called the situation 'exponential wall' [20]; If the continuum space of $\boldsymbol{r}_j$ is discretized into p points, the number of elements for the discretized wavefunction is $M = p^N_{\rm elec}$. In the case of $N_{\rm elec} = 36$ electrons with $p = 2$, for example, the number M is given as

$$M = p^{N_{\rm elec}} = 2^{36} = 68,719,476,736 \approx 70{\rm G} \tag{2.4}$$

and the required memory size to store the wavefunction as complex double precision complex array is estimated to be

$$16{\rm B} \times M \approx 1{\rm TB}. \tag{2.5}$$

The above estimation indicates that the above calculation can not be carried out with one hundred electrons.

Most of the present electronic state calculations are carried out with one-electron wavefunction $\phi(\boldsymbol{r})$, instead of the many-electron wavefunction Ψ. See textbooks for detail [1]. Effective equations appear as eigenvalue equations in continuum space

$$\hat{H}_{\rm eff}\phi_j = \lambda_j\phi_j \tag{2.6}$$

with the Hamiltonian operator $\hat{H}$ of

$$\hat{H}_{\rm eff} \equiv -\frac{\hbar^2}{2m_{\rm e}}\Delta + U_{\rm eff}(\boldsymbol{r}). \tag{2.7}$$

Here $m_{\rm e}$ is the mass of electron and $\hbar$ is Planck's constant ($m_{\rm e} = 9.109534 \times 10^{31}$kg, $\hbar = 1.0545887 \times 10^{-34}$Js). The scalar function $U_{\rm eff}(\boldsymbol{r})$ is the potential function for electrons and differs among materials. Since the potential function includes the quantum mechanical interactions between electrons, Eq. (2.6) is a nonlinear equation. The solution of Eq. (2.6) gives the eigenpairs of $\{\lambda_j, \phi_j(\boldsymbol{r})\}$. The eigenvalues of $\{\lambda_j\}$ are real and the index j is determined as $\lambda_1 \leq \lambda_2 \leq \lambda_3, ...$ The lowest $N_{\rm elec}$ eigenpairs are those for the electrons in material. The j-th eigenvalue and eigenfunctions means the energy and state (wavefunction) of the j-th electron, respectively. The wavefunctions satisfy the orthogonality relation of

$$\int \phi_k(\boldsymbol{r})\phi_l(\boldsymbol{r})d\boldsymbol{r} = \delta_{kl}, \tag{2.8}$$

unless the eigenvalues are degenerated ($\lambda_k \neq \lambda_l$). The $N_{\rm elec}$-th eigenvalue or eigenfunction is usually called, highest occupied (HO) energy or state, respectively.

The numbers of electrons $N_{\rm elec}$ in material and is usually proportional to the number of atoms N ($N_{\rm elec} \propto N$).

It is noteworthy that the one-electron wavefunction theory corresponds to an approximation on the many-electron wavefuntion by the determinant form called Slater determinant. The case of $N_{\rm elec} = 2$, for example, is written as

$$\Psi(\boldsymbol{r}_1, \boldsymbol{r}_2) := \frac{1}{\sqrt{2}} \begin{vmatrix} \phi_1(\boldsymbol{r}_1) & \phi_2(\boldsymbol{r}_1) \\ \phi_1(\boldsymbol{r}_2) & \phi_2(\boldsymbol{r}_2) \end{vmatrix} = \frac{1}{\sqrt{2}} (\phi_1(\boldsymbol{r}_1)\phi_2(\boldsymbol{r}_2) - \phi_2(\boldsymbol{r}_1)\phi_1(\boldsymbol{r}_2)). \quad (2.9)$$

The determinant form satisfies the mathematical relation of

$$\Psi(\boldsymbol{r}_1, \boldsymbol{r}_2) = -\Psi(\boldsymbol{r}_2, \boldsymbol{r}_1), \quad (2.10)$$

known as a Fermion property of electrons.

2.3.2 Derivation of generalized eigenvalue problem

Here a generalized eigenvalue problem (GEP) is derived by discretization of real space. The problem will be the central linear algebraic problem in this thesis.

When an electronic wavefunction $\phi_k(\boldsymbol{r})$ is expressed by the linear combination with given basis functions of $\{\chi_j(\boldsymbol{r})\}_{j=1,\ldots M}$

$$\phi_k(\boldsymbol{r}) = \sum_j y_{jk} \chi_j(\boldsymbol{r}), \quad (2.11)$$

a generalized eigenvalue problem of

$$H\boldsymbol{y}_k = \lambda_k S\, \boldsymbol{y}_k. \quad (2.12)$$

appears with $M \times M$ Hermitian matrices of H and S that are defined as

$$H_{ij} \equiv \int \chi_i^*(\boldsymbol{r}) \hat{H}_{\rm eff} \chi_j(\boldsymbol{r}) d\boldsymbol{r} \quad (2.13)$$

$$S_{ij} \equiv \int \chi_i^*(\boldsymbol{r}) \chi_j(\boldsymbol{r}) d\boldsymbol{r}. \quad (2.14)$$

The matrix S is positive definite from the definition. The matrices H and S are called Hamiltonian and the overlap matrices, respectively.

In the thesis, the real-space atomic-orbital representation is used and the matrix dimension is proportional to the number of atoms ($M \propto N$). A basis function $\chi_j(\boldsymbol{r})$ is a localized function, called atomic orbital, of which center is located at the position of one atom (nucleus). In addition, the basis functions are chosen as real functions and the matrices of H and S are real symmetric. Since the basis functions are normalized, all the diagonal elements of S are the unity ($S_{ii} = 1$) and all the off-diagonal elements satisfy the inequality of $|S_{ij}| < 1 (i \neq j)$. The orthogonality relation in continuum real space, Eq. (2.8), is transformed into

$$\boldsymbol{y}_k^{\rm T} S\, \boldsymbol{y}_l = \delta_{kl}. \quad (2.15)$$

2.3.3 Numerical aspect of large-scale calculations

As explained in the previous subsections, the GEP of Eq. (2.12) gives the mathematical foundation of electronic structure calculations or quantum mechanical calculations of materials, in which an electron is treated as a quantum mechanical 'wave'. Fig. 2.1(a) shows an example of the wavefunction. The number of the required eigenvalues is, at least, on the order of the number of the electrons or the atoms in calculated materials.

When one think about large scale calculations, however, a potential difficulty in parallelism is the orthogonality relation between eigenvectors. Since the eigenvectors should satisfy an orthogonality relation of Eq. (2.15), an explicit orthogonalization procedure is required. An orthogonalization procedure requires an operation cost of ($O(N^3)$) and can be a bottleneck in parallelism. Since the orthogonality is inherent in quantum mechanics, the above potential difficulty appears commonly among the large-scale electronic state calculations.

Here an order-N electronic state calculation code ELSES [7, 21] is explained briefly, since the present research is a theoretical extension of the methods developed previously for ELSES. The simulations use novel 'order-N' linear-algebraic methods in which the computational cost is 'order-N' ($O(N)$) or is proportional to the number of atoms N. Their mathematical foundation is sparse-matrix (Krylov-subspace) solvers. The detailed theories are explained later in this thesis. Efficient massively parallel computation is found in Fig. 2.1, a strong scaling benchmark on the K computer [22, 23] with one hundred million atoms or one-hundred-nanometer scale materials. The simulated materials are a nano-composite carbon solid with $N = 103,219,200$ or $M = 412,876,800$ [22] and an amorphous-like conjugated polymer with $N = 102,238,848$ or $M = 230,776,128$ [23]. All the calculations in this thesis were realized with first-principles-based modeled (tight-binding-form) theory. The details can be found in Ref. [21] and will be explained later in this thesis.

2.4 Notes on physical and methodological concepts

2.4.1 Participation ratio

Participation ratio (PR) was introduced in theoretical condensed matter physics, mainly for the theory of localized electron wavefunctions [24, 25, 26, 27].

Originally the PR is defined for a normalized wavefunction $\phi(\boldsymbol{r})$ as

$$PR^{(\mathrm{org})}(\phi) \equiv \left(\int |\phi(\boldsymbol{r})|^4 d\boldsymbol{r} \right)^{-1} . \tag{2.16}$$

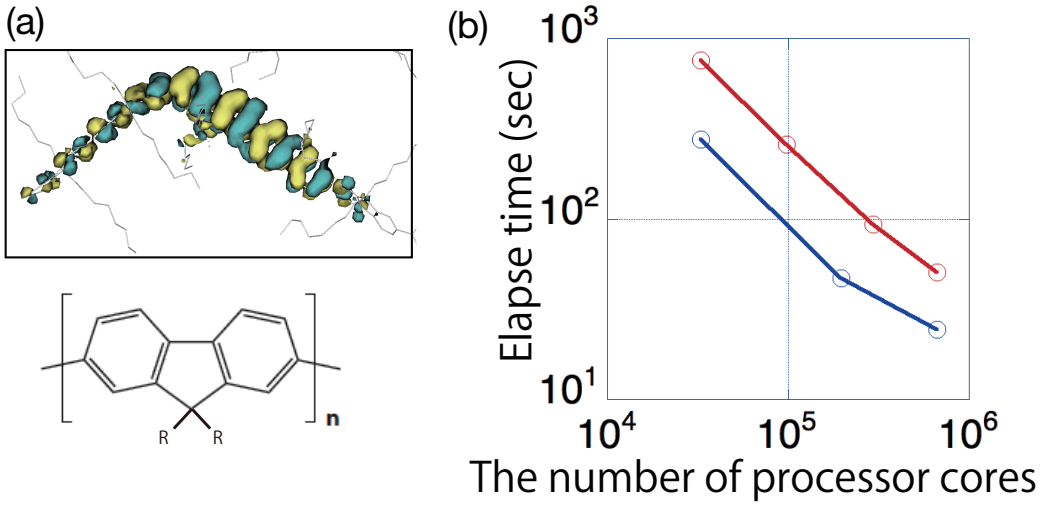

Figure 2.1: (a) The upper panel is a π-type electronic wavefunction in an amorphous-like conjugated polymer (poly-((9,9) dioctyl fluorine)). The lower panel shows the atomic structure (R≡C_8H_{17}) [22]. (b) Strong scaling plot by ELSES for one-hundred-million-atoms calculations on the K computer. [22, 23] The calculated materials are a nano-composite carbon solid (the upper line) and the amorphous-like conjugated polymer (the lower line). The number of used processor nodes is from P = 4,096 to 82,944 (full nodes of the K computer).

Suppose D is a closed area whose volume is Ω and ϕ is uniformly distributed in D as

$$\phi(\boldsymbol{r}) = \begin{cases} \frac{1}{\sqrt{\Omega}} & (\boldsymbol{r} \in D) \\ 0 & (\text{otherwise}). \end{cases} \tag{2.17}$$

Then the PR of ϕ is

$$PR^{(\text{org})}(\phi) = \left(\frac{1}{\Omega^2} \int_D d\boldsymbol{r} \right)^{-1} = \Omega. \tag{2.18}$$

Namely, the PR indicates how the wavefunction spreads in space.

An analogue of the PR for a finite dimensional vector is defined. In context of the thesis, it is defined for an eigenvector $\boldsymbol{y}$ of the generalized eigenvalue equation Eq. (2.12) as

$$PR(\boldsymbol{y}) \equiv \frac{\left(\boldsymbol{y}^T S \boldsymbol{y}\right)^2}{\sum_j |y_j|^4}. \tag{2.19}$$

Note that the definition is invariant under scaling. The PR indicates a measure of the number of non-zero elements, namely, how broadly the elements exist on the

indices. For example, the case of $S = I$ and

$$\boldsymbol{y} \equiv \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, 0, 0, ..., 0 \right)^T, \tag{2.20}$$

gives the PR as

$$PR(\boldsymbol{y}) = \left\{ \sum_j |y_j|^4 \right\}^{-1} = \left\{ 3 \left(\frac{1}{\sqrt{3}} \right)^4 \right\}^{-1} = 3. \tag{2.21}$$

In cases of $S \neq I$, the value of PR can be smaller than one ($PR(\boldsymbol{y}) < 1$) and the PR does not indicate the measure of the number of non-zero elements. In calculation of PRs for the eigenvectors of a generalized eigenvalue problem, such cases appear in high eigenvalue regions. For example, the case of

$$H = \begin{pmatrix} 0 & -t \\ -t & 0 \end{pmatrix}, \quad S = \begin{pmatrix} 1 & s \\ s & 1 \end{pmatrix} \tag{2.22}$$

with $t > 0$ and $0 < s < 1$. The eigenpairs are obtained as

$$\lambda_1 = \frac{-t}{1+s}, \quad \boldsymbol{y}_1 = \frac{1}{\sqrt{2(1+s)}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{2.23}$$

and

$$\lambda_2 = \frac{t}{1-s}, \quad \boldsymbol{y}_2 = \frac{1}{\sqrt{2(1-s)}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}. \tag{2.24}$$

The PR of the eigenvectors are given by

$$PR(\boldsymbol{y}_1) = \left\{ 2 \left(\frac{1}{\sqrt{2(1+s)}} \right)^4 \right\}^{-1} = \left\{ \frac{1}{2(1+s)^2} \right\}^{-1} = 2(1+s)^2 \tag{2.25}$$

and

$$PR(\boldsymbol{y}_2) = \left\{ 2 \left(\frac{1}{\sqrt{2(1-s)}} \right)^4 \right\}^{-1} = \left\{ \frac{1}{2(1-s)^2} \right\}^{-1} = 2(1-s)^2. \tag{2.26}$$

The latter value can be smaller than one ($PR(\boldsymbol{y}_2) < 1$), when $s > 1 - 1/\sqrt{2}$.

However, only low eigenvalue regions are interesting in most cases and the value of PR in these regions usually larger than one and indicate the measure of the number of non-zero elements.

It is noteworthy that the unit vector of $\boldsymbol{y} \equiv (1, \boldsymbol{0})^T$ satisfies the normalization condition of $\boldsymbol{y}^T S \boldsymbol{y} = 1$ and gives the PR of $PR(\boldsymbol{y}) = 1$.

2.4.2 Mobility

In this section, mobility of a carrier in semiconductors is derived by Einstein's relation as in many textbooks, for example, Ref. [28]. Mobility is an important physical quantity of semiconductor and means the ability of a carrier to move through a material. Since a material with higher mobility value can be a better device material, the material exploration for higher mobility is crucial for material design.

Here the derivation is explained in the one-dimensional case for a carrier, hole or exited electron. The theoretical extension to two- and three-dimensional cases is straightforward. The mass, charge, position of the carrier is denoted as m, q and x, respectively. The sign of q is denoted as $s \equiv \mathrm{sign}(q)$ ($s = +1$ for $q > 0$ and $s = -1$ for $q < 0$). The velocity is denoted as $v \equiv dx/dt$. The potential for the carrier is denoted as $V(x)$. The electric field is given as

$$E(x) \equiv -\frac{dV}{dx}. \tag{2.27}$$

The density of carriers, denoted as $\rho(x)$, is given as

$$\rho(x) = \rho_0 \exp\left(-\frac{sqV(x)}{k_\mathrm{B}T}\right) \tag{2.28}$$

by the Boltzmann distribution with the temperature T and Boltzmann's constant k_B.

The drift current is derived from Newton's equation of motion

$$m\frac{dv}{dt} = qE - \frac{m}{\tau}v. \tag{2.29}$$

The second term of the right hand side is a modeled friction force with the time constant of $\tau(> 0)$ called mean free time. A stationary state results in $dv/dt = 0$ and

$$v = \frac{q\tau}{m}E. \tag{2.30}$$

The drift current is defined as

$$j_\mathrm{drift} \equiv \rho q v = \frac{\rho q^2 \tau}{m}E. \tag{2.31}$$

If the mobility μ is defined as

$$\mu \equiv s\frac{q\tau}{m}, \tag{2.32}$$

one obtains

$$j_{\text{drift}} = s\rho q\mu E. \tag{2.33}$$

Hereafter, Eq. (2.33) is used as the definition of the mobility μ.

The diffusion current, on the other hand, is defined as

$$j_{\text{diff}} \equiv -sqD\frac{d\rho}{dx} \tag{2.34}$$

with the diffusion coefficient $D(>0)$. The diffusion current appears from the particle flow from a higher density region into a lower density region.

The total current is given as

$$\begin{aligned} j_{\text{tot}} &\equiv j_{\text{drift}} + j_{\text{diff}} \\ &= sq\rho\mu E - sqD\frac{d\rho}{dx} \\ &= -sq\rho\mu\frac{dV}{dx} - sqD\frac{d\rho}{dV}\frac{dV}{dx} \\ &= -sq\frac{dV}{dx}\left\{\rho\mu + D\frac{d\rho}{dV}\right\} \end{aligned} \tag{2.35}$$

If no voltage is imposed, the total current is zero ($j_{\text{tot}} = 0$) and one obtains

$$\begin{aligned} \mu &= -D\frac{1}{\rho}\frac{d\rho}{dV} \\ &= -D\frac{d}{dV}(\log\rho) \\ &= D\frac{d}{dV}\left(\frac{sqV}{k_{\text{B}}T}\right) \\ &= \frac{Dsq}{k_{\text{B}}T} \\ &= \frac{D|q|}{k_{\text{B}}T}. \end{aligned} \tag{2.36}$$

Eq. (2.36) is Einstein's relation.

2.4.3 Diffusion constant

In diffusion phenomena, a characteristic diffusion speed called diffusion constant D, which has dimension of L^2T^{-1}, can be obtained. In the one dimensional diffusion equation

$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2}, \tag{2.37}$$

the solution of the initial value problem from $f(x,0) = \delta(x)$ is

$$f(x,t) = \frac{1}{2\sqrt{\pi D t}} \exp\left(-\frac{x^2}{4Dt}\right), \tag{2.38}$$

which is the normal distribution with variance of $\sigma^2 = 2Dt$. Inversely the diffusion constant D is determined from the solution $f(x,t)$ as

$$D = \frac{\langle x^2(t)\rangle}{2t} \tag{2.39}$$

where

$$\langle x^2(t)\rangle \equiv \int x^2 f(x,t) dx. \tag{2.40}$$

2.4.4 Atomic unit

The atomic unit (a.u.) is widely employed in electronic state theory papers and is used throughout this thesis, except where indicated. The atomic unit is defined so that

$$m_\mathrm{e} = 1, \quad \hbar = 1, \quad \frac{e^2}{4\pi\epsilon_0} = 1, \tag{2.41}$$

where $m_\mathrm{e}, e, \hbar$ and ϵ_0 are the mass of electron, the charge of electron ($e > 0$), Planck's constant, and the permittivity of vacuum, respectively. The length atomic unit

$$a_\mathrm{B} \equiv \frac{\hbar^2}{m_\mathrm{e}} \left(\frac{e^2}{4\pi\epsilon_0}\right)^{-1} \approx 0.52918 \times 10^{-10}\mathrm{m} \tag{2.42}$$

is called Bohr radius. The energy atomic unit

$$E_\mathrm{H} \equiv \frac{m_\mathrm{e}}{\hbar^2} \left(\frac{e^2}{4\pi\epsilon_0}\right)^{2} \approx 4.3598 \times 10^{-18}\mathrm{J} \tag{2.43}$$

is called Hartree unit. The time atomic unit is given by

$$T_\mathrm{au} \equiv \frac{\hbar}{E_\mathrm{H}} \approx 2.4189 \times 10^{-17}\mathrm{s} \approx \frac{1}{40}\mathrm{fs}. \tag{2.44}$$

Several energy conversion relations among the atomic unit, electron volt (eV) and Kelvin (K) are noted below;

$$1\,\mathrm{a.u.} = 27.211\,\mathrm{eV}, \quad 1\,\mathrm{eV} = 11,604\,\mathrm{K}. \tag{2.45}$$

Chapter 3

Hybrid solver for massively parallel eigenvalue computation

3.1 Concept of hybrid solver

The present chapter presents a hybrid solver collection, EigenKernel [29], for massively parallel eigenvalue computation, as a foundation of large-scale electronic state calculations. Nowadays ScaLAPACK[18, 19] [1] is the *de facto* standard solver library for parallel computations but several routines give severe bottlenecks in the computational speed with current massively parallel architectures. Novel solver libraries were proposed so as to overcome the bottlenecks. Since the performance of numerical routines varies significantly with problems and architectures, the best performance is achieved, when one constructs an optimal 'hybrid' among the libraries.

The concept of hybrid solver is illustrated in Fig. 3.1. It is a numerical middleware and has a unique data interface to real applications. One can choose the optimal workflow for each problem without any programming effort.

The present thesis focuses on dense-matrix solvers for generalized eigenvalue problems (GEPs) in the form of

$$A\boldsymbol{y}_k = \lambda_k B\boldsymbol{y}_k \tag{3.1}$$

with the given $M \times M$ real-symmetric matrices of A and B. The matrix B is positive definite. The eigenvalues $\{\lambda_k\}$ and the eigenvectors $\{\boldsymbol{y}_k\}$ will be calculated. The computational cost is $O(M^3)$ or is proportional to M^3. The present hybrid solvers are constructed among ScaLAPACK and the two newer libraries of ELPA [30, 31, 32] [2], and EigenExa [33, 34, 35, 36]. The ELPA and EigenExa libraries are written

[1] ScaLAPACK = Scalable Linear Algebra PACKage

[2] ELPA = Eigenvalue soLvers for Petascale Applications

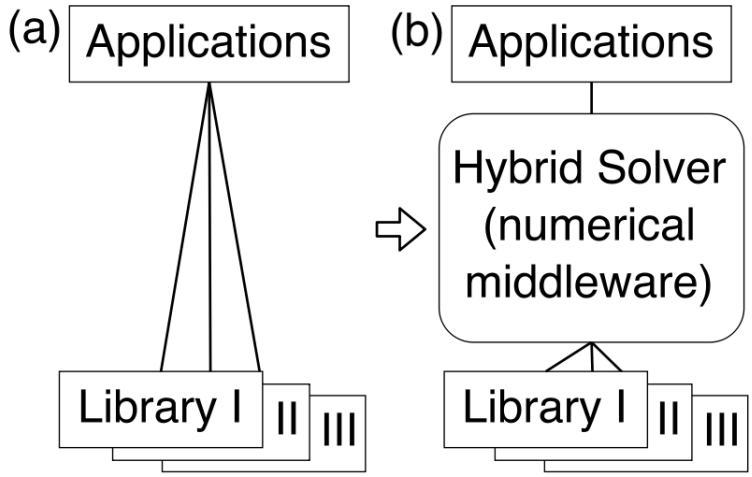

Figure 3.1: Concept of hybrid solver; Structure of the program code (a) without and (b) with hybrid solver or numerical middleware.

in Fortran and appeared in 2000's for efficient massively parallel computations.

The present dense-matrix solvers are complementary methods to the order-N calculations, because the order-N calculation gives approximate solutions, while the dense-matrix solvers give numerically exact ones with a heavier ($O(M^3)$) computational cost. The use of the two methods will lead us to fruitful researches. The exact solutions are important, for example, when the system has many nearly degenerated eigen pairs and one would like to distinguish them. The exact solutions are important also as reference data for the development of fine approximate solvers.

The matrices of A and B in the present benchmark appear on 'ELSES Matrix Library'. [37] See Sec. 3.6.4 for details of ELSES Matrix Library. The benchmark was carried out with the data files of 'NCCS430080', 'VCNT22500' 'VCNT90000' and 'VCNT1008000' for the matrix sizes of M=22,500, M=90,000, M=430,080, M=1,008,000, respectively. A large matrix data (> 0.5GB) is uploaded as a set of split files for user's convenience.

The physical origin of the matrices is explained briefly. The files in the present benchmark are carbon materials within modeled tight-binding-form theories based on *ab initio* calculations. The matrix of 'NCCS430080' appears in the material research on a nano-composite carbon solid (NCCS) [38]. An sp-orbital form [39] is used and the system contains $N = M/4 = 107,520$ atoms. The other files are generated for thermally vibrated single-wall carbon nanotubes (VCNTs) within a supercell. An spd-orbital form [40] is used and each system contains $N = M/9$ atoms. The VCNT systems were prepared, so as to generate matrices systematically in different size with similar eigenvalue distributions. These matrices were

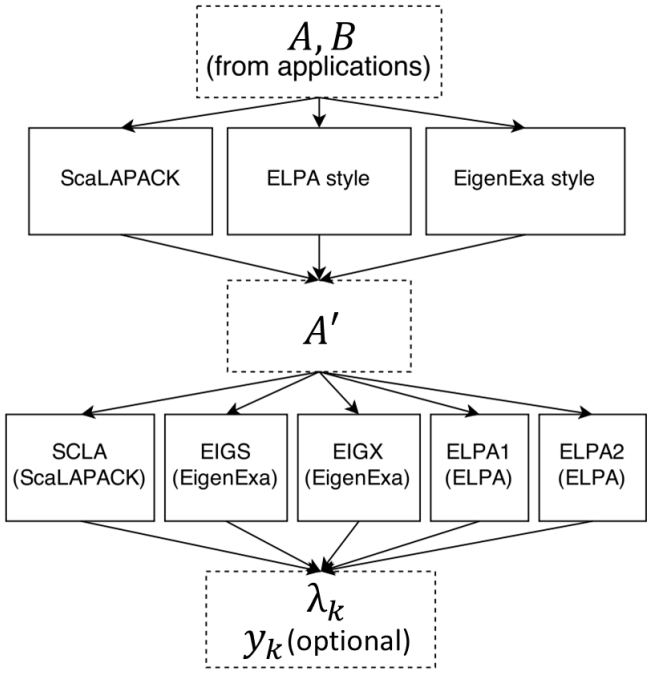

Figure 3.2: Workflow of the hybrid GEP solver.

used for the investigation on π-electron materials with the present dense-matrix solver and the order-N solver. [3]

3.2 Parallel eigenvalue solvers of ScaLAPACK, EigenExa and ELPA

A hybrid solver is constructed, when a routine is chosen for each subprocedure from ScaLAPACK, EigenExa and ELPA. The code was developed as a general middleware that can be connected not only to ELSES but also to any real application software, as in Fig. 3.1. A mini-application was also developed and used in the present benchmark. In the benchmark, ScaLAPACK was used as a built-in library on each machine. EigenExa in the version 2.2a [4] and ELPA in the version 2014.06.001 were used. ELPA and EigenExa call some ScaLAPACK routines.

[3] The present matrices are sparse, which does not lose the generality of the benchmark, since the cost of the dense matrix solver is not dependent on the number of non-zero elements of the matrix.

[4] The present EigenExa package does not include the GEP solver. The GEP solver routine for EigenExa in the present thesis is that of the version 2.2b of KMATH_EIGEN_GEV [41] that shares the SEP solver routine with the EigenExa package.

3.3 Mathematical formulation

The GEP of Eq. (3.1) can be written in a matrix form of

$$AY = BY\Lambda, \tag{3.2}$$

where the matrix $\Lambda \equiv \mathrm{diag}(\lambda_1, \lambda_2, \dots)$ is diagonal and the matrix $Y \equiv (\boldsymbol{y}_1\, \boldsymbol{y}_2\, \cdots)$ satisfies $Y^T BY = I$. In the solvers, the GEP of Eq. (3.1) is reduced to a standard eigenvalue problem (SEP) of

$$A'Z = Z\Lambda, \tag{3.3}$$

where the reduced matrix A' is real symmetric [14] and the matrix of $Z \equiv (\boldsymbol{z}_1\, \boldsymbol{z}_2\, \cdots)$ contain eigenvectors of A'. The reduction procedure can be achieved, when the Cholesky factorization of B gives the Cholesky factor U as an upper triangle matrix:

$$B = U^T U. \tag{3.4}$$

The reduced matrix A' is defined by

$$A' = U^{-T} A U^{-1}. \tag{3.5}$$

The eigenvectors of the GEP, written as $Y \equiv (\boldsymbol{y}_1\, \boldsymbol{y}_2\, \cdots)$, are calculated from those of the SEP by

$$Y = U^{-1} Z. \tag{3.6}$$

This procedure is usually called backward transformation.

The GEP solver is decomposed into the two subprocedures of (a) the solver of the SEP in Eq. (3.3) and (b) the reduction from the GEP to the SEP ($(A, B) \Rightarrow A'$) and the backward transformation ($Z \Rightarrow Y$). The subprocedures (a) and (b) are called 'SEP solver' and 'reducer', respectively, and require $O(M^3)$ operations.

Figure 3.2 summarizes the workflows of the possible hybrid solvers. A hybrid solver is constructed, when one choose the routines for (a) the SEP solver and (b) the reducer, respectively.

For (a) the SEP solver, five routines are found in the base libraries; One routine is a ScaLAPACK routine (routine name in the code : 'pdsyevd') that uses the conventional tridiagonalization algorithm. [42] The ELPA or EigenExa library contains a SEP solver routine based on the tridiagonalization algorithm. The routine in ELPA is called 'ELPA1' (routine name in the code : 'solve_evp_real') in this thesis, as in the original paper [31], and the one in EigenExa called 'Eigen_s' or 'EIGS' (routine name in the code : 'eigen_s'). ELPA and EigenExa also contain the novel SEP solvers based on the narrow-band reduction algorithms without

the conventional tridiagonalization procedure. The solvers are called 'ELPA2' (routine name in the code : 'solve_evp_real_2stage') for the ELPA routine and 'Eigen_sx' or 'EIGX' (routine name in the code : 'eigen_sx') for the EigenExa routine in this thesis. See the papers [34, 32] for details.

For (b) the reducer, three routines are found in the base libraries and are called ScaLAPACK style, ELPA style, and EigenExa style reducers in this thesis. In the ScaLAPACK style, the Cholesky factorization, Eq. (3.4) is carried out and then the reduced matrix A', defined in Eq. (3.5), is generated by a recursive algorithm (routine name 'pdsygst') without explicit calculation of U^{-1} nor U^{-T}. Details of the recursive algorithm are explained, for example in Ref. [43]. In the ELPA style, the Cholesky factorization (routine name: 'cholesky_real') is carried out, as in the ScaLAPACK style, and the reduced matrix A' is generated by the explicit calculation of the inverse (triangular) matrix $R \equiv U^{-1}$ (routine names : 'invert_trm_real') and the explicit successive matrix multiplication of $A' = (R^T A)R$ (routine names: 'mult_at_b_real') [31] [5]. In the EigenExa style, the Cholesky factorization is not used. Instead, the SEP for the matrix B

$$BW = WD, \tag{3.7}$$

is solved by the SEP solver (Eigen_sx), with the diagonal matrix of $D \equiv \mathrm{diag}(d_1, d_2, ...)$ and the unitary matrix of $W \equiv (\boldsymbol{w}_1\, \boldsymbol{w}_2\, ...)$. A reduced SEP in the form of Eq. (3.3) is obtained by

$$\begin{aligned} A' &= (D^{-1/2}W^T)A(WD^{-1/2}) && (3.8)\\ Y &= WD^{-1/2}Z, && (3.9) \end{aligned}$$

because of $Z = D^{1/2}W^T Y$ and $W^{-T} = W$. Equation (3.9) is solved by the SEP solver (Eigen_sx).

Though the SEP solver of Eq. (3.3) requires a larger operation cost than the Cholesky factorization (See Fig.1 of Ref. [44], for example), the elapsed time can not be estimated only from the operation costs among the modern supercomputers.

The benchmark of the hybrid GEP solvers was carried out for the eight workflows listed in Table 3.1. In general, a potential issue is the possible overhead of the data conversion process between libraries. This issue will be discussed in Sec. 3.5.2.

[5] The benchmark was carried out in an ELPA style reduction algorithm. The ScaLAPACK routine of 'pdtrmm' is used for the multiplication of the triangular matrix R from right, while a sample code in the ELPA package uses the ELPA routine ('mult_at_b_real'). The difference is ignored, since the elapsed time of the above procedure is not dominant.

Table 3.1: List of the workflows in the benchmark. The routine names for the SEP solver and the reducer are shown for each workflow. Abbreviations are shown within parentheses.

Workflow	SEP solver	Reducer
A	ScaLAPACK (SCLA)	ScaLAPACK (SCLA)
B	Eigen_sx (EIGX)	ScaLAPACK (SCLA)
C	ScaLAPACK (SCLA)	ELPA
D	ELPA2	ELPA
E	ELPA1	ELPA
F	Eigen_s (EIGS)	ELPA
G	Eigen_sx (EIGX)	ELPA
H	Eigen_sx (EIGX)	Eigen_sx (EIGX)

3.4 Benchmark result

Strong scaling benchmarks are investigated for the hybrid solvers. The elapsed times were measured for (i) the full eigenpair calculation (T_{full}) and (ii) the 'eigenvalue-only' calculation (T_{evo}). In the latter case, the elapsed time is ignored for the calculation of the eigenvectors. The two types of calculations are important among electronic structure calculations. [31] The present benchmark ignores small elapsed times of the initial procedure for distributed data and the comments on them will appear in Sec. 3.5.1.

The benchmark was carried out on three supercomputers; the K computer at Riken, Fujitsu FX10 and SGI Altix ICE 8400EX. The K computer has a single SPARC 64 VIIIfx processor (2.0GHz, 8-core) on node. The FX10 is Oakleaf-FX of the University of Tokyo. Fujitsu FX10 is the successor of the K computer and has a single SPARC64 IXfx processor (1.848 GHz, 16-core) on each node. [6] SGI Altix ICE 8400EX of Institute for Solid State Physics of the University of Tokyo was also used. It is a cluster of Intel Xeon X5570 (2.93GHz, 8-core). The byte-per-flop value (B/F) is B/F=0.5, 0.36 or 0.68, for the K computer, FX10 or SGI Altix, respectively. The numbers of used processor nodes P are set to be square numbers ($P = q^2$) except in Sec. 3.4.3, since the ELPA paper [31] reported that the choice of a (near-)square number for P can give better performance.

When the non-traditional SEP solver algorithm of ELPA is used on Altix, one can choose an optimized low-level routine using SSE instructions ('REAL_ELPA_KERNEL_SSE') and a generic routine

[6] Additional options of the K computer and FX10 are explained; An MPI process shape on the Tofu interconnect was not specified. The rank directory feature to alleviate I/O contention was used.

('REAL_ELPA_KERNEL_GENERIC'). [31] The optimized code can run only on the Intel-based architectures compatible to SSE instructions and was prepared so as to accelerate the backtransformation subroutine. Among the results on Altix, the 'ELPA2' solver and the workflow D on Altix are those with the optimized routine, while the 'ELPA2′' solver and the workflow D' are those with the generic routine.

3.4.1 Result with the matrix size of $M = 430,080$

The benchmark with the matrix size of $M = 430,080$ was carried out for up to $P = 10{,}000$ nodes on the K computer. The elapsed times for P=10,000 nodes is shown in Table 3.2. The elapsed times for all the cases are shown in Fig. 3.3 for the (a) full ($T_{\rm full}$) or (b) eigenvalue-only ($T_{\rm evo}$) calculations. The decomposed times are also shown in Fig. 3.3 (c) for the SEP solver ($T_{\rm SEP}$) and the reducer ($T_{\rm RED}$) ($T_{\rm full} = T_{\rm SEP} + T_{\rm RED}$). Table 3.3 shows the decomposed time of the SEP solvers for P=10,000. A SEP solver routine is decomposed into three subroutines of (i) the tridiagonalization or narrow-band reduction ('TRD/BAND'), (ii) the divide and conquer algorithms for the tridiagonal or narrow-band matrices ('D&C') so as to compute the eigenvalues, and (iii) the backtransformation of eigenvectors ('BACK') so as to compute the eigenvectors of the GEP. One can observe several features on the results; (I) In the full calculation benchmark (Fig. 3.3(a)), the best data, the smallest elapsed time, appears in the workflow G for P=10,000. The workflow G is the hybrid solver that uses the 'Eigen_sx' SEP solver in EigenExa and the ELPA style reducer, since these routines are the best among the SEP solvers and the reducers, respectively, as shown in Fig. 3.3(c) and Table 3.3. In Table 3.2, the speed ($T_{\rm full}^{-1}$) of the workflow G is approximately four times faster than that of the conventional workflow A (11,634 sec) / (2,734 sec) $\approx$ 4.3).

(II) Fig. 3.3 (c) shows that the ELPA style reducer gives significantly smaller elapsed times than those of ScaLAPACK and those of EigenExa. The elapsed time for P=10,000 is $T_{\rm RED}$ = 1,261 sec with the ELPA style reducer and is $T_{\rm RED}$ = 2,157 sec with the EigenExa reducer. The elapsed time with the EigenExa reducer is governed by that of the SEP solver for Eq. (3.7) ($T_{\rm SEP}$ = 1,473 sec in Table 3.3). (III) In the eigenvalue-only calculation (Fig. 3.3(b)), the best data, the smallest elapsed time, appears in the workflow D for P=10,000. The workflow D is the solver that uses the 'ELPA2' SEP solver and the ELPA style reducer and the eigenvector calculation consumes a large elapsed time of $T_{\rm vec}$; $T_{\rm vec} \equiv T_{\rm full} - T_{\rm evo}$ = (4,242 sec) - (2,227 sec) = (2,015 sec) in Table 3.2. The time $T_{\rm vec}$ is contributed mainly by the backward transformation subroutine ($T_{\rm BACK}$ =1,892 sec) in Table 3.3, because the backward transformation subroutine in ELPA2 uses a characteristic two-step algorithm (See Sec. 4.3 of Ref. [31]).

Table 3.2: Selected results of the benchmark. The elapsed time for the full (eigenpair) calculation (T_{full}) and that for the eigenvalue-only calculation (T_{evo}) with the workflows. The recorded time is the best data among ones with different numbers of the used nodes. The number of used nodes (P) for the best data is shown within parentheses. The best data among the workflows are labelled by '[B]'. The saturated data are labelled by '[S]'. The workflow D' on Altix is that without the SSE optimized routine of the 'ELPA2' SEP solver. See the text for details.

Size M/Machine	WF	T_{full} (sec)	T_{evo} (sec)
1,000,080/FX10	*G*	39,919 (P = 4,800)	35,103 (P = 4,800)
430,080/K	*A*	11,634 (P = 10,000)	10,755 (P = 10,000)
	B	8,953 (P = 10,000)	8,465 (P = 10,000)
	C	5,415 (P = 10,000)	4,657 (P = 10,000)
	D	4,242 (P = 10,000)	2,227 (P = 10,000)[B]
	E	2,990 (P = 10,000)	2,457 (P = 10,000)
	F	2,809 (P = 10,000)	2,416 (P = 10,000)
	G	2,734 (P = 10,000)[B]	2,355 (P = 10,000)
	H	3,595 (P = 10,000)	3,147 (P = 10,000)
90,000/K	*A*	590 (P = 4,096)	551 (P = 4,096)
	B	493 (P = 1,024)[S]	449 (P = 1,024)[S]
	C	318 (P = 4,096)	298 (P = 4,096)
	D	259 (P = 4,096)	190 (P = 4,096)[B]
	E	229 (P = 4,096)[B]	194 (P = 4,096)
	F	233 (P = 4,096)	210 (P = 4,096)
	G	258 (P = 4,096)	240 (P = 4,096)
	H	253 (P=4,096)	236 (P=4,096)
90,000/FX10	*A*	1,248 (P = 1,369)	1,183 (P = 1,369)
	B	691 (P = 1,024)[S]	648 (P = 1,024)[S]
	C	835 (P = 1,369)	779 (P = 1,369)
	D	339 (P = 1,369)	166 (P = 1,024)[B][S]
	E	262 (P = 1,369)	233 (P = 1,024)[S]
	F	250 (P = 1,369)[B]	222 (P = 1,369)
	G	314 (P = 1,024)[S]	283 (P = 1,024)[S]
	H	484 (P=1,369)	456 (P=1,369)
90,000/Altix	*A*	1,985 (P = 256)	1,675 (P = 256)
	B	1,883 (P = 256)	1,586 (P = 256)
	C	1,538 (P = 256)	1,240 (P = 256)
	D	1,621 (P = 256)	594 (P = 256)
	D'	2,621 (P = 256)	585 (P = 256)[B]
	E	1,558 (P = 256)	1,287 (P = 256)
	F	1,670 (P = 256)	1,392 (P = 256)
	G	1,453 (P = 256)[B]	1,170 (P = 256)
	H	2,612 (P=256)	2,261 (P=256)

(Continuation of Table 3.2)

Size M/Machine	WF	$T_{\rm full}$ (sec)	$T_{\rm evo}$ (sec)
22,500/K	*A*	65.2 (P = 1,024)	59.6 (P = 256)
	B	45.8 (P = 1,024)[S]	43.2 (P = 1,024)[S]
	C	41.7 (P = 2,025)	37.8 (P = 2,025)
	D	28.4 (P = 2,025)	22.6 (P = 1,024)
	E	28.3 (P = 2,025)[B]	22.6 (P = 1,024)[B]
	F	28.8 (P = 1,024)[S]	26.9 (P = 1,024)[S]
	G	29.7 (P = 1,024)[S]	27.8 (P = 1,024)[S]
	H	39.3(P=1024)[S]	37.5(P=1024)[S]
22,500/FX10	*A*	126.2 (P = 256)	118.1 (P = 256)
	B	71.3 (P = 256)[S]	67.1 (P = 256)[S]
	C	103.5 (P = 256)[S]	96.3 (P = 256)[S]
	D	30.5 (P = 529)[B]	24.4 (P = 529)[B]
	E	34.3 (P = 256)[S]	31.2 (P = 256)[S]
	F	32.1 (P = 529)	29.4 (P = 529)
	G	45.3 (P = 529)	42.5 (P = 529)
	H	74.9(P=529)	72.2 (P=529)
22,500/Altix	*A*	51.4 (P = 256)	42.1 (P = 256)
	B	70.0 (P = 256)	50.7 (P = 256)
	C	45.6 (P = 256)	35.5 (P = 256)
	D	41.8 (P = 256)	22.3 (P = 256)[B]
	D′	59.6 (P = 256)	21.8 (P = 256)[B]
	E	32.3 (P = 256)[B]	26.7 (P = 256)
	F	48.5 (P = 256)	37.3 (P = 256)
	G	57.2 (P = 256)	39.6 (P = 256)
	H	71.2 (P=256)	64.1 (P=256)

Table 3.3: Decomposition of the elapsed time (sec) of the SEP solvers with M=430,080 and $P = 10,000$. See the text for the subroutine names of 'TRD/BAND' , 'D&C' and 'BACK'.

SEP solver	TRD/BAND	D&C	BACK	Total ($T_{\rm SEP}$)
SCLA	3,055	465	633	4,152
ELPA2	966	141	1,892	2,999
ELPA1	1,129	138	400	1,667
EIGS	1,058	196	265	1,521
EIGX	828	390	255	1,473

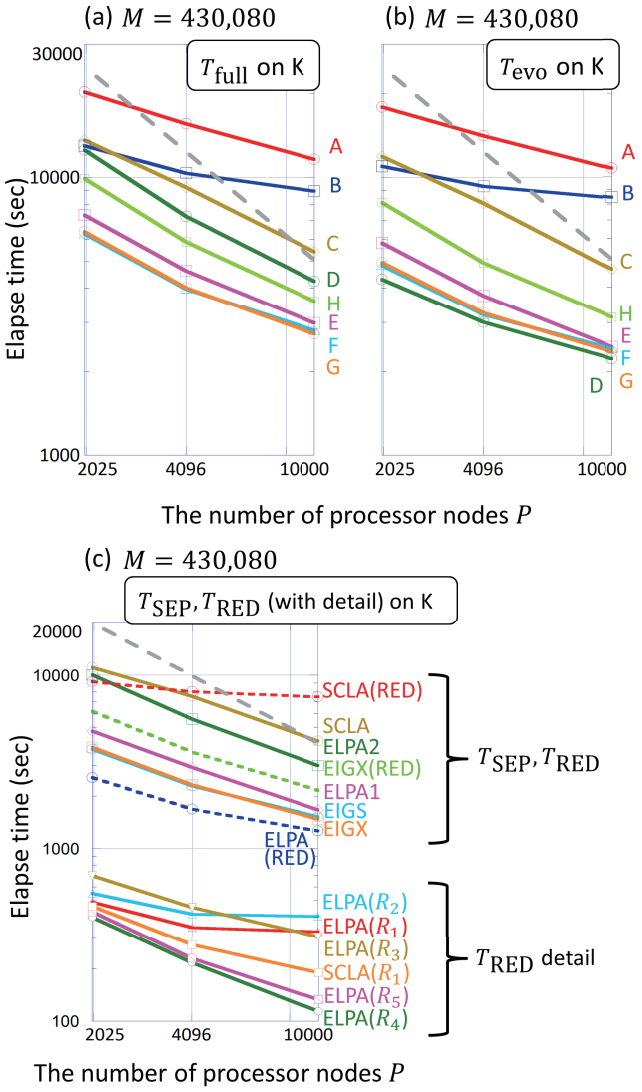

Figure 3.3: Results with M=430,800 on the K computer. The elapsed times are plotted with the workflows for the (a) full (T_{full}) and (b) eigenvalue-only (T_{evo}) calculations. (c) The decomposed times for the SEP solver (T_{SEP}) and for the reducer (T_{RED}) are plotted. The routines for the reducers is labeled by '(RED)'. Detailed decomposed times for subprocedures of the ELPA style reducer and the Cholesky decomposition in the ScaLAPACK style reducer are also plotted in (c). The ideal speedup in parallelism is drawn as a dashed gray line.

3.4.2 Benchmark with the matrix sizes of M=90,000, 22,500

The benchmark with the smaller matrix sizes of $M = 90,000$ and 22,500 are also investigated. The maximum number of used processor nodes is $P_{\rm max}$ = 4,096, 1,039 and 256, on the K computer, FX10, and Altix, respectively. [7] Figures 3.4 and 3.5 show the data with M=90,000 and with M=22,500, respectively. The decomposed times are shown in Fig. 3.6. Table 3.2 shows the best data for each workflow among the different numbers of used nodes.

The results will help general simulation researchers to choose the solver and the number of used nodes, since the elapsed times in Table 3.2 are less than a half hour and such calculations are popular 'regular class' jobs among systematic investigations. [8]

Here, the results are discussed; (I) Table 3.2 shows that the smallest elapsed time in the full calculation appears among the workflows with the ELPA style reducer (the workflows D, E, F, and G) and that in the eigenvalue-only calculation appears with the workflow D. The above features are consistent to the results in the previous subsection. (II) Unlike the result in the previous subsection, the speed up is sometimes saturated. An example is observed in Fig. 3.5 (a), in the full calculation with M=22,500 on the K computer, because the elapsed time in the workflow F gives a minimum as the function of P at P=1,024. The decomposition analysis of Fig. 3.6(b) indicates that the saturation occur both for the SEP solver and the reducer, which implies that the improvement both on the SEP solver and the reducer is desirable. The saturated cases are marked in Table 3.2 with the label of '[S]'. [9] (III) Finally, the SSE-optimized routine in the workflow D is compared with the generic routine in the workflow D' in the case of $M = 90,000$ on Altix with P =256. The SSE-optimized routine is prepared only in the backward transformation process. Since the process with the SSE-optimized routine or the generic one gives the elapsed time of T_{BACK} = 929 sec or T_{BACK} = 1,872 sec, respectively, the process is accelerated with the SSE-optimized routine by $1,872$ sec / 929 sec ≈ 2.02. As shown in Table 3.2, the full calculation is accelerated with the SSE-optimized routine by $2,621$ sec / $1,621$ sec ≈ 1.62.

[7] It is observed on Altix that the 'ELPA2' and 'ELPA2'' SEP solver required non-blocking communication requests beyond the default limit number of $N_{\rm MPI\_MAX} = 16,384$ and the job stopped with an MPI error message. Then the limit number was increased to $N_{\rm MPI\_MAX} = 1,048,576$, the possible maximum of the machine by the environment variable 'MPI_REQUEST_MAX' and the calculations were completed.

[8] One should remember that supercomputers are usually shared by many researchers who run many calculations in similar problem sizes successively and/or simultaneously.

[9] No saturation is found on Altix, unlike on the K computer and FX10, partially because the maximum number of used nodes ($P_{\rm max}$ = 256) is smaller.

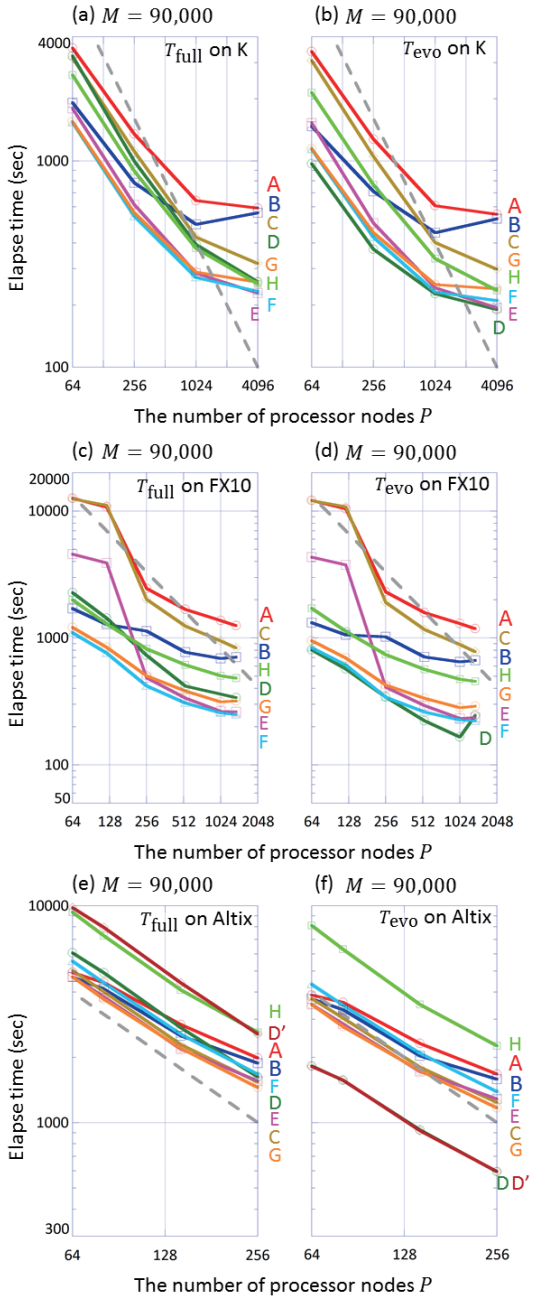

Figure 3.4: Benchmark with the matrix size of M=90,000, (I) on the K computer for the (a) full (eigenpair) and (b) eigenvalue-only calculation, (II) on FX10 for the (c) full and (d) eigenvalue-only calculation, (III) on Altix for the (e) full and (f) eigenvalue-only calculation. The ideal speedup in parallelism is drawn as a dashed gray line.

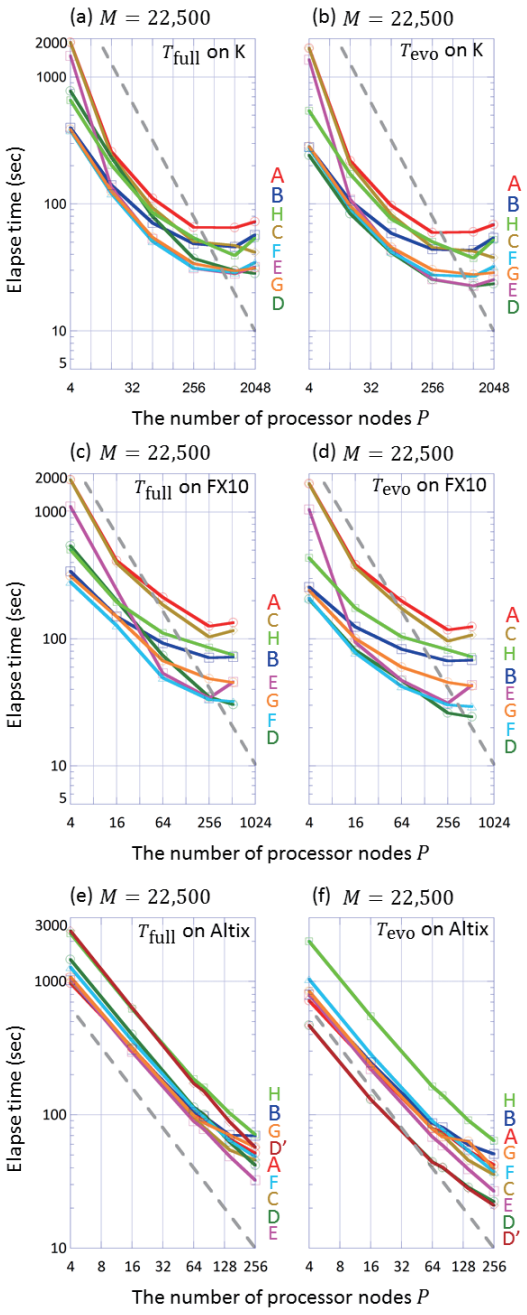

Figure 3.5: Benchmark with the matrix size of M=22,500, (I) on the K computer for the (a) full (eigenpair) and (b) eigenvalue-only calculation, (II) on FX10 for the (c) full and (d) eigenvalue-only calculation, (III) on Altix for the (e) full and (f) eigenvalue-only calculation. The ideal speedup in parallelism is drawn as a dashed gray line.

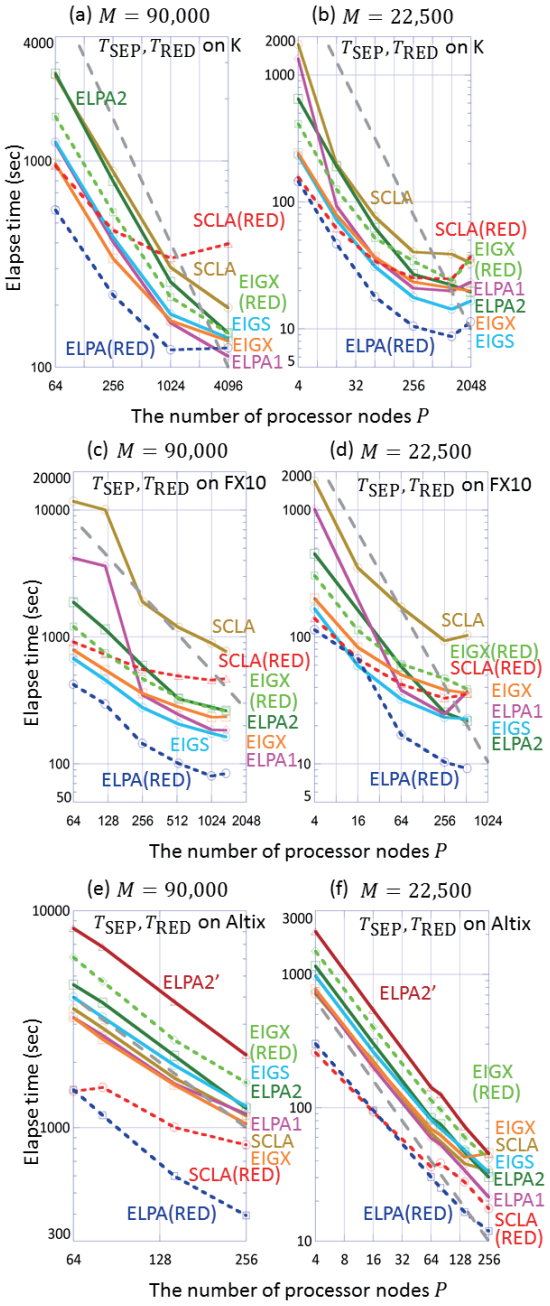

Figure 3.6: Decomposition analysis of the elapsed time into those of the SEP solver and the reducer (I) on the K computer with (a) M=90,000 and (b) M=22,500, (II) on FX10 with (c) M=90,000 and (d) M=22,500, (III) on Altix with (c) M=90,000 and (d) M=22,500. The routines for the reducers is labeled by '(RED)'. The 'ELPA2'' SEP solver is that without the SSE optimized routine. The ideal speedup in parallelism is drawn as a dashed gray line.

3.4.3 Benchmark for a million dimensional matrix

Finally, the benchmark for a million dimensional matrix is discussed. A press release at 2013 [45] reported, as a world record, a benchmark of a million dimensional SEP carried out by EigenExa, in approximately one hour, on the full (82,944) nodes of the K computer. An eigenvalue problem with a million dimensional matrix (M=10^6) seems to be the practical limitation of the present supercomputer, owing to the $O(M^3)$ operation cost.

We calculated a million dimensional GEP at Dec. 2014 on the full (4,800) nodes of Oakleaf-FX. Since computational resource was limited, only one calculation was carried out with the workflow G, because it gives the best data among those with $M = 430,080$ in Table 3.2. The calculation finished in approximately a half day, as shown in Table 3.2 ($T_{\rm full}$ = 39,919 sec and $T_{\rm evo}$ = 35,103 sec). The elapsed time of the reducer ($T_{\rm RED} = T_{\rm full} - T_{\rm SEP}$ = 15,179 sec) is smaller than but comparable to that of the SEP solver ($T_{\rm SEP} = 24,740$). The benchmark proved that the present code qualifies as software applicable to massively parallel computation with up to a million dimensional matrix.

The million dimensional GEP was also solved with the workflow G by the K computer. The elapsed time was $T_{\rm elaps} = 9,939$ sec with $n_{\rm node} = 41,472$ nodes and $T_{\rm elaps} = 5,516$ sec on the full system (with $n_{\rm node} = 82,944$ nodes).

3.5 Discussions

3.5.1 Preparation of initial distributed data

In the benchmark, the initial procedures including file reading are carried out for the preparation of distributed data. Its elapsed time is always small and is ignored in the previous section. [10] These procedures, however, may consume significant elapsed times, when the present solver is used as a middleware with real applications. The discussions on such cases are beyond the present scope, since they depend on the program structure of the real applications. Here, several comments are added for real application developers; In general, the matrix data cost is, at most, $O(M^2)$ and the operation cost is $O(M^3)$ in the dense-matrix solvers and one should consider a balance between them. In the case of $M = 430,080$, for example, the required memory size for all the matrix elements is 8 B $\times M^2 \approx$ 1.5 TB, which can not be stored on a node of the K computer. Therefore, the data should be always distributed. In the real application (ELSES), the initial distributed data

[10] In the case of the workflow G on the K computer with M=430,080 and P=10,000, for example, the elapsed time of the initial procedures is $T_{\rm ini}$ =123sec and is much smaller than that of the total computation ($T_{\rm tot} = 2,734$sec. See Table. 3.2). It is noteworthy that the present matrices are sparse, as explained in Sec. 3.1.

Table 3.4: The elapsed times for data conversion; '(b → 1)', '(1 → b)' and 'T_{RED}' are the times in seconds for, the conversion process from block cyclic into cyclic distributions, the inverse process and the whole reducer procedure, respectively. The saturated data of T_{RED} are labelled by '[S]'. The 'ratio' is ((b → 1) + (1 → b)) / T_{RED}.

Size M	Machine(P)	(b → 1)	(1 → b)	T_{RED}	ratio[%]
1,008,000	FX10(4,800)	51.4	51.7	8,208	1.26
430,080	K(10,000)	13.4	6.48	1,261	1.58
90,000	K(4,096)	6.89	0.797	124[S]	6.21
	FX10(1,369)	1.89	0.973	84.0[S]	3.41
	Altix(256)	2.01	2.02	394	1.02
22,500	K(2,025)	0.571	0.610	11.3[S]	10.4
	FX10(529)	0.328	0.176	9.20	5.48
	Altix(256)	0.120	0.279	11.9	3.35

is prepared, when only the required elements are generated and stored on each node.

3.5.2 Data conversion overhead

As explained in Sec. 3.3, several workflows require data conversion processes between distributed data formats, since ScaLAPACK and ELPA use block cyclic distribution with a given block size $n_{\mathrm{block}}(> 1)$ and EigenExa uses cyclic distribution ($n_{\mathrm{block}} \equiv 1$). In the present benchmark, the block size n_{block} in ScaLAPACK and ELPA was set to be $n_{\mathrm{block}} = 128$, a typical value. Consequently, the workflows B, F, G require data conversion processes. In the present thesis, the elapsed time of the conversion procedures is included in the reducer part (T_{red}).

Table 3.4 shows the elapsed time for the data conversion. The elapsed times are shown in the cases with the maximum numbers of used nodes ($P = P_{\mathrm{max}}$) among the present benchmark. Two data conversion procedures are required. One is the conversion from the block cyclic distribution into the cyclic distribution, shown as '(b → 1)' in Table 3.4 and the other is the inverse process shown as '(1 → b)'. The two procedures are carried out, commonly, by the 'pdgemr2d' routine in ScaLAPACK.

Table 3.4 indicates that the overhead of the data conversion procedures is always small and is not the origin of the saturation. In general, the conversion requires an $O(M^2)$ operation cost, while the calculation in a dense-matrix solver requires an $O(M^3)$ operation cost. The fact implies the general efficiency of hybrid solvers, at least, among dense-matrix solvers.

3.5.3 Reducer analysis and future outlook

The decomposition analysis of the ELPA-style reducer is focused on, since the ELPA-style reducer is fastest among the three libraries. Figure 3.3 (c) shows the case on the K computer with M=430,080. The elapsed times of the subprocedures of the ELPA-style reducer are plotted; 'ELPA(R_1)' is the Cholesky factorization of Eq. (3.4), 'ELPA(R_2)' is the explicit calculation of the inversion $R = U^{-1}$ of the Cholesky factor U, 'ELPA(R_3)' and 'ELPA(R_4)' are the successive matrix multiplication of Eq. (3.5) and 'ELPA(R_5)' is the backward transformation of eigenvectors by matrix multiplication of Eq. (3.6). The elapsed times of the Cholesky factorization in the ScaLAPACK style reducer is also plotted as 'SCLA(R_1)' as a reference data. The same decomposition analysis is carried out also for other cases, as shown in Fig. 3.7. One can observe that the Cholesky factorization of the ELPA-style reducer does not scale and sometimes is slower than that of the ScaLAPACK reducer. In particular, the saturation of the ELPA-style reducer is caused by that of the Cholesky factorization in Fig. 3.7 (a)(b)(c).

The above observation implies that the reducer can be a serious bottleneck in the next-generation (exa-scale) supercomputers, though not in the present benchmark. One possible strategy is the improvement on the Cholesky factorization for better scalability and another is the development of a reducer without the Cholesky factorization, as in the EigenExa-style reducer.

As a future outlook, the present code for the hybrid solvers is planned to be extended by introducing the solvers with different mathematical foundations. A candidate is the parallel block Jacobi solver [46, 47]. Since the solver is applicable only to standard eigenvalue problems, the hybrid solver enables us to use the solver in generalized eigenvalue problems.

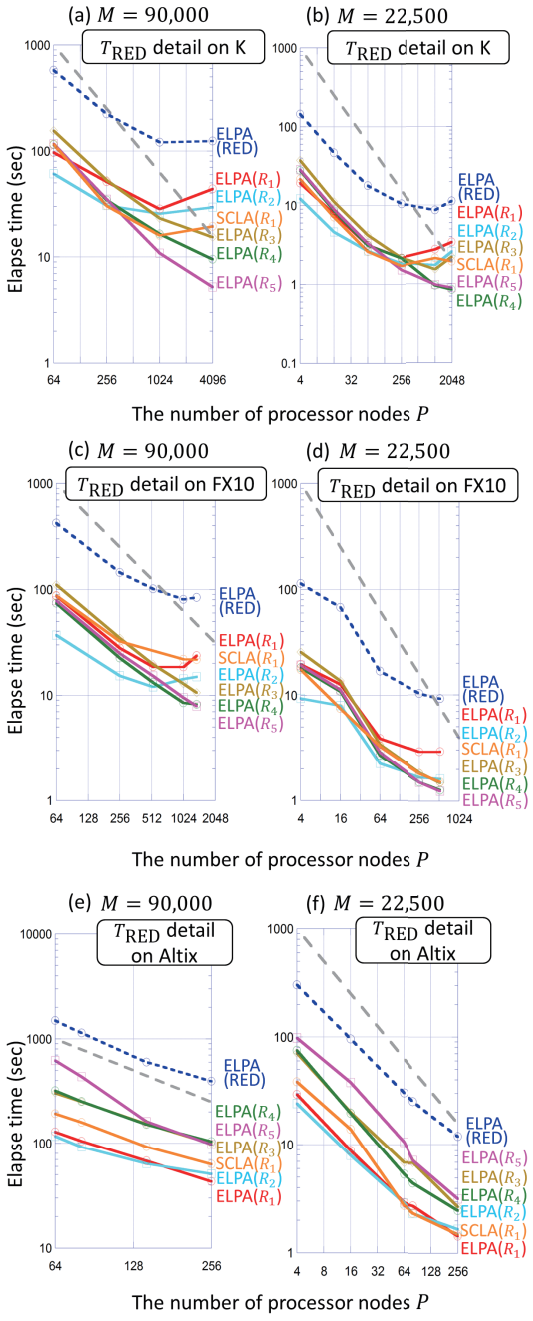

Figure 3.7: Decomposition analysis of the elapsed time of subprocedures of the ELPA style reducer and the Cholesky factorization in the ScaLAPACK style reducer (I) on the K computer with (a) M=90,000 and (b) M=22,500, (II) on FX10 with (c) M=90,000 and (d) M=22,500, (III) on Altix with (c) M=90,000 and (d) M=22,500. The ideal speedup in parallelism is drawn as a dashed gray line.

3.6 Details of the code

3.6.1 Execution flow of EigenKernel

The code of the hybrid eigenvalue problem solver collection, named EigenKernel, is released at GitHub [29]. EigenKernel can be built to a static library and a standalone executable. The executable form of EigenKernel is called eigbench. execution flow of eigbench is as follows. The list mostly corresponds to the source file `src/main.f90`.

1. Read commandline arguments

   Read commandline arguments in the standard Unix style. eigbench can solve both of SEP and GEP in almost the same manner. In the minimum, eigbench takes one option (-s) to choose the solver name and one (in SEP) or two (in GEP) arguments of matrix file paths. Names of GEP solvers have the prefix of 'general_' like 'general_scalapack'. In commandline arguments, output file paths of eigenvalues (`-o`), inversed participation ratios (see Sec. refSEC-PR-INTRO) of eigenvectors (`-i`), eigenvectors (`-d`) and execution log file (`-l`) can be specified. The range of the indices of eigenvectors to be output can also be specified (`-p <start>,<end>`).

2. Read matrix files in MatrixMarket format and broadcasting

   Input file format in eigbench is the MatrixMarket format [48]. An input file is read only on the master node (0-th node in MPI) and broadcast to all the nodes. After broadcasting, matrix data are converted into a format for the main calculation. As explained in Sec. 3.5.2, EigenKernel uses block cyclic or (non-block) cyclic format in the main calculation.

3. Main solver routine call

   Call the specified SEP or GEP solver. The data conversion process runs when needed as explained in Sec. 3.5.2. When EigenKernel is used in the static library form, users can call only the main solver routines.

4. (Partially optional) output eigenvalues, inversed participation ratios and eigenvectors

   Output calculated eigenvalues, inversed participation ratios of eigenvectors and eigenvectors itself. Output of eigenvectors is optional process executed only when the `-p` option is specified because in most cases the eigenvector matrix is dense even if the input matrices are sparse and therefore requires large disk space. Also, output of eigenvectors is parallelized by MPI. Output eigenvector indices are partitioned into the compute nodes evenly in cyclic

manner. This parallelization is especially effective in systems which have a local file system for each compute node.

5. (Optional) compute residual norms and orthogonality of calculated eigenpairs

   EigenKernel has two accuracy checking routines. One is to compute residual norms

   $$r_k \equiv \|A\boldsymbol{y}_k - \lambda_k B\boldsymbol{y}_k\| \ \ (k = 1, \ldots, M), \tag{3.10}$$

   and the other is to compute orthogonality of eigenvectors

   $$t \equiv \|Y^T BY - I\|_F \tag{3.11}$$

   where $\|\cdot\|_F$ is the Frobenius norm.

3.6.2 Quick start of EigenKernel

In minimum configuration, EigenKernel can be built with an MPI Fortran compiler and ScaLAPACK (without ELPA and EigenExa). Settings depend on a machine environment should be written in `Makefile.in`. The sample `Makefile.in` in the repository supposes there are mpif90 over gfortran as a fortran compiler and libgomp as an OpenMP library. Following commands will test solving a generalized eigenvalue problem with the matrix size of $M = 30$.

```
tar zxvf eigenkernel-*.tar.gz
cd eigenkernel-*
cp Makefile.in.gfortran.noext Makefile.in
make
mpirun -np 4 bin/eigbench -s general_scalapack \
  matrix/ELSES_MATRIX_BNZ30_A.mtx matrix/ELSES_MATRIX_BNZ30_B.mtx
```

After executing eigbench, there are output files named eigenvalues.dat, ipratios.dat, log.json. eigenvalues.dat and ipratios.dat contain computed eigenvalues and inversed participation ratios of computed eigenvectors respectively. log.json contains execution information such as given commandline options in the JSON format and elapsed time for some routines. In the execution command -s <solver> is a mandatory option to specify the solver routine. The general_scalapack solver is, of course, a pure ScaLAPACK solver. The last two arguments are paths to input matrix files in the MatrixMarket format. Note that the input matrices must be real symmetric and moreover the latter one must be positive definite. You can also solve standard eigenvalue problems.

```
mpirun -np 4 bin/eigbench -s scalapack \
  matrix/ELSES_MATRIX_VCNT400std_A.mtx
```

To utilize full functions of EigenKernel, it should be built linking ELPA and EigenExa. After installing them, edit Makefile.in and set LIBS variable properly to indicate the paths where .a and .mod are installed. Then EigenKernel with ELPA and EigenExa should be rebuilt like

```
emacs Makefile.in  # Edit $LIBS properly
make clean
make WITH_EIGENEXA=1 WITH_ELPA=1
```

If the rebuild is succeeded, truly hybrid solvers can be selected by the `-s` option.

```
mpirun -np 4 bin/eigbench -s general_elpa_eigensx \
  matrix/ELSES_MATRIX_VCNT900_A.mtx \
  matrix/ELSES_MATRIX_VCNT900_B.mtx
```

3.6.3 Use with real applications

EigenKernel can be built into a Fortran library as well as the standalone mini-application eigbench. For example, one of optional solvers in ELSES uses EigenKernel. The library is automatically generated in default build setting. A minimal example of calling EigenKernel library to solve a positive symmetric generalized eigenvalue problem is as follows.

```
integer :: ierr
type(process) :: proc
type(eigenpairs_types_union) :: eigenpairs
call mpi_init(ierr)
call setup_distribution(proc)
call reduce_generalized(dim, A, desc_A, B, desc_B)
call eigen_solver_scalapack_all(proc, desc_A, A, eigenpairs)
call recovery_generalized(dim, dim, B, desc_B, &
  eigenpairs%blacs%Vectors, eigenpairs%blacs%desc)
call mpi_finalize(ierr)
```

Call of the subroutines ‘mpi_init’ and ‘mpi_finalize’ is mandatory in MPI programs. ‘setup_distribution’ determines the processor grid layout in ScaLAPACK. ‘reduce_generalized’ and ‘recovery_generalized’ execute the reduction and the inverse reduction between GEP and SEP in the ScaLAPACK style, respectively. ‘eigen_solver_scalapack_all’ solves the reduced SEP. After calculation, eigenpairs are stored in the variable ‘eigenpairs’. Variables ‘A’, ‘B’, ‘desc_A’ and ‘desc_B’

are the standard distributed matrices and their descriptors in ScaLAPACK. There are also subroutines of solvers using external libraries like ELPA or/and EigenExa. They have a unified interface and users can easily exchange solver routines.

3.6.4 ELSES matrix library

ELSES Matrix Library [37] is a collection of matrix data generated by ELSES for quantum material simulations. They appears in standard or generalized eigenvalue problems. It contains from small (dimension $M = 30$) to very large ($M = 1,008,000$) matrices. The matrices are sparse, real-symmetric or Hermitian. The matrix data are recorded in the MatrixMarket format [48]. The atomic unit is used for energy unit. Some matrix packages contain a list of all eigenvalues and inversed participation ratios of eigenvectors.

Chapter 4

Extreme scalability of 10^8-atom simulation for organic materials

4.1 Problem and previous works

4.1.1 Ultra-flexible device and organic materials

Ultra-flexible devices are based on organic materials and play a crucial role among next-generation IoT products, such as display, battery and sensor. Organic materials form flexible atomic structures and enable ultra-thin, light, flexible (wearable) devices with a low fabrication cost [49, 50, 51]. Alan J. Heeger, Alan G. MacDiarmid and Hideki Shirakawa won the Nobel Prize in Chemistry in 2000 for their pioneering research on organic polymer devices in 1970's. The atomic structure of organic materials is disordered [52, 53, 54, 55] and the simulation of huge, 100-nanometer-scale, disordered systems are required. Nowadays many electronic state calculation codes are available but such large-scale calculations are far beyond their computational limit.

The present thesis reports that a new electronic state calculation code ELSES [7] is based on novel linear algebraic algorithms suitable to massive parallelism [56, 57, 58, 21, 59, 60, 23, 4, 5], and was applied to the above challenging industrial problem. Since the algorithm has a highly parallelizable mathematical structure, the code shows excellent results both in the strong scaling and the time-to-solution. The code realized calculations with one-hundred-million (10^8) atoms or 100-nm-scale systems, the largest system among electronic state calculations. Since the algorithms are mathematical, the method is applicable to various materials, such as semiconductor and metal. The algorithm is applicable also to other computational science area. The code has been developed for large-scale and/or fast electronic state calculations by academic-industrial collaboration. A recent collaboration research is one for organic device materials (Refs. [21, 60, 5] and

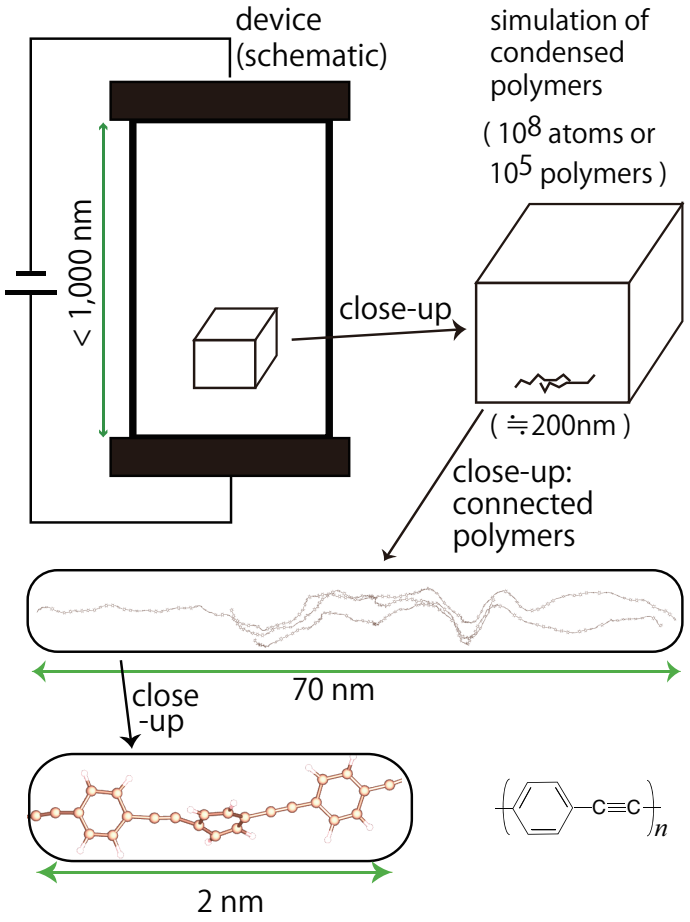

Figure 4.1: Overview of the present material research on organic polymer device; The comparison of the length scales in real device and the present simulation of poly-(phenylene-ethynylene) (PPE).

the present work) with Sumitomo Chemical Co., Ltd, and another is one for battery materials [61, 62] with Toyota Motor Corporation. The other application researches can be found in the reference lists of the above papers.

Figure 4.1 indicates the overview of the present material research. The length scales of real device and the present simulation are compared.

4.1.2 Electronic wavefunction in organic materials

The electric conductivity in organic devices and related materials like graphene stems from the characteristic electronic wavefunction called π wavefunction. A π wavefunction lies, typically, on a benzene ring. Figure 4.2(a) shows a π wavefunc-

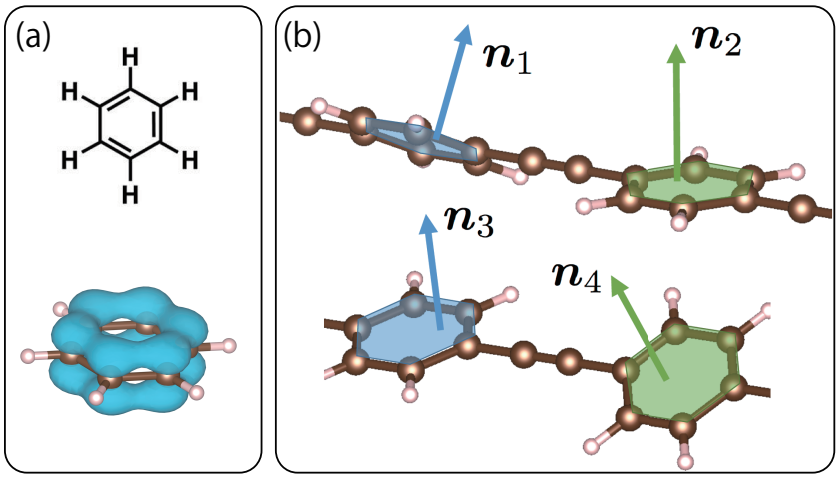

Figure 4.2: Organic materials and π electronic wavefunction. (a) Atomic structure of benzene and an example of π electronic wavefunction. (b) A schematic figure of condensed organic polymers in a disordered structure. The vectors of $\boldsymbol{n}_1, \boldsymbol{n}_2, \boldsymbol{n}_3, \boldsymbol{n}_4$ are defined as ones perpendicular to the benzene ring planes.

tion in benzene (C_6H_6). The charge of the π wavefunction of $|\phi(\boldsymbol{r})|^2$ is depicted as isosurfaces. The wavefunction extends to the direction perpendicular to the plane of benzene, which is the origin of the strong anisotropy in electric conductivity. Figure 4.2(b) shows schematically the structure of condensed polymers in real devices. Disorder appears both in intra- and inter-chain structures. Here the vectors of $\boldsymbol{n}_1, \boldsymbol{n}_2, \boldsymbol{n}_3, \boldsymbol{n}_4$ are depicted as ones perpendicular to the benzene ring planes. When neighboring two vectors of $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ are almost parallel ($\boldsymbol{n}_i \parallel \boldsymbol{n}_j$), the π wavefunctions on their benzene rings connect easily with each other and the electrical current can propagate between them.

In conclusion, quantum simulations with π-type wavefunctions are crucial for the organic electronic devices, which is a challenging problem in computational science.

4.1.3 Concept of order-N method

'Order-N' method is a general name of the large-scale calculation methods in which the computational cost is O(N) or proportional to the number of atoms N. The present code is one of them. Other order-N calculation codes were also developed [63, 64, 65, 66].

A key concept for the order-N method was stated by Walter Kohn. His paper in 1996 shows that the above potential difficulty in electronic state calculation can

be avoided, when the theory is not based on an eigenvalue equation [67].

The theory [67] focuses on a physical quantity defined as

$$\langle X \rangle \equiv \sum_k f(\varepsilon_k)\, \boldsymbol{y}_k^{\mathrm{T}} X \boldsymbol{y}_k, \tag{4.1}$$

with a given sparse real-symmetric matrix X. Equation (4.1) is found in elementary textbooks of electronic state calculations. Here the function of $f(\varepsilon)$ is a weight function, called Fermi function, and is defined as

$$f(\varepsilon) \equiv \left\{ 1 + \exp(\frac{\varepsilon - \mu}{\tau}) \right\}^{-1}. \tag{4.2}$$

The weight function is a 'smoothed' step function with a smoothing parameter $\tau(> 0)$, because the Heaviside step function will appear in the limiting case of $\tau \to +0$ ($f(\varepsilon) = 1(\varepsilon < \mu)$ and $f(\varepsilon) = 0(\varepsilon > \mu)$). The smoothing parameter τ has a physical meaning of temperature of electrons. The parameter μ is the chemical potential and the value should be determined so that the number of electrons is the correct one. The case in $X = H$, for example, gives the electronic energy

$$\langle H \rangle \equiv \sum_k f(\varepsilon_k)\, \varepsilon_k. \tag{4.3}$$

Other quantities, such as the force on atom and the density of states (eigenvalue histogram), are also described in the above form.

A quantity in Eq.(4.1) is transformed into the trace form of

$$\langle X \rangle = \mathrm{Tr}[\rho X] \tag{4.4}$$

with the density matrix

$$\rho \equiv \sum_k f(\varepsilon_k)\, \boldsymbol{y}_k\, \boldsymbol{y}_k^{\mathrm{T}}. \tag{4.5}$$

The order-N property can be found, as follows; A density matrix element ρ_{ji} is *not* required when $X_{ij} = 0$, because the element ρ_{ji} does not contribute to the physical quantity

$$\mathrm{Tr}[\rho X] = \sum_{i,j} \rho_{ji} X_{ij}, \tag{4.6}$$

even if its value is nonzero ($\rho_{ji} \neq 0$). Consequently, the number of the required density matrix elements ρ_{ji} is O(N). The above principle is called 'quantum locality' or 'nearsightedness principle' [67].

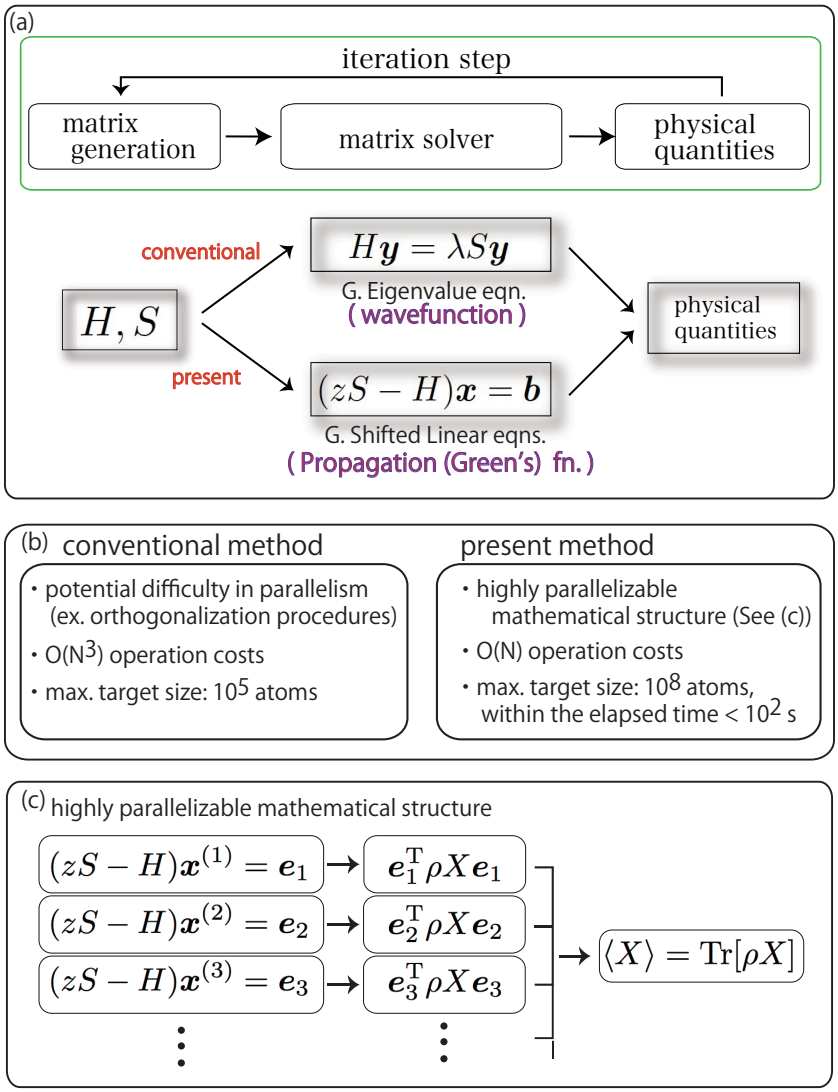

Figure 4.3: Overview of the algorithm for large-scale electronic state calculations in the extreme strong scaling; (a) The ground algorithm design. (b) Comparison between the conventional method with eigenvalue equation and the present method with generalized shifted linear equations. (c) Schematic figure of the highly parallelizable mathematical structure.

4.1.4 Highly parallelizable mathematical structure

The above formulation has a highly parallelizable mathematical structure and the original problem is decomposed mathematically into parallel subproblems. The

trace in Eq. (4.4) can be decomposed as

$$\langle X \rangle = \mathrm{Tr}[\rho X] = \sum_{j}^{M} \boldsymbol{e}_j^{\mathrm{T}} \rho X \boldsymbol{e}_j, \tag{4.7}$$

with the j-th unit vector of $\boldsymbol{e}_j \equiv (0, 0, 0, ...1_j, 0, 0, ..., 0)^{\mathrm{T}}$. Here the quantity of $\boldsymbol{e}_j^{\mathrm{T}} \rho X \boldsymbol{e}_j$ is called 'projected physical quantity', because the quantity is defined by the projection onto the vector of $\boldsymbol{e}_j$. The essence of the parallelism is the fact that the projected physical quantity of $\boldsymbol{e}_j^{\mathrm{T}} \rho X \boldsymbol{e}_j$ is calculated almost independently among different indices of j.

4.2 Ground algorithm design

4.2.1 Generalized shifted linear equations

The ground algorithm design of the present method is summarized in Fig. 4.3(a). The method is based not on the eigenvalue equation of Eq. (2.12) but on linear equations in the form of

$$(zS - H)\boldsymbol{x} = \boldsymbol{b}. \tag{4.8}$$

Here z is a (complex) energy value and the matrix of $(zS - H)$ can be non-Hermitian. The vector $\boldsymbol{b}$ is an input and the vector $\boldsymbol{x}$ is the solution vector. A set of linear equations in the form of Eq. (4.8) with different energy values $(z = z_1, z_2, ...)$ is called generalized shifted linear equations. The equations in the case of $S = I$ are called shifted linear equations.

The generalized shifted linear equations are solved on an iterative Krylov-subspace solver. Such solvers are sometimes called shifted Krylov-subspace solvers. Krylov subspace is defined as the linear space of

$$K_\nu(A; \boldsymbol{b}) \equiv \mathrm{span}[\boldsymbol{b}, A\boldsymbol{b}, A^2\boldsymbol{b}, ..., A^{\nu-1}\boldsymbol{b}], \tag{4.9}$$

with a given vector $\boldsymbol{b}$ and a given square matrix A. Krylov subspace solver is the solver in which the solution is calculated within a Krylov subspace. An example is Conjugate Gradient method and the dimension of the subspace ν is the number of iterations.

The use of Eq. (4.8) results in the Green's (propagation) function formalism, since the solution $\boldsymbol{x}$ of Eq. (4.8) is written formally as

$$\boldsymbol{x} = G\boldsymbol{b} \tag{4.10}$$

with the Green's function $G \equiv (zS - H)^{-1}$. The Green's function and the eigenvectors holds the relationship of

$$G(z) = \sum_{k}^{M} \frac{\boldsymbol{y}_k \boldsymbol{y}_k^{\mathrm{T}}}{z - \varepsilon_k}. \tag{4.11}$$

The density matrix is also given by the Green's function as

$$\rho = \frac{-1}{\pi} \int_{-\infty}^{\infty} f(\varepsilon) \,\mathrm{Im}[G(\varepsilon + i0)] \, d\varepsilon. \tag{4.12}$$

The use of the (generalized) shifted linear equations, instead of eigenvalue equations, is a powerful strategy among large scale calculations. Such methodologies have been investigated, in particular, from 2000's, partially because the strategy is suitable to parallelism. Since the solver algorithms are mathematical, they are applicable to many scientific areas, such as, QCD [68], large-scale electronic state calculation [56, 57, 58, 21], quantum many-body electron problem [69], nuclear shell model problem [70], first-principle electronic excitation problem [71], and first-principle transport calculation [72].

4.2.2 Calculation of projected physical quantity

In the present method, the formalism has a highly parallelizable mathematical structure and the projected physical quantity of $\boldsymbol{e}_j^{\mathrm{T}} \rho X \boldsymbol{e}_j$, should be calculated as explained in Sec. 4.1.4. Figure 4.3(b) shows the comparison between the conventional method with eigenvalue equation and the present method with generalized shifted linear equations. The parallel procedure is illustrated in Fig. 4.3(c).

In the code, the projected physical quantity of $\boldsymbol{e}_j^{\mathrm{T}} \rho X \boldsymbol{e}_j$ is obtained from the generalized shifted linear equations of

$$(zS - H)\boldsymbol{x}^{(j)} = \boldsymbol{e}_j \tag{4.13}$$

within an iterative Krylov-subspace solver, called multiple Arnoldi solver [21]. In short, Eq. (4.13) is solved within the subspace of

$$\mathcal{L}_\nu(\boldsymbol{e}_j) \equiv K_{\nu/2}(H; \boldsymbol{e}_j) \oplus K_{\nu/2}(H; S^{-1}\boldsymbol{e}_j) \tag{4.14}$$

with an even number of ν. The number ν is typically, $\nu = 30 - 300$ and the calculations in the present thesis was carried out with $\nu = 30$ as in the previous one [21]. The second term in the right hand side of Eq. (4.14) appears so as to satisfy several conservation laws [21]. A reduced (small) $\nu \times \nu$ eigenvalue equation is solved and the solution vector of Eq. (4.13) is given by

$$\boldsymbol{x}^{(j)} := G^{(j)}(z)\boldsymbol{e}_j \tag{4.15}$$

with

$$G^{(j)}(z) \equiv \sum_{m}^{\nu} \frac{\boldsymbol{v}_m^{(j)} \boldsymbol{v}_m^{(j)\mathrm{T}}}{z - \varepsilon_m^{(j)}}. \tag{4.16}$$

Here $\varepsilon_m^{(j)}$ and $\boldsymbol{v}_m^{(j)}$ is an eigenvalue and eigenvector of the reduced equation ($m = 1, 2, ...\nu$). When the Green's function of G in Eq. (4.12) is replaced by $G^{(j)}(z)$ in Eq. (4.16), the projected physical quantity with the index of j is given by

$$\begin{aligned} \boldsymbol{e}_j^{\mathrm{T}} \rho X \boldsymbol{e}_j &:= \frac{-1}{\pi} \int_{-\infty}^{\infty} f(\varepsilon)\, \mathrm{Im}[\boldsymbol{e}_j^{\mathrm{T}} G^{(j)}(\varepsilon + i0) X \boldsymbol{e}_j]\, d\varepsilon \\ &= \sum_{m}^{\nu} f(\varepsilon_m^{(j)}) \boldsymbol{e}_j^{\mathrm{T}} \boldsymbol{v}_m^{(j)}\, \boldsymbol{v}_m^{(j)\mathrm{T}} X \boldsymbol{e}_j. \end{aligned} \tag{4.17}$$

Equation (4.17) will be exact, if the subspace dimension of ν increases to the original matrix dimension ($\nu = M$). As an additional technique in large-scale calculations, the real-space projection method [21] was also used. In short, the Krylov subspace is generated by a Hamiltonian matrix projected in real space $H^{(j)} \equiv P^{(j)} H P^{(j)}$, where the projection matrix $P^{(j)}$ projects a function onto the spherical region whose center is located at the atomic position of the j-th atomic basis function ($\chi_j(\boldsymbol{r})$). The radius of the spherical region is determined with an input integer parameter κ, so that the region contains κ atoms or more. The same technique is used also for the overlap matrix S. The value of κ is set to $\kappa = 100$ in the present thesis as in the previous one [21]. As results, numerical problems in the form of Eq. (4.13) are solved with the matrix size of, typically, $M' = 200 - 400$ among the simulations of the present thesis.

4.3 Methodological details for large scale calculations

4.3.1 Quantum molecular dynamics simulation for organic polymers

The present code is written in Fortran 90 with the MPI/OpenMP hybrid parallelization and the main purpose is quantum molecular dynamics simulation. Here an electron is treated as a quantum mechanical wave in the Green's (propagation) function formalism, while an atom (a nucleus) is treated as a classical particle in Newtonian equation of motion

$$M_I \frac{d^2 \boldsymbol{R}_I}{dt^2} = \boldsymbol{F}_I, \tag{4.18}$$

where M_I and $\boldsymbol{R}_I$ are the mass and the position of the I-th atom and $\boldsymbol{F}_I$ is the force on the I-th atom. Other variables, such as the electronic charge on each atom $\{q_I\}_I$, can be calculated also in the dynamical simulation. As technical details, the quantum molecular dynamics simulations are realized with first-principles-based modeled (tight-binding-form) theory. See Refs. [21, 5] and the references therein for details. In the code, these variables that consume O(N) memory cost are stored redundantly among nodes. All the matrix elements of H and S are generated from these variables on each node without any inter-node communications. Several matrix elements are generated redundantly among nodes. The force and charge on each atom is calculated in the trace form of Eq. (4.4)(See Ref. [21] for details). The time evolution was realized numerically by the velocity-Verlet algorithm. The finite-temperature simulations were realized by a thermostat method.

4.3.2 Details in parallelization

The details in parallelization are explained here, so as to clarify the condition of an efficient parallelism. In the present parallel computation, the parallel subproblems for calculating the projected physical quantity of $\boldsymbol{e}_j^{\mathrm{T}}\rho X\boldsymbol{e}_j$ should be carried out, as in Fig. 4.3(c). In the code, the basis index j is treated as a composite index. When a wavefunction is expressed by atomic orbitals in Eq. (2.11), a couple of atomic orbitals $\chi_j(\boldsymbol{r})$ are prepared at each atom. These atomic orbitals are different in shape. Among the present calculations, one (s-type) orbital is prepared at each hydrogen (H) atom, while four (s-, p_x-, p_y-, p_z-type) atomic orbitals at each carbon (C) atom. Consequently, the basis index j is the composite indices of the atom index J and the orbital index β ($j \equiv j(J,\beta)$). The projected physical quantity of $\boldsymbol{e}_j^{\mathrm{T}}\rho X\boldsymbol{e}_j$ is calculated under the double loop of J and β. In the present code, the outer loop, the loop with the atom index (J) is parallelized both by the MPI and OpenMP methods, and a projected physical quantity of $\boldsymbol{e}_j^{\mathrm{T}}\rho X\boldsymbol{e}_j$ is calculated within a thread (a CPU core). An efficient parallel computation is realized on the condition that the number of atoms is larger than that of cores ($N > n_{\mathrm{core}}$).

4.3.3 Sparsity of matrices

Sparsity of the matrices of H_{ij} and S_{ij} are explained briefly. See Ref. [21] and reference therein for details. As explained in the previous subsection, the indices i and j are the composite indices of the atom indices I and J and the orbital indices α and β, respectively ($i \equiv i(I,\alpha)$, $j \equiv j(J,\beta)$)). Therefore, an element of the matrices H and S is expressed by the four indices as $H_{I\alpha;J\beta}$ and $S_{I\alpha;J\beta}$, respectively. Since a matrix element value decreases quickly and monotonically as the function of the inter-atomic distance between the I-th and J-th atoms (r_{IJ}), a cutoff distance r_{cut} was introduced. A matrix element, $H_{I\alpha;J\beta}$ or $S_{I\alpha;J\beta}$, is ignored, if $r_{IJ} > r_{\mathrm{cut}}$, which

makes the matrices to be sparse. In the present thesis, the cutoff distance $r_{\rm cut}$ is set to be $r_{\rm cut} = 5{\rm au}(\approx 0.2646{\rm nm})$ for diamond crystal and $r_{\rm cut} = 10{\rm au}(\approx 0.5292{\rm nm})$ for condensed polymers. A longer cutoff distance is used for condensed polymers, so as to include the interaction between polymers.

4.3.4 MPI Communication

The communication among nodes is required, *only when* a summation is performed in the form of Eq. (4.7), as shown in Fig. 4.3(c). See Ref. [21] for detailed formulations. The code is written in the MPI/OpenMP hybrid parallelization and the summation is carried out hierarchically; First, the summation is carried out on each node by OpenMP directives and then the summation is carried out between nodes by `MPI_Allreduce()`. The pure MPI parallelism is also possible but consumes larger memory costs.

4.3.5 Use of Extensible Markup Language (XML) in File I/O

In our simulations, the main input and output files are written in the format of Extensible Markup Language (XML), since the XML format is simple, flexible and widely used on the Internet [23]. For example, the minimum information for an atom is written as follows;

```
<atom element="C">
  <position unit="angstrom"> 1.0d0 0.0d0 0.0d0 </position>
</atom>
```

The above description means that a carbon atom is located at the position of $(x, y, z) = (1, 0, 0)$, where Angstrom unit (1 Angstrom = 10^{-10} m) is used. The method for reading XML files should be chosen properly, according to the purpose. In our simulations, the XML files are read by two methods, Document Object Model (DOM) method and Simple API for XML (SAX) method. In general, the DOM method is easier in programming and results in huge memory and time consumption for large-size data, while the SAX method is more difficult in programming and results in tiny memory and time consumption. Two input files, configuration and structure XML files should be prepared for each simulation and they are quite different in their file size. (i) The configuration XML file describes calculation conditions, such as temperature of the system. A typical file consists of several tens of lines and the file size does not depend on the system size N. The configuration file is read by the DOM method, since its file size is always tiny. (ii) The structure XML file describes the atomic structure data, as shown in the above example. The structure XML file is read by the DOM method, since the file size is proportional to the system size N and can be huge. Since the atomic structure data

contains three (x, y, z) components in the double precision (8 B) value for each atom, the required data size with 10^7 (= 10 M) atoms is estimated to be 3×8 B $\times 10$ M = 240 MB. A typical size of the structure XML file with 10^7 atoms is one GB. In addition, the parallel file reading is used for large-scale calculations with split XML files for the structure file and gives a significant acceleration [38]. The K computer and FX10 support the parallel file IO, called 'rank directory' function, at the hardware level and are suitable to the parallel file reading with split XML files.

4.3.6 Performance tuning on the K computer

Performance tuning on two bottlenecks, matrix-vector multiplication (matvec) in the iterative Krylov-subspace solver and frequent access to shared disk space, was carried out on the K computer as follows.

For old implementation, performance profiling using the basic profiler on the K computer showed that matvec routine in the iterative Krylov-subspace solver took 39% execution time in the whole program. Because parallelism is already utilized by higher levels of the algorithm (see Sec. 4.3.2), only single node tuning was enough. Because the matvec routine in old implementation had a loop over four indices of matrix elements (see Sec. 4.3.3), loop length was short and locality of reference was low, which led to low efficiency. Accordingly, the matvec routine was split into two subroutines. The one is to generate matrix data in compressed row storage (CRS) format, and the other is to compute matvec with the CRS format matrix. Now the latter subroutine only has a loop over two indices.

Moreover, matrix data formats other than CRS are implemented for matvec routine, because the most efficient matrix data format depends on sparsity of matrices. In the present code, the conventional quadruple loop (called 'quad'), CRS format ('crs') and dense format ('dense') are implemented. For CRS and dense format, matvec routines that utilize symmetry of matrices are also implemented ('crs-sym' and 'dense-sym', respectively). The dense matvec routines use LAPACK library for efficiency. When sparsity of matrices is high and low, the CRS matvec routines and the dense ones will be efficient, respectively. The user can select an optimal matvec routine with a configuration file.

The elapsed times per step T_{elaps} by the five matvec routines (quad, crs, dense, crs-sym and dense-sym) were measured, for two systems that have the same number of atoms N=1,228,800 and different sparsity. The systems are a disordered systems of condensed polymers called 'P1' and the ideal diamond solid called 'D1'. Details of the systems and definition of T_{elaps} will be described in Sec. 4.4.2. The matrix generated from 'D1' is more sparse than 'P1' case. The number of nodes n_{node} was fixed to $2,592$. Table 4.1 shows comparison of T_{elaps} by the matvec routines. When the best matvec routines for each system are chosen, T_{elaps}

is 1.3 – 1.6 times faster than cases of the conventional quadruple loop code. It was also found that a dense matvec is suitable for the dense system 'P1', and a sparse matvec is suitable for the sparse system 'D1', as expected. However, utilization of symmetry of matrices did not contribute much to speed up because the matrix size for each node was small.

Table 4.1: The measured elapsed times T_{elaps} (sec) by the five matvec routines for the ideal diamond solid ('D1') and the condensed polymer system with ('P1'). The best elapsed time for each system is written in bold. Note that the matrices generated from the two systems have different size though the number of atoms N is the same, due to difference of elements.

System	quad	crs	crs-sym	dense	dense-sym
P1	16.5	11.2	11.3	10.6	**10.4**
D1	15.5	11.6	**11.5**	22.8	20.8

On the K computer, access to a disk space that is shared by all the compute nodes happens when, for example, nameless file in Fortran is used. Frequent access to the shared disk space can cause unexpected performance decrement. To avoid this, an environment variable TMPDIR which specifies the directory to create temporary files can be set to '.', which means the local filesystem of compute nodes.

4.4 Benchmarks and their analysis on the K computer

The present benchmarks were carried out so as to show (i) the extreme strong scaling, on the full system of the K computer, and (ii) the time-to-solution qualified for a practical research. Our target value of the time-to-solution is $T_{\text{elaps}} = 10^2$s for the elapsed time per step in a quantum molecular dynamics simulation, so that a dynamical simulation of $n_{\text{step}} = 10^3$ steps can be executed within one day ($T_{\text{elaps}} n_{\text{step}} = 10^5$s $\approx$ one day).

4.4.1 Architecture

The architecture of the K computer is explained briefly. The K computer achieved 10 PetaFLOPS Linpack performance for the first time in the world.

The K computer consists of 82,944 compute nodes which have one CPU, one Inter Connect Controller (ICC) , which is responsible for interconnection between

compute nodes, and 16GB off-chip memory. Each CPU has 8 cores and 6MB L2 cache shared by cores and each core has 4 double precision FMA units, which can be driven by 2×2way SIMD operations, and frequency is 2GHz. Therefore, total double precision peak performances are 16 GigaFLOPS per core, 128 GigaFLOPS per CPU and 10.62 PetaFLOPS per whole system, respectively [73].

The interconnection between compute nodes of the K computer is named 'Tofu' which constructs physical 6 dimensional mesh/torus network topology. User can use it as logical 3 dimensional torus network with redundant links which contribute robustness of the connection and flexibility of the operation. Each compute node has 6 links for 3 dimensional directions to connect neighboring nodes for each direction. Each link has 5GB/s full-duplex bandwidth for each direction.

The official operation of the K computer started at September 2012, the K computer has been kept one of the largest and fastest super computer in the world. Actually, the K computer keeps top level position in major benchmark rankings. For example, according to the TOP500 list NOVEMBER 2015 [74], the K computer is ranked in No.4 as well as No.1 in Graph500 list November 2015 [75] which is a benchmark for capability of graph analysis and No.2 in November 2015 HPCG Results [76] which is a benchmark for applications with iterative matrix solver in Conjugate Gradient method.

The results show that the K computer is applicable to general purpose computing.

In general massive parallel computations, collective communication uses synchronizations among compute nodes, which requires extremely high overhead caused by increased number of compute nodes. To address the problem, the ICC of the K computer provides hardware barrier function which is applied to `MPI_Barrier()`,
`MPI_Allreduce()`, and so on. In particular, the feature is strongly effective in reduction of `MPI_Allreduce()`.

4.4.2 Conditions of benchmarks

The benchmark jobs were executed in the MPI/OpenMP hybrid parallelization. The number of the MPI processes is set to that of the compute nodes and the number of the OpenMP threads is set to be eight, the number of cores per compute node. All the numerical calculations were carried out in double precision. The jobs were executed by specifying the three-dimensional node geometry on the K computer ($n_{\mathrm{node}} \equiv n_{\mathrm{node}}^{(x)} \times n_{\mathrm{node}}^{(y)} \times n_{\mathrm{node}}^{(z)}$) for optimal performance or minimum hop count. The number of used nodes (node geometry) is listed below; n_{node} $= 2,592 (= 12 \times 12 \times 18)$, $5,184 (= 12 \times 18 \times 24)$, $10,368 (= 18 \times 24 \times 24)$, $20,736 (= 24 \times 27 \times 32)$, $41,472 (= 27 \times 32 \times 48)$, and $82,944 (= 32 \times 48 \times 54$, the full system). The `MPI_Allreduce()` optimized on the K computer were used

[77].

The simulations were carried out for three disordered systems of condensed polymers, poly-(phenylene-ethynylene) (PPE). The three systems are called 'P100', 'P10' and 'P1' and contain N=101,606,400 ($\approx 10^8$ or 100M), N=10,137,600 ($\approx$ 10M) and N=1,228,800 ($\approx$1M) atoms, respectively. The periodic boundary condition is imposed. The size of the periodic simulation box is 134 nm $\times$ 134 nm $\times$ 209 nm for the 'P100' system. Among them, the systems with $N = 10^8, 10^7$ atoms are the main targets for the full-system calculation. As another reference data for an 'ideal' system without structural disorder, the simulations were carried out also for the ideal diamond solid called 'D100' that contains $N = 106,168,320$ ($\approx 10^8$ or 100M) atoms in the ideal periodicity, because the structural disorder can be a source of load imbalance.

A calculation job contains the initial preparation procedure including the reading of the input (atomic structure) file and the simulation procedure for five steps, as shown in Fig. 4.3(a). In the benchmark, a finite-temperature dynamics simulation was carried out with $N = 10^6, 10^7$ atoms, while a 'snapshot' simulation, a successive calculation of the given atomic structures with slightly different values of volume, was carried out with $N = 10^8$ atoms. The snapshot was carried out with $N = 10^8$ atoms, because of the limitation of built-in memory size (16GB per node). For example, the position data $\boldsymbol{R}_I \equiv (x_I, y_I, z_I)$ with $N = 10^8$ atoms occupies the memory size of 8B $\times 3N$ = 2.4GB. The present snapshot calculation consumes 9 GB per node. A dynamical simulation should store additional variables such as velocity and force, and requires a larger memory size. The memory size is estimated to be 60 GB per node, which exceeds the limitation of built-in memory. The above difference between the simulations does not change the conclusions of the present discussions on elapsed time, since the elapsed time is dominated by that for electronic state calculation.

The elapsed time per step (T_{elaps}) is recorded by averaging the elapsed times among the four steps except the first step, because the first step includes several initial procedures, unlike the other steps. Since the elapsed time for the initial preparation procedure and the first step is in the same order of that for one step, it is negligible in practical dynamical simulations with $n_{\text{time}} = 10^3$ steps or more.

4.4.3 Preparation of disordered structures of condensed polymers

The initial atomic structures for the disordered systems were generated from classical molecular dynamics simulations by GROMACS [78]. GROMACS is free software for various kind of classical molecular dynamics simulations and highly parallelized with MPI. Classical simulations work faster but do not treat elec-

tronic (quantum) waves that are responsible for the device property explained in Sec. 4.1.2.

The detailed procedure for generating the three initial disordered structures of condensed polymers 'P100', 'P10' and 'P1' was as follows. At first a straight polymer chain of PPE with 100 monomers ($N = 1200$ atoms) is generated. Then copies of the chain are bundled in parallel on a lattice. The number of copies is 84672, 8448 and 1024 for 'P100', 'P10' and 'P1' respectively. For further details, when the direction of the chain axis is set to z-axis, the number of copies for each axis (n_x, n_y, n_z) is $(168, 168, 3)$, $(44, 48, 4)$ and $(32, 32, 1)$ for 'P100', 'P10' and 'P1' respectively. The numbers are determined so that the entire bundle approximates a cubic region. In the bundles each chain is randomly rotated around the z-axis and random displacement along the z-axis is imposed. By the method, structures which have moderate density are generated without MD simulations with pressure coupling.

After that molecular dynamics simulations by GROMACS were carried out for the bundle structures to generate totally disordered systems. Generalized AMBER Force Field (GAFF) [79, 80] was used for the force field. In the simulations the periodic boundary condition is imposed. Total simulation time is 10 ps and temperature is $1,200$ K. Simulation time step is 10^{-4} ps and therefore the number of time steps is 10^5. This simulation condition is enough for relaxation of the initial lattice bundle configuration. All the simulations were carried out on SGI ICE XA whose each compute node consists of two Intel Xeon 2.5 GHz (12 cores) processor. The elapsed time was 12,109 sec on 36 nodes for 'P100', 4,024 sec on 9 nodes for 'P10' and 737 sec on 4 nodes for 'P1'.

4.4.4 Analysis on strong scaling and time-to-solution

The elapsed time per step was analyzed so as to observe the strong scaling property and the qualified time-to-solution. The measured elapsed time is summarized in Table 4.2. Here the parallel efficiency ratio α is defined by

$$\alpha \equiv \frac{T_{\mathrm{elaps}}(n_0)/T_{\mathrm{elaps}}(n_{\mathrm{node}})}{n_{\mathrm{node}}/n_0} \tag{4.19}$$

with $n_0 \equiv 2592$. For example, the parallel efficiency ratio α with 10^8 atoms and the maximum number of nodes ($n_{\mathrm{node}} = 82,944$) is $\alpha = 0.92$ for 'D100' and $\alpha = 0.75$ for 'P100'.

Table 4.2: The measured elapsed times $T_{\rm elaps}$ (sec) for ideal diamond solid with 10^8 atoms ('D100') and condensed polymer systems with 10^8 atoms ('P100'), with 10^7 atoms ('P10') and with 10^6 atoms ('P1'). The ideal or measured speed-up ratio is shown inside the parenthesis.

$n_{\rm node}$	D100	P100	P10	P1
2,592 (1)	1001.4 (1)	741.1 (1)	81.4 (1)	10.3 (1)
5,184 (2)	502.2 (1.99)	378.5 (1.96)	43.7 (1.86)	5.95 (1.73)
10,368 (4)	252.6 (3.96)	195.2 (3.80)	24.3 (3.35)	3.28 (3.14)
20,736 (8)	127.9 (7.83)	103.0 (7.19)	11.4 (7.14)	1.96 (5.26)
41,472 (16)	65.6 (15.3)	57.1 (13.0)	6.32 (12.9)	1.25 (8.23)
82,944 (32)	34.1 (29.4)	30.9 (24.0)	3.60 (22.6)	0.84 (12.2)

Table 4.3: Communication time $T_{\rm comm}$ and barrier time $T_{\rm barr}$ of the elapsed time $T_{\rm elaps}$. See the caption of Table 4.2 for notations. The values are listed as $T_{\rm comm}$ / $T_{\rm barr}$ (sec).

$n_{\rm node}$	D100	P100	P10	P1
2,592	1.04 / 7.16	1.60 / 28.14	0.382 / 6.69	0.0617 / 2.68
5,184	1.04 / 3.34	1.60 / 20.75	0.378 / 4.86	0.0689 / 1.85
10,368	1.05 / 2.22	1.61 / 14.97	0.384 / 4.34	0.0734 / 1.13
20,736	1.05 / 1.34	1.61 / 9.05	0.218 / 3.31	0.0712 / 0.674
41,472	1.06 / 1.03	1.64 / 6.87	0.215 / 2.18	0.0727 / 0.409
82,944	1.06 / 0.485	1.65 / 5.96	0.218 / 1.28	0.0613 / 0.227

Table 4.4: Communication time $T_{\rm comm}$ and barrier time $T_{\rm barr}$ of the elapsed time $T_{\rm elaps}$ (sec) for the molecular dynamics simulation in the 'P10' case.

$n_{\rm node}$	$T_{\rm elaps}$	$T_{\rm comm}$	$T_{\rm barr}$
2,592	81.8	0.635	6.84
5,184	43.9	0.637	4.95
10,368	24.5	0.674	4.42
20,736	14.4	0.676	3.44
41,472	9.38	0.696	2.37
82,944	6.62	0.677	1.45

The data of Table 4.2 is plotted in Fig. 4.4. One can find that the target time-to-solution of $T_{\rm elaps} \approx 10^2$s is fulfilled by $n_{\rm node}$ = 20,736 and 2,592 for the polymer systems with 10^8 and 10^7 atoms, respectively. The above results indicate that the present simulation method is qualified for practical device material research with 10^7-10^8 atoms.

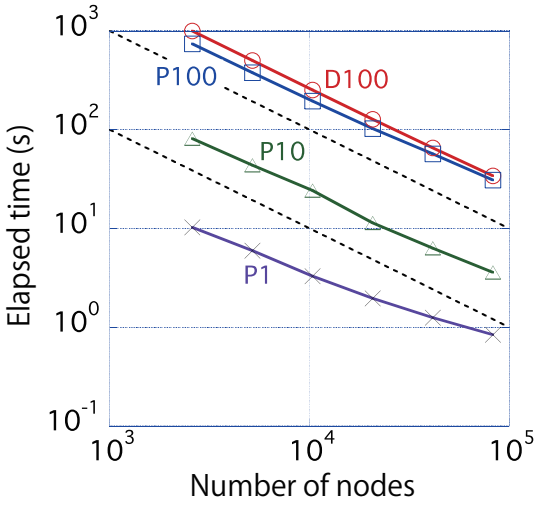

Figure 4.4: Strong scaling benchmarks for ideal diamond solid with 10^8 atoms ('D100') and condensed polymer systems with 10^8 atoms ('P100'), with 10^7 atoms ('P10') and with 10^6 atoms ('P1'). Dashed lines are drawn for ideal scaling.

Since the computational cost is proportional to the number of atoms in the order-N method, the number of atoms per cores $N_{\mathrm{A/C}} \equiv N/n_{\mathrm{node}}$ is an important parameter for qualified simulations. The above two cases with the qualified time-to-solution ($T_{\mathrm{elaps}} \approx 10^2$s) give, commonly, the value of $N_{\mathrm{A/C}} \equiv N/n_{\mathrm{node}} \approx$500. The condition of $N_{\mathrm{A/C}} = 500$ will be a typical condition for qualified simulations.

Figure 4.4 indicates also that the strong scaling property is found in all cases, because the elapsed time T_{elaps} decreases monotonically as the function of the number of used nodes. As explained in Sec. 4.3.2, an efficient parallelism appears on the condition that the number of atoms per core is larger than one ($N_{\mathrm{A/C}} \equiv N/n_{\mathrm{core}} > 1$). In the case of 'P1', for example, the number of atoms per core is $N_{\mathrm{A/C}} \approx 1.85$ with the maximum number of nodes ($n_{\mathrm{node}} \equiv 82,944$). The value of $N_{\mathrm{A/C}} \approx 1.85$ is close to the lower limit and the simulation shows a lower value of the parallel efficiency ratio ($\alpha = 0.38$).

4.4.5 Detailed analysis

Detailed analysis was carried out for the effect of communication time and load imbalance. In the simulations, not only the total elapsed time T_{elaps}, but also the accumulated MPI communication time T_{comm}, and the accumulated barrier time T_{barr}, are recorded at every step on all nodes. The barrier time includes the time to wait for other processors. Table 4.3 shows the average value among the nodes

for $T_{\rm elaps}$ and $T_{\rm comm}$ and the maximum value for $T_{\rm barr}$. The communication time $T_{\rm comm}$ is consumed by inter-node data communications, while the barrier time $T_{\rm barr}$ appears from a load imbalance.

Figure 4.5 plots the data in Table 4.3. Three points are discussed; (i) The communication time $T_{\rm comm}$ is almost constant among all the four cases, because the communication time is consumed by `MPI_Allreduce()` within a tree algorithm. (ii) Among all the cases, the barrier time $T_{\rm barr}$ seems to include a term $T_{\rm barr}^{(1)}$ that is proportional to the total elapsed time $T_{\rm elaps}$ ($T_{\rm barr}^{(1)} \propto T_{\rm elaps}$), though the ratio of $T_{\rm barr}^{(1)}/T_{\rm elaps}$ is not serious. (iii) Among the condensed polymer systems (Figs. 4.5(b)-(d)), the barrier time $T_{\rm barr}$ is always much larger than the communication time $T_{\rm comm}$ and can affect the efficiency in parallelism with the maximum number of nodes ($n_{\rm node} = 82,944$).

To end up this section, two comments are addressed; (I) The further tuning should be focused mainly on single-core calculations, since the most routines are executed as single-core calculations as in Fig.4.3(c). The profiler reported that the performance is 2.3 % of the peak for the 'P100' case with $n_{\rm node} = 82,944$ in Table 4.2. The severest limitation in the present calculations is the memory size of the K computer (16GB per node) and the present code was written in the memory-saving style, in which the memory cost should be minimized and the time cost is sometimes sacrificed. Since the situation can differ among materials and/or architectures, a possible way is to add another workflow in the time-saving style. The routines can be classified into those for the generation of matrix elements and for the Krylov subspace solver as in Fig.4.3(a). The matrix-vector multiplication gives a large fraction of the total elapsed time, as usual in a Krylov-subspace solver, and a typical fraction is 21 % among the present condensed polymer systems. The result suggests that the matrix generation part gives a larger fraction. (II) Table 4.4 shows the detailed elapsed time for the molecular dynamics simulation in the 'P10' case, the possible maximum size (See Sec. 4.4.2). For example, the elapsed time per molecular dynamics time step is $T_{\rm elaps}^{\rm (MD)} = 81.8$ sec or 6.62 sec in $n_{\rm node}$ =2,592 or 82,944, respectively. Fig. 4.6 plots the data in Table 4.4 in the same manner of Fig. 4.5(c). For comparison, Fig. 4.6 also shows the data in Fig. 4.5(c), the data with the electronic structure calculation part. The elapsed time is much smaller than the target time-to-solution (10^2s) and the method is qualified well for a real research. However, non-negligible time costs appear in the total elapsed time ($T_{\rm elaps}^{\rm (MD)}$) among the cases with $n_{\rm node} > 2 \times 10^4$, because of the additional routine for MD simulation. Therefore, further tuning of the code for faster MD simulations is needed.

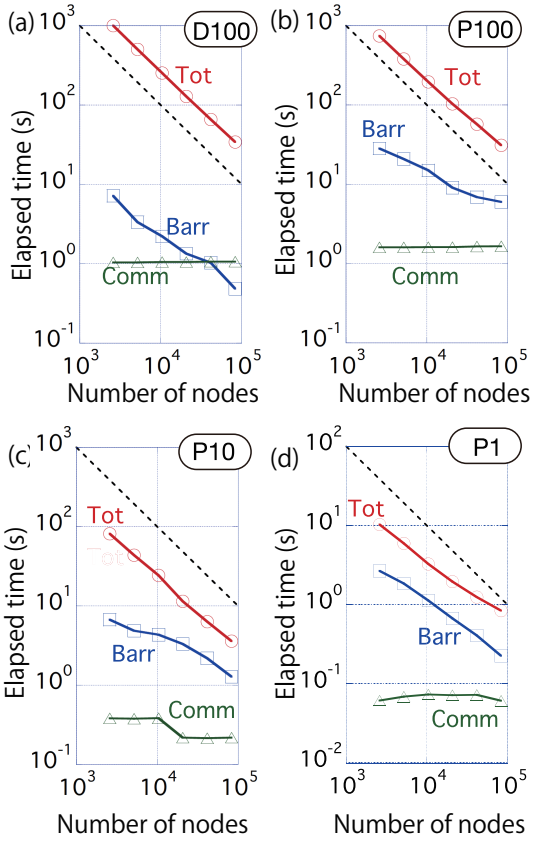

Figure 4.5: Decomposition analysis of the elapsed times. The total elapsed time $T_{\rm elaps}$, the communication time $T_{\rm comm}$ and the barrier time $T_{\rm barr}$ are plotted for (a) ideal diamond solid with 10^8 atoms ('D100') and condensed polymer systems (b) with 10^8 atoms ('P100'), (c) with 10^7 atoms ('P10') and (d) with 10^6 atoms ('P1'). Dashed lines are drawn for ideal scaling.

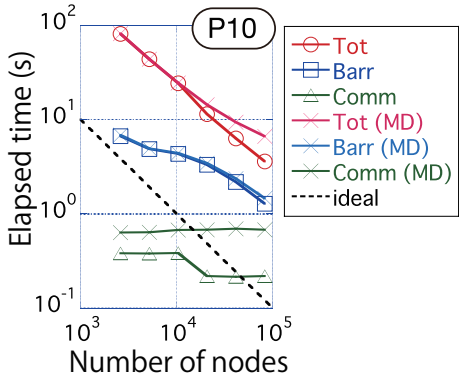

Figure 4.6: Decomposition analysis of the elapsed time for the MD simulation in the 'P10' case. The total elapsed time $T^{\rm (MD)}_{\rm elaps}$ (Tot(MD)), the barrier time $T^{\rm (MD)}_{\rm barr}$ (Barr(MD), and the communication time $T^{\rm (MD)}_{\rm comm}$ (Comm(MD)) are plotted per MD step in the same manner of Fig. 4.5(c). The data for the electronic state calculation (Tot, Barr, Comm) are also plotted for comparison.

4.5 Wavepacket dynamics simulation

To investigate transport property of organic device materials, quantum wavepacket dynamics simulations are carried out. The simulation is based on a Schrödinger-type equation

$$\partial_t \Psi = -iH_{\text{eff}}\Psi \tag{4.20}$$

for a hole wavefunction $\Psi(r,t)$ with a modelled Hamiltonian H_{eff}. In general, the time interval of simulation step is much smaller than that in the molecular dynamics simulation ($h_{\text{WP}} \ll h_{\text{MD}}$) and is $h_{\text{WP}} = 0.1$ fs or smaller. A typical post-simulation analysis is the estimation of mobility explained in Sec. 2.4.2. The estimation procedure is explained; After the wavepacket simulation, the mean square displacement (MSD) of the wavepacket is calculated at each time step ($\langle r^2(t)\rangle$). Then the MSD behavior is plotted as the function of time. See Fig. 4.7 for typical graphs. One can find a linear behavior in Fig. 4.7 and the diffusion constant D can be fitted as

$$D = \lim_{t\to\infty} \frac{\langle r^2(t)\rangle}{2t}. \tag{4.21}$$

Finally mobility μ is given as

$$\mu = \frac{De}{k_{\text{B}}T} \tag{4.22}$$

under Einstein's relation in Sec. 2.4.2.

In a previous paper, [55] a wavepacket dynamics simulation was carried out for an organic polymer, in which a π-orbital-only model is constructed for disordered atomic structures in dynamical simulations. The analysis of the result gives mobility. The present method is a theoretical generalization of the previous one. The methodological details will be explained in rest of this section and a brief outline is explained here. The initial wavepacket $\Psi(r, t = 0)$ is set as an eigenstate, such as the HO state. Two kinds of dynamical simulation methods are constructed; The first simulation method is called 'multi-time-scale simulation' or the combined simulation between molecular dynamics and wavepacket dynamics simulations. In the simulation, a double loop structure is used in the code for the time evolution; The outer loop is used for the atomic thermal motion, a slower dynamics, while the inner loop is the quantum wavepacket dynamics, as explained in Sec. 4.5.1. The second simulation method is called 'simulation with modeled atomic motion', in which the atomic motion is introduced as a modeled stochastic term in the wavepacket dynamics. The latter method gives a faster simulation.

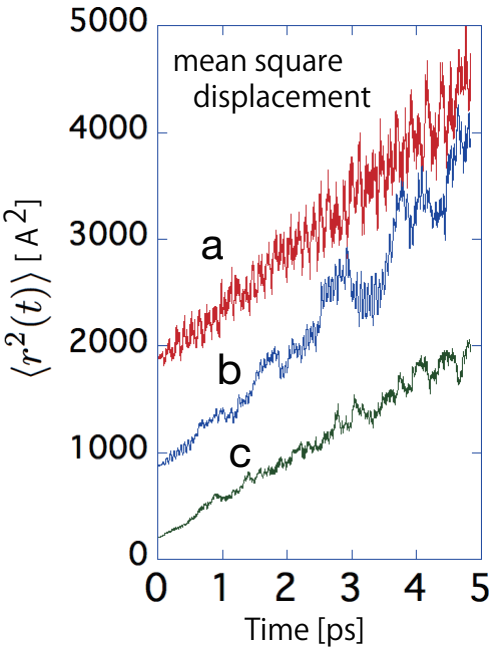

Figure 4.7: Examples of the mean square displacement during the wavepacket dynamics simulations. The three simulated samples of (a)(b)(c) are disordered organic polymer (poly-(phenylene-ethynylene);PPE) with disordered structures by the thermal motion.

4.5.1 Concept of multi-time-scale quantum wavepacket simulations

Rest of this section is devoted to the investigation on the practical methods of quantum wavepacket dynamics, as a real-time non-stationary simulation, in which the atomic position and the wavepacket are updated iteratively. The method should be beyond the adiabatic theory and is not yet well established. The purpose of the present method is to construct a modeled theory for a practical large-scale device material simulation within a moderate computational cost. The simulated system is a material with a single carrier, hole or excited electron, and the carrier is described as a wavepacket. Such simulations were carried, for decades, on site models, in which a (small) molecule is treated as a site. See Ref. [81] for example.

In the present thesis, a multi-time-scale simulation is adopted, since the atomic motion is much slower than the wavepacket dynamics. In other words, the situation is assumed to be *nearly* adiabatic. Therefore, the time evolution is realized by a double loop structure in the code; The atomic position is updated in the outer loop, while the wavepacket is updated in the inner loop. The time interval of the outer loop is set to be $h = 40\text{au}(\approx 1\text{fs})$, a typical value for molecular dynamics

simulation, while the time interval of the inner loop should be set to be $h_{\mathrm{WPD}} \ll h$.

4.5.2 Basic mathematical formulation of generalized eigenvalue equation

The basic mathematical formulation is summarized for the generalized eigenvalue equation

$$H_0 \boldsymbol{y}_k = \lambda_k S \boldsymbol{y}_k. \tag{4.23}$$

The eigenvectors satisfy the orthogonality relation

$$\boldsymbol{y}_k^\dagger S \boldsymbol{y}_l = \delta_{kl}. \tag{4.24}$$

Eq. (4.23) is rewritten as

$$\sum_j H_{ij} Y_{jk} = \lambda_k \sum_j S_{ij} Y_{jk} \tag{4.25}$$

or

$$HY = SY\Lambda, \tag{4.26}$$

where $\boldsymbol{y}_k \equiv (Y_{1k}, Y_{2k}, ..., Y_{Mk})$, $\Lambda \equiv \mathrm{diag}(\lambda_1, \lambda_2, ...\lambda_M)$. Eq. (4.24)) is rewritten as

$$Y^\dagger S Y = I \tag{4.27}$$

or

$$Y^{-1} = Y^\dagger S. \tag{4.28}$$

The wavepacket vector on the LCAO basis $\boldsymbol{u} \equiv (u_1, u_2, ..., u_M)^{\mathrm{T}}$ is expressed by the linear combination of the eigenvectors

$$u_i = \sum_k \alpha_k Y_{ik} \tag{4.29}$$

or

$$\boldsymbol{u} = Y\boldsymbol{\alpha}. \tag{4.30}$$

with $\boldsymbol{\alpha} \equiv (\alpha_1, \alpha_2, ..., \alpha_M)^{\mathrm{T}}$. Eqs. (4.30) and (4.28) give

$$\boldsymbol{\alpha} = Y^{-1}\boldsymbol{u} = Y^\dagger S \boldsymbol{u}. \tag{4.31}$$

4.5.3 Multi-time-scale simulation (I)

Our multi-time-scale simulation is explained. The wavepacket dynamics, the inner loop, is considered for the time period of

$$nh \leq t \leq (n+1)h, \tag{4.32}$$

where n indicates the MD time step or the iteration number of the outer loop. The wavepacket is written as the linear combination of atomic orbitals $\{\chi_j^{(n)}(\boldsymbol{r})\}$ as

$$\Psi(\boldsymbol{r}, t) = \sum_j u_j^{(n)}(t)\chi_j^{(n)}(\boldsymbol{r}). \tag{4.33}$$

The suffix of '(n)' indicates that the used atomic orbital basis set is that for the atomic position at the n-th MD time step ($t = nh$). The atomic orbitals $\{\chi_j^{(n)}(\boldsymbol{r})\}$ are real functions. Here we consider an effective Schrödinger-type equation of the wavepacket

$$\partial_t \Psi = -i\hat{H}_{\text{eff}}\Psi \tag{4.34}$$

with an effective Hamiltonian operator $\hat{H}_{\text{eff}}$. Eqs. (4.33) and (4.34) lead us to the equation of

$$S^{(n)}\dot{\boldsymbol{u}}^{(n)} = -iH^{(n)}\boldsymbol{u}^{(n)}, \tag{4.35}$$

with the definitions of

$$H_{ij}^{(n)} \equiv \int \chi_i^{(n)}(\boldsymbol{r})\hat{H}_{\text{eff}}\chi_j^{(n)}(\boldsymbol{r})d\boldsymbol{r} \tag{4.36}$$

$$S_{ij}^{(n)} \equiv \int \chi_i^{(n)}(\boldsymbol{r})\chi_j^{(n)}(\boldsymbol{r})d\boldsymbol{r}. \tag{4.37}$$

As a practical methodology, the matrix of $H^{(n)}$ is set to be that in Eq. (4.23) ($H^{(n)} = H_0^{(n)}$). When the generalized eigenvalue equation

$$H^{(n)}\boldsymbol{y}_k^{(n)} = \lambda_k^{(n)} S^{(n)}\boldsymbol{y}_k^{(n)} \tag{4.38}$$

is solved, the wavepacket vector $\boldsymbol{u}^{(n)}(t)$ is expressed by the eigenvectors of Eq. (4.38) as

$$\boldsymbol{u}^{(n)}(t) = \sum_k \alpha_k^{(n)}(t)\boldsymbol{y}_k^{(n)} \tag{4.39}$$

with the coefficient vector of $\boldsymbol{\alpha}^{(n)} \equiv (\alpha_1^{(n)}, \alpha_2^{(n)}, ...\alpha_M^{(n)})^{\mathrm{T}}$. Since Eqs. (4.35) and (4.39) are reduced to

$$\dot{\alpha}_k^{(n)}(t) = -i\lambda_k^{(n)}\alpha_k^{(n)}(t), \tag{4.40}$$

the time evolution of $\alpha_k^{(n)}(t)$ is solved analytically as

$$\boldsymbol{\alpha}^{(n)}((n+1)h) = D^{(n)}\boldsymbol{\alpha}^{(n)}(nh) \tag{4.41}$$

with the diagonal matrix of

$$D^{(n)} \equiv \mathrm{diag}(e^{-i\lambda_1^{(n)}h}, e^{-i\lambda_2^{(n)}h}, ..., e^{-i\lambda_M^{(n)}h}). \tag{4.42}$$

At the time of $t = (n+1)h$, the wavepacket Ψ should be expressed as the linear combination for the updated atom position ($\boldsymbol{R}(t = (n+1)h)$)

$$\Psi(\boldsymbol{r}, t) = \sum_j u_j^{(n+1)}(t)\chi_j^{(n+1)}(\boldsymbol{r}) \tag{4.43}$$

for the time period of

$$(n+1)h \leq t \leq (n+2)h \tag{4.44}$$

and a relation between $\boldsymbol{u}^{(n+1)}((n+1)h)$ and $\boldsymbol{u}^{(n)}((n+1)h)$ is needed for the simulation. A reasonable relation is

$$\boldsymbol{u}^{(n+1)}((n+1)h) := \boldsymbol{u}^{(n)}((n+1)h), \tag{4.45}$$

because the relation will be exact, when all the atoms are not mobile and the basis set is unchanged ($\chi_j^{(n+1)}(\boldsymbol{r}) = \chi_j^{(n)}(\boldsymbol{r})$).

Since Eqs. (4.30) and (4.31) give

$$\boldsymbol{u}^{(n)}((n+1)h) = Y^{(n)}\boldsymbol{\alpha}^{(n)}((n+1)h) \tag{4.46}$$

$$\boldsymbol{\alpha}^{(n+1)}((n+1)h) = Y^{(n+1)\dagger}S^{(n+1)}\boldsymbol{u}^{(n+1)}((n+1)h), \tag{4.47}$$

the relation between $\boldsymbol{\alpha}^{(n+1)}((n+1)h)$ and $\boldsymbol{\alpha}^{(n)}((n+1)h)$ is obtained as

$$\boldsymbol{\alpha}^{(n+1)}((n+1)h) = A^{(n)}\boldsymbol{\alpha}^{(n)}((n+1)h) \tag{4.48}$$

with

$$A^{(n)} \equiv Y^{(n+1)\dagger}S^{(n+1)}Y^{(n)}. \tag{4.49}$$

Eqs. (4.41) and (4.48) result in

$$\boldsymbol{\alpha}^{(n+1)}((n+1)h) = A^{(n)}D^{(n)}\boldsymbol{\alpha}^{(n)}(nh). \tag{4.50}$$

Finally, the normalization procedure is imposed on the resultant vector of $\boldsymbol{\alpha}^{(n+1)}((n+1)h)$, since Eq. (4.45) introduces a deviation in the normalization condition, when

$S^{(n+1)} \neq S^{(n)}$. All the multi-time-scale simulations in Sec. 4.6.2 were carried out by the above formulation.

In Eq. (4.30), the eigenvectors for the expansion of the wavepacket vector can be partially selected. This approximation is called eigenvector filtering. Namely, a matrix of filtered eigenvectors

$$Y_{\rm f} = (\boldsymbol{y}_l \; \boldsymbol{y}_{l+1} \; ... \; \boldsymbol{y}_{l+m-1}) \quad \in C^{M\times m} \tag{4.51}$$

is used instead of Y. Small m is advantageous in computational cost, and unphysical artifacts can be removed when the filtering region $l, \ldots, l+m-1$ is adequately selected. Although $Y_{\rm f}^{-1}$ is no longer exists in this case, the relations $Y_{\rm f}^{\dagger} S\, Y_{\rm f} = I_m$ and $Y_{\rm f}^{+} = Y_{\rm f}^{\dagger} S$ hold instead of Eqs. (4.27) and (4.28), respectively, where $Y_{\rm f}^{+}$ is the pseudo-inverse of $Y_{\rm f}$. In general, for any (not necessarily square) matrix A, its pseudo-inverse A^{+} solves the least square problem

$$\min_{\boldsymbol{x}} ||A\boldsymbol{x} - \boldsymbol{b}||_2 \tag{4.52}$$

by $\boldsymbol{x} = A^{+}\boldsymbol{b}$. In this case, $A = Y_{\rm f}$. By these relations, almost the same formulation with the original case is valid. Hereafter the subscript f is dropped.

4.5.4 Multi-time-scale simulation (II)

Although the methodology in the previous subsubsection can realize a multi-time-scale simulation code, the present subsubsection will introduce several corrections for better description of the wavepacket dynamics. The corrections were introduced from the energetic aspect based on the fact that, since the wavepacket dynamics is motivated from the atomic thermal motion, the energy of wavepacket $\boldsymbol{u}^{\rm T} H \boldsymbol{u}$ should fluctuate only within the energy scale of the atomic thermal motion. The energy discontinuity can appear in Eq. (4.48) and the correction is introduced by

$$\boldsymbol{\alpha}^{(n+1)}((n+1)t) = \tilde{A}\boldsymbol{\alpha}^{(n)}((n+1)t) \tag{4.53}$$

$$\tilde{A}_{ij} \equiv \tilde{\delta}(\lambda_i^{(n+1)} - \lambda_j^{(n)}) A_{ij} \tag{4.54}$$

with the 'smoothed' delta function of

$$\tilde{\delta}(x) \equiv \frac{1}{\sqrt{2\pi}b} e^{-(1/2)(x/b)^2}. \tag{4.55}$$

The energy parameter $b(>0)$ should be given in the order of the thermal fluctuation. The correction is introduced so that the transition between different eigenstates ($j \to i$) can occur only when the two eigenenergies is so close that

$$|\lambda_i^{(n+1)} - \lambda_j^{(n)}| \leq b. \tag{4.56}$$

Another correction is introduced by the fact that the thermal relaxation effect should induce an energy flow from the wavepacket into the atomic motion. The standard equilibrium state theory gives the carrier energy distribution in the Boltzmann distribution ($\propto \exp(-|\lambda - \lambda_{\rm F}|/k_{\rm B}T)$). If the k-th eigenenergy is far from the Fermi level ($|\lambda_k - \lambda_{\rm F}| \gg k_{\rm B}T$), its component α_k tends to be relaxed into other levels nearer the Fermi level. Such relaxation effect is realized, when the diagonal matrix $D^{(n)}$ is replaced by

$$\tilde{D}^{(n)} \equiv \mathrm{diag}(e^{-i\lambda_1^{(n)}h-\gamma_1 h}, e^{-i\lambda_2^{(n)}h-\gamma_2 h}, ...e^{-i\lambda_k^{(n)}h-\gamma_k h}, .., e^{-i\lambda_M^{(n)}h-\gamma_M h}). \tag{4.57}$$

with the definition of

$$\gamma_k \equiv \frac{|\lambda_k - \lambda_{\rm F}|}{k_{\rm B}T\tau}. \tag{4.58}$$

The relaxation effect is governed by the time parameter $\tau(> 0)$, since the k-th eigenstate component α_k decays as

$$\propto \exp\left(-\frac{h}{\tau}\frac{|\lambda_k - \lambda_{\rm F}|}{k_{\rm B}T}\right) \tag{4.59}$$

with the time interval of h. In short, the parameter τ means a relaxation time, since the k-th eigenstate component with $|\lambda_k - \lambda_{\rm F}| \approx k_{\rm B}T$ decays in the time order of τ. It might be noteworthy that Eq. (4.57) can be understood, formally, as the introduction of the imaginary part of the eigenenergy

$$\tilde{D}_{kk}^{(n)} \equiv \exp\left(-i\lambda_k^{(n)}h - \gamma_k h\right) = \exp\left(-i(\lambda_k^{(n)} - i\gamma_k)h\right). \tag{4.60}$$

The above two corrections modify Eq. (4.50) into

$$\boldsymbol{\alpha}^{(n+1)}((n+1)h) \;\; = \;\; \tilde{A}^{(n)}\tilde{D}^{(n)}\boldsymbol{\alpha}^{(n)}(nh). \tag{4.61}$$

The method with the corrections is now under test, in which the test simulations are running with several different values of the parameter set. The computational cost of the corrections is negligible. Quantitative discussion will be a future topic.

4.5.5 Simulation with modeled thermal atomic motion

The present subsection explains the simulation method with modeled atomic motion, in which the atomic thermal fluctuation is treated as a modeled stochastic term in the wavepacket dynamics. In the simulation, only the input matrices are required only for the initial structure $(H^{(0)}, S^{(0)})$. When Eq.(4.35) is solved, the Hamilton matrix at the n-th MD step, $H^{(n)}$, is determined by

$$H^{(n)} := H^{(0)} + H^{(n)}_{\rm therm} \tag{4.62}$$

and $S^{(n)} := S^{(0)}$. The matrix $H^{(n)}_{\rm therm}$ is introduced for the atomic thermal fluctuation term. The matrix was chosen to be the diagonal and the element is

$$\left(H^{(n)}_{\rm therm}\right)_{ii} = \frac{k_{\rm B}T}{2}\cos 2\pi x_i \tag{4.63}$$

with the temperature of T and a random number of x_i. The random numbers of $\{x_i\}_i$ are updated with the time interval of MD ($h = 1$fs) or the time scale of the atomic thermal motion. The same random number is applied for the bases on the same atom. The above method will be exact in the zero-temperature limit, in which the matrices of $(H^{(n)}, S^{(n)})$ are unchanged throughout the simulation.

This method runs faster than the multi-time-scale simulation in the previous subsections. We believe that the two methods are complementary. Since our purpose is the systematic survey among different samples, so as to obtain insights for better device materials and their fabrication process, the problem is the balance between the computational cost and the accuracy. If the simulation runs faster, one can investigate a dynamics simulation among many sample in different structures. The number of the samples is crucial, in particular, among data scientific research and the present method should be fruitful.

4.5.6 Calculation of diffusion coefficient

As explained in Sec. 2.4.2, the mobility μ is proportional to the diffusion coefficient D (Eq. (2.36)). In the present thesis, D is determined from time evolution of the wavepacket Eq. (4.33) as follows. At first the Mulliken charge [82] for j-th atomic orbital $m_j(t)$ is defined as

$$m_j(t) \equiv \mathrm{Re}\left[\overline{u^{(n)}_j(t)} S^{(n)} \boldsymbol{u}^{(n)}(t)\right]. \tag{4.64}$$

Note that a normalization condition

$$\sum_j m_j(t) = 1 \tag{4.65}$$

is satisfied due to the normalization condition of $\boldsymbol{\alpha}(t)$, Eqs. (4.30) and 4.27). Denoting the coordinate of the atom which the j-th atomic orbital belongs to by $\boldsymbol{r}_j(t)$, the mean square displacement (MSD) $\langle r(t)^2\rangle$ is determined as

$$\langle r(t)^2\rangle \equiv \sum_j m_j(t)\|\boldsymbol{r}_j(t) - \langle \boldsymbol{r}(t)\rangle\|^2 \tag{4.66}$$

where $\langle \boldsymbol{r}(t)\rangle$ is the mean coordinate of the atoms weighted by the Mulliken charge

$$\langle \boldsymbol{r}(t)\rangle \equiv \sum_j m_j(t)\boldsymbol{r}_j(t). \tag{4.67}$$

In the same manner as Sec. 2.4.3, diffusion constant for the wavepacket simulations is determined as the slope of

$$d(t) \equiv \langle r(t)^2 \rangle / 2. \tag{4.68}$$

Because $d(t)$ is generally not a linear function of time, the slope is calculated by least-square fitting. Moreover, trajectories of $d(t)$ by wavepacket simulations from different initial atomic structures should be averaged for reliable values of D.

4.6 Organic material simulations

Organic materials play a crucial role among next-generation IoT products, such as display, battery and sensor, since they form flexible atomic structures and enable ultra-thin, light, flexible (wearable) devices with a low fabrication cost. Since the atomic structure of organic device materials is disordered, the simulation of such materials requires 100-nanometer-scale systems, which is beyond the computational limit of the present standard electronic structure calculations.

The transport of organic device is governed by π-type wavefunctions that appears typically on benzene-ring regions. Figure 4.8(a) shows an example of π-type wavefunctions in an organic polymer, poly-((9,9)-dioctyl fluorene) with $n = 2$. The transport property stems from both ballistic and diffusive conduction mechanisms and can be simulated by wavepacket dynamics. In an ideally ordered sample, for example, several wavefunctions are extended throughout the sample and the sample shows ballistic conduction. In a fairly disordered sample, on the other hand, wavefunctions are localized and the sample shows diffusive conduction motivated by the atomic thermal motion.

4.6.1 Quantum molecular dynamics simulation

Quantum molecular dynamics simulations were carried out for condensed organic polymers using the order-N method. Figure 4.9 shows an example of bundle-like poly-(phenylene vinylene) (PPV), The sample consists of 169 PPV polymers and each polymer has $n = 50$ monomers.

The length of a polymer is $L \approx 40$nm. The total number of atoms is $N = 117,962$. The van der Waals force [83] is included in the simulation. The parallel computation was carried out by 960 nodes (15,360 cores) of the supercomputer Oakleaf-FX of the University of Tokyo. The time interval of simulation step is $h_{\mathrm{MD}} = 1$ fs. A finite-temperature simulation in $T = 600$ K was performed for $n_{\mathrm{step}} = 5000$ iteration steps or the period of hMD nstep = 5ps. The total elapsed time is approximately 10 hours.

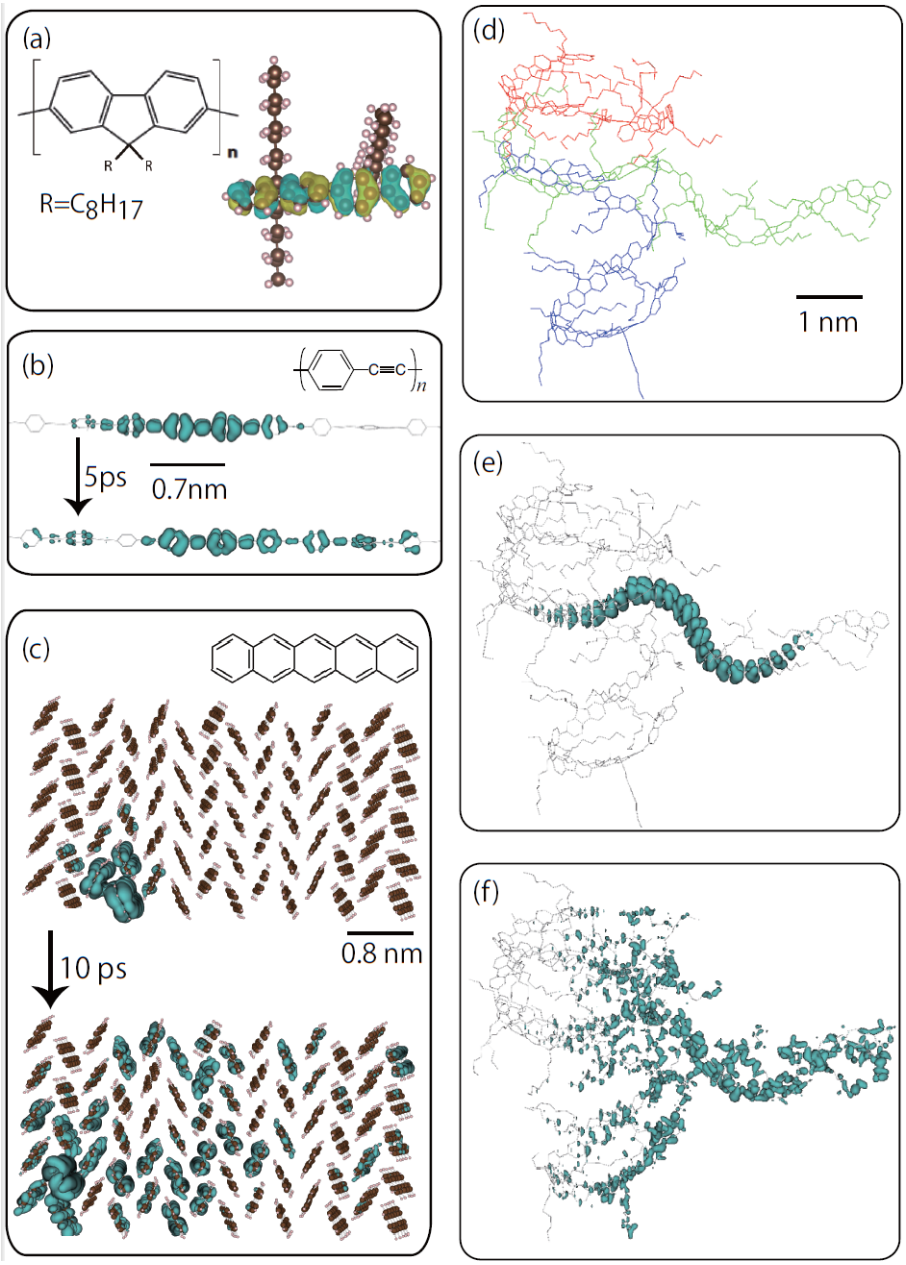

Figure 4.8: (a) Example of organic polymer: poly-((9,9) dioctyl-fluorene) (PFO) with $n = 2$. The HO state is drawn as an example of π wavefunction. (b) Quantum wavepacket dynamics on a polymer of poly-(phenyleneethynylene). The number of monomer unit of $n = 1000$ and the length of the polymer chain is $L \approx 700$ nm. The figure is a close-up. (c) Quantum wavepacket dynamics on pentacene thin film with single layer. (d) A condensed polymer of poly-((9,9)-dioctyl fluorene). The three polymers in a periodic cell form an amorphous-like structure. The three polymers are painted in different colors for better understanding. (e)-(f) Quantum wavepacket dynamics on the condensed polymers. The initial state at $t = 0$ (e) and the final state at $t = 1$ ps (f) are drawn.

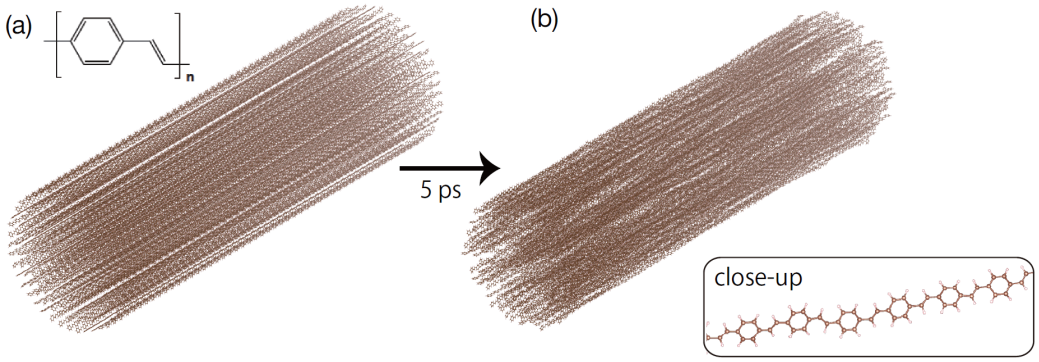

Figure 4.9: Quantum molecular dynamics of poly-(phenylene vinylene) (PPV) at the initial time ($t = 0$) and the final time ($t = 5$ps).

4.6.2 Transport calculations with wavepacket dynamics simulations

Several results for transport calculation of organic materials are discussed below. The calculations were carried out with the multi-time-scale simulation, except where indicated. Figure 4.8(b) shows the wavepacket dynamics simulation in organic polymers of poly-(phenyleneethynylene), where the maximum polymer lengths is $L = 700$ nm the figure is shown as a close-up. A characteristic semi-localized wavepacket was observed, since the wavepacket spreads over several monomer units but does not over the whole region of the polymer. Preliminary results on calculated mobility by the present method are consistent to the experimental trend [55] in which the mobility of meta (zig-zag) type polymers is larger than that of para (linear) type polymers. The mobility values were measured as the average of three samples in the simulation method with modeled atomic motion. The calculated mobility values of the meta- and para-type polymer are $(\mu_{\rm meta}, \mu_{\rm para}) = (1.41, 0.98)$ [cm^2 / V s], which are comparable to the experimental values of $(\mu_{\rm meta}, \mu_{\rm para}) = (2.1, 0.75)$ [cm^2 / V s]. Figure 4.8(c) shows the wavepacket dynamics simulation in a disordered pentacene thin film with a single layer. The periodic boundary condition is imposed on in the upper and lower area in Fig. 4.8(c). The wavefunction propagates from a molecule into another. Figure 4.8(d)-(f) shows condensed polymers of poly-((9,9)-dioctyl fluorene). The initial structure was generated by a classical molecular dynamics simulation, so as to save the total computational time in research. Three polymers in a periodic cell form an amorphous-like structure, as shown in Fig. 4.8(d). The number of atoms in a polymer is $N_{\rm polymer} = 692$ and the total number of atoms is $N = 692 \times 3 = 2076$. Figure 4.8(e)-(f) shows a result of wavepacket dynamics.

The initial wavepacket is localized in one polymer and extends over different polymers. More quantitative discussions are under development and will be a future topic.

4.7 Data scientific research for prescreening

In this section, a machine learning approach to the research of organic polymer device materials is presented [1]. The purpose is a pre-screening procedure of atomic structures before large-scale quantum wavepacket dynamics calculations. Atomic structures of condensed polymers are highly varied due to their flexibility and disorder. Therefore we should filter hopeful structures in the sense of high mobility before we actually execute quantum wavepacket dynamics calculations. The previous simulation research [5, 6] implies that the relative rotation angles between benzene rings strongly affect locality of π-electron wavefunctions and then mobility.

As a method for the pre-screening procedure, clustering of disordered polymer structures by locality of π-electron wavefunctions was carried out. K-means method, which is a simple clustering method [84], is used. K-means method can be found in standard textbooks of machine learning, such as Ref. [85]. At first PPE polymer chains with 240 atoms (20 monomers) whose relative rotation angles between adjacent benzene rings are given from the normal distribution with standard deviation of 20 / 60 degree are generated 100 times for each class (total $N = 200$ samples). After the rotations, small fluctuations taken from the normal distribution with standard deviation of 0.01 Åis given to all the coordinates of all atoms to remove degeneracy. For each structure ($i = 1, \ldots, N$), a singleshot calculation by ELSES generates Hamiltonian $H^{(i)}$ and overlap matrix $S^{(i)}$, and the generalized eigenvalue problem is solved numerically as

$$H^{(i)}\boldsymbol{y}_j^{(i)} = \lambda_j^{(i)} S^{(i)}\boldsymbol{y}_j^{(i)} \quad (j = 1, \ldots, M). \tag{4.69}$$

The matrix size M, which is determined by the number of each element in the atomic structure, was $M = 714$. Then the K-means clustering method is applied for the atomic structures using the list of the participation ratios of the eigenvectors defined in Sec. 2.4.1

$$p^{(i)} := (PR(\boldsymbol{y}_1^{(i)}), \ldots, PR(\boldsymbol{y}_M^{(i)}))^T \tag{4.70}$$

as a feature vector for the i-th structure.

[1] H. Imachi, Y. M. Tsai, T. Hoshi, and W. Wang, A data scientific research on organic polymer device materials, The 19th Asian Workshop on First-Principles Electronic Structure Calculations, National Chiao Tung University, Hsinchu, Taiwan, 31. Oct.- 2. Nov., 2016.

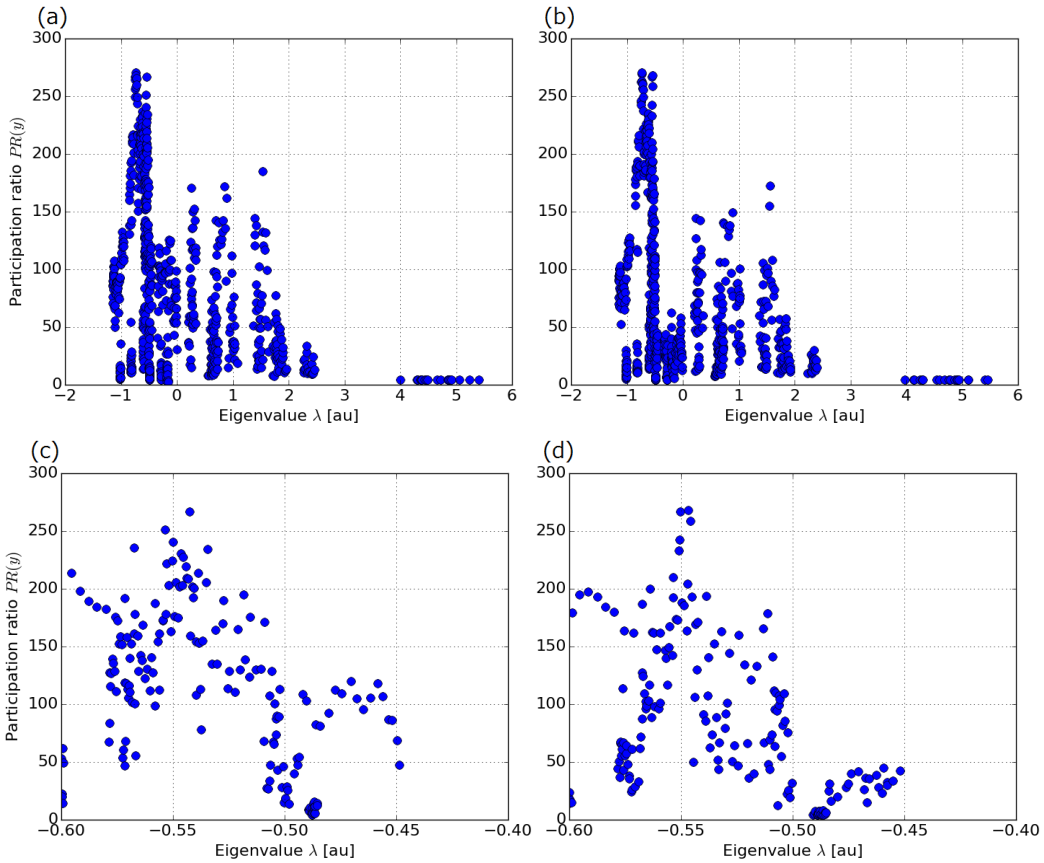

Figure 4.10: Eigenvector participation ratio diagrams of different atomic structure classes. The x-axis is the eigenvalue and the y-axis is the participatin ratio of the correspoiding eigenvector. (a) and (b) are diagrams of a structure with adjacent rotation angles in standard deviation of 20 and 60 degree, respectively. (c) and (d) are the same diagrams in partial eigenvalue range ($-0.6 \leq \lambda \leq -0.4$ au).

Figure 4.10 shows typical examples of the eigenvector participation ratios for the two structure classes. It indicates that participation ratios of π-electrons are strongly affected by the disorder in rotation angles. When the disorder in rotation angles is small (Fig. 4.10(a)), π-electrons are delocalized and their participation ratios are high, and vice versa, because the participation ratio of an eigenvector indicates its spatial extent. For each structure classes, a typical shape of π electronic wavefunction is visualized in Fig. 4.11. One can confirm the relation between the disorder in rotation angles and the participation ratio, because in Fig. 4.11(b), the wavefunction is more localized than in Fig. 4.11(a) due to the high rotation angles at the edges of the wavefunction.

The number of clusters, a parameter for the K-means algorithm, was two.

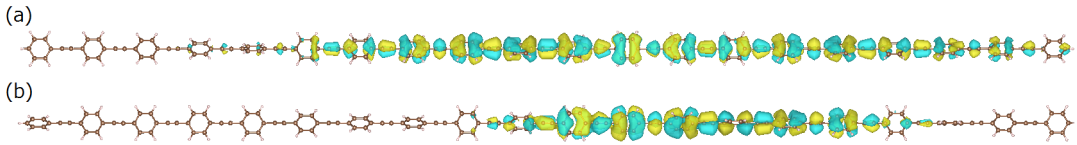

Figure 4.11: Typical atomic structures and π electronic wavefunctions of different atomic structure classes. Structures of (a) and (b) have adjacent rotation angles in standard deviation of 20 and 60 degree, respectively. The participation ratios of the wavefunctions (a) and (b) are 122.7 and 48.9, respectively.

Figure 4.12 shows the result of the K-means clustering. Each atomic structure is plotted on the plane by two statistical quantities with markers corresponding to the cluster labels given by the K-means algorithm. As a result, the two clusters perfectly matched to the structure classes. Namely, all the structures with adjacent rotation angles in standard deviation of 20 degree are clustered into one group, and all the structures with adjacent rotation angles in standard deviation of 60 degree are clustered into the other group.

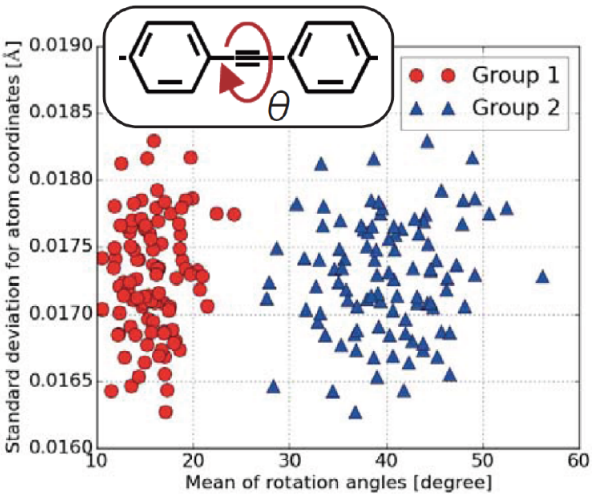

Figure 4.12: K-means clustering of polymer structures by their eigenvector participation ratios. Each structure is plotted by two statistical quantities. The x-axis is the mean of the rotation angles between adjacent benzene rings. The y-axis is the resulting standard deviation of the small fluctuation to the coordinates. Two markers, red circle and blue triangle, correspond to the two clusters labeled by the K-means algorithm.

The result of K-means clustering indicates that the PRs of the eigenvectors in-

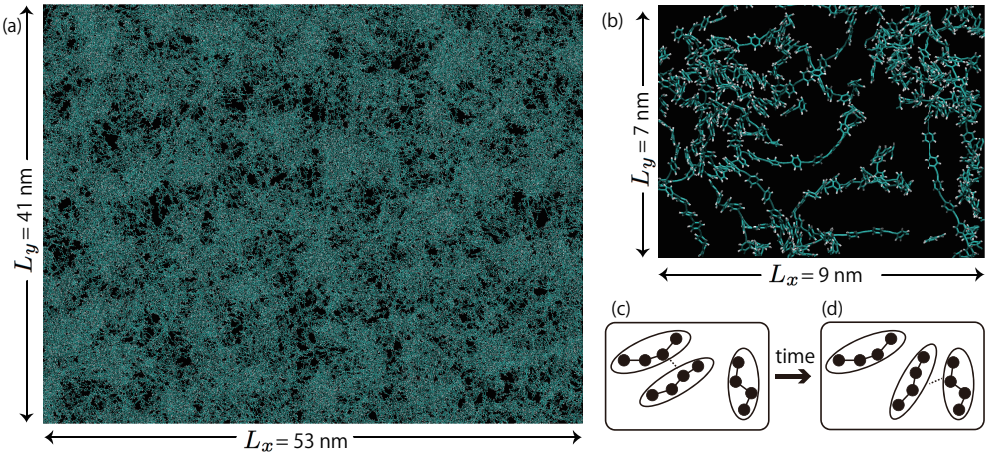

Figure 4.13: Visualization of partial regions in a 10^8-atom system of condensed polymers (PPE). The system has the periodic cell lengths of (265nm, 206nm, 239nm). (a) The region has the edge lengths of (L_x, L_y, L_z)=(53 nm, 41 nm, 48 nm) and contains approximately 8×10^5 atoms. (b) The region has the edge lengths of (L_x, L_y, L_z)=(9 nm, 7 nm, 8 nm) and contains approximately 4×10^3 atoms. (c) (d) Schematic figures of a local network of connected polymers.

herit the disorder property in atomic structures of polymers, and inversely, atomic structures can be classified using the PRs before executing wavepacket simulations for them.

4.8 10^8-atom simulation

This section is devoted to the scientific investigation with the 'P100', 10^8 atom (largest) system of condensed organic polymers. A smaller system with 10^5 atoms was calculated as a finite temperature simulation by 10^4 cores and the whole elapsed time is 10 hours for 5000 iteration steps, as reported in Sec. 4.5.1

Such a dynamical simulation is, however, impractical with 10^8 atoms at the present day and this section demonstrates a part of the possible future research.

4.8.1 Network analysis of electronic wavefunctions

A large-scale data analysis was carried out for a fundamental research of electronic devices, so as to characterize the propagation of electronic wave through a disordered structure.

Figures 4.13(a)(b) show partial regions of a 10^8 atoms and one can observe that

the structure is disordered. In general, electronic wavefunctions are localized in a disordered structure. The electrical current can propagate among polymers that are 'connected' locally by π electronic wavefunctions, as explained in Sec. 4.1.2. Therefore, the local network of connected polymers should be investigated.

Figures 4.13(c)(d) show schematically small local networks of connected polymers. Three polymers are drawn and atoms are depicted as filled circles. The figures include a small local network that consists of two polymers connected by a dashed line. The networks are dynamical, as seen later in this section.

A network analysis on connected polymer networks was carried out from electronic states, since a connection between polymers is formed by the electronic waves that spread over the polymers. The analysis was carried out with the Green's function G obtained by the parallel order-N simulation, as follows; Stage I: The order-N simulation gives a 'connectivity' matrix of B_{IJ}

$$B_{IJ} \equiv \sum_{\alpha} \sum_{\beta} \rho_{I\alpha;J\beta} H_{J\beta;I\alpha} \tag{4.71}$$

where I and J are the atom indices. The connectivity matrix is called integrated crystal orbital Hamiltonian population (ICOHP) among physics papers [86, 59]. The quantity is a partial sum of the electronic energy $\langle H \rangle$ in Eq. (4.3) ($\langle H \rangle = \sum_{IJ} B_{IJ}$). Since the matrix elements are calculated always during the parallel order-N simulation, the elements can be obtained independently among nodes, without any additional operation or communication cost. A matrix element B_{IJ} has a physical meaning of a local bonding energy between the I-th and J-th atoms; If the value of $|B_{IJ}|$ is significantly large, the two atoms are 'connected' by electronic wave. The matrix B is sparse, since only atoms within a short distance can be 'connected' with each other. Stage II: Since every atom belongs to one of polymers, the connectivity matrix for polymers is defined by

$$B_{PQ}^{(\mathrm{poly})} \equiv \sum_{I}^{\in P} \sum_{J}^{\in Q} B_{IJ}, \tag{4.72}$$

where the summation of $\sum_I^{\in P}$, for example, means the summation among the atoms that belong to the P-th polymer. If an element $B_{PQ}^{(\mathrm{poly})}$ shows a meaningful non-zero value, the P-th and Q-th polymers are connected by electronic wave. The matrix $B^{(\mathrm{poly})}$ is sparse. The dimension of $B^{(\mathrm{poly})}$ is equal to the number of polymers $N^{(\mathrm{poly})} = 83,349$ and is much smaller than that of B ($N = 10^8$). Stage III: As a coarse grained analysis, the eigenvalue equation of $B^{(\mathrm{poly})}\boldsymbol{z} = \lambda \boldsymbol{z}$ in the matrix dimension of $N^{(\mathrm{poly})}$ was solved by the parallel eigenvalue solver [4]. Then the participation ratio $PR(\boldsymbol{z})$ is computed for each eigenvector $\boldsymbol{z}$.

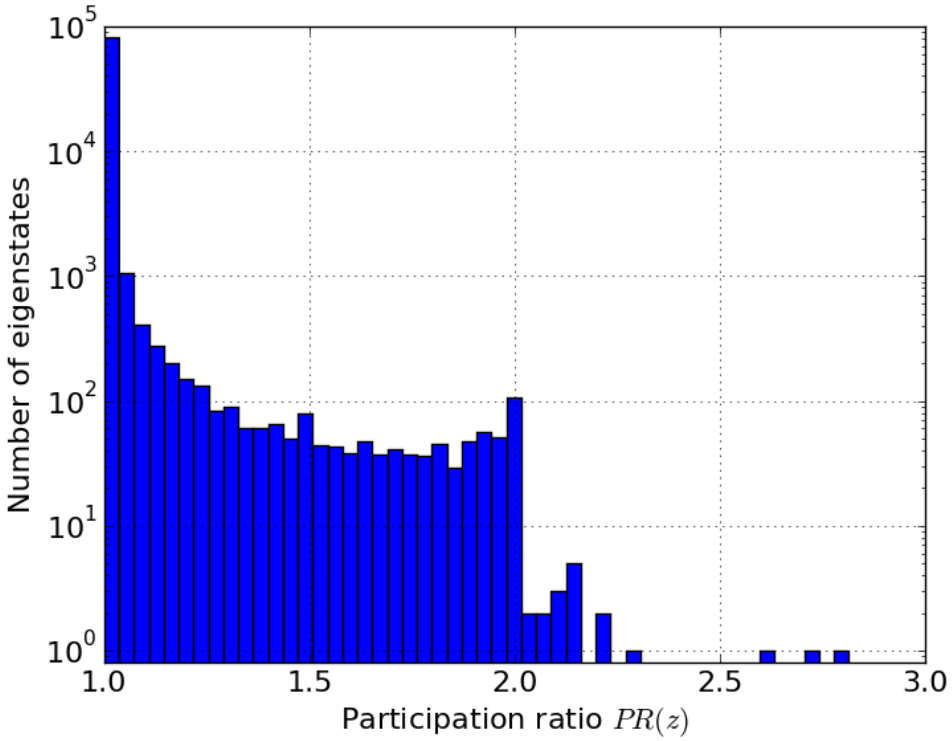

Figure 4.14: Participation ratio histogram of the eigenvectors $\{z_k\}$ of the connectivity matrix for the PPE polymers $B^{(\text{poly})}$. The highest participation ratio $PR(z_{61731}) = 2.814$ is achieved for the eigenvector z_{61731}.

Table 4.5: The number of eigenvectors z_k of the connectivity matrix for the PPE polymers $B^{(\text{poly})}$ in some characteristic ranges of the participation ratio $PR(z_k)$. Note that $PR(z_k) \geq 1.0$ holds for any k because the standard eigenvalue problem is considered here (See Sec. 2.4.1).

Range of $PR(z_k)$	Count of eigenvectors z_k
$1.0 \leq PR(z_k) < 1.05$	80,521
$1.05 \leq PR(z_k) < 1.5$	2,135
$1.5 \leq PR(z_k) < 2.0$	654
$2.0 \leq PR(z_k) < 2.5$	36
$2.5 \leq PR(z_k)$	3

Fig. 4.14 shows the participation ratio histogram for the eigenvectors of $B^{(\mathrm{poly})}$, and Table 4.5 summarizes the histogram by selecting some characteristic ranges. Several eigenvectors with participation ratios $PR(\boldsymbol{z}) \sim 3.0$ are found, which means the presence of small local polymer networks with several (about 3) polymers, as illustrated schematically in Fig. 4.13(c). Such networks are responsible for the electrical current, which will be confirmed in Sec. 4.8.2. Fig. 4.14 and Table 4.5 also show the almost all of eigenvectors have the participation ratios $PR(\boldsymbol{z}) \sim 1.0$, which means that many other polymers are almost isolated or are not connected to others.

4.8.2 Quantum wavepacket dynamics simulation on polymer networks

The quantum wavepacket dynamics simulation was carried out, so as to confirm that electronic wave can propagate on the small polymer networks detected in the previous subsection. From the eigenvectors $\boldsymbol{z}$ with high PR value $PR(\boldsymbol{z})$, the small polymer networks were extracted as isolated systems by selecting several polymer indices which have the largest absolute values. Then the wavepacket simulations were carried out for them.

Several tens of the wavepacket simulations were carried out in different atomic structures and initial wavepackets ($\Psi(\boldsymbol{r}, t=0)$). Figure 4.15 shows a typical result, in which the dynamical simulation was carried out approximately for 1 ps. The initial structure is extracted from the eigenvector $\boldsymbol{z}_{61731}$ with the highest participation ratio shown in Fig. 4.14. Because the participation ratio is $PR(\boldsymbol{z}_{61731}) = 2.814 \sim 3$, 3 polymers are selected by the 3 indices with the largest absolute values in $\boldsymbol{z}_{61731}$. The simulation consumes six hours with 128 nodes. The wavepacket as well as the atom positions change dynamically, though the atomic motion is much slower than that of the wavepacket. In Fig. 4.15, the initially localized wavepacket propagates first through one polymer (Fig. 4.15(a)→(b)) and then propagates to others (Fig. 4.15(b)→(c)). The above behavior is reasonable, because the connection between polymers is much weaker than that within a polymer. The above observation confirms that the network analysis is fruitful for the investigation of charge propagation in organic device materials.

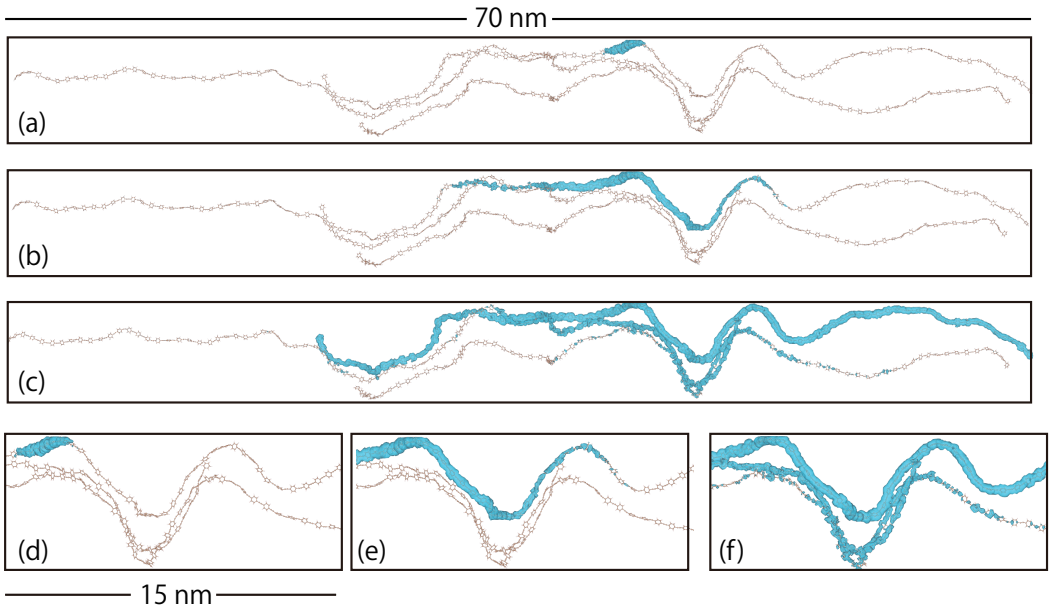

Figure 4.15: Quantum wavepacket dynamics of the condensed polymers. The charge density $|\Psi(\boldsymbol{r},t)|^2$ is plotted at (a) $t = 0$, (b) $t = 50$ and (c) $t = 948$fs, respectively. Figures (d), (e) or (f) is a close up of (a), (b) or (c), respectively.

Chapter 5

Summary and future outlook

Numerical linear algebraic methods were developed for large-scale electronic state calculations on modern massively parallel supercomputers, for example, the full system of the K computer.

The optimal hybrid generalized eigenvalue solver was constructed with the three parallel dense-matrix solver libraries of ELPA, EigenExa and ScaLAPACK. The hybrid solvers with ELPA and EigenExa give better benchmark results than the conventional ScaLAPACK library. The code was developed as a middleware and a mini-application and is available online. The benchmark was carried out with up to a million dimensional matrix on the K computer and other supercomputers. The decomposition analysis of the elapsed time reveals a potential bottleneck part on next-generation (exa-scale) supercomputers, which indicates the guidance for future development of the algorithms and the codes.

An extreme scalability was realized by the code optimization based on the novel Krylov-subspace solver. The simulation was carried out with 10^8-atom simulations for flexible (organic) electronic devices, key devices of next-generation IoT products. The mathematical foundation is generalized shifted linear equations ($(zS - H)\boldsymbol{x} = \boldsymbol{b}$), instead of conventional eigenvalue equations. Since the foundation has a highly parallelizable mathematical structure, the benchmark shows an extreme strong scaling on the full system of the K computer and a qualified time-to-solution for material researches in industrial purpose. The method was applied to the condensed polymer materials in academia-industrial collaboration. A combined research of the quantum simulation and the data analysis reveals that polymers form small local networks and electronic waves propagate on the networks. The present research will give insights of flexible devices and their fabrication process. Since the present method is general, the combined research of huge (100-nm-scale) quantum material simulations and the post-simulation data analysis will be fruitful not only for organic devices but also for many other industrial materials.

Finally two points are discussed as the future outlook; The first point is to develop efficient numerical method for the real-time quantum dynamics simulations with better computational speed and higher reliability. Very recently, for example, we have developed a mini-application for a novel structure-preserving numerical solver [87] with the two-dimensional non-linear Schrödinger equation [1]. The second point is the convergence between simulation and machine learning. Preliminary results are reported on organic polymer device materials in this thesis. The convergence will realize the industrial application of a high-throughput material design.

[1] S. Kudo, H. Imachi, Y. Miyatake, Y. Yamamoto, and T. Hoshi, Application of the mathematical-structure-preserving method to computational material science, The 2nd CDSMI Symposium : Creation of new functional Devices and high-performance Materials to Support next-generation Industries, The University of Tokyo, 6-7. Dec. 2016.

Acknowledgment

The author thanks to Takeo Hoshi, the supervisor of this work, and colleagues for their devoted guidance and discussions. The author also thanks to Toshiyuki Imamura in RIKEN Advanced Institute of Computational Science (AICS) and Takeshi Fukaya in Hokkaido University for fruitful discussions on EigenExa. The author has greatly benefited from Kiyoshi Kumahata, Masaaki Terai, Kengo Miyamoto, Kazuo Minami and Fumiyoshi Shoji in RIKEN AICS for helpful advice on performance tuning of ELSES. Tomofumi Tada in Tokyo institute of Technology gives the author invaluable comments about quantum wavepacket simulation methods. The author is deeply grateful to Yusaku Yamamoto and Shuhei Kudo in The University of Electro-Communications, and Shao-Liang Zhang, Tomohiro Sogabe and Dongjin Lee in Nagoya University, and Takafumi Miyata in Fukuoka institute of technology for insightful comments on numerical linear algebra algorithms. The author appreciates the suggestions on structure-preserving methods by Yuto Miyatake in Nagoya University and Takayasu Matsuo in The University of Tokyo. The author also appreciates precise feedback by Wei-Chung Wang and Yuhsiang Tsai in National Taiwan University. The author also thanks to Masaya Ishida (Sumitomo Chemical Co.) for providing several atomic structure data.

The K computer of RIKEN was used in the research projects of hp150144, hp150281, hp160066 and hp160222.

The supercomputer Oakleaf-FX of The University of Tokyo was used in the research project of 14-NA04 in 'Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures' in Japan, in the 'Large-scale HPC Challenge' Project, Information Technology Center, The University of Tokyo and Initiative on Promotion of Supercomputing for Young or Women Researchers, Supercomputing Division, Information Technology Center, The University of Tokyo.

The supercomputer of Kyoto University was used in Collaborative Research Project for Young, Women Scientists, Institute for Information Management and Communication, Kyoto University.

The author also used the supercomputers SGI altix ICE 8400EX and SGI ICE XA/UV hybrid system at the Institute for Solid State Physics of The University of Tokyo and the supercomputers at the Research Center for Computational Science,

Okazaki.

Bibliography

[1] R. M. Martin, "Electronic Structure: Basic Theory and Practical Methods," Cambridge University Press, Cambridge, 2004.

[2] http://www.gaussian.com/

[3] http://www.vasp.at/

[4] H. Imachi and T. Hoshi, "Hybrid numerical solvers for massively parallel eigenvalue computation and their benchmark with electronic structure calculations," J. Inf. Process. 24, pp. 164–172, 2016.

[5] H. Imachi, S. Yokoyama, T. Kaji, Y. Abe, T. Tada, and T. Hoshi, "One-hundred-nm-scale electronic structure and transport calculations of organic polymers on the K computer," AIP Conf. Proc. 1790, 020010, 4pp., 2016.

[6] T. Hoshi, H. Imachi, K. Kumahata, M. Terai, K. Miyamoto, K. Minami, and F. Shoji, "Extremely scalable algorithm for 10^8-atom quantum material simulation on the full system of the K computer," In Proceedings of ScalA16: Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, held as a part of SC16, Nov. 13, 2016, Salt Lake City, Utah, USA.

[7] http://www.elses.jp/

[8] http://openmp.org/

[9] http://mpi-forum.org/

[10] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," Parallel Comput. 22, 6, pp. 789–828, 1996.

[11] A, J. Van der Steen, "Overview of recent supercomputers," Publication of the NCF, 2008.

[12] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," In Proceedings of the April 18-20, 1967, spring joint computer conference (AFIPS '67 Spring), 1967.

[13] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[14] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed., Johns Hopkins University Press, Baltimore, 2013.

[15] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., Society for Industrial and Applied Mathematics, Philadelphia, 2003.

[16] L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and R. C. Whaley, "An Updated Set of Basic Linear Algebra Subprograms (BLAS)," ACM Trans. Math. Soft., Vol. 28, Issue 2, pp. 135–151, 2002.

[17] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, "LAPACK Users' Guide," Society for Industrial and Applied Mathematics, Philadelphia, 1999.

[18] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, "ScaLAPACK Users' Guide," Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[19] `http://www.netlib.org/scalapack/`

[20] W. Kohn, "Nobel Lecture: Electronic structure of matter - wave functions and density functionals," Rev. Mod. Phys. 71, 1253, 1999.

[21] T. Hoshi, S. Yamamoto, T. Fujiwara, T. Sogabe, and S.-L. Zhang, "An order-N electronic structure theory with generalized eigenvalue equations and its application to a ten-million-atom system," J. Phys. Condens. Matter **21**, 165502, 2012.

[22] T. Hoshi, K. Yamazaki, and Y. Akiyama, "Novel Linear Algebraic Theory and One-Hundred-Million-Atom Electronic Structure Calculation on The K Computer," JPS Conf. Proc. **1**, 016004, 2014.

[23] T. Hoshi, T. Sogabe, T. Miyata, D. Lee, S. L. Zhang, H. Imachi, Y. Kawai, Y. Akiyama, K. Yamazaki, and S. Yokoyama, "Novel linear algebraic theory and one-hundred-million-atom quantum material simulations on the K computer," PoS(IWCSE2013) 065, 13pp, 2014.

[24] R. J. Bell and P. Dean, "Atomic vibrations in vitreous silica," Disc. Faraday Soc. 50, pp. 55–61, 1970.

[25] R. J. Bell, "The dynamics of disordered lattices," Rep. Prog. Phys. 35, 1315, 1972.

[26] D. J. Thouless, "Electron in disordered systems and the theory of localization," Phys. Rep. 13, 93, 1974.

[27] F. Wegner, "Inverse Participation Ratio in 2 + ε Dimensions," Z. Physik B 36, 209, 1980.

[28] B. L. Anderson and R. L. Anderson, "Fundamentals of Semiconductor Devices," McGraw-Hill, New York, 2005.

[29] EigenKernel; `https://github.com/eigenkernel/eigenkernel/`

[30] `http://elpa.rzg.mpg.de/`

[31] A. Marek, V. Blum, R. Johanni, V. Havu, B. Lang, T. Auckenthaler, A. Heinecke, H. J. Bungartz, and H. Lederer, "The ELPA Library – Scalable Parallel Eigenvalue Solutions for Electronic Structure Theory and Computational Science," J. Phys. Condens. Matter **26**, 213201, 2014.

[32] T. Auckenthaler, V. Blum, H. J. Bungartz, T. Huckle, R. Johanni, L. Krämer, B. Lang, H. Lederer, P. R. Willems, "Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations," Parallel Computing, Vol. 37, Issue 12, pp. 783–794, 2011.

[33] `http://www.aics.riken.jp/labs/lpnctrt/index_e.html`

[34] T. Imamura, S. Yamada, and M. Yoshida, "Development of a high-performance eigensolver on a peta-scale next-generation supercomputer system," Prog. Nucl. Sci. Technol., Vol. 2, pp. 643–650, 2011.

[35] T, Imamura, Y. Hirota, T. Fukaya, S. Yamada, and M. Machida, "EigenExa: high performance dense eigensolver, present and future," 8th International Workshop on Parallel Matrix Algorithms and Applications (PMAA14), Lugano, Switzerland, 2014; `http://www.aics.riken.jp/labs/lpnctrt/index_e.html`

[36] T. Imamura, "The EigenExa Library – High Performance & Scalable Direct Eigensolver for Large-Scale Computational Science," ISC 2014, Leipzig, Germany, 2014.

[37] http://www.elses.jp/matrix/

[38] T. Hoshi, Y. Akiyama, T. Tanaka, and T. Ohno, "Ten-million-atom electronic structure calculations on the K computer with a massively parallel order-*N* theory," J. Phys. Soc. Jpn. 82, 023710, 4pp, 2013.

[39] G. Calzaferri and R. Rytz, "The Band Structure of Diamond," J. Phys. Chem. 100, pp. 11122–11124, 1996.

[40] J. Cerdá and F. Soria, "Accurate and transferable extended Hckel-type tight-binding parameters," Phys. Rev. B **61**, pp. 7965–7971, 2000.

[41] http://www.aics.riken.jp/labs/lpnctrt/KMATH_EIGEN_GEV_e.html

[42] F. Tisseur and J. Dongarra, "Parallelizing the divide and conquer algorithm for the symmetric tridiagonal eigenvalue problem on distributed memory architectures," SIAM J. Sci. Comput., Vol. 20, Issue 6, pp. 2223–2236, 1999.

[43] J. Poulson, R. van de Geijn, and J. Bennighof, "Parallel algorithms for reducing the generalized hermitian-definite eigenvalue problem," FLAME Working Note #56, The University of Texas at Austin, Department of Computer Science, Tech. Rep. TR-11-05, 2011.

[44] M. P. Sears, K. Stanley, and G. Henry, "Application of a High Performance Parallel Eigensolver to Electronic Structure Calculations," In Proceedings of the 1998 ACM/IEEE conference on Supercomputing (SC '98), Orlando, FL, 1998.

[45] RIKEN, Press Release at 5. Dec. 2013 (in Japanese); available from http://www.riken.jp/pr/press/2013/20131205_1/

[46] Y. Takahashi, Y. Hirota, and Y. Yamamoto, "Performance of the block Jacobi method for the symmetric eigenvalue problem on a modern massively parallel computer," In Proceedings of ALGORITMY 2012, pp. 151–160, 2012.

[47] Y. Yamamoto, L. Zhang, and S. Kudo, "Convergence analysis of the parallel classical block Jacobi method for the symmetric eigenvalue problem," JSIAM Letters, Vol. 6, pp. 57–60, 2014.

[48] http://math.nist.gov/MatrixMarket/

[49] M. Lefenfeld, G. Blanchet, and J. A. Rogers, "High-performance contacts in plastic transistors and logic gates that use printed electrodes of DNNSA-PANI doped with single-walled carbon nanotubes," Adv. Mater. 14, pp. 1188–1191, 2003.

[50] M. C. Choi, Y. Kim, and C. S. Ha, "Polymers for flexible displays: from material selection to device applications," Prog. Polym. Sci. 33, pp. 581–630, 2008.

[51] Sony Corporation, "Sony develops a 'Rollable' OTFT-driven OLED display that can wrap around a pencil," News Release, 26. May. 2010; `http://www.sony.net/SonyInfo/News/Press/201005/10-070E/index.html`

[52] H. Bässler, "Localized states and electronic transport in single component organic solids with diagonal disorder," Phys. Stat. Solidi B 107, pp. 9–54, 1981.

[53] S. H. Chen, H. L. Chou, A. C. Su, and S. A. Chen, "Molecular packing in crystalline poly(9,9-di-*n*-octyl-2,7-fluorene)," Macromolecules **37**, pp. 6833–6838, 2004.

[54] R. Noriega, J. Rivnay, K. Vandewal, F. P. V. Koch, N. Stingelin, P. Smith, M. F. Toney, and A. Salleo, "A general relationship between disorder, aggregation and charge transport in conjugated polymers," Nature Mater. **12**, pp. 1038–1044, 2013.

[55] J. Terao, A. Wadahama, A. Matono, T. Tada, S. Watanabe, S. Seki, T. Fujihara, and Y. Tsuji, "Design principle for increasing charge mobility of π-conjugated polymers using regularly localized molecular orbitals," Nat. Commun. 4, 1691, 2013.

[56] R. Takayama, T. Hoshi, and T. Fujiwara, "Krylov subspace method for molecular dynamics simulation based on large-scale electronic structure theory," J. Phys. Soc. Jpn. **73**, pp. 1519–1524, 2004.

[57] R. Takayama, T. Hoshi, T. Sogabe, S. L. Zhang, and T. Fujiwara, "Linear algebraic calculation of the Green's function for large-scale electronic structure theory," Phys. Rev. B **73**, 165108, 2006.

[58] H. Teng, T. Fujiwara, T. Hoshi, T. Sogabe, S. L. Zhang, and S. Yamamoto, "Efficient and accurate linear algebraic methods for large-scale electronic structure calculations with nonorthogonal atomic orbitals," Phys. Rev. B **83**, 165103, 2011.

[59] T. Hoshi, Y. Akiyama, T. Tanaka, and T. Ohno, "Ten-million-atom electronic structure calculations on the K computer with a massively parallel order-N theory," J. Phys. Soc. Jpn. 82, 023710, 2013.

[60] T. Hoshi, K. Yamazaki, and Y. Akiyama, "Novel linear algebraic theory and one-hundred-million-atom electronic structure calculation on the K computer," JPS Conf. Proc. 1, 016004, 2014.

[61] S. Nishino, T. Fujiwara, H. Yamasaki, S. Yamamoto, and T. Hoshi, "Electronic structure calculations and quantum molecular dynamics simulations of the ionic liquid PP13-TFSI," Solid State Ionics **225**, pp. 22–25, 2012.

[62] S. Nishino, T. Fujiwara, and H. Yamasaki, "Nanosecond quantum molecular dynamics simulations of the lithium superionic conductor $Li_{4-x}Ge_{1-x}P_xS_4$," Phys. Rev. B **90**, 024303, 2014.

[63] J. M. Soler, E. Artacho, J. D. Gale, A. García, J. Junquera, P. Ordejön, and D. Sänchez-Portal, "The SIESTA method for ab initio order-*N* materials simulation," J. Phys. Condens. Matter **14**, 2745, 2002.

[64] C. K. Skylaris, P. D. Haynes, A. A. Mostofi, and M. C. Payne, "Introducing ONETEP: Linear-scaling density functional simulations on parallel computers," J. Chem. Phys. 122, 084119, 2005.

[65] T. Ozaki, "O(*N*) Krylov-subspace method for large-scale ab initio electronic structure calculations," Phys. Rev. B 74, 245101, 2006.

[66] M. J. Gillan, D. R. Bowler, A. S. Torralba, and T. Miyazaki, "Order-*N* first-principles calculations with the conquest code," Comp. Phys. Commun. **177**, 14, 2007.

[67] W. Kohn, "Density functional and density matrix method scaling linearly with the number of atoms," Phys. Rev. Lett. **76**, pp. 3168–3171, 1996.

[68] A. Frommer, "BiCGStab(*l*) for Families of Shifted Linear Systems," Computing **70**, pp. 87–109, 2003.

[69] S. Yamamoto, T. Sogabe, T. Hoshi, S. L. Zhang, and T. Fujiwara, "Shifted Conjugate-Orthogonal-Conjugate-Gradient Method and Its Application to Double Orbital Extended Hubbard Model," J. Phys. Soc. Jpn., **77**, 114713, 2008.

[70] T. Mizusaki, K Kaneko, M. Honma, and T. Sakurai, "Filter diagonalization of shell-model calculations," Phys. Rev. C **82**, 024310, 2010.

[71] F. Giustino, M. L. Cohen, and S. G. Louie, "GW method with the self-consistent Sternheimer equation," Phys. Rev. E **81**, 115105, 2010.

[72] S. Iwase, T. Hoshi, and T. Ono, "Numerical solver for first-principles transport calculation based on real-space finite-difference method," Phys. Rev. E **91**, 063305, 2015.

[73] K. Yamamoto, A. Uno, H. Murai, T. Tsukamoto, F. Shoji, S. Matsui, R. Sekizawa, F. Sueyasu, H. Uchiyama, M. Okamoto, N. Ohgushi, K. Takashina, D. Wakabayashi, Y. Taguchi, and M. Yokokawa, "The K computer Operations: Experiences and Statistics," In Proceedings of the International Conference on Computational Science (ICCS2014), pp. 576–585, 2014.

[74] TOP500 list; `http://www.top500.org/`

[75] The Graph 500 List; `http://www.graph500.org/`

[76] HPCG Benchmark; `http://www.hpcg-benchmark.org/index.html`

[77] T. Adachi, N. Shida, K. Miura, S. Sumimoto, A. Uno, M, Kurokawa, F. Shoji, and M. Yokokawa, "The design of ultra scalable MPI collective communication on the K computer," Comput. Sci. Res. Dev. **28**, pp. 147–155, 2013.

[78] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl, "GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation," J. Chem. Theory Comput., 4 (3), pp. 435-447, 2008.; http://www.gromacs.org/

[79] J. Wang, W. Wang, P. A. Kollman, and D. A. Case, "Automatic atom type and bond type perception in molecular mechanical calculations," J. Mol. Graph. Model. 25, 247260, 2006.

[80] J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, and D. A. Case, "Development and testing of a general AMBER force field," J. Comp. Chem. 25, pp. 1157–1174, 2004.

[81] A. Troisi and G. Orlandi, "Charge-transport regime of crystalline organic semiconductors: diffusion limited by thermal off-diagonal electronic disorder," Phys. Rev. Lett. 96, 086601, 2006.

[82] R. S. Mulliken, "Electronic Population Analysis on LCAO-MO Molecular Wave Functions. I," J. Chem. Phys. 23, pp. 1833–1840, 1955.

[83] F. Ortmann, F. Bechstedt, and W. G. Schmidt, "Semiempirical van der Waals correction to the density functional description of solids and molecular structures," Phys. Rev. B **73**, 205101, 2006.

[84] J. MacQueen, "Some methods for classification and analysis of multivariate observations," In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, pp. 281–297, 1967.

[85] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer-Verlag New York, Inc., Secaucus, 2006.

[86] R. Dronskowski and P. E. Blöchl, "Crystal orbital Hamilton populations (COHP): energy-resolved visualization of chemical bonding in solids based on density-functional calculations," J. Phys. Chem. **33**, 97, 1993.; `http://www.cohp.de/`

[87] Y. Miyatake and J. C. Butcher, "A characterization of energy-preserving methods and the construction of parallel integrators for Hamiltonian systems," SIAM J. Numer. Anal., 54, 1993, 2016.