Chapter 3
Scalable Eigen-Analysis Engine for Large-Scale Eigenvalue Problems

Tetsuya Sakurai, Yasunori Futamura, Akira Imakura, and Toshiyuki Imamura

Abstract Our project aims to develop a massively parallel Eigen-Supercomputing Engine for post-petascale systems. Our Eigen-Engines are based on newly designed algorithms that are suited to the hierarchical architecture in post-petascale systems and show very good performance on petascale systems including K computer. In this paper, we introduce our Eigen-Supercomputing Engines: z-Pares and EigenExa and their performance.

3.1 Introduction

Our project aims to develop a massively parallel Eigen-Supercomputing Engine for post-petascale systems. This Eigen-Engine is to be developed based on newly designed algorithms that are suited to the hierarchical architecture in post-petascale systems. Our Eigen-Engine is expected to overcome issues of scalability and fault tolerance in conventional eigensolvers. To achieve the aim and provide a high-performance Eigen-Engine that can contribute to practical applications, six research groups are organized, and a variety of studies and developments are conducted through collaboration with researchers in applied mathematics HPC and application fields. Our Eigen-Engine will make it possible to do extensive scale scientific computations that are impossible today and open the door to innovation in various fields of science and industry.

We have collaboratively developed two Eigen-Engines: z-Pares for sparse eigenvalue problems and EigenExa for dense eigenvalue problems. These engines consist of building blocks developed in the project, and their performance have been evaluated in actual applications.

T. Sakurai (✉) · Y. Futamura · A. Imakura
University of Tsukuba, Tsukuba, Ibaraki, Japan
e-mail: sakurai@cs.tsukuba.ac.jp; futamura@cs.tsukuba.ac.jp; imakura@cs.tsukuba.ac.jp

T. Imamura
RIKEN Center for Computational Science, Kobe, Hyogo, Japan
e-mail: imamura.toshiyuki@riken.jp

M. Sato (ed.), *Advanced Software Technologies for Post-Peta Scale Computing*,
https://doi.org/10.1007/978-981-13-1924-2_3

3.2 Sparse Eigen-Super Computing Engine

Here, we consider complex moment-based eigensolvers and their high-performance software: z-Pares for solving the following generalized eigenvalue problem

$$A\boldsymbol{x}_i = \lambda_i B\boldsymbol{x}_i, \quad A, B \in \mathbb{C}^{n\times n}, \quad \boldsymbol{x}_i \in \mathbb{C}^n \setminus \{\boldsymbol{0}\}, \quad \lambda_i \in \Omega \subset \mathbb{C}, \tag{3.1}$$

with sparse matrices A and B, where $zB - A$ is non-singular in a boundary Γ of the target region Ω.

3.2.1 Complex Moment-Based Eigensolvers

3.2.1.1 Basic Concepts

As one of the powerful algorithms for solving (3.1), a complex moment-based eigensolver has been proposed by Sakurai and Sugiura in 2003 [37]. The basic concept is to introduce the rational function

$$r(z) := \widetilde{\boldsymbol{v}}^{\mathrm{H}}(zB - A)^{-1}B\boldsymbol{v}, \quad \boldsymbol{v}, \widetilde{\boldsymbol{v}} \in \mathbb{C}^n,$$

whose poles are the eigenvalues of the generalized eigenvalue problem: $A\boldsymbol{x}_i = \lambda_i B\boldsymbol{x}_i$, and compute all poles located in Ω by solving Hankel generalized eigenvalue problem with complex moments

$$\mu_k := \frac{1}{2\pi \mathrm{i}} \oint_\Gamma z^k r(z) \mathrm{d}z$$

using the method proposed by Kravanja et al. [33]. Now, there are several improvements and variants including direct extensions of Sakurai and Sugiura's approach [20–22, 25, 27, 39] and the FEAST eigensolver developed by Polizzi [36] and its improvements [15, 32, 44, 49, 50].

Let $L, M \in \mathbb{N}$ be the input parameters and $V \in \mathbb{C}^{n\times L}$ be an input matrix. We define $S_k \in \mathbb{C}^{n\times L} (k = 0, 1, \ldots, M-1)$ as follows:

$$S_k := \frac{1}{2\pi \mathrm{i}} \oint_\Gamma z^k (zB - A)^{-1} BV \mathrm{d}z. \tag{3.2}$$

Complex moment-based eigensolvers are mathematically designed based on the properties of the matrices S_k. Then, practical algorithms are derived by approximating the contour integral (3.2) using the numerical integration rule:

$$\widehat{S}_k := \sum_{j=1}^{N} \omega_j z_j^k (z_j B - A)^{-1} BV \mathrm{d}z, \tag{3.3}$$

where z_j is a quadrature point and ω_j is its corresponding weight.

The algorithms of complex moment-based eigensolvers comprise the following three steps:

Step 1. Solve N linear systems with L right-hand sides:

$$(z_j B - A)W_j = BV, \quad j = 1, 2, \ldots, N. \tag{3.4}$$

Step 2. Construct complex moment matrices $\widehat{S}_k (k = 0, 1, \ldots, M-1)$ and others, from $W_j (j = 1, 2, \ldots, N)$.

Step 3. Extract the target eigenpairs from the complex moment matrices.

The most time-consuming part of the complex moment-based eigensolvers is Step 1 that is solving the linear systems (3.4). For solving the linear systems, these eigensolvers have hierarchical parallelism.

Layer 1. Contour paths can be independently performed.
Layer 2. Each linear system can be solved in parallel.
Layer 3. The linear systems can be independently solved.

Because of the hierarchical structure of the algorithms, these methods are expected to achieve high scalability [13, 19, 27, 31, 32, 43, 48]. The algorithm on GridRPC systems is also considered [38, 40].

These methods have been implemented in the form of the high-performance parallel software: z-Pares [52] and FEAST [10], respectively.

3.2.1.2 Theoretical Aspect

The complex moment-based eigensolvers can be regarded as projection methods using a subspace constructed by the contour integral (3.3). The property of the subspace is well analyzed using the so-called filter function:

$$f(\lambda_i) := \sum_{j=1}^{N} \frac{\omega_j}{z_j - \lambda_i},$$

which approximates a band-path filter for the target region Ω. Using the filter function, error analyses of the complex moment-based eigensolvers were given in [15, 23, 24, 44]. An error resilience technique and an accuracy deterioration technique have also been discussed in [16, 28] using the results of the error analyses.

The relationship among typical complex moment-based eigensolvers was also analyzed in [24] focusing on the subspace. The block SS–RR method [21] and the FEAST eigensolver [44] are projection methods directory for solving the target eigenvalue problem (3.1), whereas the block SS–Hankel method [20], Beyn [3], and the block SS–Arnoldi methods [22] are projection methods for solving an implicitly constructed standard eigenvalue problem (see [24] for the details). A map of the relationship among the contour integral-based eigensolvers is presented in Fig. 3.1.

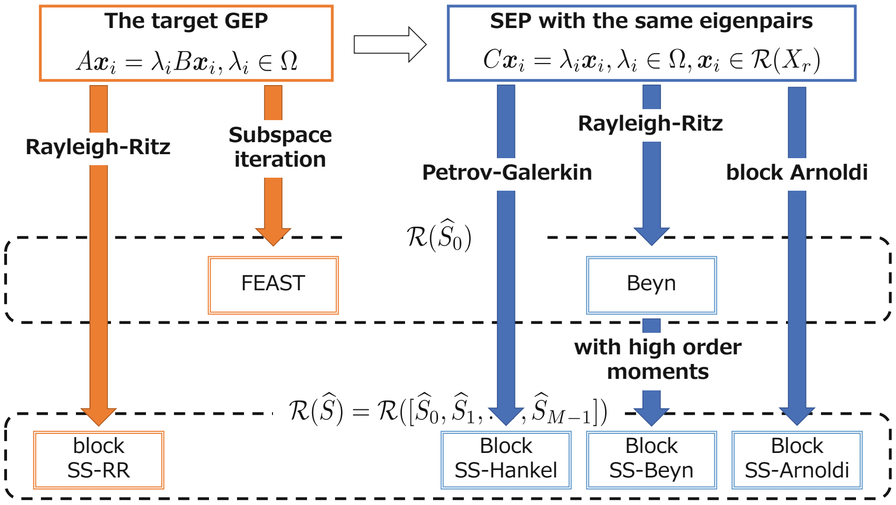
The target GEP
$Ax_i = \lambda_i Bx_i, \lambda_i \in \Omega$
SEP with the same eigenpairs
$Cx_i = \lambda_i x_i, \lambda_i \in \Omega, x_i \in \mathcal{R}(X_r)$
Rayleigh-Ritz
Subspace iteration
Rayleigh-Ritz
Petrov-Galerkin
block Arnoldi
$\mathcal{R}(\widehat{S}_0)$
FEAST
Beyn
with high order moments
$\mathcal{R}(\widehat{S}) = \mathcal{R}([\widehat{S}_0, \widehat{S}_1, \ldots, \widehat{S}_{M-1}])$
block SS-RR
Block SS-Hankel
Block SS-Beyn
Block SS-Arnoldi

Fig. 3.1 A map of the relationship among the contour integral-based eigensolvers

3.2.1.3 Extension to Nonlinear Eigenvalue Problems

The complex moment-based eigensolvers were extended to solve nonlinear eigenvalue problems (NEPs):

$$T(\lambda_i)\boldsymbol{x}_i = \boldsymbol{0}, \quad \boldsymbol{x}_i \in \mathbb{C}^n \setminus \{\boldsymbol{0}\}, \quad \lambda_i \in \Omega \subset \mathbb{C},$$

where the matrix-valued function $T : \Omega \to \mathbb{C}^{n\times n}$ is holomorphic in some open domain Ω.

The block SS–Hankel [1, 2], block SS–RR [51], and block SS–CAA methods [26] are simple extensions of the GEP solvers. Improving technique of the numerical stability of the block SS–RR method for solving NEP was also studied in [7, 8].

As another type of complex moment-based nonlinear eigensolvers, Beyn proposed a method based on Keldysh's theorem and the singular value decomposition [3]. Also, van Barel and Kravanja proposed an improvement of the Beyn method using the canonical polyadic (CP) decomposition [46].

3.2.2 Distributed Parallel Sparse Eigensolver Package z-Pares

3.2.2.1 Introduction

z-Pares is a package for solving generalized eigenvalue problems (3.1). The symmetries and definitenesses of the matrices can be exploited suitably. z-Pares computes eigenvalues inside a user-specified contour path and the corresponding eigenvectors. The most important feature of z-Pares is two-level message passing interface (MPI)-distributed parallelism.

3.2.2.2 Features

The main features of z-Pares are described below.

- Implemented in Fortran 90/95
- Solves standard eigenvalue problems $A\boldsymbol{x} = \lambda\boldsymbol{x}$ and generalized eigenvalue problems $A\boldsymbol{x} = \lambda B\boldsymbol{x}$
- Computes eigenvalues located in an interval or a circle and the corresponding (right) eigenvectors
- Both real and complex types are supported
- Single precision and double precision are supported
- Both sequential and distributed parallel MPI builds are available
- Two-level distributed parallelism can be employed by using a pair of MPI communicators
- Reverse communication mechanism is used to ensure the package accept any matrix data structure
- Interfaces for dense and sparse CSR format are available (only with one-level-distributed parallelism)

3.2.2.3 Dependences

z-Pares depends on following packages:

- BLAS/LAPACK
- Message passing interface (MPI-2 standard)
- MUMPS 4.10.0 (optional)

BLAS/LAPACK should be installed, and MPI is needed for the parallel version of z-Pares. MUMPS is required to use the sparse CSR interface.

3.2.2.4 Basic Concepts of z-Pares

Here we show the schematic illustration of a numerical contour integration (3.3) in Fig. 3.2. As described in Fig. 3.2, numerical quadrature with N quadrature points is used to approximate the contour integral. The basis of the subspace which is used for extracting eigenpairs is computed by solving linear systems with multiple right-hand sides. In this section, matrix V is called a source matrix, and its column vector is called a source vector.

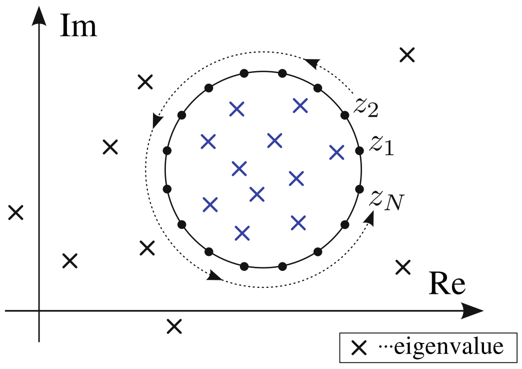

Fig. 3.2 z-Pares computes eigenvalues located inside a contour path on the complex plane (Blue cross)

Because the linear systems can be solved independently, the computations can be embarrassingly parallelized. Additionally, each linear system can be solved in parallel.

3.2.2.5 Two-Level MPI Parallelism

Above the parallelism of quadrature points, there is independent parallelism if multiple contour paths are given. Here we define three levels of parallelism:

- **Top level**: Parallelism of computations on contour paths
- **Middle level**: Parallelism of computations on quadrature points
- **Bottom level**: Parallelism of computations for solving linear systems

z-Pares uses the middle- and bottom-level parallelism by employing a pair of MPI communicators. The MPI communicators that manage the middle level and the bottom level are called *the higher-level communicator* and *the lower-level communicator*, respectively. Because the top-level parallelism can be implemented completely without communications, we have not added the implementation of this level to the feature of z-Pares. Users should manage the top-level parallelism by theirselves if necessary.

The above descriptions are shown in Fig. 3.3.

For the Rayleigh-Ritz procedure and the residual calculations, matrix-vector multiplications (mat-vec) of A and B must be done for multiple vectors. The higher-level communicator manages the parallelization in performing mat-vec for different vectors. The lower-level communicator manages the parallelization for one mat-vec.

3.2.2.6 `zpares_prm` Derived Type

The derived type `zpares_prm` plays a central role in the use of z-Pares. `zpares_prm` consists of components that represent several input and output

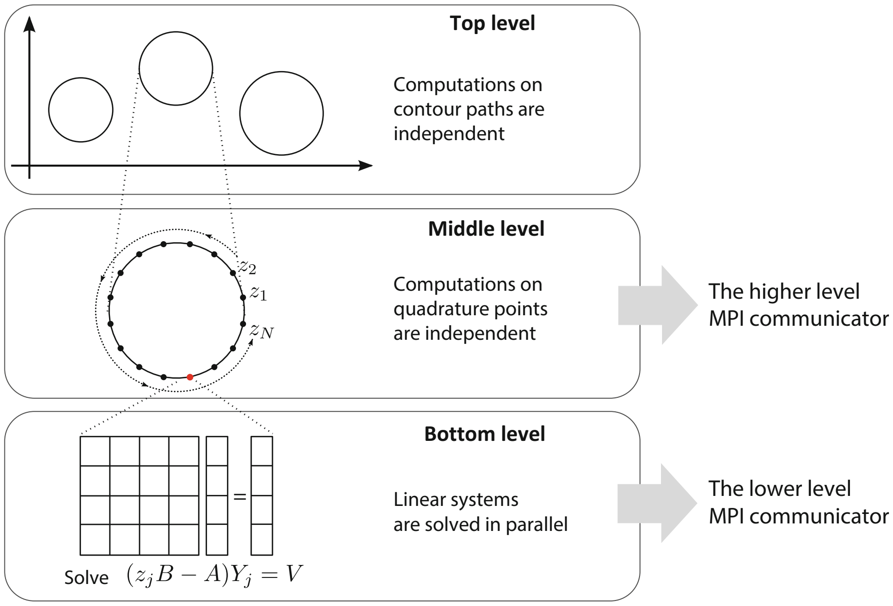
Top level
Computations on contour paths are independent
Middle level
Computations on quadrature points are independent
z_2
z_1
z_N
The higher level MPI communicator
Bottom level
Linear systems are solved in parallel
The lower level MPI communicator
Solve $(z_jB - A)Y_j = V$

Fig. 3.3 Three levels of parallelism and two-level MPI communicator

parameters and inner variables. See z-Pares users' guide for more details. In the rest of this section, an entry of `zpares_prm` is referred to as `prm`.

3.2.2.7 Reverse Communication Interface

z-Pares basically delegates tasks of

- Solving linear systems with multiple right-hand sides $(z_jB - A)Y_j = BV$
- Performing matrix-vector multiplications of A and B

to the user, because an efficient algorithm and matrix data structure are seriously problem dependent.

z-Pares delegates the tasks by using the reverse communication interface (RCI) rather than the modern procedure pointer or an external subroutine.

When using RCI, the user code communicates with the z-Pares subroutine in the following manner:

1. Reverse communication flag `prm%itask` is initialized with `zpares_init` before entering the loop of 2.
2. The z-Pares subroutine is repeatedly called until
   `prm%itask == ZPARES_TASK_FINISH`
3. In the loop of 2., the tasks indicated by `prm%itask` are completed with the user's implementation

When using RCI, the user need not define global, COMMON, or module variables to share information (such as matrix data) with the subroutine given to the package, in contrast to manners using the procedure pointer or external subroutine. RCI is also used in eigensolver packages such as ARPACK and FEAST.

To briefly describe a user code using RCI, a skeleton code for solving complex non-Hermitian problem is given below.

Listing 3.1 Usage of reverse communication interface

```
do while ( prm%itask /= ZPARES_TASK_FINISH )
   call zpares_zrcigegv &
        (prm, nrow_local, z, mwork, cwork, left, right, num_ev, eigval, X, res, info)

   select case (prm%itask)
   case(ZPARES_TASK_FACTO)

      ! Here, the user factorizes (z*B - A)
      ! At the next return from zpares_zrcigegv,
      ! prm%itask==ZPARES_TASK_SOLVE with the same z is returned

   case(ZPARES_TASK_SOLVE)

      ! i = prm%ws; j = prm%ws+prm%nc-1
      ! Here, user solves (z*B - A) X = cwork(:,i:j)
      ! The solution X should be stored in cwork(:,i:j)

   case(ZPARES_TASK_MULT_A)

      ! iw = prm%ws; jw = prm%ws+prm%nc-1
      ! ix = prm%xs; jx = prm%xs+prm%nc-1
      ! Here, the user performs matrix-vector multiplications:
      ! mwork(:,iw:jw) = A*X(:,ix:jx)

   case(ZPARES_TASK_MULT_B)

      ! iw = prm%ws; jw = prm%ws+prm%nc-1
      ! ix = prm%xs; jx = prm%xs+prm%nc-1
      ! Here, the user performs matrix-vector multiplications:
      ! mwork(:,iw:jw) = B*X(:,ix:jx)

   end select
end do
```

`ZPARES_TASK_FINISH`, `ZPARES_TASK_FACTO`, `ZPARES_TASK_SOLVE`, `ZPARES_TASK_MULT_A`, and `ZPARES_TASK_MULT_B` are defined as module variables of the type `integer,parameter` of the `zpares` module. Tasks delegated to the user are indicated by these values.

Implementing a linear solver is a heavy task for users. To allow users to get started with z-Pares easily, we provide two interfaces for a specific matrix data structure:

- Dense interface using LAPACK
- Sparse CSR interface using MUMPS

3.2.2.8 Efficient Implementations for Specific Problems

In the above descriptions, we have used the subroutines for complex non-Hermitian problems. In z-Pares, efficient implementations are given for exploiting specific features of the problem. The following features are considered:

- Symmetry or hermiticity of matrices A and B
- Positive definiteness of B
- $B = I$ (standard eigenvalue problem)

We recommend the user to let z-Pares consider these features by setting appropriate parameters to obtain maximum efficiency.

A stochastic estimation method of eigenvalue distribution in a given domain is proposed in [12, 34]. This method is used to evaluate appropriate parameters. Some properties of the contour integral-type methods are considered to determine efficient parameters in [41, 42].

3.3 EigenExa: Development of a Dense Solver

3.3.1 Introduction

Eigenvalue calculation is a significant tool for scientific numerical simulation and engineering analysis. High-performance and highly reliable software must be available. As the size of problems to be solved and available computer resources become substantial, the practical eigenvalue solver is expected to change accordingly.

When this post-petascale CREST project was initiated, we reviewed the trend of the future hardware technology, microprocessor, accelerator unit, memory module, and interconnect network. We supposed that the following must be an essential requirement to build a next-generation, so-called exascale, supercomputer system:

- CPU socket
  8CPUs+1000FPUs,
  On-chip shared memory,
  1.25TFLOP/s
- Compute node
  8sockets,
  64GB shared memory,
  20TFLOP/s
- System
  10^5nodes,
  6.4PB memory,
  2EFLOP/s

In a rough sketch of the design mentioned above, the system has a heterogeneous and hierarchical combination of CPU modules and memories interconnected. We also predicted that such a complication of hierarchical hardware results in a new hierarchical parallel programming style. In fact, two programming languages were critical issues at the beginning of the project: MPI for a representative tool of distributed parallelism and OpenMP for thread parallelism among computational cores in a shared memory fashion.

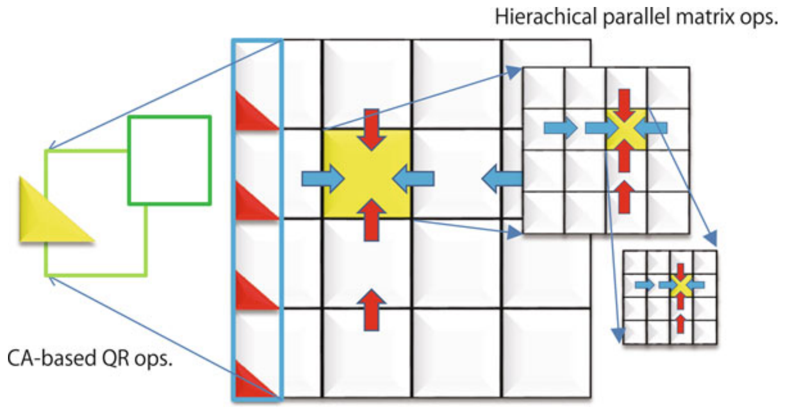

Fig. 3.4 Concept of hierarchical and multi-layered approach in dense linear algebra

As parallel numerical linear algebra software, not only eigenvalue calculation, the configuration of ScaLAPACK + LAPACK + BLAS has been known as de facto standard not changed significantly for nearly 20 years since the 1990s. In the above hierarchization, vertical (upper and lower) parallelism could be handled by a combination of the existing numerical linear algebra software, but it was quite hard to perform higher parallel control by combining flexible parallelism and parallel execution in the same horizontal layer.

What is more, in 2010, a pragmatic innovation was required from not only the software environment but also the parallel algorithm. For one thing, it was necessary to collaborate with a highly parallel/concurrent language to support parallel/concurrent processing over multiple parallel layers and software runtime. Naturally, the emergence of numerical algorithms having multilevel parallelism in every hierarchy was demanded. In fact, there is a need to respond to the emergence of times when hardware has parallelism ranging from hundreds to tens of thousands. In our implementation of algorithms crossing over multiple layers of a dense matrix representation, we decided to realize by the configuration method being aware of any hierarchy as shown in the figure. In particular, we decided to develop mainly block algorithms corresponding to multiple memory hierarchies found in (i) typical high-speed implementation of matrix-matrix products and (ii) local and global communication avoidance, for example, by the CAQR-type approach (Fig. 3.4).

Here, we would like to summarize a brief review of the solver development project, which was promoted during the 2010s. As mentioned previously, ScaLAPACK released in the 1990s keeps still in the position as the de facto standard in distributed parallel environments. Meanwhile, thread parallelism has also to be taken into consideration at the same time by the multicore processor that appeared around 2010. On the other hand, in the HPC community, the development of numerical libraries specialized for thread parallelism for many-core environment equipped with dozens to thousands of cores, the MAGMA project, and the PLASMA project, has been started.

Focused on the trend of numerical eigenvalue libraries, it had been known that xSYEVD of ScaLAPACK based on Cuppen's divide and conquer method had a significant advantage in the distributed parallel environment. Nowadays, a successor routine of xSYEVR, which adopts a different mathematical algorithm of MR3 (multiple relatively robust representations), is still naive implementation. However, the new algorithm has $O(N^2)$ complexity and a promising way (has another name of "the holy grail") to reduce the computational cost regarding flops. As still ScaLAPACK as old design in 1990s, the developer needs to improve thread parallelism. Since 2010, there have been several signs of progress in parallel eigenvalue solvers, such as ELPA by the German team, Elemental, DPLASM by the University of Tennessee, our EigenExa library, and QDWH-based library by KAUST. Each eigensolver library has a unique aspect in the numerical algorithm and parallel implementations. For example, the ELPA library insists significance of a so-called two-stage algorithm, which is a promising way to resolve the matters of narrow memory bandwidth and non-negligible network latency. Also, Elemental employs a brand-new parallel implementation of the MR3 algorithm.

In the rest of the section, we report the status summary of the EigenExa library, which was developed in the CREST project.

3.3.2 Brief History of the Dense Solver Project

Since 2010, the H4ES (=$\mathsf{H}^4\mathsf{ES}$, high performance, high scalability, high portability, and high-reliability Eigen-Supercomputing engine) project conducted by Prof. Tetsuya Sakurai has been kicked off supported by one of the national grant CREST JST. We organized a mini-research group to devote to the development of a dense eigenvalue solver for a post-petascale supercomputer system at the University of Electro-Communications and later at RIKEN Advanced Institute for Computational Science (AICS) from 2012 to the present.

The activities initiated by T. Imamura had not been started since the H4ES project but another CREST project led by Dr. Masahiko Machida, Japan Atomic Energy Agency (2005–2010). The present dense solver project inherited the primary results from the Earth Simulator system, which was the world's largest vector supercomputer [47]. The library was optimized with a combination of a very conservative long-vector-oriented technique and a modern cache-oriented technique, so-called vectorization and efficient cache reuse, respectively. After the Earth Simulator age, we comprehensively scrapped and built up the vector code to multi-threading processing code [30], which was deployed on each system of T2K cluster. The T2K cluster was designed as a commodity supercomputer. Then, the code was ported to the K computer with the help of Fujitsu, and we named the eigensolver library **EigenExa**. The K computer was the first 10 petaFLOPS system housed at RIKEN AICS and is still keeping rank 10 in the top 500 benchmark (Nov. 2017). The K computer has a unique interconnect, namely, Tofu interconnect, and more than 80,000 SPARC64 VIIIfx processors consist of the heart of the system. We observed a big impact of high-performance eigensolver on the RSDFT code, which won the Gordon Bell prize in SC2011 [17], even though the solver was very early version and not optimized so well. Currently, the EigenExa library version 2.4p1 is the latest release [9].

Since 2014, RIKEN AICS has started a national project to develop a flagship system, so-called the post-K system. For that, we continue to improve the EigenExa library toward emerging supercomputer systems, such as Oakforest-PACS, which is the rank 9 system in the top 500 benchmark (Nov. 2017) and is hosted at joint center between the University of Tokyo and University of Tsukuba. Since another aspect on the extreme computing implies not only capability computing but capacity computing, we recognized the necessity of the high-performance eigensolver with a broad variety of parallelism and parallel scaling. It is an entirely challenging work for applied mathematics, computer science, and engineering.

3.3.3 Our Approaches in Parallel Algorithm

As shown, we have been developing an eigenvalue library EigenExa, which consists of multiple eigensolvers, toward next-generation distributed memory parallel supercomputers [30]. For performance improvement of the solvers, it is critical to identify the significant performance bottleneck and remove it on the highly parallel environment [17]. We exploit several algorithmic and implementation techniques to remove bottleneck which comes from data communication among or through a great number of computing nodes.

Main components of EigenExa are two driver routines and one reducer routine from generalized eigenvalue problem (GEVP) to standard eigenvalue problem (SEVP). Two driver routines are:

- `eigen_s`: a conventional scheme, and
- `eigen_sx`: a novel one-stage scheme.

We exploited a novel one-stage scheme for the implementation of `eigen sx`, and its outline is summarized as follows (see also Fig. 3.5).

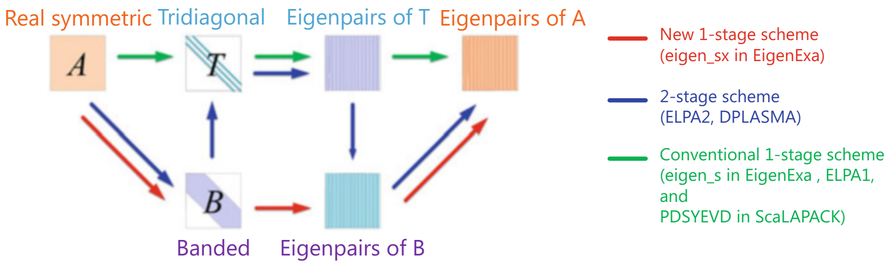

Fig. 3.5 Schematics of Numerical schemes for dense symmetric eigenvalue problem

1. Transform a matrix A to a pentadiagonal matrix P by similarity transformation with an orthogonal matrix V (forward transformation, which consists of multiple Householder transformation): $V^\top AV = P$,
2. Compute the eigenpairs of P by the divide-and-conquer (DC) algorithm for a banded matrix: $P = U\Lambda U^\top$ (where U is an orthogonal matrix and Λ is a diagonal matrix),
3. Compute the eigenvectors of A (back transformation): $Q = VU$.

3.3.3.1 Householder Band Reduction

The first and the third steps are implemented based on Bischof's band reduction algorithm, so-called successively band reduction (SBR) [4, 5]. The parallelization is performed in an MPI/OpenMP hybrid fashion. In `eigen sx`, a pentadiagonal form is used as the intermediate matrix to reduce the internode communication cost, while the tridiagonal form is used in most of the high-performance eigensolvers. Core manipulations are based on block Householder transformation.

1. Compute a block reflector u corresponding to w and an associated lower triangular matrix C, somehow, such that they hold $(I - uCu^\top)w = ER$. Here, E is a unit matrix, and R is an upper triangular matrix.
2. Compute $v_0 := Au$, and $S := Cu^\top v_0 C^\top$.
3. Compute $v := (vC^\top - \frac{1}{2}uS)$.
4. Update $A := A - uv^\top - vu^\top$.

3.3.3.2 Divide and Conquer for a Banded Matrix

The routine for the second step is implemented and modified for the pentadiagonal matrix based on the routine PDSTEDC in ScaLAPACK [6, 45], which is for solving an eigenvalue problem of a tridiagonal matrix. The principle of the algorithm is "single perturbation of a diagonal matrix" defined as $M = D + \rho uu^\top$. For a banded matrix, we can summarize the algorithm as follow. As you can see, Step 2 can be done recursively.

1. Divide $P := P_1 \oplus P_2 + U\Sigma U^\top$, here Σ is a diagonal matrix, and K refers to the half value of the bandwidth of matrix P.
2. Compute eigenproblems P_1 and P_2, somehow.
3. Transform $P_1 \oplus P_2 + U\Sigma U^\top \rightarrow D_1 \oplus D_2 + V\Sigma V^\top$ by similarity transformation.
4. Set $D = D_1 \oplus D_2$.
5. for i=1, . . . , K
6. Solve a single perturbation problem $F := D + \sigma_i v_i v_i^\top$.
7. Set $D := Q^\top FQ$. Here, Q is corresponding eigenvectors of matrix F.
8. Set $V_{i+1:K} := Q[v_{i+1}, \cdots, v_K]$
9. Return the eigenvalues in the diagonal of D and the corresponding eigenvectors in matrix Q.

3.3.3.3 Back Transformation

The third step employs the compact WY representation to accelerate the computational performance as most of the modern solvers do. A core part of the block Householder transformation with the WY representation is as follows.

1. Construct a triangular matrix C corresponding to block reflector from $u = [u_1, u_2, \cdots, u_b]$, such that $I - uCu^\top = (I - u_b u_b^\top) \cdots (I - u_2 u_2^\top)(I - u_1 u_1^\top)$.
2. Update $X := (I - uCu^\top)X = X - uC(u^\top X)$.

Since all the reflector vectors $u = [u_1, u_2, \cdots]$ were already computed in the forward transformation, data redistribution (or broadcasting) of them has many variations. We introduced one of the communication hiding techniques (CH), the overlap of communication, and computation to reduce the communication overhead hiding behind computation.

3.3.4 Performance

In the project, we evaluated and analyzed the feasibility of the algorithm and parallel implementation. Also, performance of `eigen_s` and `eigen_sx` driver routines is investigated using the K computer [29] housed in RIKEN AICS (Project number hp120170 and ra000005). The benchmark presented in this article was done by using EigenExa version 2.5 Release Candidate (development code "c4"), which was developed in February 2018 under the support of KAKENHI grand-in-aid (15H02709).

3.3.4.1 Performance on the K Computer

Figure 3.6 shows a strong scaling benchmark results demonstrated on the K computer. We consider problems of dimension 10,000 to 130,000, and we selected a Frank matrix defined by $(A)_{ij} = \max(i, j)$ as a test matrix, which has eigenvalues represented analytically as $\lambda_k = 1/2(1 - \cos(\pi(2k + 1)/(2N + 1)))$. EigenExa performs on the K computer in a hybrid MPI/OpenMP parallel fashion 8threads/1process deployed on a node with varying the number of processes from 32 to 4096 processes. Our EigenExa libraries yielded excellent performance than the ELPA2 solver does. However, in case of a middle-sized problem, N = 10,000, performance improvement stacked up when more than 200 processes were used because the problem size and the amount of required computational counts were not enough for such large number of processes. However, in the cases of more larger dimensional matrices, N = 50,000 and 130,000, we observed the gradual but acceptable performance improvement up to 4096 processes.

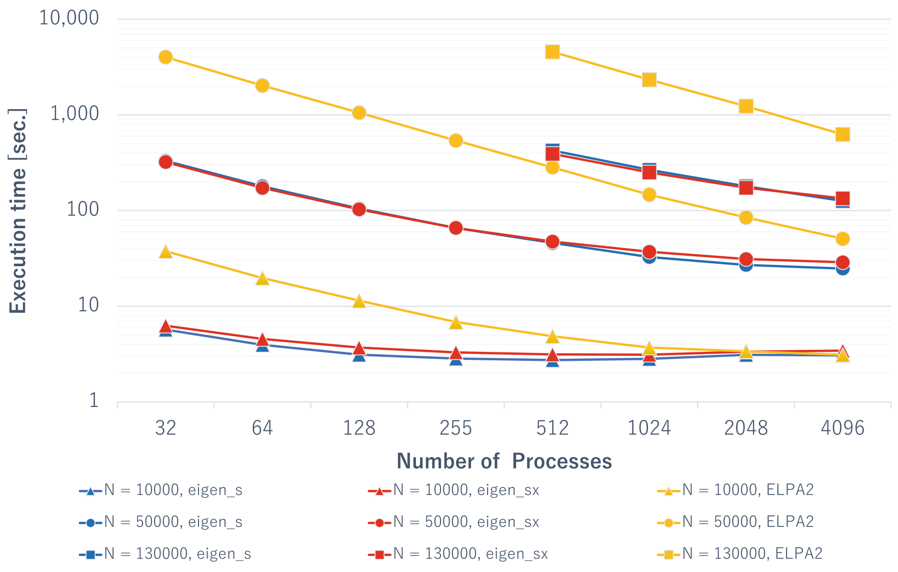

Fig. 3.6 Strong scaling benchmark on the K computer

3.3.4.2 Ultra-Scale Benchmark

The most outstanding result on the development of EigenExa was the success of a full diagonalization of a one million-dimensional matrix by using the whole system of the K computer. The actual log is shown in Fig. 3.7. This ultra-scale benchmark was done under the conditions:

1. EigenExa version 1.0 (`eigen_sx` driver routine).
2. The test matrix was generated as a random matrix symmetrized, $(A + A^\top)/2$.
3. Fujitsu software environment was K-1.2.0-14.
4. Full node of the K computer, i.e., 82944 nodes, was occupied during the job.
5. MPI/OpenMP hybrid parallelism. 1MPI process/node, 8threads/1MPI process.
6. Job time stamp was "Wed Aug 14 23:16:05 JST 2013."

Another experiment on Jan 16, 2014, revealed an accuracy of such a huge-sized eigenvalue problem. The relative residual was $\max_i \|Ax_i - \lambda_i x_i\|_1/N\|A\|_1 = 5.99 \times 10^{-16}$, and the orthogonal error was $\|X^\top X - I\|_F/N = 2.16 \times 10^{-16}$. These observations exhibit that the parallel algorithm adopted in the current implementation works feasibly when even matrix size and parallelism grow in super-scale. Furthermore, the algorithm and parallel implementation are trustable in the exascale era.

From the results through our current and past benchmark tests, we clarified the parallel performance improvement and the performance bottleneck of the solvers on the highly parallel environment. These achievements provide us with deep insight and perspectives of a high-performance numerical library toward the future supercomputer systems such as the post-K computer.

```
NUM.OF.PROCESS= 82944 ( 288 288 )
NUM.OF.THREADS= 8
calc (u,beta)     503.0970594882965
mat-vec (Au)      1007.285000801086 661845.1244051798
2update (A-uv-vu) 117.4089198112488 5678160.294281102
calc v            0.000000000000000
v=v-(UV+VU)u      328.3385872840881
UV post reduction 0.6406571865081787
COMM_STAT
   BCAST  ::  424.3022489547729
   REDUCE ::  928.1299135684967
   REDIST ::  0.000000000000000
   GATHER ::  78.28400993347168
TRD-BLK 1000000 1968.435860157013 677356.7583893638 GFLOPS
TRD-BLK-INFO 1000000   48
before PDSTEDC 0.1448299884796143
PDSTEDC 905.2210271358490
MY-REDIST1 1.544256925582886
MY-REDIST2 14.75343394279480
RERE1 4.861211776733398E-02
COMM_STAT
   BCAST  ::  4.860305786132812E-02
   REDUCE ::  2.155399322509766E-02
   REDIST ::  0.000000000000000
   GATHER ::  0.000000000000000
PDGEMM 532.6731402873993 5417097.565200453 GFLOPS
D&C 921.8044028282166 3130319.580211733 GFLOPS
TRBAK= 573.9026420116425  COMM= 533.7601048946381
    573.9026420116425 3484911.644577213 GFLOPS
    182.3303561210632 5484550.248648792 GFLOPS
    152.0370917320251 6577342.335399065 GFLOPS
    0.1022961139678955 7.379654884338379
COMM_STAT
   BCAST  ::  229.3666801452637
   REDUCE ::  234.4477448463440
   REDIST ::  0.000000000000000
   GATHER ::  0.000000000000000
TRBAKWY 573.9029450416565
TRDBAK 1000000 573.9216639995575 3484796.141101135 GFLOPS
Total 3464.162075996399 1795203.448396145 GFLOPS
Matrix dimension =  1000000
Internally required memory =  480502032  [Byte]
Elapsed time =  3464.187163788010  [sec]
```

Fig. 3.7 Console output of the full-diagonalization benchmark of a one million-dimensional matrix

3.3.5 Related Sub-projects

To develop a dense eigenvalue solver, mathematical and computational innovations are required. Several topics were conducted in the project and interacted with other projects.

3.3.5.1 CholeskyQR2

Orthogonalization by QR factorization is one of the critical issues for the eigen decomposition or internal Householder transformation. Fukaya and Yamamoto et al. proposed the CholeskyQR2 algorithm, which factorizes a tall-skinny matrix in a QR representation, where matrix Q holds $Q^\top Q = I$ and R is an upper triangular matrix.

1. R_0 :=Chol($A^\top A$), st. $A^\top A = R_0^\top R_0$.
2. $Q_0 := AR_0^{-1}$.
3. R_1 :=Chol($Q_0^\top Q_0$).
4. $Q := Q_0 R_1^{-1}$, $R := R_1 R_0$.

They elucidated that CholeskyQR2, which is an algorithm performing CholeskyQR twice, gives excellent accuracy and computing speed in most practical cases. In their experiments using 16,384 nodes of the K computer, CholeskyQR2 outperformed TSQR nearly threefold in computing time for a 4,194,304 $\times$ 64 matrix [14].

3.3.5.2 Performance Modeling

It is inevitable to establish a methodology of the performance prediction model for emerging large-scale systems. In the development of EigenExa, we investigated two styles of performance modeling: (i) an empirical base-function approximation [11] and (ii) the LP method, which introduced suppress of overfitting with nonnegative constraint [35].

In the above empirical study, we reported the evaluation on a Fujitsu PRIME-HPC FX10, which was mainly focused on investigating the differences between the two driver routines. The obtained results were expected to be useful for not only for a future EigenExa library but other parallel dense matrix computations. In contrast to the empirical way, we examined that the LP method predicted more accurately than the LASSO method, which is often used in the sparse modeling field. The LP method exhibited the valid base functions of the EigenExa library systematically from the products of $\{N^3, N^2, N\}$ and $\{1, 1/\sqrt{p}, 1/p\}$, and then we obtained a model function of the collective communication represented by a simple linear combination of the base functions:

$$c_1 \frac{N^2}{\sqrt{p}} + c_2 N^3 + c_3 N.$$

Since theoretical discussions do not provide the term of N^3, it suggests that a significant scale eigenvalue problem might tend to incur severe performance degradation due to (i) non-parallelized parts and (ii) increasing communication overhead. Since similar arguments held for another empirical approximation modeling, we recognized the significance to continue the topics in the future.

3.3.5.3 High-Precision Calculation

The higher precision calculations often demanded in practical engineerings and quantum chemistry to obtain more accurate eigenvalues or identify the algebraic duplicity of a cluster of eigenmodes. For that, it is essential to take account of Bailey's double-double arithmetic as a quadruple precision format from the viewpoint of accuracy and performance. We evaluated the performance of the high-performance quadruple precision eigensolver libraries QPEigenK on the K computer. The latest version of QPEigenK performs exhibiting excellent scalability. We observed that the elapsed time to solve an eigenproblem with $n = 10{,}000$ was 118 seconds on 16384 nodes of the K computer, whereas it was 31 times longer than the case of a double precision solver [18].

Currently, the standardization of IEEE754 half precision format has induced other arguments of computing precision in numerical libraries. Not only high precision but flexible or selectable precision may become significant in future computing with the help of new hardware such as an FPGA device. Also, we expect that the topic could be enhanced to the new research world of reproducible computing, which guarantees identical computing result on any circumstances, anywhere and anytime.

3.4 Conclusion

In this paper, we introduced massively parallel Eigen-Supercomputing Engines: z-Pares and EigenExa, for post-petascale systems. Our Eigen-Engines ware based on newly designed algorithms that are suited to the hierarchical architecture in post-petascale systems and showed very good performance on petascale systems including K computer.

References

1. Asakura, J., Sakurai, T., Tadano, H., Ikegami, T., Kimura, K.: A numerical method for nonlinear eigenvalue problems using contour integrals. JSIAM Lett. **1**, 52–55 (2009)
2. Asakura, J., Sakurai, T., Tadano, H., Ikegami, T., Kimura, K.: A numerical method for polynomial eigenvalue problems using contour integral. Jpn. J. Indust. Appl. Math. **27**, 73–90 (2010)
3. Beyn, W.-J.: An integral method for solving nonlinear eigenvalue problems. Linear Algebra Appl. **436**, 3839–3863 (2012)
4. Bischof, C., et al.: A framework for symmetric band reduction. ACM Trans. Math. Softw. (TOMS) **26**, 581–601 (2000)
5. Bischof, C., et al.: Algorithm 807: the SBR toolbox – software for successive band reduction. ACM Trans. Math. Softw. (TOMS) **26**, 602–616 (2000)
6. Blackford, L.S., et al.: ScaLAPACK Users' Guide. Society for Industrial and Applied Mathematics, Philadelphia (1997)

7. Chen, H., Imakura, A., Sakurai, T.: Improving backward stability of Sakurai-Sugiura method with balancing technique in polynomial eigenvalue problem. Appl. Math. **62**, 357–375 (2017)
8. Chen, H., Maeda, Y., Imakura, A., Sakurai, T., Tisseur, F.: Improving the numerical stability of the Sakurai-Sugiura method for quadratic eigenvalue problems. JSIAM Lett. **9**, 17–20 (2017)
9. EigenExa Homepage: http://www.aics.riken.jp/labs/lpnctrt/en/projects/eigenexa/
10. FEAST Eigenvalue Solver: http://www.ecs.umass.edu/~polizzi/feast/
11. Fukaya, T., Imamura, T.: Performance evaluation of the EigenExa Eigensolver on Oakleaf-FX: Tridiagonalization Versus Pentadiagonalization. In: Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (PDSEC 2015), pp. 960–969 (2015)
12. Futamura, Y., Tadano, H., Sakurai, T.: Parallel stochastic estimation method of eigenvalue distribution. JSIAM Lett. **2**, 127–130 (2010)
13. Futamura, Y., Sakurai, T., Furuya, S., Iwata, J.-I.: Efficient algorithm for linear systems arising in solutions of eigenproblems and its application to electronic-structure calculations. In: Proceedings of the 10th International Meeting on High-Performance Computing for Computational Science (VECPAR 2012), pp. 226–235 (2013)
14. Fukaya, T., Nakatsukasa, Y., Yanagisawa, Y., Yamamoto, Y.: CholeskyQR2: a simple and communication-avoiding algorithm for computing a tall-skinny QR factorization on a large-scale parallel system. In: Proceedings of the 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA14), pp. 31–38 (2014)
15. Güttel, S., Polizzi, E., Tang, T., Viaud, G.: Zolotarev quadrature rules and load balancing for the FEAST eigensolver. SIAM J. Sci. Comput. **37**, A2100–A2122 (2015)
16. Hasegawa, T., Imakura, A., Sakurai, T.: Recovering from accuracy deterioration in the contour integral-based eigensolver. JSIAM Lett. **8**, 1–4 (2016)
17. Hasegawa, Y., et al.: First-principles calculations of electron states of a silicon nanowire with 100,000 atoms on the K computer. In: Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, Article No. 1 (2011)
18. Hirota, Y., Yamada, S., Imamura, T., Sasa, N., Machida, M.: Performance of quadruple precision eigenvalue solver libraries QPEigenK & QPEigenG on the K computer. In: Proceedings of the International Supercomputing Conference (ISC'16). HPC in Asia Poster Session (2016)
19. Ide, T., Toda, K., Futamura, Y., Sakurai, T.: Highly parallel computation of eigenvalue analysis in vibration for automatic transmission using Sakurai-Sugiura method and K computer. SAE Technical Paper, 2016-01-1378 (2016)
20. Ikegami, T., Sakurai, T., Nagashima, U.: A filter diagonalization for generalized eigenvalue problems based on the Sakurai-Sugiura projection method. J. Comput. Appl. Math. **233**, 1927–1936 (2010)
21. Ikegami, T., Sakurai, T.: Contour integral eigensolver for non-Hermitian systems: a Rayleigh-Ritz-type approach. Taiwan. J. Math. **14**, 825–837 (2010)
22. Imakura, A., Du, L., Sakurai, T.: A block Arnoldi-type contour integral spectral projection method for solving generalized eigenvalue problems. Appl. Math. Lett. **32**, 22–27 (2014)
23. Imakura, A., Du, L., Sakurai, T.: Error bounds of Rayleigh–Ritz type contour integral-based eigensolver for solving generalized eigenvalue problems. Numer. Algorithms **71**, 103–120 (2016)
24. Imakura, A., Du, L., Sakurai, T.: Relationships among contour integral-based methods for solving generalized eigenvalue problems. Jpn. J. Ind. Appl. Math. **33**, 721–750 (2016)
25. Imakura, A., Sakurai, T.: Block Krylov-type complex moment-based eigensolvers for solving generalized eigenvalue problems. Numer. Algorithms **75**, 413–433 (2017)
26. Imakura, A., Sakurai, T.: Block SS–CAA: a complex moment-based parallel nonlinear eigensolver using the block communication-avoiding Arnoldi procedure. Parallel Comput. **74**, 34–48 (2018)
27. Imakura, A., Futamura, Y., Sakurai, T.: Structure-preserving block SS–Hankel method for solving Hermitian generalized eigenvalue problems. In Proceedings of 12th International Conference on Parallel Processing and Applied Mathematics (PPAM2017) (2017, accepted)

28. Imakura, A., Futamura, Y., Sakurai, T.: Structure-preserving technique in the block SS–Hankel method for solving Hermitian generalized eigenvalue problems. In: Wyrzykowski, R., Dongarra, J., Deelman, E., Karczewski, K. (eds.) Parallel Processing and Applied Mathematics. PPAM 2017. Lecture Notes in Computer Science, vol. 10777, pp. 600–611. Springer, Cham (2017)
29. Imamura, T., et al.: Current status of EigenExa, high-performance parallel dense eigensolver. In: EPASA2018 (2018)
30. Imamura, T., et al.: Development of a high performance eigensolver on the Peta-Scale next generation supercomputer system. Prog. Nucl. Sci. Technol. **2**, 643–650 (2011)
31. Iwase, S., Futamura, Y., Imakura, A., Sakurai, T., Ono, T.: Efficient and Scalable Calculation of Complex Band Structure using Sakurai-Sugiura Method, In SC'17 proceeding of the International Conference for High Performance Computing, Networking, Storage and Analysis, 17, 2017 (accepted).
32. Kestyn, J., Kalantzis, V., Polizzi, E., Saad, Y.: PFEAST: a high performance sparse eigenvalue solver using distributed-memory linear solvers, In SC'16 proceeding of the International Conference for High Performance Computing, Networking, Storage and Analysis, vol. 16 (2016)
33. Kravanja, P., Sakurai, T., van Barel, M.: On locating clusters of zeros of analytic functions. BIT **39**, 646–682 (1999)
34. Maeda, Y., Futamura, Y., Sakurai, T.: Stochastic estimation method of eigenvalue density for nonlinear eigenvalue problem on the complex plane. JSIAM Lett. **3**, 61–64 (2011)
35. Orii, S., Imamura, T., Yamamoto, Y.: Performance Prediction of Large-Scale Parallel Computing by Regression Model with Non-Negative Model Parameters, IPSJ SIG Technical Reports (High Performance Computing), Vol. 2016-HPC-155, No. 9, pp. 1–9 (2016). (in Japanese)
36. Polizzi, E.: A density matrix-based algorithm for solving eigenvalue problems, Phys. Rev. B **79**, 115112 (2009)
37. Sakurai, T., Sugiura, H.: A projection method for generalized eigenvalue problems using numerical integration. J. Comput. Appl. Math. **159**, 119–128 (2003)
38. Sakurai, T., Hayakawa, K., Sato, M., Takahashi, D.: A parallel method for large sparse generalized eigenvalue problems by OmniRPC in a grid environment. Lect. Notes Comput. Sci. **3732**, 1151–1158 (2005)
39. Sakurai, T., Tadano, H.: CIRR: a Rayleigh-Ritz type method with counter integral for generalized eigenvalue problems. Hokkaido Math. J. **36**, 745–757 (2007)
40. Sakurai, T., Kodaki, Y., Tadano, H., Takahashi, D., Sato, M., Nagashima, U.: A parallel method for large sparse generalized eigenvalue problems using a grid RPC system. Fut. Gen. Comput. Syst. Appl. Distrib. Grid Comput. **24**, 613–619 (2008)
41. Sakurai, T., Asakura, J., Tadano, H., Ikegami, T.: Error analysis for a matrix pencil of Hankel matrices with perturbed complex moments. JSIAM Lett. **1**, 76–79 (2009)
42. Sakurai, T., Futamura, Y., Tadano, H.: Efficient parameter estimation and implementation of a contour integral-based eigensolver. J. Algorithms Comput. Tech. **7**, 249–269 (2013)
43. Shimizu, N., Utsuno, Y., Futamura, Y., Sakurai, T.: Stochastic estimation of nuclear level density in the nuclear shell model: an application to parity-dependent level density in ^{58} Ni. Phys. Lett. B **753**, 13–17 (2016)
44. Tang, P.T.P., Polizzi, E.: FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection. SIAM J. Matrix Anal. Appl. **35**, 354–390 (2014)
45. Tisseur, F., et al.: A Parallel Divide and Conquer Algorithm for the Symmetric Eigenvalue Problem on Distributed Memory Architectures. SIAM J. Sci. Comput. **20**, 2223–2236 (1999)
46. van Barel, M., Kravanja, P.: Nonlinear eigenvalue problems and contour integrals. J. Comput. Appl. Math. **292**, 526–540 (2016)
47. Yamada, S., et al.: High-Performance Computing for Exact Numerical Approaches to Quantum Many-Body Problems on the Earth Simulator (SC06) (2006)
48. Yamazaki, I., Ikegami, T., Tadano, H., Sakurai, T.: Performance comparison of parallel eigensolvers based on a contour integral method and a Lanczos method. Parallel Comput. **39**, 280–290 (2013)

49. Yin, G.: A randomized FEAST algorithm for generalized eigenvalue problems, arXiv:1612.03300 [math.NA] (2016)
50. Yin, G., Chan, R.H., Yeung, M.-C.: A FEAST algorithm for generalized non-Hermitian eigenvalue problems. Numer. Linear Algebra Appl. **24**, e2092 (2017)
51. Yokota, S., Sakurai, T.: A projection method for nonlinear eigenvalue problems using contour integrals. JSIAM Lett. **5**, 41–44 (2013)
52. z-Pares: Parallel Eigenvalue Solver. http://zpares.cs.tsukuba.ac.jp/