JSIAM Letters Vol.10 (2018) pp.77–80 ©2018 Japan Society for Industrial and Applied Mathematics JSIAM Letters

Performance evaluation of the Sakurai-Sugiura method with a block Krylov subspace linear solver for large dense Hermitian-definite generalized eigenvalue problems

Takahiro Yano[1], Yasunori Futamura[1], Akira Imakura[1] and Tetsuya Sakurai[1]

[1] Department of Computer Science, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan

E-mail *yano@mma.cs.tsukuba.ac.jp*

Received September 19, 2018, Accepted October 28, 2018

Abstract

Contour-integral based eigensolvers have been proposed for efficiently exploiting the performance of massively parallel computational environments. In the algorithms of these methods, inner linear systems need to be solved and its calculation time becomes the most time-consuming part for large-scale problems. In this paper, we consider applying a contour-integral based method to a large dense problem in conjunction with a block Krylov subspace method as an inner linear solver. Comparison of parallel performance with the contour-integral based method with a direct linear solver and a ScaLAPACK's eigensolver is shown using matrices from a practical application.

Keywords generalized eigenvalue problem, dense matrix, contour integral method, block Krylov subspace method, distributed computing

Research Activity Group Algorithms for Matrix / Eigenvalue Problems and their Applications

1. Introduction

We consider a dense Hermitian-definite generalized eigenvalue problem (GEP),

$$A\boldsymbol{x}_i = \lambda_i B\boldsymbol{x}_i, \quad i = 1, \ldots, m, \quad m \leq n,$$

where A, $B \in \mathbb{C}^{n\times n}$ are dense Hermitian matrices and B is positive definite. We compute all m eigenvalues in $\mathcal{I} = [a, b]$. It is assumed there are no eigenvalues at a and b. This kind of problem appears in scientific problems, e.g. electronic structure calculations.

To solve a large scale eigenproblem, one may consider using a distributed parallel subroutine implementing a method based on the Householder tridiagonalization in e.g. the ScaLAPACK library and running it on a many-node cluster. Parallel performance of currently available libraries scales to a certain extent; however, in some cases, their scalability is not satisfactory. That is, specifically, in a case where a highly expensive computational kernel (other than the eigensolver) that scales on a large number of computational nodes exists in the application program and the eigensolver poorly exploits the performance of that amount of parallel resources.

On the other hand, state-of-the-art parallel computing environments equip GPUs and/or many-core CPUs. Implementing algorithms that require fine-grained communication like methods based on the Householder tridiagonalization is becoming more and more difficult for such new complex architectures.

Because of this situation, in this paper, we consider evaluating the performance of the Sakurai-Sugiura method (SSM) [1] in conjunction with a block Krylov subspace method as an inner linear solver. As shown in the subsequent sections, owing to its coarse-grained parallelism and simplicity of the computationally dominant kernel, the approach possibly outperforms conventional Householder-based approaches when a large number of computational resources are available.

2. The Sakurai-Sugiura method with Rayleigh-Ritz procedure

Let Γ be a positively oriented Jordan curve which intersects the real axis at a and b of $\mathcal{I}$. Let L, $M \in \mathbb{N}$ be input parameters such that $LM \geq m$ and L must be larger than the maximum algebraic multiplicity of the eigenvalues in $\mathcal{I}$. We construct a complex moment $S := [S_0, \ldots, S_{M-1}] \in \mathbb{C}^{n\times LM}$ by

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

where $k = 0, \ldots, M-1$ and $V \in \mathbb{C}^{n\times L}$ is called a source matrix that is usually generated by random numbers.

For discretizing the integral,

$$\widehat{S}_k := \sum_{j=1}^{N} w_j z_j^k \left(z_j B - A\right)^{-1} BV$$

is used for approximating (1), where z_j is a quadrature point and w_j is a corresponding quadrature weight. We also define $\widehat{S} := [\widehat{S}_0, \ldots, \widehat{S}_{M-1}]$.

We employ the Rayleigh-Ritz procedure for extracting eigenpairs from $\widehat{S}$ [2]. Let $Q \in \mathbb{C}^{n\times K}$ be a matrix whose columns are orthonormal bases of Range($\widehat{S}$), where K is the numerical rank of $\widehat{S}$. Then, we trans-

form the original problem to the projected problem, $Q^{\mathrm{H}}AQ\boldsymbol{t}_i = \theta_i Q^{\mathrm{H}}BQ\boldsymbol{t}_i$. Here, $(\lambda_i, \boldsymbol{x}_i) = (\theta_i, Q\boldsymbol{t}_i)$.

The SSM has hierarchical parallelism which consists of the following layers. **Layer 1**: Multiple intervals can be set for the SSM, then, each interval can be computed simultaneously. **Layer 2**: For each interval, the solutions of N linear systems of order n are required, however, they can be computed simultaneously because they are independent of each other. **Layer 3**: Parallel linear solvers can be used for solving each linear system.

3. The SSM with block Krylov strategy for dense GEPs

3.1 Motivation

Generally, Hermitian-definite dense GEPs are solved by the following steps: 1) Reduction to a standard eigenproblem (SEP) using the Cholesky factorization of B, 2) Tridiagonalization of the SEP by Householder transformations, 3) Eigenpair computation for the tridiagonal form e.g. the bisection method (followed by inverse iterations), 4) Back-transformation of eigenvectors to the original GEP. This type of method is sometimes referred to as a *direct method*. As a ScaLAPACK subroutine, the driver routine PZHEGVX is provided for Hermitian-definite GEPs and uses the bisection method for computing eigenvalues of the tridiagonal form. Thus, it can selectively compute the eigenvalues in a specified interval. However, in distributed computing, the communication related to a level-2 BLAS type kernel used for the Householder transformation becomes a bottleneck. Moreover, the bisection method and the inverse iteration are hard to scale and become an obstacle to parallel speedup. Several efforts have been made to develop highly efficient libraries for direct eigensolvers e.g. [3, 4].

On the other hand, for dense eigenproblems, total sequential computational complexity of the SSM (which is commonly used for sparse eigenproblems) is expected to be greater than direct eigensolvers because multiple linear systems of order n need to be solved in the algorithm of the SSM. However, if one can employ a highly scalable implementation of a linear solver, and at the same time, fully exploit the hierarchical parallelism of the SSM with a large amount of computational resources, the SSM possibly outperforms direct eigensolvers.

In a multi-GPU parallel environment, the distributed parallel performance of the SSM with a direct linear solver for dense eigenproblems is evaluated [5]. An iterative linear solver is also an option while load imbalance may occur at the Layer-1 and -2 parallelisms of the SSM due to possible imbalance of iteration counts. Block Krylov subspace linear solvers are used for solving systems with simultaneously given right-hand sides. In the dense setting, block Krylov subspace linear solvers have good property because their computationally dominant kernel is the matrix-matrix product (GEMM) consist of $n \times n$ and $n \times L$ dense matrices (in this study, we always set the number of right-hand sides for a block Krylov subspace linear solver as the parameter L of the SSM). Obtaining distributed scalability of GEMM is substantially easier than the kernels (e.g. the Householder transformation) of direct eigensolvers.

Moreover, accuracy of the computed eigenpairs can be adjusted by appropriately setting the parameters of the SSM. Thus, we can controll the computational cost according to the demanded accuracy in the application. Accuracy of the computed eigenpairs can be controlled indirectly by appropriately choosing the tolerance for the residual norms of the inner linear systems. This is a main advantage of iterative linear solvers over the direct linear solvers. Therefore, in case where the application accepts rough eigenpairs, the combination of the SSM and a block Krylov subspace iterative solver can be a promising approach for large dense GEPs. Preconditioning for (block) Krylov subspace methods is crucial for the performance; however, performance comparison with preconditioning is beyond the scope of this paper.

3.2 Implementation

In the SSM, the contour path is assumed to be symmetric with respect to the real axis. Then, with a natural ordering of the indices, all pairs of quadrature points have the relations

$$\overline{z_j} = z_{N-j+1}, \quad j = 1, \ldots, N \tag{2}$$

(when N is an even number). Let the coefficient matrix of a linear system with respect to the quadrature point z_j be $C_j = z_j B - A$. From the Hermitian symmetries of A and B and the relations (2), $C_{N-j+1} = C_j^{\mathrm{H}}$ is satisfied. This indicates, in a case where a direct solver used for the linear systems, the LU factors of C_j can be used to solve the linear system of C_{N-j+1}. On the other hand, the bi-Lanczos type (block) Krylov subspace linear solvers can simultaneously solve linear systems of C_j and C_{N-j+1} as the primary and dual systems (by setting initial solutions properly). Because of this characteristic, in this study, we choose a block BiCG type method as the iterative method for the inner linear systems. Specifically, we employ Block BiCGrQ [6] which orthogonalizes the residual matrix at every iterations. For the computationally dominant GEMM kernel involving an $n \times n$ matrix, we use the 2D block distribution.

4. Numerical experiments

In this section, we show the performance comparison of the SSM with Block BiCGrQ, the SSM with the direct linear solver and a ScaLAPACK's direct eigensolver by some numerical experiments. The experiments are performed on the Oakforest-PACS, a huge Knights Landing Xeon Phi cluster operated by JCAHPC, up to 1024 nodes. For all measurements, we assigned four processes per node and 64 OpenMP threads per process with the compact affinity. Also, MCDRAM mode is set as cache.

For the experiments, we used a Hermitian-definite generalized eigenvalue problem obtained from the electronic structure calculation software SIRIUS [7]. The degree of the matrices n is 95951. We compute 1960 eigenvalues in $[-10, -0.1]$ and corresponding eigenvectors. The eigenvalue distribution is shown in Fig. 1.

For the SSM, we set multiple intervals, $[-10.0, -9.0]$, $[-5.0, -4.0]$, $[-4.0, -3.0]$ and $[-1.0, -0.1]$. We call them Interval 1, 2, 3 and 4, respectively. Parameters L, M and

N are set as 256, 8 and 16, respectively in all intervals.

We implemented the method using z-Pares, a software library which implements the SSM, and its new C++ interface. z-Pares itself supports the hierarchical parallelism; however only the Layer-3 parallelism is used in the experiments. Intel Compiler 18.0.1 and Intel MPI 2018 Update 1 are used as the compiler and the MPI environment, respectively. Also we used Intel MKL 2018.0 Update 1 as BLAS, LAPACK and ScaLAPACK.

4.1 Experiment 1

In Experiment 1, we compare scalability of Block BiCGrQ and the direct linear solver of ScaLAPACK, PZGETRF and PZGETRS, for the linear systems, whose right-hand side has L vectors, in the SSM. Experiment 1 is a preliminary experiment for estimating the whole computation time of the SSM in Experiment 2 and investigates how much the both methods can utilize the Layer-3 parallelism. For Block BiCGrQ, we measured the computation time for solving the linear system at a quadrature point on $4^2, 8^2, 12^2, \ldots, 32^2$ nodes. The selected quadrature point is the one which requires the largest iteration count of Block BiCGrQ among all intervals. The iteration counts with respect to all quadrature points are measured in a preliminary experiment. Note that we assume the number of split blocks in the 2D block distribution is the same for rows and columns. Thus, the number of nodes is set as square numbers. Here, we show the scalability of Block BiCGrQ only with tolerance 10^{-6} because the scalability should be same in the different tolerance. For the direct linear solver of ScaLAPACK, we measured the computation time, which includes the LU factorization and two calls of the forward and backward substitutions, at a quadrature point on the same number of nodes. The block sizes of ScaLAPACK's 2D block cyclic distribution, mb and nb, are set as 1024. Then, we calculated the scalability with reference to the result on 16 nodes for the both methods. The results of Experiment 1 are shown in Table 1 and Fig. 2. In Table 1, BB-GEMM, BB-Other, SL-LU and SL-FBS are the measured computation time for the distributed GEMM kernel of Block BiCGrQ, the other part of Block BiCGrQ, PZGETRF and two calls of PZGETRS, respectively.

From Fig. 2, the scalability of Block BiCGrQ is better than that of ScaLAPACK's direct linear solver. From Table 1, PZGETRF does not scale well and PZGETRS does not scale at all. This is because the utilization of a large number of OpenMP threads makes the local computation time too small to take advantage of the distributed parallelism. On the other hand, distributed GEMM kernel scales well and the other part of Block BiCGrQ slightly scales. Thus, Block BiCGrQ for dense matrices is more appropriate than ScaLAPACK's direct linear solver on Knights Landing Xeon Phi clusters.

4.2 Experiment 2

In Experiment 2, we evaluate computation time of the SSM with Block BiCGrQ (SSM-BB), the SSM with the direct linear solver (SSM-SL) and PZHEGVX, eigensolver of ScaLAPACK. For SSM-BB and SSM-SL, we estimate the computation time using the result of Experiment 1. For SSM-BB, we measured the iteration counts of every quadrature points of all intervals with three tolerances for Block BiCGrQ: $10^{-6}, 10^{-8}$ and 10^{-10} only on 64 nodes. Here, we show the assumed parallelisms of the layers for the estimation in Table 2. P_{int}, P_{q}, P_{LS} and P_{total} are the number of (groups of) nodes used for the parallelism of interval, quadrature points and linear solver and the total number of nodes, respectively. Note that Oakforest-PACS has 8208 nodes in total. The parallel scalability of the SSM is estimated as if Oakforest-PACS has more than 32768 nodes.

The computation time on P_{total} nodes, $t_{\mathrm{total}}(P_{\mathrm{total}})$, can be estimated by $t_{\mathrm{total}}(P_{\mathrm{total}}) := \max_{i,j} t_{\mathrm{LS}}^{(i,j)}(P_{\mathrm{LS}}) + t_{\mathrm{other}}(P_{\mathrm{LS}})$, where $t_{\mathrm{LS}}^{(i,j)}(P_{\mathrm{LS}})$ and $t_{\mathrm{other}}(P_{\mathrm{LS}})$ are computation time for solving the linear system on the j-th quadrature point in Interval i and the other part of z-Pares on P_{LS} nodes for using the Layer-3 parallelism, respectively. Note that full use of the Layer-1 and -2 parallelisms is assumed. For SSM-BB, $t_{\mathrm{LS}}^{(i,j)}(P_{\mathrm{LS}})$ is estimated by using the measured iteration count and the average computation time for a loop of Block BiCGrQ on P_{LS} nodes measured in Experiment 1. For SSM-SL, we assumed that the computation time of the direct linear solver is the same as the result of Experiment 1 at all of the quadrature points in all intervals. $t_{\mathrm{other}}(P_{\mathrm{LS}})$ is measured in a preliminary experiment. For PZHEGVX, we measured the computation time on the same number of nodes in Experiment 1 with $(mb, nb) = (1024, 1024)$. The result is shown in Fig. 3.

On a small number of nodes, PZHEGVX is the fastest; however, the computation time stagnates around hundreds of nodes. On the other hand, SSM-BB and SSM-SL are about 3 times slower than PZHEGVX on 16 nodes; although, both methods become faster than PZHEGVX

Table 1. Computation time of the both linear solvers in seconds.

#node	BB-GEMM	BB-Other	SL-LU	SL-FBS
16	728.1	139.0	565.3	57.7
64	218.3	100.9	369.9	66.4
144	201.3	94.2	334.2	70.8
256	154.9	67.9	316.8	64.4
400	115.1	72.0	317.7	65.8
576	87.6	63.7	316.2	65.0
784	65.1	59.7	310.4	61.7
1024	56.8	57.5	308.9	58.9

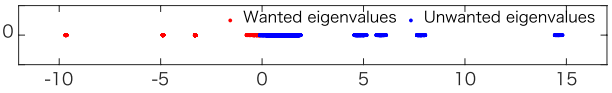

Fig. 1. Eigenvalue distribution of the target problem.

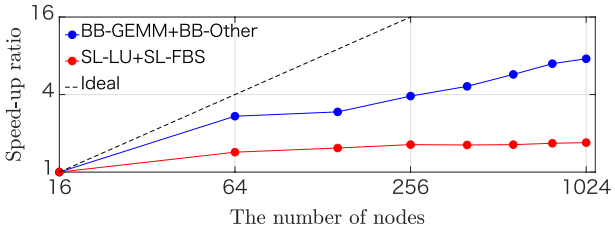

Fig. 2. Scalability of Block BiCGrQ and ScaLAPACK's PZGETRF and PZGETRS.

Table 2. Node usage of the SSM for each parallelism.

P_{int}	P_{q}	P_{LS}	P_{total}	P_{int}	P_{q}	P_{LS}	P_{total}
1	1	16	16	4	8	64	2048
2	1	16	32	4	8	144	4608
4	1	16	64	4	8	256	8192
4	2	16	128	4	8	400	12800
4	4	16	256	4	8	576	18432
4	8	16	512	4	8	784	25088
				4	8	1024	32768

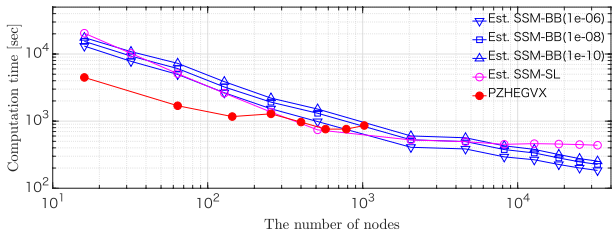

Fig. 3. Computation time comparison with PZHEGVX, SSM-SL and SSM-BB.

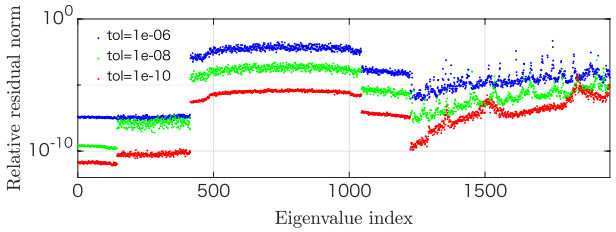

Fig. 4. Relative residual norms of computed eigenpairs by SSM-BB with 3 tolerances.

on over hundreds nodes. Because SSM-SL uses the direct linear solver, it scales linearly when P_{int} increases and also scales almost linearly when P_{q} increases. However, its computation time does not reduce significantly due to the poor scalability of the linear solver. SSM-BB has load imbalance on the Layer-1 and Layer-2 parallelisms due to the differences of the iteration counts. Thus, it does not scale linearly and is slower than SSM-SL on 512 nodes, the case of maximum use of the Layer-1 and -2 parallelisms. However, it becomes the fastest on thousands of nodes because Block BiCGrQ has better scalability than the direct linear solver of ScaLAPACK as shown in Experiment 1. This indicates that SSM-BB will be able to exploit whole of huge computational resources effectively whereas direct eigensolvers may not scale well.

4.3 Experiment 3

In Experiment 3, we evaluate how the tolerance of the solution of the linear systems affects the accuracy of the computed eigenpairs in the SSM. The evaluation is performed by calculating relative residual norms of the computed eigenpairs in all intervals on 64 nodes for each tolerance. The result is shown in Fig. 4.

The result shows rough eigenpairs can be obtained if inaccurate solutions of linear systems are used in the SSM. On the other hand, the relative residual norms of the computed eigenpairs are not uniformly distributed over the intervals. In Intervals 3 and 4, there are many eigenvalues. If LM is not large enough, it is known that the accuracy of the computed eigenpairs degrades [8]. Moreover, there are many clustered eigenvalues in Interval 3. In this case, we should use larger L for computing more accurate eigenpairs. To achieve uniform distribution of the residual norms, parameters and intervals of the SSM should be determined by information of eigenvalue distribution, e.g. estimated eigenvalue count obtained by stochastic estimation [9].

5. Conclusion

In this paper, we evaluated the performance of the SSM with Block BiCGrQ as an inner linear solver. From numerical experiments, SSM-BB is scalable than SSM-SL and ScaLAPACK's PZHEGVX in extremely parallel environments. Also, the computation time can be reduced if the application does not need accurate eigenpairs. In addition, the use of Block BiCGrQ is also appealing in the sense that it has high portability for other architectures (e.g. GPUs) owing to the simplicity of the GEMM kernel.

Since we used only one example in the experiments of this paper, as a future work, we will perform additional experiments using matrices arising from various applications. We will also evaluate actual parallel scalability of our approach using state-of-the-art supercomputers.

Acknowledgments

The authors thank Dr. Anton Kozhevnikov and Prof. Thomas C. Schulthess for providing them the matrices.

References

[1] T. Sakurai and H. Sugiura, A projection method for generalized eigenvalue problems using numerical integration, J. Comput. Appl. Math., **159** (2003), 119–128.

[2] T. Ikegami and T. Sakurai, Contour integral eigensolver for non-Hermitian systems: a Rayleigh-Ritz-type approach, Taiwanese J. Math., **14** (2010), 825–837.

[3] 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** (2014), 213201.

[4] EigenExa, `http://www.r-ccs.riken.jp/labs/lpnctrt/en/projects/eigenexa/`.

[5] T. Yano, Y. Futamura and T. Sakurai, Multi-GPU scalable implementation of a contour-integral-based eigensolver for real symmetric dense generalized eigenvalue problems, In: Proc. 3PGCIC 2013, pp. 121–127, IEEE Computer Society, 2013.

[6] L. Du, Y. Futamura and T. Sakurai, Block conjugate gradient type methods for the approximation of bilinear form $C^H A^{-1} B$, Comput. Math. Appl., **66** (2014), 2446–2455.

[7] A. Kozhevnikov, A. G. Eguiluz and T. C. Schulthess, Toward first principles electronic structure simulations of excited states and strong correlations in nano- and materials science, In: Proc. SC'10, pp. 1–10, IEEE Computer Society, 2010.

[8] A. Imakura, L. Du and T. Sakurai, Error bounds of Rayleigh–Ritz type contour integral-based eigensolver for solving generalized eigenvalue problems, Numer. Algorithms, **71** (2016), 103–120.

[9] Y. Futamura, H. Tadano and T. Sakurai, Parallel stochastic estimation method of eigenvalue distribution, JSIAM Letters, **2** (2010), 127–130.