Efficient Algorithm for Linear Systems Arising in Solutions of Eigenproblems and Its Application to Electronic-Structure Calculations

Yasunori Futamura[1], Tetsuya Sakurai[1,2], Shinnosuke Furuya[3], and Jun-Ichi Iwata[3]

[1] Department of Computer Science, University of Tsukuba, 1-1-1 Tennodai, Tsukuba-shi, Ibaraki 305-8573, Japan
futamura@mma.cs.tsukuba.ac.jp, sakurai@cs.tsukuba.ac.jp
[2] JST CREST, 4-1-8 Hon-cho, Kawaguchi-shi, Saitama 332-0012, Japan
[3] Department of Applied Physics, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
furuya@comas.t.u-tokyo.ac.jp, iwata@ap.t.u-tokyo.ac.jp

Abstract. We consider an eigenproblem derived from first-principles electronic-structure calculations. Eigensolvers based on a rational filter require solutions of linear systems with multiple shifts and multiple right hand sides for transforming the spectrum. The solutions of the linear systems are the dominant part of the eigensolvers. We derive an efficient algorithm for such linear systems, and develop implementation techniques to reduce time-consuming data copies in the algorithm. Several experiments are performed on the K computer to evaluate the performance of our algorithm.

The first-principles electronic-structure calculation based on the density functional theory (DFT) is currently one of the best choices for understanding and predicting phenomena in material sciences. In terms of parallelization on distributed parallel computers, the real-space method in which the basic equation of DFT, the Kohn-Sham equation, is solved as a finite-difference equation is a promising idea due to small communication costs comparing to the other method such as the reciprocal-space method with fast Fourier transformations [1, 2, 3]. A real-space first-principles calculation of a large system, which consists of about 100,000 Si atoms, was performed on the K computer [3]. In such calculations, we have to solve eigenproblems of large Hermitian matrices called the Hamiltonians self consistently, because the Hamiltonian depends on the charge density that is constructed from its eigenvectors correspond to a certain number of the smallest eigenvalues. Thus it is an exterior eigenproblem.

By using the self-consistent charge density, various physical quantities can be calculated. The electronic band structure around the Fermi energy is important information on the electric current flow of the system. In order to get the band structure, eigenproblems of the self-consistent Hamiltonian with several different parameters need to be solved. However, we need only the eigenvalues within a certain range in this case. Thus it is an interior eigenproblem. In

M. Daydé, O. Marques, and K. Nakajima (Eds.): VECPAR 2012, LNCS 7851, pp. 226–235, 2013.

the self-consistent calculation, the exterior eigenproblem with a large number of eigenpairs is inevitable. On the other hand, the band structure calculation is just the interior eigenproblem with relatively small number of eigenpairs, and we can reduce much computational time by utilizing the eigensolver suitable for interior eigenproblems.

In this study, we solve this interior eigenproblem in band structure calculations with the Sakurai-Sugiura (SS) method [4]. The SS method is proposed as a solver for interior eigenproblem. In the method, solutions of linear systems with multiple shifts and multiple right hand sides (RHSs) are required. Ohno *et al.* [5] and Mizusaki *et al.* [6] solve them by a conjugate gradient (CG) type methods for linear systems with multiple shifts. They consider a restricted case of single RHS. We extend these approaches to deal with both multiple shifts and multiple RHSs on an additional degree of freedom. In addition, we introduce implementation techniques to reduce time-consuming data copies of the dominant part in our approach.

The present paper is organized as follows. Section 1 describes a brief introduction of the Sakurai–Sugiura method and linear systems solved in the method. We propose an algorithm of a CG type method for multiple shifts and multiple RHSs in Section 2. We also describe implementation techniques to reduce time-consuming data copies for the algorithm. In Section 3, we show the performance evaluation of our algorithm on the K computer. Conclusions and future work are presented in Section 4.

1 Linear Systems in the Sakurai–Sugiura Method

The Sakurai–Sugiura method is an eigensolver which seeks eigenvalues in specified closed curve and their corresponding eigenvectors. Let $A = A^{\mathrm{H}} \in \mathbb{C}^{n\times n}$. Let us describe the Rayleigh-Ritz projection type method for standard eigenproblem. In the SS method, we calculate matrices

$$S_k \equiv \frac{1}{2\pi \mathrm{i}} \int_{\Gamma} z^k \left(zI - A\right)^{-1} V \mathrm{d}z$$

where, $V \in \mathbb{C}^{n\times L}$ is an arbitrary nonzero matrix, Γ is an Jordan curve, i is the imaginary unit, I is the n dimensional unit matrix, L is called block size usually $L << n$, $z \in \mathbb{C}$ and $k = 0, 1, \ldots, M-1$. Assume that L is greater than maximum multiplicity of eigenvalues in Γ, $L \times M$ is greater than number of eigenvalues in Γ and $M \leq N$. So as to calculate S_k numerically, the N-point trapezoidal rule is applied, and we approximate S_k by

$$\hat{S}_k \equiv \sum_{j=0}^{N-1} w_j \zeta_j^k (z_j I - A)^{-1} V, \tag{1}$$

where z_j and w_j are a quadrature point and a weight, respectively. The condition for ζ_j and w_j is given in [5]. Let $S \equiv [\hat{S}_0, \hat{S}_1, \ldots, \hat{S}_{m-1}] \in \mathbb{C}^{n\times(LM)}$ and

$U \in \mathbb{C}^{n\times(\ell m)}$ be an unitary matrix given by the QR decomposition of $\hat{S}$. Let $\tilde{\lambda}$ and $\tilde{\boldsymbol{x}}$ be eigenvalues and corresponding eigenvectors that obtained by diagonalizing $U^{\mathrm{H}}AU$. Eigenpairs of A are approximated by $\lambda \approx \tilde{\lambda}\boldsymbol{x} \approx U\tilde{\boldsymbol{x}}$. To calculate (1), we should solve linear systems with multiple shifts and multiple RHSs. This process often become the most time-consuming part of the SS method.

2 Solver for Linear Systems with Multiple Shifts and Multiple Right Hand Sides

In the SS method, the linear systems with multiple shifts and multiple RHSs should be solved. From this section, we refer to the target shifted linear systems as

$$(A + \sigma_j I)X_j = B, \quad j = 0, 1, \ldots, N-1. \tag{2}$$

Here, $X_j, B \in \mathbb{C}^{n\times L}$. We refer $AX = B$ as the seed system. Ohno *et al.* [5] and Mizusaki *et al.* [6] solve them by conjugate gradient (CG) type methods in case of $L = 1$. They compare the SS method with a widely used method, the Lanczos method, and found that the methods are comparable. When seed system is Hermitian, the linear systems with multiple shifts can be solved by the shifted CG method [7] even if σ_j are complex numbers [5]. Using the shift invariance of the Krylov subspace, the update of solution vectors for shifted systems can be performed without time-consuming matrix-vector products, i.e. matrix-vector products are only required for the seed system. In this study, we deal with multiple RHSs in addition to multiple shifts to reduce the iteration count by exploiting this additional degree of freedom. A GMRES algorithm for both multiple shifts and multiple RHSs was proposed by Darnell *et al.* [8]. Since we consider the case that the seed system is Hermitian, we choose the CG method as the base method. Thus, we propose the CG method for multiple shifts and multiple RHSs. We refer to the approach shown in [5] as the conventional approach.

2.1 Shifted Block CG-rQ Method

We derive the CG method for multiple shifts and multiple RHSs by extending the block CG method [9] for shifted systems. The block CG method solves systems with multiple RHSs by using the block Krylov subspace [10]. In the block CG method, the search space is extended by L basis per iteration. The block CG method often requires fewer iteration count than the CG method. Several techniques and variants to stabilize the block CG method are presented in [9, 11, 12]. Dubrulle [12] showed that a variant BCGrQ (we refer this as the block CG-rQ method) is the best variant in terms of execution time by numerical experiments. Therefore we choose the block CG-rQ method as the base method of extension for shifted systems. In a similar way as the standard Krylov subspace, the block Krylov subspace also has the shift invariance. Thus there is a relation between

the orthonormalized residual matrix Q_k in the block CG-rQ method for the seed system and the residual matrix of $R_k^{\sigma_j}$ in the block CG method for the shifted systems, that is $R_k^{\sigma_j} = \xi_k^{\sigma_j} Q_k$. Here, $\xi_k^{\sigma_j} \in \mathbb{C}^{L \times L}$. By using this relation, an algorithm of the block CG-rQ method for multiple shifts can be obtained. We refer to this algorithm as the shifted block CG-rQ (SBCGrQ) method . The pseudo code of the algorithm is shown in Fig.1. Note that the time-consuming matrix-vector products with $(A + \sigma_j I)$ do not appear in the algorithm of the SBCGrQ method. The computational cost for the SBCGrQ method is much smaller than that of the case that block CG-rQ method is applied for each shifted system.

If a preconditioner is applied, preconditioned coefficient matrices of shifted linear systems are no longer shifted matrices in general. Thus applicable preconditioners are limited (e.g. the incomplete LU preconditioner can not be applied) for block Krylov subspace methods that use the shift invariance. For this reason we omit considering preconditioners in this study.

1: $X_0^{\sigma_j} = O_{n \times L}$, $\xi_{-1}^{\sigma_j} = \alpha_{-1} = I_L$,
2: $Q_0 \rho_0 = \mathrm{qr}(B)$
3: $\xi_0^{\sigma_j} = \Delta_0 = \rho_0$, $P_0^{\sigma_j} = P_0 = Q_0$
4: **for** $k = 0, 1, \ldots$ until solutions converge **do**
5: $\quad \alpha_k = \left(P_k{}^{\mathrm{H}} A P_k\right)^{-1}$
6: $\quad X_{k+1} = X_k + P_k \alpha_k \Delta_k$
7: $\quad Q_{k+1} \rho_{k+1} = \mathrm{qr}(Q_k - A P_k \alpha_k)$
8: $\quad \Delta_{k+1} = \rho_{k+1} \Delta_k$
9: $\quad P_{k+1} = Q_{k+1} + P_k \rho_{k+1}^{\mathrm{H}}$
10: $\quad$ **for** $j = 0, 1, \ldots, N-1$ **do**
11: $\quad\quad \xi_{k+1}^{\sigma_j} = \rho_{k+1} \left[I_L + \sigma_j \alpha_k + \left\{ \rho_k - \xi_k^{\sigma_j} \left(\xi_{k-1}^{\sigma_j}\right)^{-1} \right\} \left(\alpha_{k-1}\right)^{-1} \rho_k^{\mathrm{H}} \alpha_k \right]^{-1} \xi_k^{\sigma_j}$
12: $\quad\quad \alpha_k^{\sigma_j} = \alpha_k \left(\rho_{k+1}\right)^{-1} \xi_{k+1}^{\sigma_j}$
13: $\quad\quad \beta_k^{\sigma_j} = \alpha_k \left(\rho_{k+1}\right)^{-1} \xi_{k+1}^{\sigma_j} \left(\xi_k^{\sigma_j}\right)^{-1} \left(\alpha_k\right)^{-1} \rho_{k+1}^{\mathrm{H}}$
14: $\quad\quad X_{k+1}^{\sigma_j} = X_k^{\sigma_j} + P_k^{\sigma_j} \alpha_k^{\sigma_j}$
15: $\quad\quad P_{k+1}^{\sigma_j} = Q_{k+1} + P_k^{\sigma_j} \beta_k^{\sigma_j}$
16: $\quad$ **end for**
17: **end for**

Fig. 1. Pseudo code of the SBCGrQ method. $O_{n \times L}$ is the $n \times L$ dimensional zero matrix. I_L is the L dimensional unit matrix. $\mathrm{qr}(C)$ indicates the QR decomposition of matrix C.

To implement the SBCGrQ method for distributed parallel computers, we introduce the row-wise distribution. We implement our distributed parallel code with Message Passing Interface (MPI). In row-wise distribution, matrix-matrix product with a Hermitian transpose matrix in the third line and the QR decomposition in the 7th line are performed with MPI_Allreduce to sum local results. The parallel implementation for the matrix-vector products AP_k depends on the application. The calculations in lines 8,11-13 are replicated. Other lines can be executed without MPI communications.

2.2 Efficient Implementation with Recurrence Unrolling

In the SS method, a number of shifted systems should be solved. In such a case, computational cost for lines 11-15 becomes dominant. Especially lines 14,15 are the most time-consuming part of the algorithm. In addition, the computational cost of lines 14,15 increases $O(L^2)$ with increasing L. We reduce execution time for this computation by following techniques. Fig.2 shows an naive implementation of the 9th line. Note that we reuse the memory area of the variables with subscript k for corresponding variables with subscript $k+1$. We use simplified notations of the two BLAS subroutines ZGEMM and ZCOPY. Here, ZGEMM(A,B,C) operates $C \leftarrow AB+C$ and ZCOPY(A,B) operates $B \leftarrow A$. To

```
ZGEMM(P_k^{σ_j}, α_k^{σ_j}, X_{k+1}^{σ_j})
ZCOPY(Q_{k+1}, T)
ZGEMM(P_k^{σ_j}, β_k^{σ_j}, T)
ZCOPY(T, P_{k+1}^{σ_j})
```

Fig. 2. Naive implementation. $T \in \mathbb{C}^{n \times L}$ is a temporary variable.

exploit the efficiency of the cache blocking of ZGEMM, we operate the products $P_k^{\sigma_j}\alpha_k^{\sigma_j}$ and $P_k^{\sigma_j}\beta_k^{\sigma_j}$ in block as $P_k^{\sigma_j}[\alpha_k^{\sigma_j}, \beta_k^{\sigma_j}]$. The drawback of this approach is that additional 2 ZCOPY calls for X^{σ_j} are required. We reduce the total number of ZCOPY calls by unrolling the recurrences for X_{k+1} and P_{k+1}. The recurrences can be unrolled as

$$X_{k+1}^{\sigma_j} = X_{k-u}^{\sigma_j} + \sum_{h=0}^{u-1} Q_{k-h}\gamma_h^{\sigma_j} + P_{k-u}\gamma_u^{\sigma_j}$$

and

$$P_{k+1}^{\sigma_j} = Q_{k+1} + \sum_{h=0}^{u-1} Q_{k-h}\delta_h^{\sigma_j} + P_{k-u}^{\sigma_j}\delta_u^{\sigma_j}.$$

Here,

$$\begin{cases} \gamma_0^{\sigma_j} = \alpha_k^{\sigma_j} \\ \gamma_h^{\sigma_j} = \alpha_{k-h}^{\sigma_j} + \beta_{k-h}^{\sigma_j}\gamma_{h-1}^{\sigma_j} \end{cases},$$

$$\begin{cases} \delta_0^{\sigma_j} = \beta_k^{\sigma_j} \\ \delta_h^{\sigma_j} = \beta_{k-h}^{\sigma_j}\delta_{h-1}^{\sigma_j} \end{cases}$$

and

$$\theta_h^{\sigma_j} = [\gamma_h^{\sigma_j}, \delta_h^{\sigma_j}].$$

Fig.3 shows the implementation which uses these relations. By this implementation, the total number of ZCOPY calls is reduced from $2K$ to $4K/u$ when $u > 2$ since ZCOPY is only called every u iterations. Here K is the number of iterations which is required to satisfy the stopping criterion. Simular to the implementation in Fig.2, we reuse the memory area of the variables with subscript $k-u$ for corresponding variables with subscript $k+1$. The problem is that the implementation shown in Fig.3 requires an additional memory requirement, mainly that of Q_{k-h} $(h = 0, 1, \ldots, u-1)$. Note that this memory requirement is comparable with that of $X_k^{\sigma_j}$ and $P_k^{\sigma_j}$ $(j = 0, 1, \ldots, N-1)$ when $u \approx N$.

```
if mod(k + 1,u + 1)= 0 then
  ZCOPY(X^{σ_j}_{k−u}, T2(:,1:L))
  ZCOPY(Q_{k+1}, T2(:,L + 1:2L))
  for h = 0, 1, ..., u − 1 do
    ZGEMM(Q_{k−h}, θ_h, T2)
  end for
  ZGEMM(P^{σ_j}_{k−u}, θ_u, T2)
  ZCOPY(T2(:,1:L), X^{σ_j}_{k+1})
  ZCOPY(T2(:,L + 1:2L), P^{σ_j}_{k+1})
end if
```

Fig. 3. Implementation with recurrence unrolling. $T_2 \in \mathbb{C}^{n \times 2L}$ is a temporary variable.

3 Numerical Experiments

In this section, we perform numerical experiments to evaluate the efficiency of the SBCGrQ method and the recurrence unrolling technique described in the previous section. In the experiments, all examples are performed on the K computer. The K computer is a distributed memory supercomputer system which has more than 80,000 compute nodes. It is currently under development at the RIKEN Advanced Institute for Computational Science as a Japanese national project. A SPARC64TM VIIIfx CPU which has eight cores is equipped for a compute node. The clock frequency and the peak performance of the CPU are 2 GHz and 128 giga-flops, respectively. The target system is a silicon nanowire which consists of 9924 Si atoms [3]. The dimension of the Hamiltonian matrix A is $n = 8,719,488$. Our code is compiled with Fujitsu Fortran Compiler. We describe common parameter setting for all experiments as follows. The contour pass for the SS method is a circle with a center of 0.05 and a radius of 0.01. The number of quadrature points is $N = 32$. The RHS vectors are generated by random numbers. We executed the experiments with 768 MPI processes and each MPI process had 8 OpenMP threads. Note that the results of the numerical experiments are tentative since they are obtained by early access to the K computer.

First, we evaluate the execution time of the SS method, the number of eigenvalues that can be obtained by the SS method, and the iteration count and the execution time for the SBCGrQ method. The results of experiments are shown in Table 1. The parameter u is set to $u = 32$. Table 1 shows the elapsed time for the SS method is mostly occupied by the solutions of the linear systems with the SBCGrQ method in all cases. Large *#eig* is obtained by large L. This result is predictable since large subspace is given by large number of RHSs. The remarkable thing is that although the number of linear systems to be solved increase L-fold, *linsol_time* does not. This trend is mainly supported by the behavior that *#iter* decreases with increasing L as is the case in the block CG method [11]. We have succeeded to extend this feature for multiple shifts by developing the SBCGrQ method. Note that the case $L = 1$ and the conventional approach described in [5] are equivalent except that scaling of the vectors are different and the conventional approach was not implemented with recurrence unrolling. Thus, we can find in the column *Speed-up* for the case $L = 32$ that the SBCGrQ method is more than five times faster than the case that the shifted CG method is sequentially applied to each RHS if these is no significant difference in the iteration count for different RHSs.

Table 1. *#iter* and *linsol_time* are iteration count and elapsed time for SBCGrQ method, respectively. *#eig* is the number of eigenvalues derived in contour pass with relative residuals less than 1e−2. *SS_time* is elapsed time for the SS method. *Speed_up* is the speed-up ratio of average elapsed time for one RHS comparing to $L = 1$, i.e. $(128.2 \times L)$ / *linsol_time*.

L	1	2	4	8	16	32	64
#iter	10626	10560	9999	8382	6501	4455	4026
#eig	10	21	43	82	159	212	271
SS_time [sec]	131.8	197.7	247.0	395.2	442.5	721.1	1714.6
linsol_time [sec]	128.2	195.3	246.3	349.3	432.8	698.1	1600.5
Speed-up	1	1.31	2.08	2.93	4.74	5.87	5.12

Next we see the detailed data that support the remarkable results described above. Fig.4 shows the results of experiments to see the behaviors of the dominant parts of the SBCGrQ method with increasing L. *Matvec* is the elapsed time of the matrix-vector products with A in the 5th line of Fig.1. *QR* is the elapsed time of the QR decomposition for the 7th line of Fig.1. *Shift* is the elapsed time of the calculations for lines 11-15 of Fig.1. Note that the time data are average data for one RHS of one iteration. *Matvec* slightly decreases with increasing L since latency for communication was reduced by sending or receiving L-fold data at once. *QR* increases with increasing L since the computational cost increases $O(L^2)$. The most time-consuming item *Shift* decreases until $L = 16$. This result indicates that the efficiency of cache blocking of ZGEMM hides the growth of the computational cost. However, *Shift* increases when $L = 32, 64$ due to the high

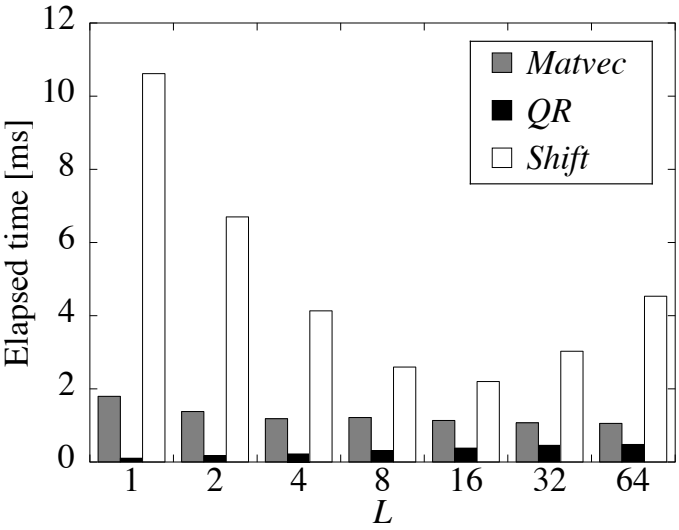

Fig. 4. Details of elapsed time for *linsol_time*

complexity. Fig.5 shows the results of experiments to see the behaviors of the dominant parts of *Shift* with increasing u of the recurrence unrolling technique. The number of RHSs is fixed to $L = 32$. *Square* is the elapsed time for calculations that involve L dimensional square matrices in lines 11-13 of Fig.1. *ZCOPY*

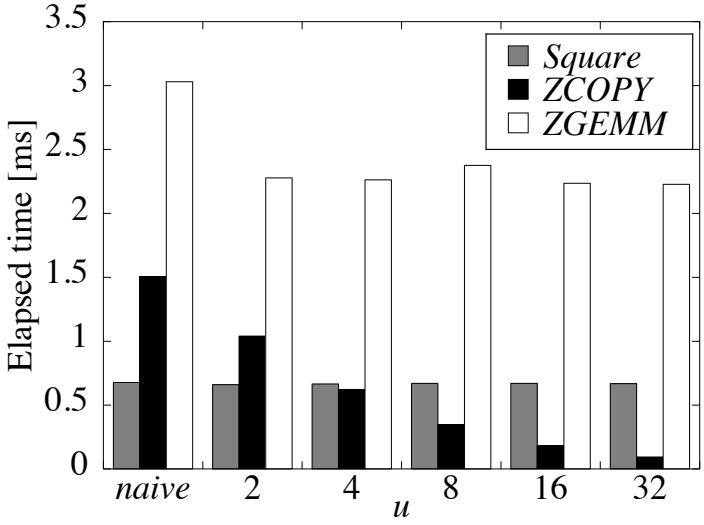

Fig. 5. Details of elapsed time for *Shift*

and *ZGEMM* are the elapsed time for ZCOPY and ZGEMM in Fig.2 or Fig.3 that implement the calculations for lines 14-15 of Fig.1. Note that the time data indicates average data for one shift of one iteration. The computational cost for *Square* other than *naive* is larger than that of *naive* due to calculations for θ_h. Practically, the elapsed time of all cases rarely different since this additional computational cost is negligible. We can find that elapsed time for ZGEMM is reduced by the recurrence unrolling technique. This is because the cache hit ratio is improved by merging two calls of ZGEMM into once. Moreover the elapsed time for ZCOPY decreases linearly with increasing u, since ZCOPY is only called every u iterations. We can find in these details that the efficient use of ZGEMM and the reduction of total call for time-consuming ZCOPY contribute to the remarkable efficiency of the SBCGrQ method.

4 Conclusions and Future Work

We have proposed a CG type method for linear systems with multiple shifts and multiple RHSs and efficient implementation techniques that reduce time-consuming data copies in the method. The proposed method can be used for linear systems that arise in solutions of eigenproblems by an interior eigensolver such as the SS method. We utilized the proposed method for the electronic-structure calculation of a large system which consists of about 10,000 Si atoms. We have found that the proposed method solves the linear systems more than five times faster than the conventional approach and have shown how much our implementation techniques contribute to efficiency of the proposed method. For future work, we will apply the proposed method in unprecedented simulations to clarify important physical properties.

Acknowledgments. This research was supported in part by a Grant-in-Aid for Scientific Research of Ministry of Education, Culture, Sports, Science and Technology, Japan, Grant number: 21246018 and 21105502. The results of the numerical experiments are obtained by early access to the K computer at the RIKEN Advanced Institute for Computational Science.

References

1. Chelikowsky, J.R., Troullier, N., Wu, K., Saad, Y.: Higher-order finite-difference pseudopotential method: An application to diatomic molecules. Phys. Rev. B 50(16), 11355–11364 (1994)
2. Iwata, J.I., Takahashi, D., Oshiyama, A., Boku, T., Shiraishi, K., Okada, S., Yabana, K.: A massively-parallel electronic-structure calculations based on real-space density functional theory. J. Comput. Phys. 229(6), 2339–2363 (2010)
3. Hasegawa, Y., Iwata, J.I., Tsuji, M., Takahashi, D., Oshiyama, A., Minami, K., Boku, T., Shoji, F., Uno, A., Kurokawa, M., Inoue, H., Miyoshi, I., Yokokawa, M.: First-principles calculations of electron states of a silicon nanowire with 100,000 atoms on the K computer. In: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2011, pp. 1:1–1:11. ACM, New York (2011)

4. Sakurai, T., Sugiura, H.: A projection method for generalized eigenvalue problems using numerical integration. J. Comput. Appl. Math. 159(1), 119–128 (2003)
5. Ohno, H., Kuramashi, Y., Sakurai, T., Tadano, H.: A quadrature-based eigensolver with a krylov subspace method for shifted linear systems for Hermitian eigenproblems in lattice QCD. JSIAM Letters 2, 115–118 (2010)
6. Mizusaki, T., Kaneko, K., Honma, M., Sakurai, T.: Filter diagonalization of shell-model calculations. Phys. Rev. C 82(2), 024310 (2010)
7. Jegerlehner, B.: Krylov space solvers for shifted linear systems. arXiv:hep-lat/9612014v1 (1996)
8. Darnell, D., Morgan, R.B., Wilcox, W.: Deflated GMRES for systems with multiple shifts and multiple right-hand sides. Linear Algebra Appl. 429(10), 2415–2434 (2007)
9. O'Leary, P.D.: The block conjugate gradient algorithm and related methods. Linear Algebra Appl. 29, 293–322 (1980); Special Volume Dedicated to Alson S. Householder
10. Gutknecht, M.H., Schmelzer, T.: The block grade of a block Krylov space. Linear Algebra Appl. 430(1), 174–185 (2009)
11. Nikishin, A.A., Yeremin, A.Y.: Variable block CG algorithms for solving large sparse symmetric positive definite linear systems on parallel computers, I: General iterative scheme. SIAM J. Matrix Anal. Appl. 16, 1135–1153 (1995)
12. Dubrulle, A.A.: Retooling the method of block conjugate gradients. Electron Trans. Numer. Anal. 12, 216–233 (2001)