Fault-tolerant LOBPCG for nuclear CI calculations

Meiyue Shao[1], Dossay Oryspayev[2], Chao Yang[3], Pieter Maris[4], and Brandon Cook[5]

[1]School of Data Science, Fudan University, Shanghai 200433, China
[2]Computational Science Initiative, Brookhaven National Laboratory, Upton, NY 11973, USA
[3]Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
[4]Department of Physics and Astronomy, Iowa State University, Ames, IA 50011, USA
[5]National Energy Research Scientific Computing Center, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

June 6, 2023

Abstract

Exascale computing platforms with millions of compute units and with thousands of nodes are predicted to experience frequent faults which interrupt applications' execution. In this context resilience against faults becomes important. We examine user and software level fault mitigation strategies in a distributed LOBPCG algorithm targeting nuclear CI calculations. In particular, we present and evaluate one strategy that keeps the total number of fault-tolerant LOBPCG iterations close to that of the standard LOBPCG algorithm ran on a fault-free machine.

Keywords. Fault-tolerance, sparse symmetric eigensolver, LOBPCG, nuclear configuration interaction

1 Introduction

The highest number of cores based on current top 10 supercomputers from the June 2022 list of `top500.org` is about 10.7 million cores. At this scale and beyond the occurrence of faults will be a norm [3], and not be treated as an exception. The study performed by [4] shows that as the number of components of these large-scale systems increases the reliability and *Mean Time Between Failures (MTBF)* of HPC systems decreases.

According to [3], fault resilience techniques can be broadly classified into three categories—*Hardware Resilience*, *Resilient Systems Software*, and *Application-Based Resilience*. The technique we present in this work falls under the Application-Based Resilience category. In particular, here, we present a fault resilient application level algorithmic technique in the context of a distributed locally optimal block preconditioned conjugate gradient (LOBPCG) implementation, which in case of a single MPI process failure allows for dynamic recovery of the execution process of the application and converge in approximately the same number of iterations as if no

fault has occurred. Moreover, if the program runs without any fault, our technique does not consume additional resources for storage or computations. The technique is validated on real world matrices generated in nuclear physics configuration interaction (CI) calculations [5, 10, 12], but are not restricted for these specific types of matrices.

The rest of the paper is organized as follows. In Section 2, we briefly review nuclear CI calculations and the LOBPCG algorithm. In Section 3, we make assumption on the fault model that we will handle. Then we discuss strategies for recovery from a fault in the LOBPCG algorithm in Section 4. Finally, in Section 5 we use simulated examples to verify the effectiveness of our fault-tolerant LOBPCG algorithm.

2 LOBPCG for Nuclear CI

2.1 Nuclear CI

For a nuclei with P protons and N neutrons, the nuclear structure is described by the Schrödinger equation

$$H\Psi(r_1, \ldots, r_{P+N}) = \Psi(r_1, \ldots, r_{P+N})\lambda, \tag{1}$$

where H is the Hamiltonian, $\Psi(r_1, \ldots, r_{P+N})$ is the wavefunction, and λ is the corresponding energy. In order to numerically compute a few lowest energy levels of (1), the wavefunction is expanded as a linear combination of products of proton and neutron Slater determinants:

$$\Psi(r_1, \ldots, r_{P+N}) = \sum_s \alpha_s \Phi_s^{\text{proton}}(r_1, \ldots, r_P)\Phi_s^{\text{neutron}}(r_{P+1}, \ldots, r_{P+N}), \tag{2}$$

where

$$\Phi^{\text{proton}}(r_1, \ldots, r_P) = \frac{1}{\sqrt{P!}} \det \begin{bmatrix} \phi_{p_1}(r_1) & \phi_{p_2}(r_1) & \cdots & \phi_{p_P}(r_1) \\ \phi_{p_1}(r_2) & \phi_{p_2}(r_2) & \cdots & \phi_{p_P}(r_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{p_1}(r_P) & \phi_{p_2}(r_P) & \cdots & \phi_{p_P}(r_P) \end{bmatrix}$$

is the proton Slater determinant of P protons, and the neutron Slater determinant Φ^{neutron} is defined similarly.

In many nuclear structure software (e.g., MFDn [8, 12] and Bigstick [5]), the single-particle wavefunctions $\phi_a(r)$'s are chosen to be eigenfunctions of a 3D harmonic oscillator. Then each single-particle wavefunction is indexed by four quantum numbers (n, l, j, m_j), i.e.,

$$a \leftrightarrow (n, l, j, m_j).$$

where j and m_j satisfy the constraints

$$j = \left| l \pm \frac{1}{2} \right|, \qquad m_j \in \{-j, ..., j\}.$$

In practice, an energy truncation scheme that keeps many-body states (i.e., basis functions in (2)) with

$$\sum_{i=1}^{P+N} (2n_i + l_i) \in \left\{N_0, N_0 + 2, \ldots, N_0 + N_{\max}\right\},$$

is often used as a criterion to truncate the series expansion (2), where N_0 is the smallest total oscillator quanta, and $N_{\max}$ is an even number specified by the user. In addition, many-body

states are further filtered by the so-called *M-scheme* so that

$$\sum_{i=1}^{P+N} (m_j)_i = \begin{cases} 0, & \text{if } P+N \text{ is even,} \\ 1/2, & \text{if } P+N \text{ is odd.} \end{cases}$$

Once the many-body states are determined, there are different ways to organize them into smaller groups so that the discretized Hamiltonian admits certain block structures [2, 5]. For instance, we can partition many-body states according to total oscillator quanta and total m_j values for protons and neutrons. Many-body states within the same group are ordered contiguously in the discretized Hamiltonian. It has been observed that certain block structures are beneficial for numerical computation, especially for constructing effective preconditioners in the eigensolver [10].

2.2 The LOBPCG algorithm

The locally block optimal preconditioned conjugate gradient (LOBPCG) algorithm [6] is an iterative eigensolver that computes a few smallest eigenvalues and corresponding eigenvectors of a real symmetric matrix A by solving the trace minimization problem

$$\min_{X^\top X = I_k} \operatorname{tr}(X^\top A X).$$

It performs Rayleigh–Ritz procedure iteratively over the subspace $\operatorname{span}\{X^{(i)}, T^{-1}R^{(i)}, X^{(i-1)}\}$, where $R^{(i)} = (I - X^{(i)}X^{(i)\top})AX^{(i)}$ and T, respectively, are the residual and the preconditioner at ith iteration. This method has been used in a number of applications [7], including nuclear configuration interaction calculations implemented in the software package MFDn [10].

For many applications, the matrix A is often large and sparse. Both A and approximations to X and other intermediate quantities are partitioned and distributed among many MPI processes. A distributed version of the LOBPCG algorithm has been implemented in large-scale nuclear configuration interaction (CI) software MFDn [10, 12]. This work mainly concerns fault-tolerant strategies of LOBPCG targeting nuclear CI calculation, though the strategies are applicable for a much wider range of problems.

When one of the MPI processes fails temporarily due to hardware or system software issues, the matrix and vector data kept on that MPI process is lost. Rather than restarting the computation from scratch, after the failed MPI process is recovered, we would like to make use of information available on other MPI processes to achieve fast recovery in time to solution. We shall discuss recovery strategies in Section 4. Our strategies require no checkpointing and can recover on the fly in case when MPI process failure is only temporary.

2.3 Localized eigenvectors

For nuclear CI calculations, additional localization properties of the eigenvectors can be exploited to accelerate the convergence of the LOBPCG algorithm. Since the series expansion (2) converges, the coefficients α_s's usually decay reasonably rapidly. This fact holds for a wide range of discretization if the basis involved in the series expansion is a Schauder basis. In addition, the basis is ordered with asymptotically increasing numerical values (i.e., Rayleigh quotients). As a consequence, typically the eigenvectors correspond to the lowest eigenvalues are nearly localized, in the sense that most entries have small magnitudes except for a few leading entries. Figure 1 illustrates a typical distribution on the magnitudes of the eigenvectors in nuclear CI calculations.

Since most trailing entries of the desired eigenvectors have small magnitudes, dropping these entries naturally yields approximate eigenvectors that can be used as a good initial guess for the

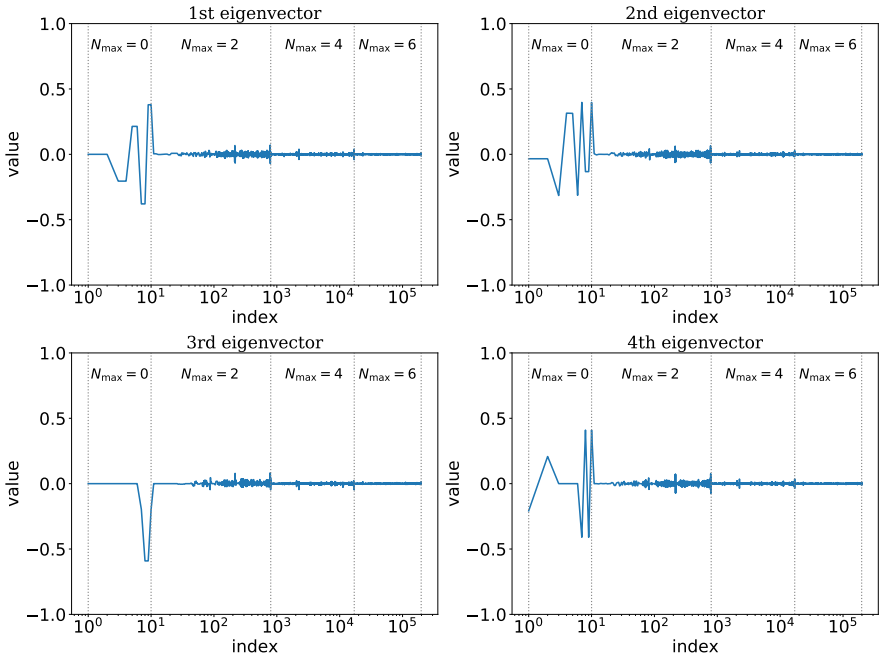

Figure 1: The eigenvectors correspond to four lowest eigenvalues for ^{6}Li with $N_{\max} = 6$. The sizes of the Hamiltonian for $N_{\max} = 0$, $N_{\max} = 2$, $N_{\max} = 4$, and $N_{\max} = 6$ are 10×10, 800×800, $17{,}040 \times 17{,}040$, and $197{,}822 \times 197{,}822$, respectively.

LOBPCG algorithm. Based on this idea, a multi-level LOBPCG algorithm has been proposed in [10] to accelerate nuclear CI calculations. We shall further exploit the localization property in the context of fault-tolerant computation in Section 4.

3 Fault Model

In nuclear CI calculations large-scale parallel computing is usually required due to the enormous problem size. Both the eigenvectors and the Hamiltonian, if ever explicitly constructed, need to be stored in a distributed manner. In this work we assume that the eigenvectors, as well as other auxiliary vectors in the LOBPCG algorithm, admit a 1-D distribution. More precisely, each MPI process stores a block row of the vectors.

In order to handle possible faults in large-scale parallel computing, we make the following assumptions. The fault model we consider in this work assumes that a fault occurs on one particular node during the execution of the LOBPCG algorithm. When the fault occurs, all data residing on the faulty compute node is lost, and all remaining nodes are notified about the fault. The compute node is then quickly rebooted, or we are provided with a new one by the underlying system. New MPI processes are restored on the node, and connection with all remaining MPI processes in the network is re-established. We shall make use of data stored on remaining nodes to resume the computation.

4 Algorithmic-Based Fault-Tolerant Strategy

To recover from a fault, it is necessary to reconstruct the matrix elements of A, along with the components in the (approximate) eigenvectors. In many applications [5], most matrix elements are not explicitly stored—they are generated on the fly when performing matrix–vector multiplications. In this case, only a small amount of information that describes the matrix needs to be restored. In some other applications [8, 12], the faulty compute node has to regenerate the matrix elements assigned to it. Fortunately, in nuclear CI calculations matrix elements can usually be generated locally and independently, i.e., without involving other MPI processes, though sometimes to accelerate the recovery process distributing the work among other MPI processes is a possibility as well. Therefore, recovering matrix elements is in general a relatively easy task. As such we shall focus on how to recover computed eigenvectors within the LOBPCG algorithm.

4.1 A strategy by filling zeros

Convergence of the LOBPCG algorithm partially depends on how close the initial guess is to the desired eigenvector solution. When a good approximation is available, the algorithm may converge in a few iterations. If a node or a processor failure occurs shortly before LOBPCG converges, we would like to recover the approximate solution produced right before the fault, and use that as the starting guess of a new run initiated after the failing node is recovered.

Suppose $X^{(i)}$ is the approximate eigenvector obtained at the ith LOBPCG iteration. Without loss of generality, let us assume that $X^{(i)}$ is partitioned as

$$\begin{bmatrix} X_1^{(i)} \\ X_2^{(i)} \end{bmatrix}, \tag{3}$$

where $X_2^{(i)}$ is assumed to have been lost on some MPI ranks, and $X_1^{(i)}$ still remains on other working MPI ranks.

As we have mentioned in Section 2.3, it is natural that most entries of the eigenvector have relatively small magnitudes in the CI calculation. If the lost data corresponds to these small entries, it is possible to ignore these entries and replace them by zeros. That is, when the failed MPI processes are recovered, we can simply replace the $X_2^{(i)}$ block of (3) by zero, and use

$$\begin{bmatrix} X_1^{(i)} \\ 0 \end{bmatrix}$$

as the approximate eigenvectors. Then we restart the LOBPCG algorithm with this approximation. A similar idea has been adopted in [10] to construct a good initial guess in the multi-level LOBPCG algorithm for nuclear CI calculations. We remark that in order to adopt this strategy, the LOBPCG algorithm needs to be carefully implemented so that it can properly handle non-orthogonal vectors. Otherwise explicit orthogonalization would be required.

However, it is unlikely for such a simple strategy to always produce a good approximation. If the faulty compute node contains dominant entries of an eigenvector, filling zeros introduces large approximation errors. This sort of recovery strategy will generate a bad approximation and hence cause slow convergence of the LOBPCG algorithm. Surprisingly, even if the faulty node only contains tiny entries of the eigenvectors, in practice we often observe limited advantage of this strategy compared to a new run of LOBPCG with the original initial guess.

4.2 A shift-and-invert strategy

Another strategy is to treat $X_2^{(i)}$ as an unknown in the following eigenvalue equation

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} X_1^{(i)} \\ X_2^{(i)} \end{bmatrix} = \begin{bmatrix} X_1^{(i)} \\ X_2^{(i)} \end{bmatrix} \operatorname{diag}\{\lambda_1^{(i)}, \ldots, \lambda_k^{(i)}\}. \tag{4}$$

When LOBPCG is close to convergence, equation (4) holds approximately (with a small residual norm). Therefore, the $X_2^{(i)}$ block is expected to be restored with good quality by solving this linear system. The second block row of (4) is a Sylvester equation of $X_2^{(i)}$ that can be solved column-wise through

$$(A_{22} - \lambda_j^{(i)} I) X_2^{(i)}(:, j) = -A_{21} X_1^{(i)}(:, j), \tag{5}$$

where the notation $(:, j)$ denotes the jth column. This strategy has been discussed in [1, 10], though the discussion in [10] was not tailored to fault-tolerance.

We remark that the right-hand side of (5) can be easily computed even if the operation $x \mapsto A_{21}x$ is not directly applicable, as we can always perform the matrix–vector multiplication

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} X_1^{(i)} \\ 0 \end{bmatrix} = \begin{bmatrix} A_{11} X_1^{(i)} \\ A_{21} X_1^{(i)} \end{bmatrix} \tag{6}$$

and extract the lower block from the output. The matrix–vector multiplication (6), which requires MPI communications, is not really expensive because it is cheaper compared to one step of the LOBPCG algorithm. Compared to the entire eigenvalue problem, equation (5) is a small linear system and hence is relatively cheap to solve. Since (4) and (5) only hold approximately, there is no need to solve (5) very accurately. Once the right-hand side of (5) has been computed, $X_{2,j}^{(i)}$ can be solved by an iterative algorithm (e.g., MINRES [9]). If only one MPI process encounters a fault, and A_{22} and $X_2^{(i)}$ reside on the same MPI process, equation (5) can be solved locally without MPI communications. Similar to the strategy that sets $X_2^{(i)}$ to zero, recovered eigenvectors need to be orthogonalized unless the LOBPCG implementation can properly handle non-orthogonal vectors.

Table 1: List of test cases. The column '$\texttt{iter}_{\text{free}}$' contains the number of iterations required by a fault-free LOBPCG algorithm.

Nuclei	$N_{\max}$	Matrix Dimension	Nonzero Elements	$\texttt{iter}_{\text{free}}$
^{5}He	8	271,023	114,566,624	81
^{7}He	6	347,157	95,016,100	60
^{6}Li	6	197,822	53,577,955	61
^{7}Li	6	663,527	211,712,541	83
^{8}Li	4	103,364	17,693,519	71
^{9}Li	4	152,791	27,261,582	62

4.3 Ordering

In large-scale calculations, natural ordering of many-body states is not a good choice since the work load is often not balanced among MPI processes. For instance, the MFDn software used to use a cyclic distribution of groups of many-body states to achieve balanced work load [11]. A zigzag ordering was later introduced in MFDn to slightly improve the load balance compared to the cyclic ordering. The Bigstick software also uses a similar idea to distribute groups of many-body states [5]. While these ordering strategies were proposed mainly for load balance, they are also beneficial when fault-tolerance is taken into account.

In natural ordering, many-body states with the smallest indices, which are the most dominant ones, are all located on the same MPI process. If this process encounters a fault, entries correspond to the most dominant states, which typically have large magnitudes, are lost. Since the eigenvectors are often localized as discussed in Section 2.3, we can only restore the missing information using entries with small magnitudes. The linear system (5) is very ill-conditioned because the inverse of the underlying linear operator has a large norm. Consequently, it can be expected that the quality of recovered entries is not satisfactory.

The situation becomes different if cyclic or zigzag ordering is adopted. In cyclic or zigzag ordering, consecutive groups of many-body states are distributed over different MPI processes so that they are not lost all together according to our fault model. Therefore, when encountering faults, cyclic and zigzag orderings have better chances for quicker recovery compared to the natural ordering. Based on these ordering strategies one can expect better results compared to natural ordering even in case of utilizing a random ordering. We shall see this from results of numerical experiments presented in Section 5.

5 Numerical Experiments

In the following, we use *simulated* experiments to illustrate the effectiveness of our fault-tolerant LOBPCG algorithm. The experiments were performed on the Linux cluster `Andes` at Oak Ridge Leadership Computing Facility (OLCF).[1]

We choose six test cases as listed in Table 1 with two-body interaction. Five smallest eigenvalues are computed with a relative residual 10^{-5} for each test case. When the shift-and-invert strategy is used, we solve (5) with a relative residual 10^{-6} using MINRES with up to 20 iterations. The number of iterations required by a fault-free LOBPCG algorithm, denoted by $\texttt{iter}_{\text{free}}$, is also listed in Table 1 for each test case.

The program is run with 15 processes and one of them encounters a fault so that about 6.7% of the data is lost. Then we artificially introduce exactly one fault on a certain process (by default,

[1] `https://docs.olcf.ornl.gov/systems/andes_user_guide.html`

the 0th process unless otherwise specified) at jth iteration (for $1 \leq j \leq \texttt{iter}_{\text{free}}$) and record the total number of iterations for the fault-tolerant LOBPCG algorithm to converge. If a new LOBPCG run from scratch is performed after the occurrence of a fault, the remaining number of iterations is expected to be $\texttt{iter}_{\text{free}}$ so that the total number of iterations is $\texttt{iter}_{\text{free}} + j$, which grows linearly with the recovery point j.

We first use ^{5}He with $N_{\max} = 8$ as a concrete example. Figure 2 shows the number of iterations of the fault-tolerant LOBPCG algorithm in different settings. Our fault-tolerant strategies are helpful in the sense that they require fewer iterations compared to a new LOBPCG run from scratch which takes $\texttt{iter}_{\text{free}} + j$ iterations, where j is the recovery point. The easiest situation for recovery is that trailing entries under natural ordering are lost because the missing entries are the least important ones. Even in this simplest case, Figure 2(a) demonstrates that the strategy of filling zeros is not satisfactory because the total number of iterations eventually grows with the recovery point j. The shift-and-invert strategy works perfectly in this case according to Figure 2(c) because the total number of iterations is always close to that from a fault-free run.

Figures 2(b)–(f) illustrate the influence of the ordering. For cyclic, zigzag, or even a random ordering, the shift-and-invert strategy behaves similarly compared to the simplest case—the total number of iterations is close to that from a fault-free run. However, if the leading entries from natural ordering are lost, the total number of iterations grows with the recovery point j, indicating that the recovery is not much better compared to a new run of LOBPCG from scratch. Hence, natural ordering is risky in practice, if the process with leading entries encounters a fault.

An interesting observation is that when j is around $20 \sim 30$ in this example, the total number of iterations for a fault-tolerant LOBPCG can even be lower than $\texttt{iter}_{\text{free}}$. Though detailed analysis of this behavior is beyond the scope of this paper.

Test results for more nuclei are shown in Figure 3. All strategies behave consistently among these nuclei. The strategy of filling zeros is always problematic even if only trailing entries are lost. When the shift-and-invert strategy with appropriate ordering is used, the total number of iterations for a fault-tolerant LOBPCG algorithm is close to that of a fault-free one.

Finally, we demonstrate that the quality of our fault-tolerant strategies also depends on the portion of lost data. To this end, we run the largest test case, ^{7}Li with $N_{\max} = 6$, using p processes with p_0 faulty ones for $(p_0, p) \in \big\{(8, 15), (1, 15), (8, 120), (1, 120)\big\}$. Intuitively, the total number of iterations increases when the portion of lost data increases. This is confirmed by the test results shown in Figure 4. Even in the extremely difficult case with $(p_0, p) = (8, 15)$, where more than half of the data is lost, our recovery strategies are still able to show reasonably good results, in the sense that they are cheaper compared to a new run of LOBPCG from scratch. Though the total number of iterations eventually grows with the recovery point in this case, the shift-and-invert strategy with appropriate ordering is still helpful, and is always better than the strategy of filling zeros. This suggests that our shift-and-invert strategy is quite robust for recovery. When the portion of lost data becomes smaller, the shift-and-invert strategy with appropriate ordering works much better. For large-scale computations in practice, we expect these strategies to work even better since the portion of lost data becomes much smaller.

6 Conclusions

In nuclear CI calculations eigenvectors corresponding to a few lowest eigenvalues are usually highly localized. By taking into account this property, we develop a fault-tolerant LOBPCG algorithm on distributed-memory architectures. When an appropriate ordering is adopted, using carefully recovered approximate eigenvectors to restart the LOBPCG iteration can significantly reduce the number of iterations required to reach convergence compared to restarting the whole

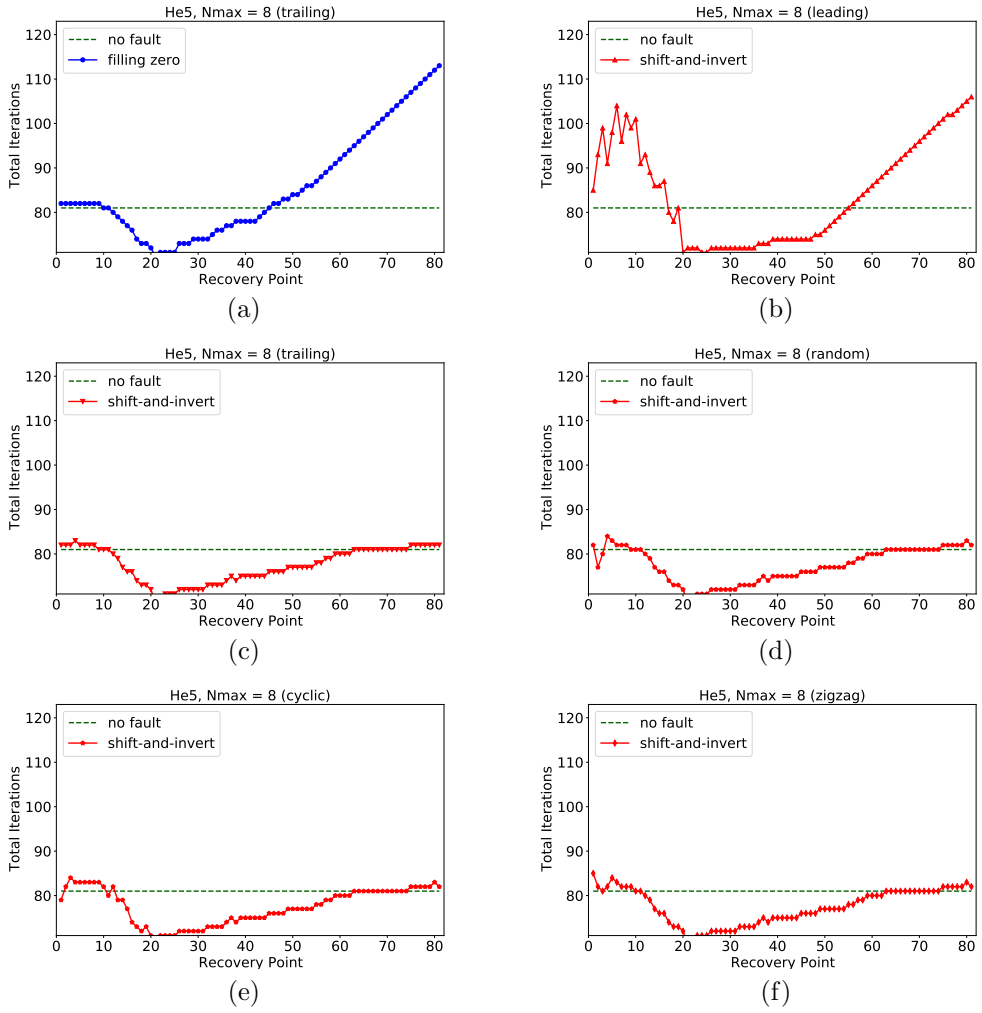

Figure 2: Number of iterations required by the fault-tolerant LOBPCG algorithm for ^{5}He with $N_{\max} = 8$ under different ordering.

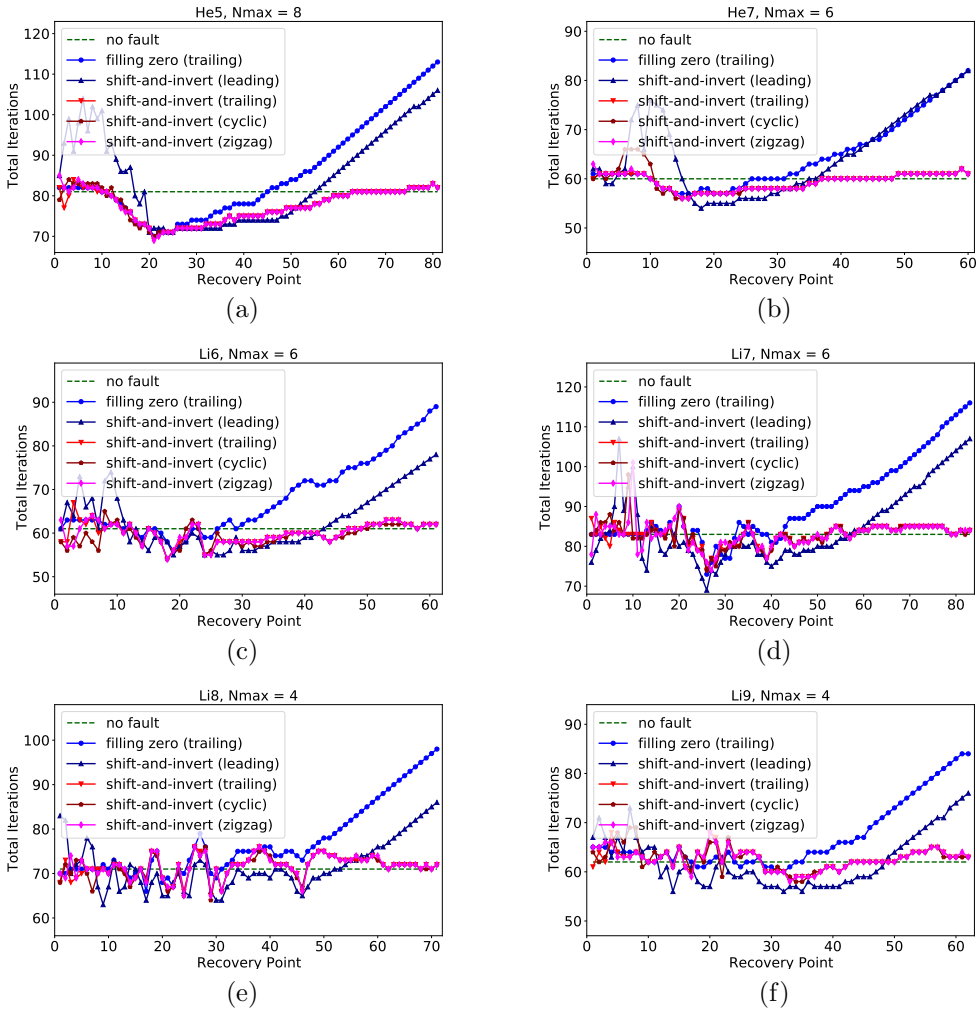

Figure 3: Number of iterations required by the fault-tolerant LOBPCG algorithm for nuclei listed in Table 1.

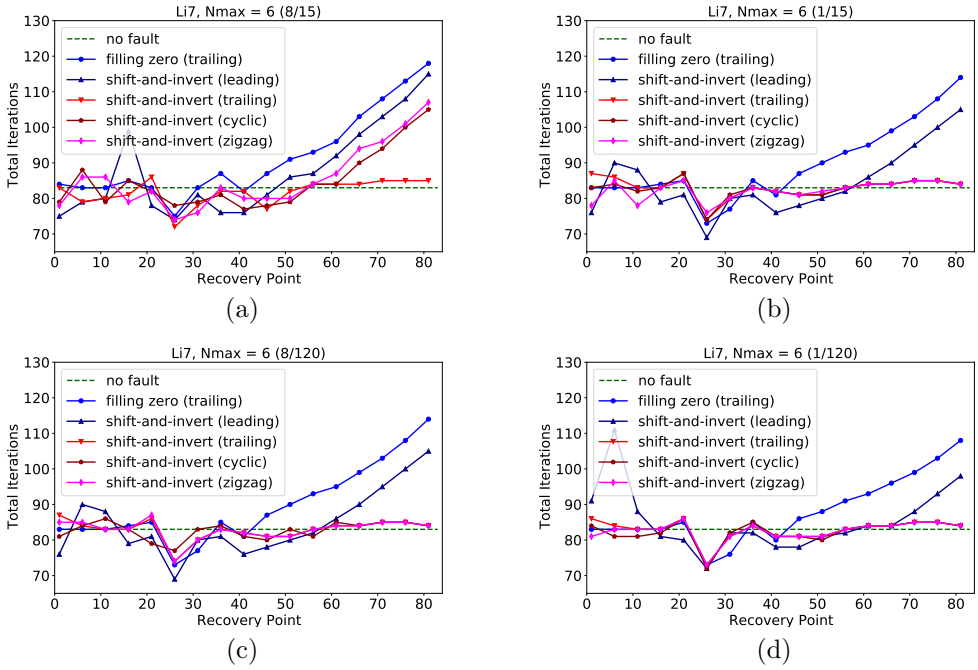

Figure 4: Number of iterations required by the fault-tolerant LOBPCG algorithm for ^{7}Li with $N_{\max} = 6$ running on p processes, with p_0 faulty processes for $(p_0, p) \in \{(8,15),(1,15),(8,120),(1,120)\}$.

calculation from the scratch. The recovery procedure is relatively cheap, and most computations do not involve inter-node communication. Unlike check-pointing, our algorithm does not perform additional work unless a fault has actually been encountered. Though our algorithm is proposed for nuclear CI calculations, the technique can be applied to a wider range of problems discretized using Schauder bases—as long as an appropriate ordering is used. The technique is also applicable to alternative eigensolvers that can take advantage of good initial guesses.

The numerical experiments in this work are proof-of-concept. We use several small-scale test cases with a relatively large lost rate to validate the effectiveness of our proposed algorithm. We expect that our strategies will remain effective for large-scale problems of practical interests, because the portion of lost data decreases as the number of processes grows. Incorporating the fault-tolerant LOBPCG algorithm into a production code of nuclear CI calculations is planned as our future work.

Acknowledgments

This work was supported in part by the U. S. Department of Energy (DOE) under grants No. DE-SC0018223 and DE-SC0023495 (SciDAC/NUCLEI) and the National Energy Research Scientific Computing Center (NERSC) Exascale Science Applications Program (NESAP). Computational resources were provided by Oak Ridge Leadership Computing Facility (OLCF). M. Shao's work was supported in part by the National Natural Science Foundation of China under grant No. 11971118.

References

[1] E. Agullo, L. Giraud, P. Salas, and M. Zounon. Interpolation-restart strategies for resilient eigensolvers. *SIAM J. Sci. Comput.*, 38(5):C560–C583, 2016.

[2] H. M. Aktulga, A. Buluç, S. Williams, and C. Yang. Optimizing sparse matrix-multiple vectors multiplication for nuclear configuration interaction calculations. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium*, pages 1213–1222. IEEE, 2014.

[3] P. Balaji, D. Buntinas, and D. Kimpe. Fault tolerance techniques for scalable computing. In S. U. Khan, A. Y. Zomaya, and L. Wang, editors, *Scalable Computing and Communications: Theory and Practice*, pages 737–758. Wiley-IEEE Computer Society Pr., Jan. 2013.

[4] I. P. Egwutuoha, D. Levy, B. Selic, and S. Chen. A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems. *J. Supercomput.*, 65(3):1302–1326, 2013.

[5] C. W. Johnson, W. E. Ormand, and P. G. Krastev. Factorization in large-scale many-body calculations. *Comput. Phys. Commun.*, 184:2761–2774, 2013.

[6] A. V. Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput.*, 23(2):517–541, 2001.

[7] A. V. Knyazev. Recent implementations, applications, and extensions of the locally optimal block preconditioned conjugate gradient method (LOBPCG), 2017. arXiv:1708.08354.

[8] P. Maris, H. M. Aktulga, M. A. Caprio, Ü. V. Çatalyürek, E. G. Ng, D. Oryspayev, H. Potter, E. Saule, M. Sosonkina, J. P. Vary, C. Yang, and Z. Zhou. Large-scale ab initio configuration interaction calculations for light nuclei. *J. Phys.: Conf. Ser.*, 403(1):012019, 2012.

[9] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12(4):617–629, 1975.

[10] M. Shao, H. M. Aktulga, C. Yang, E. G. Ng, P. Maris, and J. P. Vary. Accelerating nuclear configuration interaction calculations through a preconditioned block iterative eigensolver. *Comput. Phys. Commun.*, 222:1–13, 2018.

[11] P. Sternberg, E. G. Ng, C. Yang, P. Maris, J. P. Vary, M. Sosonkina, and H. V. Le. Accelerating configuration interaction calculations for nuclear structure. In *SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*. IEEE, 2008.

[12] J. P. Vary, R. Basili, W. Du, M. Lockner, P. Maris, D. Oryspayev, S. Pal, S. Sarker, H. M. Aktulga, E. Ng, M. Shao, and C. Yang. Ab initio no core shell model with leadership-class supercomputers. In A. M. Shirokov and A. I. Mazur, editors, *Proceedings of the International Conference 'Nuclear Theory in the Supercomputing Era—2016' (NTSE-2016), Khabarovsk, Russia, September 19–23, 2016.*, pages 15–35, 2018.