X. Hu et al. *Res Math Sci* (2019) 6:29
https://doi.org/10.1007/s40687-019-0187-z

Research in the Mathematical Sciences

RESEARCH

Randomized and fault-tolerant method of subspace corrections

Xiaozhe Hu[1], Jinchao Xu[2*] and Ludmil T. Zikatanov[2]

[*]Correspondence: xu@math.psu.edu
[2]Department of Mathematics, The Pennsylvania State University, University Park, PA 16802, USA
Full list of author information is available at the end of the article
The work of Hu was partially supported by NSF (DMS-1620063). The work of Xu and Zikatanov was supported in part by NSF (DMS-1522615)

Abstract

In this paper, we consider the iterative method of subspace corrections with random ordering. We prove identities for the expected convergence rate and use these results to provide sharp estimates for the expected error reduction per iteration. We also study the fault-tolerant features of the randomized successive subspace correction method by rejecting corrections when faults occur and show that the resulting iterative method converges with probability one. In addition, we derive estimates on the expected convergence rate for the fault-tolerant, randomized, subspace correction method.

Keywords: Method of subspace corrections, Randomized method, Fault-tolerant method

1 Introduction

In this paper, we consider iterative methods for solving the following model problem: Given $f \in V$, find $u \in V$ such that

$$Au = f, \tag{1}$$

where V is a Hilbert space and $A : V \mapsto V$ is a symmetric positive definite (SPD) linear operator. The class of iterative methods we are interested fall into the category of the so-called the methods of subspace corrections (MSC) which have been widely studied in the past several decades, see [6,27,28]. MSC is a general framework for linear iterative methods for the solution of linear problems in Hilbert spaces. Many well-known iterative methods can be viewed in the MSC framework and, therefore, can be studied using the general theory of MSC framework, for example, multigrid (MG) method [1,8,26] and domain decomposition (DD) method [17,25].

There are two basic types of MSC depending on how the error correction in the subspaces is done: the *Parallel subspace corrections (PSC) method* corrects the error from all the subspaces simultaneously, while the *Successive subspace corrections (SSC)* method corrects one after another. The standard SSC method traverses the subspace problems in a fixed order, but one of the interesting features of the SSC method is that the ordering need not be fixed from the start and it can be chosen dynamically during the iterations. A classical example in this direction is the greedy ordering algorithm for Gauss–Seidel method by Southwell [22]. Recently, the effects of the greedy ordering on the convergence of the multiplicative Schwarz method have been studied in [7]. Other related algorithms,

© Springer Nature Switzerland AG 2019.

such as randomized Kaczmarz iterative method, have been studied in detail in [4,12–14,16,24]. A randomized Schwarz method has been discussed in [7], and a randomized coordinate decent methods for a certain class of convex optimization problems was in the focus of several recent works [11,15,18].

One of our main results is the proof of an identity for the *expected* error reduction in energy norm per iteration step of the randomized SSC method. We note that in comparison with [7], even though the convergence rate estimate is of the same order, our analysis directly shows how one step of the randomized SSC method relates to the PSC method. Arguably, the identity we introduce leads to a better understanding of the nature of product randomized algorithms and provides more insights that can lead to better algorithms. In addition, we propose and analyze the convergence of a novel SSC method with J subspaces in which the ordering of the subspace corrections is chosen every J iterations by randomly selecting a permutation of $\mathbb{J} = \{1, \ldots, J\}$. We further provide a generalization of the XZ-identity [28] which applies to the error reduction rate in energy norm for such randomized SSC method. Next, we consider a special feature of this, namely, its convergence in case of hardware and/or software failures. On the one hand, when such error occur, there is no guarantee that the iterative method can produce a reasonable approximation of the solution. On the other hand, for many PDE-based applications, solving the linear system of equations dominates the overall simulation time (more than 80% of the simulation time for large-scale simulations). Therefore, the development and analysis of fault-tolerant linear solvers with low overhead is an important and urgent issue for improving the overall reliability of the huge pool of PDE-based applications. The standard approaches for constructing fault-tolerant iterative methods usually belong to the so-called ABFT (Algorithm-Based Fault Tolerance) category and basic linear and fault-tolerant versions of nonlinear iterative methods, such as successive over-relaxation (SOR) method, conjugate gradient (CG) method, and general minimal residual (GMRes) method have been studied in [9,19–21]. Another approach, proposed in [23], relies on rejecting large hardware error propagation and improves the resilience of iterative methods with respect to silent errors. In [10], resilience for massively parallel multigrid methods has been discussed based on combining domain partitioning with geometric multigrid methods. More recently, in [2], an intrinsic fault-/error-tolerant feature of the MSC has been explored. The idea is based on introducing redundant subspaces and working with carefully designed mappings between subspaces and processors.

Our results also show how the randomization can be used to improve the reliability of the SSC method in this paper. We built our fault-tolerant, randomized SSC method on a procedure which rejects the faulty subspace corrections when errors occur. Basically, we only update the solution when there is no error and, naturally, we are able to show that this simple procedure, together with randomization, converges almost surely (with probability 1). Our results demonstrate the potential of the SSC method as a natural fault-tolerant iterative method and provide theoretical justification of the usage of the SSC method in improving the reliability of long-running large-scale PDE applications.

The reminder of the paper is organized as follows. We recall the PSC method, the SSC method, the XZ-identity, and some basic notions from probability theory in Sect. 2.1. In Sect. 3, we describe the randomized SSC method and its fault-tolerant variant. Section 4 presents our main results, i.e., the sharp identity estimates of the convergence rate and almost sure convergence of the proposed SSC methods. At the end, we give some remarks in Sect. 5 to conclude the paper.

2 Preliminaries

In this section, we introduce the notation and review some basic results and definitions from the theory of the subspace correction methods and basic notions from probability theory.

2.1 Method of subspace corrections

We recall the standard definitions and the notation commonly used in the analysis of subspace correction methods. We begin by introducing a decomposition of the vector space V which consists of subspaces $V_i \subset V$, $i = 1, 2, \ldots, J$, such that

$$V = \sum_{i=1}^{J} V_i. \tag{2}$$

This means that, for each $v \in V$, there exist $v_i \in V_i$, $i = 1, 2, \ldots, J$, such that $v = \sum_{i=1}^{J} v_i$. This representation of v may not be unique in general, namely (2) is not necessarily a direct sum.

For each i, we define $Q_i, P_i : V \mapsto V_i$ and $A_i : V_i \mapsto V_i$ by

$$(Q_i u, v_i) = (u, v_i), \quad (P_i u, v_i)_A = (u, v_i)_A, \quad \forall\, u \in V,\ v_i \in V_i, \tag{3}$$

and

$$(A_i u_i, v_i) = (A u_i, v_i), \quad \forall\, u_i, v_i \in V_i. \tag{4}$$

Q_i and P_i are both orthogonal projections and A_i is the restriction of A on V_i and is SPD. Note that, from the definitions given above we have

$$A_i P_i = Q_i A. \tag{5}$$

Indeed, $\forall\, u, v \in V$, we have $(Q_i A u, v) = (Au, Q_i v) = (u, Q_i v)_A = (P_i u, Q_i v)_A = (A_i P_i u, Q_i v) = (A_i P_i u, v)$, and therefore $A_i P_i = Q_i A$.

Since $V_i \subset V$, we have the natural inclusion operator $I_i : V_i \mapsto V$ defined by

$$(I_i u_i, v) = (u_i, v), \quad \forall\, u_i \in V_i. \quad v \in V. \tag{6}$$

We notice that $Q_i = I_i^T$ as $(Q_i u, v_i) = (u, v_i) = (u, I_i v_i) = (I_i^T u, v_i)$. Similarly, we have $P_i = I_i^*$, where I_i^* is the transpose of I_i with respect to the inner product $(\cdot,\cdot)_A$ induced by A.

If u is the solution of (1), then

$$A_i u_i = f_i, \tag{7}$$

where $u_i = P_i u$ and $f_i = Q_i f$. (7) can be viewed as the restriction of (1) on the subspace V_i, $i = 1, 2, \ldots, J$. MSC solves these subspace equations (7) iteratively. In general, these subspace equations are solved approximately. More precisely, we introduce a non-singular operator $R_i : V_i \mapsto V_i$, $i = 1, 2, \ldots, J$, which is assumed to be an approximation to A_i^{-1} in certain sense. And then the subspaces equation (7) are solved approximated by $u_i \approx \hat{u}_i = R_i f_i$.

As we pointed out in the introduction, there are two major types of MSC, depending on how the error is corrected in each subspace. These are (1) *the parallel subspace corrections*

(PSC) method, which is similar in nature to the classical Jacobi method and the subspace equations are solved in parallel as in Algorithm 1; and (2) *the successive subspace corrections (SSC) method*, similar to the classical Gauss-Seidel method and the error is corrected successively from each subspace as in Algoritm 2. From the definitions in Algorithm 1, it is easy to see that

$$u^{m+1} = u^m + B_a(f - Au^m),$$

Algorithm 1 Parallel subspace correction method

```
1: Compute the residual by $r^m = f - Au^m$,
2: Approximately solve the subspace equations $A_i e_i = Q_i r^m$ by $\hat{e}_i = R_i Q_i r^m$ in parallel,
3: Update the iteration by $u^{m+1} = u^m + \sum_{i=1}^{J} I_i \hat{e}_i$.
```

with

$$B_a = \sum_{i=1}^{J} I_i R_i Q_i = \sum_{i=1}^{J} I_i R_i I_i^t, \tag{8}$$

which is the operator corresponds to the PSC method.

Algorithm 2 Successive subspace correction method

```
1: Compute the residual by $r^m = f - Au^m$,
2: Set $v^0 = u^m$,
3: for $k = 1 \to J$ do
4:     $v^k = v^{k-1} + R_k Q_k (f - Av^{k-1})$,
5: end for
6: Update $u^{m+1} = v^J$.
```

Let us define $T_i = R_i Q_i A$, and we note that $T_i : V_i \mapsto V_i$ is symmetric with respect to $(\cdot, \cdot)_A$, nonnegative definite, and satisfies $T_i = R_i A_i P_i$. Moreover, $T_i = P_i$ if $R_i = A_i^{-1}$. Using this notation, we have

$$B_a A = \sum_{i=1}^{J} T_i, \tag{9}$$

and

$$I - B_m A = (I - T_J)(I - T_{J-1}) \dots (I - T_1). \tag{10}$$

Here, B_m is the operator approximating A^{-1} which corresponds to the SSC method. For the theoretical analysis, we define the symmetrized smoother $\bar{R}_i := R_i^t + R_i - R_i^t A_i R_i$ and subspace solver $\bar{T}_i = T_i + T_i^* - T_i^* T_i$ with T_i^* defined as $(T_i^* u, v)_A = (u, T_i v)_A, \forall u, v \in V_i$.

We now focus on the randomized and fault-tolerant versions of the SSC method given by B_m. For its convergence analysis, we will need the following well-known XZ-identity [28].

Theorem 1 *(XZ-identity) Assume that B_m is defined by the SSC method (Algorithm 2), then we have*

$$\|I - B_m A\|_A^2 = 1 - \frac{1}{c}, \tag{11}$$

where

$$c = \sup_{\|v\|_A=1} \inf_{\sum v_i = v} \sum_{i=1}^{J} \left\| \overline{T_i}^{-1/2} \left(v_i + T_i^* P_i \sum_{j>i} v_j \right) \right\|_A^2 . \tag{12}$$

Corollary 1 *In case that the subspace problems are solved exactly, i.e., $R_i = A_i^{-1}$, the X-Z identity* (11) *holds with*

$$c = \sup_{\|v\|_A=1} \inf_{\sum v_i = v} \sum_{i=1}^{J} \left\| P_i \left(\sum_{j\geq i} v_j \right) \right\|_A^2 . \tag{13}$$

3 Randomized and fault-tolerant SSC

Traditionally, the SSC method visits each subspace in a pre-determined ordering, i.e., it solves subspace problems one by one in a fixed, problem-independent order. Here we consider to choose the ordering randomly, which is a key component of the algorithms. Another component we introduce into the SSC method is the fault-tolerant ability enabled by randomization. In this section, we formulate those algorithms and their convergence analysis are discussed in the next section.

3.1 Randomized SSC

In the randomized SSC method, we randomly choose the next subspace in which the error needs to be corrected. We randomly choose the subspace, according to certain probability distribution as in Algorithm 3.

Algorithm 3 SSC method with random ordering (Version 1)

```
1: Randomly choose an index $i \in \{1, 2, \dots, J\}$ with probability $p_i = \frac{1}{J}$,
2: $u^{k+1} = u^k + R_i Q_i (f - Au^k)$
```

As discussed in [7], the cost of randomly picking i does not exceed $\mathcal{O}(\log J)$ and each update in the SSC method can be done in $\mathcal{O}(N)$ operations where N is the dimension of the vector space V. Therefore, the overall computational cost of the randomized SSC method is comparable to the standard SSC method which is a very desirable feature.

Note that in Algorithm 3, there is no guarantee that all the J subspaces are all corrected in J iterations. Therefore, we propose the second version randomized SSC method, such that the J subspaces are guaranteed to be corrected within J iterations by randomly choosing the ordering in which the error is corrected. To do this, we first consider the set of all permutations of $\mathbb{J} = \{1, 2, \dots, J\}$. Then a permutation of $\mathbb{J}$ is any bijective mapping $\sigma : \mathbb{J} \mapsto \mathbb{J}$. The idea is to randomly choose a permutation σ from the set of permutations and apply the SSC following the correction order as specified by σ. We have the randomized SSC method presented in Algorithm 4.

We note that, in Algorithm 4, the cost of randomly picking the permutation σ is $\mathcal{O}(J \log J)$ (see, e.g., [3,5] for such algorithms) which is more expensive than Algorithm 3. However, the cost of the **for** loop (steps 4–6 in Algorithm 4) is comparable with traditional PSC method (Algorithm 1). Moreover, it is reasonable to assume that $J = \mathcal{O}(N)$

Algorithm 4 SSC method with random ordering (Version 2)

```
1: Compute the residual by $r^m = f - Au^m$,
2: $v^0 = u^m$,
3: Randomly choose a permutation $\sigma$ of the indexes $\mathbb{J} = \{1, 2, \ldots, J\}$ with probability $\frac{1}{J!}$,
4: for $k = 0 \to J - 1$ do
5:     $v^{k+1} = v^k + R_{\sigma(k)} Q_{\sigma(k)} (f - Av^k)$,
6: end for
7: Update the iteration by $u^{m+1} = v^J$.
```

at the worst case (as in the multigrid method), then the overall computational cost of Algorithm 4 is $\mathcal{O}(N \log N)$ which is nearly optimal.

3.2 Fault-tolerant randomized SSC

Another feature pertaining to this randomized SSC method is its fault tolerance. During the iterative process, the correction or update may fail due to hard and/or soft errors. The hard error usually is due to a permanent node/memory crash which stops the process. A soft error is a type of error where a signal or datum is wrong. In this case, the process continues and the failures affects the following execution. Both cases, if errors are not accounted for correctly, can result in stagnating iterations, namely, there will be no error reduction during iterations.

We propose a simple approach which can handle all such scenarios (see Algorithm 5). Basically, we do not update the approximation to the solution during the iterations when error occurs. The randomization of the ordering helps to guarantee that such simple treatment leads to theoretically convergent iterative method.

Algorithm 5 Fault-tolerant SSC method with random ordering

```
1: if error occurs then
2:     $u^{k+1} = u^k$,
3: else
4:     Randomly choose an index $i \in \{1, 2, \ldots, J\}$ with probability $p_i = \frac{1}{J}$,
5:     $u^{k+1} = u^k + R_i Q_i (f - Au^k)$.
6: end if
```

Similar to Algorithm 3, the cost of randomly picking i is $\mathcal{O}(\log J)$ and each correction costs $\mathcal{O}(N)$. Therefore, the overall cost of Algorithm 5 is comparable to the cost of the traditional SSC method and Algorithm 3.

4 Convergence analysis

In this section, we present the convergence analysis of the randomized and fault-tolerant SSC methods (Algorithms 3–5). We want to emphasize that, instead of usual upper bound estimation, we give identities for the convergence rate of the randomized and fault-tolerant SSC methods. We note that in the analysis below we relate the expected convergence rate of the SSC method, to the quality of the PSC preconditioner, and the latter is obviously independent of the ordering of the subspaces. This result confirms that the expected (or the average) convergence rate of an SSC method is also independent of the ordering.

4.1 Convergence rate of the randomized SSC

First, we consider Algorithm 3 and the main result is stated in the following theorem. Here, we use B_a to denote the operator corresponding to the PSC method with $\bar{R}_i$ as the inexact subspace solver.

Theorem 2 *The Algorithm 3 converges with the expected error decay rate,*

$$\mathbb{E}(\|u - u^{k+1}\|_A^2) = \left(1 - \frac{\delta_k}{J}\right)\mathbb{E}(\|u - u^k\|_A^2) = \prod_{\ell=0}^{k}\left(1 - \frac{\delta_\ell}{J}\right)\|u - u^0\|_A^2, \tag{14}$$

where $\delta_k = \frac{\mathbb{E}((B_a A e^k, e^k)_A)}{\mathbb{E}((e^k, e^k)_A)} > 0$ *and* $e^k = u - u^k$. *Moreover, if* $\|I - T_i\|_A < 1$, *then* $\delta_k < J$.

Proof It is easy to see that, given i, we have

$$\|e^{k+1}\|_A^2 = \|(I - T_i)e^k\|_A^2 = \left(\left(I - \bar{T}_i\right)e^k, e^k\right)_A,$$

where $\bar{T}_i$ is the symmetrized version of T_i. According to the way we pick the index i, at iteration k, we define random variables $S_k(x)$ as follows:

$$S_k(\omega_k) = \omega_k,\ \omega_k = (i_1, i_2, \ldots, i_k),\ i_m \in \Omega, \quad m = 1, \ldots, k.$$

Because we choose an index $i \in \{1, 2, \ldots, J\}$ uniformly with probability $1/J$ and picking i is independent of the iteration number k, we have,

$$P(S_{k+1} = \omega_{k+1} \,|\, S_k = \omega_k) = \begin{cases} \frac{1}{J}, & \text{if } \omega_{k+1} = (\omega_k, i),\ i = 1, 2, \ldots, J \\ 0, & \text{otherwise} \end{cases}$$

Then we define, for $\omega_k = (i_1, i_2, \ldots, i_k)$,

$$X_k(\omega_k) = \|(I - T_{i_k})(I - T_{i_{k-1}}) \ldots (I - T_{i_1})e^0\|_A^2 := \|e^k\|_A^2,$$

and the conditional expectation with respect to S_{k+1} conditioned on $S_k = \omega_k$ can be computed as follows:

$$\begin{aligned}
\mathbb{E}(X_{k+1} \,|\, S_k)(\omega_k) &= \mathbb{E}(X_{k+1} \,|\, S_k = \omega_k) \\
&= \sum_{\omega_{k+1} \in \boldsymbol{\Omega}_{k+1}} X_{k+1}(\omega_{k+1}) P(S_{k+1} = \omega_{k+1} \,|\, S_k = \omega_k) \\
&= \sum_{i=1}^{J} \|(I - T_i)e^k\|_A^2 \frac{1}{J} = \sum_{i=1}^{J} \frac{1}{J}((I - \bar{T}_i)e^k, e^k)_A \\
&= \|e^k\|_A^2 - \frac{1}{J}(\sum_i^J \bar{T}_i e^k, e^k)_A = \|e^k\|_A^2 - \frac{1}{J}(B_a A e^k, e^k)_A,
\end{aligned}$$

where the last equation follows from (9) with T_i replaced by $\bar{T}_i$ since here B_a denotes the PSC method with symmetrized smoother as mentioned at the beginning of this section. Apply $\mathbb{E}(X) = \mathbb{E}(\mathbb{E}(X|Y))$, use the linearity of the expectation, and let $X = X_{k+1}$, $Y = S_k$, we then have the desired expected value with respect to S_{k+1} as follows,

$$\begin{aligned}\mathbb{E}(\|e^{k+1}\|_A^2) &= \mathbb{E}(X_{k+1}) = \mathbb{E}(\mathbb{E}(X_{k+1}|S_k))\\ &= \mathbb{E}(\|e^k\|_A^2 - \frac{1}{J}(B_aAe^k, e^k)_A)\\ &= \mathbb{E}(\|e^k\|_A^2) - \frac{1}{J}\mathbb{E}((B_aAe^k, e^k)_A)\\ &= \left(1 - \frac{1}{J}\frac{\mathbb{E}((B_aAe^k, e^k)_A)}{\mathbb{E}((e^k, e^k)_A)}\right)\mathbb{E}(\|e^k\|_A^2).\end{aligned}$$

Note that, if $\|I - T_i\|_A < 1$, by the definition of $\bar{T}_i$, we have $0 \le (\bar{T}_ie^k, e^k)_A < (e^k, e^k)_A$. Therefore, we have $(B_aAe^k, e^k)_A = (\sum_{i=1}^J \bar{T}_ie^k, e^k)_A < J(e^k, e^k)_A$ which implies that $\mathbb{E}((B_aAe^k, e^k)_A) < J\mathbb{E}((e^k, e^k)_A)$, i.e., $\delta_k < J$. This completes the proof. □

Remark 1 Now we discuss about the constant δ_k, we have

$$\lambda_{\min}(B_aA)\|e^k\|_A^2 \le (B_aAe^k, e^k)_A \le \lambda_{\max}(B_aA)\|e^k\|_A^2.$$

Use the linearity and monotonicity of expectation, we have

$$\lambda_{\min}(B_aA)\mathbb{E}(\|e^k\|_A^2) \le \mathbb{E}((B_aAe^k, e^k)_A) \le \lambda_{\max}(B_aA)\mathbb{E}(\|e^k\|_A^2).$$

Therefore, we have

$$\lambda_{\min}(B_aA) \le \delta_k \le \lambda_{\max}(B_aA),$$

and

$$1 - \frac{\delta_k}{J} \le 1 - \frac{\lambda_{\min}(B_aA)}{J}.$$

Remark 2 After J steps, we have $\mathbb{E}(\|e^J\|_A^2) \le \left(1 - \frac{\lambda_{\min}(B_aA)}{J}\right)^J \|e^0\|_A^2$ and the energy error reduction is bounded by $\left(1 - \frac{\lambda_{\min}(B_aA)}{J}\right)^J \le \exp(-\lambda_{\min}(B_aA))$.

Remark 3 Here we choose the index $i \in \{1, 2, \ldots, J\}$ based on a uniform distribution with constant probability $p_i = \frac{1}{J}$. However, one can use other choices as long as the probability does not depend on the iteration number k. Similar results can be derived and same statements also apply to the theoretical results below.

$$\mathbb{E}(\|u - u^{kJ}\|_A^2) \le \delta^k\mathbb{E}(\|u - u^0\|_A^2), \quad \delta = e^{-\lambda_{\min}(B_aA)}$$

A special case of Algorithm 3 is that the subspace corrections are exact, i.e, $R_i = A_i^{-1}$. And the following corollary is a direct consequence of Theorem 2.

Corollary 2 *Assume that the probabilities p_i, $i \in \mathbb{J}$ are independent of the iteration number k. Then Algorithm 3 with $R_i = A_i^{-1}$ converges with the following expected error reduction:*

$$\mathbb{E}(\|u - u^{k+1}\|_A^2) = \left(1 - \frac{\delta_k}{J}\right)\mathbb{E}(\|u - u^k\|_A^2) = \prod_{\ell=0}^{k}\left(1 - \frac{\delta_\ell}{J}\right)\|u - u^0\|_A^2, \tag{15}$$

where $0 < \delta_k = \frac{E\left(\sum_{i=1}^J ||P_ie^k||_A^2\right)}{\mathbb{E}(||e^k||_A^2)} < J$.

Next theorem shows that the randomized SSC method converges almost surely, i.e., converges with probability 1, if all the subspace corrections are convergent.

Theorem 3 *If $\|I - T_i\|_A < 1$ for $i = 1, 2, \ldots, J$, then Algorithm 3 converges almost surely or with probability 1, i.e., $\|e^k\|_A^2 \xrightarrow{a.s} 0$.*

Proof In order to show the almost sure convergence, we need to show that $\sum_{k=1}^{\infty} P(\|e^k\|_A^2 \geq \varepsilon) < +\infty$ for any $\varepsilon > 0$. Note that, by Markov's inequality, we have

$$P(\|e^k\|_A^2 \geq \varepsilon) \leq \frac{1}{\varepsilon}\mathbb{E}(\|e^k\|_A^2) = \frac{1}{\varepsilon}\prod_{\ell=0}^{k-1}\left(1 - \frac{\delta_\ell}{J}\right)\|e^0\|_A^2,$$

Since $\|I - T_i\|_A < 1$ and according to Remark 1, we have $\lambda_{\min}(B_a A) < \delta_\ell < \lambda_{\max}(B_a A) < J$ and then,

$$\begin{aligned}\sum_{k=1}^{\infty}\frac{1}{\varepsilon}\prod_{\ell=0}^{k-1}\left(1 - \frac{\delta_\ell}{J}\right)\|e^0\|_A^2 &\leq \frac{1}{\varepsilon}\|e^0\|_A^2\sum_{k=1}^{\infty}\left(1 - \frac{\lambda_{\min}(B_a A)}{J}\right)^{k-1} \\ &= \frac{1}{\varepsilon}\|e^0\|_A^2\frac{J}{\lambda_{\min}(B_a A)} < +\infty.\end{aligned}$$

Therefore, we have that the series $\sum_{k=1}^{\infty} P(\|e^k\|_A^2 \geq \varepsilon)$ converges. □

Next, we discuss the convergence rate for Algorithm 4. We introduce B_σ to denote the corresponding operator for J iterations using permutation σ, i.e.,

$$I - B_\sigma A = (I - T_{\sigma(J)})(I - T_{\sigma(J-1)})\ldots(I - T_{\sigma(1)}).$$

We have the following theorem.

Theorem 4 *Consider B_σ defined by Algorithm 4, we have*

$$\mathbb{E}(\|I - B_\sigma A\|_A^2) = 1 - \frac{1}{J!}\sum_{i=1}^{J!}\frac{1}{c_{\sigma_i}},$$

where c_{σ_i} is the constant c in the XZ-identity (12) of SSC using permutation σ_i.

Proof According to XZ-identity (11), for a permutation σ_i, we have

$$||I - B_{\sigma_i} A||_A^2 = 1 - \frac{1}{c_{\sigma_i}}.$$

Here, we need to introduce a different probability space $(\Omega, \mathcal{F}, P)$ where $\Omega = \{\sigma_1, \sigma_2, \ldots, \sigma_{J!}\}$, i.e., the set of all possible permutations, $\mathcal{F}$ is again the σ-algebra of Ω, and the probability P is defined by $P(a) = |a|/J!$, $a \in \mathcal{F}$. Based on this probability space, we define a random variable $X : \Omega \mapsto \mathbb{R}$ as following,

$$X(\sigma_i) = \|I - B_{\sigma_i} A\|_A^2 = 1 - \frac{1}{c_{\sigma_i}} =: x_{\sigma_i}, \quad i = 1, 2, \ldots, J!$$

X is a discrete random variable an its expectation can be computed as following

$$\begin{aligned}\mathbb{E}(\|I - B_\sigma A\|_A^2) = \mathbb{E}(X) &= \sum_{i=1}^{J!} x_{\sigma_i} P(X = x_{\sigma_i}) \\ &= \sum_{i=1}^{J!} \left(1 - \frac{1}{c_{\sigma_i}}\right) \frac{1}{J!} \\ &= 1 - \frac{1}{J!} \sum_{i=1}^{J!} \frac{1}{c_{\sigma_i}},\end{aligned}$$

which completes the proof. □

Remark 4 For a given space decomposition, different permutations of the subspaces lead to different convergence of the corresponding SSC method. According to the XZ-identity, the worse case is when $C_{\max} = \max_{1 \le i \le J!} c_{\sigma_i}$ and the corresponding permutation is denoted by σ^*. Note that

$$1 - \frac{1}{J!} \sum_{i=1}^{J!} \frac{1}{c_{\sigma_i}} \le 1 - \frac{1}{c_{\max}}.$$

The equality holds if and only if $c_{\sigma_i} = c_{\max}$ for all σ_i. The left hand side of the inequality is the convergence rate of Algorithm 4 in expectation and the right hand side of the inequality is the worse case of SSC method. Therefore, the above inequality implies that the randomized SSC method Algorithm 4 "improves" the convergence rate of the deterministic SSC method in the worst case scenario.

Here we present a simple numerical test to demonstrate this. We consider solving the model problem (1) arising from the linear finite element discretization of the Laplace problem $-\Delta u = f$ on unit square in 2D with $f = 0$ and homogeneous Dirichlet boundary conditions. Uniform meshes are used here. Therefore, we consider SSC in the geometric multigrid setting, i.e., the space decomposition (2) is a multilevel nodal decomposition (see [27,28]). For Algorithm 2, we use lexicographical ordering and compare the results with Algorithm 4. We set the initial guess to be the vector $(1, \ldots, 1)^T$ and, since the exact solution is $u = 0$, we can compute the error $\|e^m\|_A$ directly. The iterations are terminated when $\frac{\|e^m\|_A}{\|e^0\|_A} \le 10^{-6}$. The results are shown in Table 1 for different levels (column "Level" in the table). Here, mesh size $h = 2^{-\text{Level}}$, that is, a higher level means a finer mesh. For Algorithm 2, we use lexicographical ordering and report the number of iterations (column "Iter.") and the convergence rate (Conv. Rate), which is computed as $\left(\frac{\|e^m\|_A}{\|e^0\|_A}\right)^{1/m}$. As expected, since we use a multilevel nodal decomposition, the corresponding SSC converges uniformly, i.e., the number of iterations stays the same as we refine the mesh. For Algorithm 4, since we use random permutation, we repeat the experiments 10 times on each level and report the average number of iterations (Ave. #Iter.) and average convergence rate (Ave. Conv. Rate). As we can see, Algorithm 4 not only converges uniformly but also is consistently faster than Algorithm 2 on all levels. In addition, we also report how many times out of 10 experiments, Algorithm 4 actually converges in less number of iterations than Algorithm 2 (in percentage). Except the smallest size case (Level=4), Algorithm 4 is 100% faster than Algorithm 2 in terms of number of iterations. This justifies

Table 1 Comparison between Algorithms 2 and 4

	Algorithm 2		Algorithm 4		
Level	Iter	Conv. Rate	Ave. #Iter	Ave. Conv. Rate	Percentage (%)
4	12	0.316	11.8	0.301	30
5	13	0.331	12.0	0.305	100
6	13	0.332	11.8	0.292	100
7	13	0.334	11.8	0.298	100
8	13	0.334	11.7	0.291	100
9	13	0.332	11.2	0.279	100
10	13	0.330	11.3	0.283	100

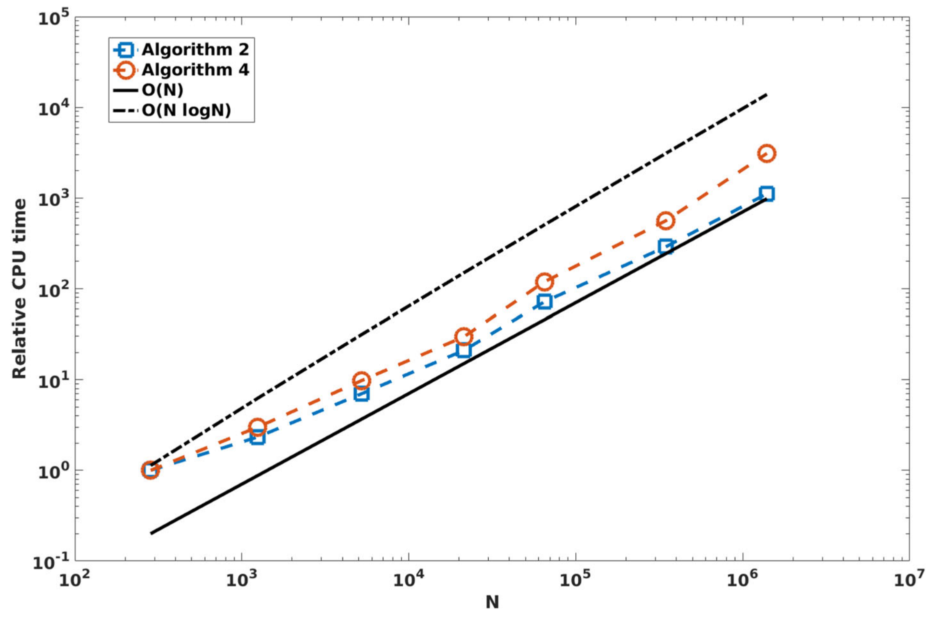

Fig. 1 Computational complexity of Algorithms 2 and 4

our statement that randomized SSC improves the convergence rate of the deterministic SSC method in the worst case scenario.

In Fig. 1, we report the computational complexity of Algorithms 2 and 4. Since the CPU time depends on the specific computational environment and implementation, we use relative CPU time to show the complexity of the algorithms here, namely, we set the CPU time for solving the smallest size problem (Level = 4) as the base 1 for each algorithm and the relative CPU time is the ratio between the CPU time for solving the problem on current level and the base CPU time. As we can see, Algorithm 2 scales like $\mathcal{O}(N)$ and Algorithm 4 scales like $\mathcal{O}(N \log N)$, confirming that Algorithm 4 is a little bit more costly. Therefore, the trade-off between the convergence rate and computational cost should be carefully considered in practice in order to achieve the best performance.

4.2 Fault-tolerant randomized SSC

In this section, we discuss the convergence rate of the fault-tolerant randomized SSC method (Algorithm 5). The main assumption is that the errors occur with probability

$\theta \in [0, 1)$. Next theorem says that the fault-tolerant randomized SSC method converges in expectation. Again, we use B_a to denote the operator corresponding to the PSC method with $\bar{R}_i = R_i^t + R_i - R_i^t A_i R_i$ as the inexact subspace solver.

Theorem 5 *Assume that error occurs with probability* $\theta \in [0, 1)$ *which is independent of* k *and how* i *is picked, then the Algorithm 5 converges with the expected convergence rate,*

$$\begin{aligned}\mathbb{E}(\|u-u^{k+1}\|_A^2) &= \left(1-\frac{(1-\theta)\delta_k}{J}\right)\mathbb{E}(\|u-u^k\|_A^2)\\ &= \prod_{\ell=0}^{k}\left(1-\frac{(1-\theta)\delta_\ell}{J}\right)\|u-u^0\|_A^2,\end{aligned} \tag{16}$$

where $\delta_k = \frac{\mathbb{E}((B_aAe^k,e^k)_A)}{\mathbb{E}((e^k,e^k)_A)} > 0$ *and* $e^k = u - u^k$. *Moreover, if* $\|I - T_i\|_A < 1$, *then* $\delta_k < J$.

Proof Note that, if there is no error (with probability $1-\theta$), for a given i, we have

$$\|e^{k+1}\|_A^2 = \|(I-T_i)e^k\|_A^2 = \left(\left(I-\bar{T}_i\right)e^k, e^k\right)_A,$$

otherwise, we have

$$\|e^{k+1}\|_A^2 = \|e^k\|_A^2.$$

As in the proof of Theorem 2, we define random variables $S_k(x)$ such that,

$$S_k(\omega_k) = \omega_k, \quad \omega_k = (i_1, i_2, \ldots, i_k),\ i_m \in \Omega, \quad m = 1, \ldots, k.$$

Because we pick i is independent of the iteration number k, we have,

$$P(S_{k+1} = \omega_{k+1} \,|\, S_k = \omega_k) = \begin{cases} \theta, & \text{if } \omega_{k+1} = (\omega_k, 0),\\ \frac{1-\theta}{J}, & \text{if } \omega_{k+1} = (\omega_k, i),\ i = 1, 2, \ldots, J\\ 0, & \text{otherwise}\end{cases}$$

Next we define for $\omega_k = (i_1, i_2, \ldots, i_k)$,

$$X_k(\omega_k) = \|(I-T_{i_k})(I-T_{i_{k-1}})\ldots(I-T_{i_1})e^0\|_A^2 := \|e^k\|_A^2,$$

Therefore, then we compute the conditional expectation as following

$$\begin{aligned}\mathbb{E}(X_{k+1}\,|\,S_k)(\omega_k) &= \mathbb{E}(X_{k+1}\,|\,S_k=\omega_k)\\ &= \sum_{\omega_{k+1}\in\boldsymbol{\Omega}_{k+1}} X_{k+1}(\omega_{k+1})P(S_{k+1}=\omega_{k+1}\,|\,S_k=\omega_k)\\ &= \theta\|e^k\|_A^2 + \sum_{i=1}^{J}\frac{(1-\theta)}{J}\|(I-T_i)e^k\|_A^2\\ &= \theta\|e^k\|_A^2 + \frac{1-\theta}{J}J(e^k,e^k)_A - \sum_{i=1}^{J}\frac{1-\theta}{J}(\bar{T}_ie^k,e^k)_A\\ &= \|e^k\|_A^2 - \frac{1-\theta}{J}(B_aAe^k,e^k)_A.\end{aligned}$$

Following the proof of Theorem 2, we apply the identity $\mathbb{E}(X) = \mathbb{E}(\mathbb{E}(X|Y))$ and use the linearity of the expectation to derive (16). This completes the proof. □

Remark 5 We can estimate the constant δ_k as in Remark 1, i.e.,

$$\lambda_{\min}(B_a A) \le \delta_k \le \lambda_{\max}(B_a A).$$

Therefore, we have

$$1 - \frac{(1-\theta)\delta_k}{J} \le 1 - \frac{(1-\theta)\lambda_{\min}(B_a A)}{J}.$$

Remark 6 After J steps, we have

$$\mathbb{E}(\|e^J\|_A^2) \le \left(1 - \frac{(1-\theta)\lambda_{\min}(B_a A)}{J}\right)^J \|e^0\|_A^2,$$

and the energy error reduction is bounded by

$$\left(1 - \frac{(1-\theta)\lambda_{\min}(B_a A)}{J}\right)^J \approx \exp\left((\theta - 1)\lambda_{\min}(B_a A)\right), \quad 0 \le \theta < 1.$$

The following corollary consider a special case of Theorem 5 that all the subspace corrections are exact, i.e, $R_i = A_i^{-1}$.

Corollary 3 *Assume that errors occurs with probability $\theta \in [0, 1)$ which is independent of k, then Algorithm 5 with $R_i = A_i^{-1}$ converges with the expected error decay rate,*

$$\begin{aligned}\mathbb{E}(\|u - u^{k+1}\|_A^2) &= \left(1 - \frac{(1-\theta)\delta_k}{J}\right)\mathbb{E}(\|u - u^k\|_A^2) \\ &= \prod_{\ell=0}^{\ell=k}\left(1 - \frac{(1-\theta)\delta_\ell}{J}\right)\|u - u^0\|_A^2, \end{aligned} \tag{17}$$

where $0 < \delta_k = E(\sum_{i=1}^J \left\|P_i e^k\right\|_A^2)/\mathbb{E}(\left\|e^k\right\|_A^2) < J$.

Next theorem shows that the fault-tolerant randomized SSC method converges almost surely or with probability 1 if all the subspace corrections are convergent.

Theorem 6 *Assume that $\|I - T_i\|_A < 1$ for $i = 1, 2, \ldots, J$. Moreover, assume that errors occur with probability $\theta \in [0, 1)$ and independent of k, Algorithm 5 converges almost surely or with probability 1, i.e., $\|e^k\|_A^2 \xrightarrow{a.s} 0$.*

Proof The proof is the same as the proof of Theorem 3, thus, omitted. □

Theorems 5 and 6 suggest that, when error occurs, simply do not update the solution or reject the update. The randomized SSC method are guaranteed to converge to the correct solution. Such property is useful for developing error resilience algorithms.

5 Conclusion

We study the convergence behavior of the randomized subspace correction methods. Instead of the usual upper bound for the convergence rate, we derived an identity for the estimation of the expect error decay rate in energy norm and also show the randomized algorithm converges almost surely if all the subspace correction converges.

We also propose another version randomized subspace correction method in which each subspace is corrected once within J iterations. We theoretically prove that it is convergent by using the XZ-identity and show how it improves the standard SSC method at the worst case in terms of the convergence rate.

In order to improve the error resilience of the subspace correction methods, we develop a fault-tolerant variant of the randomized method by rejecting any correction when error occurs. We show that the fault-tolerant iterative method based on such approach converges with probability 1 if all the subspace corrections are convergent and, moreover, we also derive a sharp identity estimate for the convergence rate. These results show the intrinsic fault-tolerant features of the subspace correction method and its potential in extreme-scale computing by introducing randomization.

Author details
[1]Department of Mathematics, Tufts University, Medford, MA 02155, USA, [2]Department of Mathematics, The Pennsylvania State University, University Park, PA 16802, USA.

Conflict of interest
On behalf of all authors, the corresponding author states that there is no conflict of interest.

Received: 1 May 2018 Accepted: 27 May 2019 Published online: 12 August 2019

References

1. Bramble, J.: Multigrid Methods. Chapman & Hall/CRC, Boca Raton (1993)
2. Cui, T., Xu, J., Zhang, C.S.: An error-resilient redundant subspace correction method. Comput. Vis. Sci. **18**(2–3), 65–77 (2017)
3. Durstenfeld, R.: Algorithm 235: random permutation. Commun. ACM **7**(7), 420 (1964). https://doi.org/10.1145/364520.364540
4. Eldar, Y.C., Needell, D.: Acceleration of randomized Kaczmarz method via the Johnson–Lindenstrauss Lemma. Numer. Algorithms **58**(2), 163–177 (2011)
5. Fisher, R.A., Yates, F.: Statistical Tables for Biological Agricultural and Medical Research. Oliver and Boyd, London (1948)
6. Griebel, M., Oswald, P.: On the abstract theory of additive and multiplicative Schwarz algorithms. Numer. Math. **70**(2), 163–180 (1995). https://doi.org/10.1007/s002110050115
7. Griebel, M., Oswald, P.: Greedy and randomized versions of the multiplicative Schwarz method. Linear Algebra Appl. **437**(7), 1596–1610 (2012)
8. Hackbusch, W.: Multigrid Methods and Applications, Springer Series in Computational Mathematics, vol. 4. Springer-Verlag, Berlin (1985)
9. Hoemmen, M., Heroux, M.A.: Fault-tolerant iterative methods via selective reliability. In: Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC). IEEE Computer Society, vol. 3, p. 9 (2011)
10. Huber, M., Gmeiner, B., Rüde, U., Wohlmuth, B.: Resilience for massively parallel multigrid solvers. SIAM J. Sci. Comput. **38**(5), S217–S239 (2016)
11. Leventhal, D., Lewis, A.S.: Randomized methods for linear constraints: convergence rates and conditioning. Math. Oper. Res. **35**(3), 641–654 (2010)
12. Liu, J., Wright, S.J.: An accelerated randomized kaczmarz algorithm. arXiv preprint arXiv:1310.2887 (2013)
13. Mansour, H., Yilmaz, O.: A fast randomized kaczmarz algorithm for sparse solutions of consistent linear systems. arXiv preprint arXiv:1305.3803 (2013)
14. Needell, D.: Randomized Kaczmarz solver for noisy linear systems. BIT Numer. Math. **50**(2), 395–403 (2010)
15. Nesterov, Y.: Efficiency of coordinate descent methods on huge-scale optimization problems. SIAM J. Optim. **22**(2), 341–362 (2012)
16. Oswald, P., Zhou, W.: Convergence analysis for Kaczmarz-type methods in a Hilbert space framework. Linear Algebra Appl. **478**, 131–161 (2015)
17. Quarteroni, A., Valli, A.: Domain Decomposition Methods for Partial Differential Equations. Oxford University Press, Oxford (1999)
18. Richtárik, P., Takáč, M.: Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. Math. Program. **144**(1–2), 1–38 (2014)
19. Roy-Chowdhury, A., Banerjee, P.: A fault-tolerant parallel algorithm for iterative solution of the laplace equation. In: International Conference on Parallel Processing, 1993. ICPP 1993, vol. 3, pp. 133–140. IEEE (1993)
20. Roy-Chowdhury, A., Bellas, N., Banerjee, P.: Algorithm-based error-detection schemes for iterative solution of partial differential equations. IEEE Trans. Comput. **45**(4), 394–407 (1996)
21. Shantharam, M., Srinivasmurthy, S., Raghavan, P.: Fault tolerant preconditioned conjugate gradient for sparse linear system solution. In: Proceedings of the 26th ACM International Conference on Supercomputing, pp. 69–78. ACM (2012)
22. Southwell, R.V.: Relaxation Methods in Engineering Science—A Treatise in Approximate Computation. Oxford University Press, Oxford (1946)
23. Stoyanov, M.K., Webster, C.G.: Numerical analysis of fixed point algorithms in the presence of hardware faults. Tech. rep., Tech. rep. Oak Ridge National Laboratory (ORNL) (2013)
24. Strohmer, T., Vershynin, R.: A randomized Kaczmarz algorithm with exponential convergence. J. Fourier Anal. Appl. **15**(2), 262–278 (2009)
25. Toselli, A., Widlund, O.: Domain Decomposition Methods-Algorithms and Theory. Springer Verlag, Berlin (2005)
26. Trottenberg, U., Oosterlee, C., Schüller, A.: Multigrid. Academic Press, Cambridge (2001)

27. Xu, J.: Iterative methods by space decomposition and subspace correction. SIAM Rev. **34**(4), 581–613 (1992). https://doi.org/10.1137/1034116
28. Xu, J., Zikatanov, L.: The method of alternating projections and the method of subspace corrections in Hilbert space. J. Am. Math. Soc. **15**(3), 573–597 (2002). https://doi.org/10.1090/S0894-0347-02-00398-3

Publisher's Note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.