J Sci Comput (2016) 68:803–825
DOI 10.1007/s10915-015-0159-4

Analysis and Practical Use of Flexible BiCGStab

Jie Chen[1] · **Lois C. McInnes**[2] · **Hong Zhang**[2]

Received: 23 April 2015 / Revised: 23 October 2015 / Accepted: 28 December 2015 /
Published online: 11 January 2016

Abstract A flexible version of the BiCGStab algorithm for solving a linear system of equations is analyzed. We show that under variable preconditioning, the perturbation to the outer residual norm is of the same order as that to the application of the preconditioner. Hence, in order to maintain a similar convergence behavior to BiCGStab while reducing the preconditioning cost, the flexible version can be used with a moderate tolerance in the preconditioning Krylov solves. We explored the use of flexible BiCGStab in a large-scale reacting flow application, PFLOTRAN, and showed that the use of a variable multigrid preconditioner significantly accelerates the simulation time on extreme-scale computers using $O(10^4)$–$O(10^5)$ processor cores.

Keywords Krylov method · BiCGStab · Variable preconditioning · Extreme-scale simulation

1 Introduction

Flexible iterative methods [4,15,16,21,24,35,37,39] for solving a linear system of equations refer to preconditioned Krylov methods where the preconditioner may vary across iterations. The flexible preconditioning strategy is also known under various terms such as *variable* or

✉ Jie Chen
chenjie@us.ibm.com

[1] IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA

[2] Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA

inexact preconditioning. A representative scenario is that the preconditioning requires a linear solve with a second iterative method, in which case "inner iterations" refer to preconditioning and "outer iterations" refer to the flexible Krylov method itself. Flexible methods are an important class of methods that offer several advantages over the use of a fixed preconditioner, one of which is the flexibility to balance the accuracy of the preconditioning solves and the speed of convergence of the outer Krylov iterations in order to reduce the total computational cost. Furthermore, in large-scale applications, the changing landscape of both scientific needs (complex physical models and couplings) and emerging extreme-scale computing systems give rise to practical preconditioners that are hierarchical or nested [8,9,11,17,18,27]. Many of these emerging preconditioners benefit from inexact inner solves and thus require the use of outer flexible Krylov methods.

Among many proposed flexible iterative methods, flexible GMRES (abbreviated as FGMRES [24]) is the most frequently used in practice. Its wide use is probably linked to the robustness that results from the long-term recurrence and global orthogonality. Compared with standard GMRES [26], even though the traditional notion of a Krylov subspace is lost, FGMRES computes an orthonormal basis of a subspace within which an optimal residual is sought. Hence, FGMRES still enjoys the residual norm minimization property, and it often shows a satisfactory convergence behavior. On the other hand, in other flexible iterative methods with short-term recurrences, such as inexact PCG [16], flexible QMR [35], and flexible BiCG [39], global (bi)orthogonality is lost, and the convergence behavior is often unpredictable unless the inner solves are sufficiently accurate so that orthogonality is nearly preserved. An idea to improve the robustness is to explicitly perform the orthogonalization as proposed for a variant of the flexible CG algorithm [21]. On the other hand, several analyses of flexible methods, using a larger Krylov subspace that includes the Arnoldi vectors, indicate that the convergence behavior can be maintained with respect to the fixed preconditioning case if the perturbation to the preconditioner grows inversely with the current residual norm [12,29,30].

BiCGStab [36] is a widely used Krylov method. In many applications, BiCGStab outperforms GMRES in terms of both solution time and memory usage, and it has become the de facto method of choice for practitioners. Although BiCGStab is akin to BiCG [13], which generates two sets of biorthogonal residual vectors that naturally form two associated Krylov subspaces, the convergence behavior of BiCGStab is harder to describe because the residual vectors alone do not span the Krylov subspace that contains them. BiCGStab can be seen as redefining the residual polynomial of BiCG by squaring the degree with a smoothing effect. BiCGStab can also be understood as a member of a family of induced dimension reduction (IDR) methods whereby the generated residuals belong to a nested sequence of shrinking subspaces [31–33], which is analogous to other Krylov methods where there is a similar subspace defined (for example, $\mathcal{K}^{\perp}$ for CG and $(A\mathcal{K})^{\perp}$ for GMRES, where $\mathcal{K}$ is the current Krylov subspace). The orthogonal complement notation for the latter methods is consistent with Saad's viewpoint of projection methods [25]; however, BiCGStab does not belong to this class.

This work is motivated by the scaling difficulty of the simulation of reacting flows in a geological application PFLOTRAN [2,19] on extreme-scale parallel computers. A well-known bottleneck in Krylov methods for solving large-scale linear systems on parallel computers is the inner-product calculations, because they require global synchronizations. The scaling of the solver starts to deteriorate when the number of processor cores increases beyond $O(10^5)$ [3]. Among several strategies, such as deferring or pipelining inner product calculations, overlapping communications with computations, and block orthogonalizations [10,14,20,23,34,40], the use of a hierarchical preconditioner that reduces the length of vectors in inner-product calculations has been demonstrated to be effective for inner-outer

GMRES iterations [18]. This discovery prompted us to generalize the idea to flexible BiCGStab iterations (FBiCGStab) [38,39], because BiCGStab has historically been the preferred solver for PFLOTRAN by domain scientists. However, the convergence behavior is not well understood. In particular, we find that FBiCGStab does not always converge if the preconditioner is not chosen appropriately (see experiments in Sect. 3 and also results in [39]).

The goal of this paper is to analyze the convergence properties of FBiCGStab, describe factors that affect its convergence, and provide guidance on its use in practice. We argue that the convergence behavior is close to that of the fixed preconditioning case if the perturbation to the preconditioner is not too large. Thus, an appropriate perturbation is key to the performance: if too large, the residual behavior of the outer iterations is unpredictable; if too small, the inner solves may be time consuming to complete. Because of the loose connection of BiCGStab with the associated Krylov subspace, this analysis is different from that for other flexible Krylov methods (see [12,29,30]). Rather, the arguments are made on orthogonality and norm minimization properties that guarantee bounds of perturbations on the iterates. Interestingly, this result can lead to a conclusion similar to that in [7,12,29,30]; that is, the convergence behavior of the flexible method can be maintained by relaxing the accuracy requirement of the preconditioning solves as the outer residual norm decreases. For demonstration, we design for PFLOTRAN a variable preconditioner based on multigrid and obtain remarkable improvement in the simulation time using $O(10^4)$–$O(10^5)$ processor cores.

The rest of the paper is organized as follows. With a brief derivation of FBiCGStab, Sect. 2 analyzes the behavior of the residual norm under flexible preconditioning and gives examples to illustrate the interpretation of the results. Then, several numerical examples are shown in Sect. 3 to demonstrate the need for a stopping criterion with a moderate tolerance in the preconditioning inner solves. Section 4 presents the successful use of FBiCGStab with multigrid preconditioners in PFLOTRAN. Concluding remarks are given in Sect. 5.

2 Algorithm and analysis

The following is the standard unpreconditioned BiCGStab algorithm for solving a linear system

$$Ax = b,$$

using x_0 as the initial vector [25]:

```
1: $r_0 = b - Ax_0$; $\bar{r}_0$ arbitrary but $(\bar{r}_0, r_0) \neq 0$
2: $p_0 = r_0$
3: **for** $j = 0, 1, \ldots$ until convergence **do**
4:     $\alpha_j = (r_j, \bar{r}_0)/(Ap_j, \bar{r}_0)$
5:     $s_j = r_j - \alpha_j Ap_j$
6:     $\omega_j = (As_j, s_j)/(As_j, As_j)$
7:     $x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j$
8:     $r_{j+1} = s_j - \omega_j As_j$
9:     $\beta_j = (r_{j+1}, \bar{r}_0)/(r_j, \bar{r}_0) \cdot \alpha_j/\omega_j$
10:    $p_{j+1} = r_{j+1} + \beta_j(p_j - \omega_j Ap_j)$
11: **end for**
```

The coefficient iterates α_j and β_j are derived based on their counterparts in BiCG for updating the residual vectors and the search direction vectors. One can show that α_j makes

$s_j \perp \bar{r}_0$ and β_j makes $p_{j+1} \perp \bar{r}_0$ for all j. Furthermore, ω_j is defined to minimize the 2-norm of the residual vector r_{j+1} given s_j and As_j.

When the algorithm is used with a preconditioner $M \approx A$, the right preconditioning consists of solving the system

$$AM^{-1}y = b, \qquad y = Mx.$$

One way to derive the preconditioned iteration is, in the above unpreconditioned version, to replace the symbol A by AM^{-1} and x_j by y_j, and then substitute y_j back by Mx_j. This introduces two auxiliary vectors $\tilde{p}_j = M^{-1}p_j$ and $\tilde{s}_j = M^{-1}s_j$, which are the only computations that require the preconditioner. We summarize this preconditioned version in Algorithm 1. It is the same as the one presented in [39].

Algorithm 1 Right preconditioned BiCGStab / Flexible BiCGStab

```
1: $r_0 = b - Ax_0$; $\bar{r}_0$ arbitrary
2: $p_0 = r_0$
3: **for** $j = 0, 1, \ldots$ until convergence **do**
4:     $\tilde{p}_j = M^{-1}p_j$
5:     $\alpha_j = (r_j, \bar{r}_0)/(A\tilde{p}_j, \bar{r}_0)$
6:     $s_j = r_j - \alpha_j A\tilde{p}_j$
7:     $\tilde{s}_j = M^{-1}s_j$
8:     $\omega_j = (A\tilde{s}_j, s_j)/(A\tilde{s}_j, A\tilde{s}_j)$
9:     $x_{j+1} = x_j + \alpha_j\tilde{p}_j + \omega_j\tilde{s}_j$
10:    $r_{j+1} = s_j - \omega_j A\tilde{s}_j$
11:    $\beta_j = (r_{j+1}, \bar{r}_0)/(r_j, \bar{r}_0) \cdot \alpha_j/\omega_j$
12:    $p_{j+1} = r_{j+1} + \beta_j(p_j - \omega_j A\tilde{p}_j)$
13: **end for**
```

One would also like to consider left preconditioning, where M^{-1} is applied to the left of the system $Ax = b$. After a change of variables, the left preconditioned version is almost the same as Algorithm 1, except that the two inner products in line 8 are changed to the ones using $(MM^T)^{-1}$-norm. In this case, ω_j minimizes the $(MM^T)^{-1}$-norm of r_{j+1}. Compared with right preconditioning, left preconditioning incurs two more applications of the preconditioner in each iteration, which increases the computational cost. Furthermore, similar to other Krylov methods, a major hurdle for developing variable preconditioners for left preconditioning is the disconnection between the preconditioned residuals and the actual residuals. Therefore, we do not consider left preconditioning in this paper.

An iterative method can be used to compute $\tilde{p}_j$ in line 4 and $\tilde{s}_j$ in line 7 of Algorithm 1, but the iterations may not run to full accuracy. In this case, Algorithm 1 becomes the flexible version of BiCGStab. The computed iterates $\tilde{p}_j$ and $\tilde{s}_j$ under inexact preconditioning will carry on their error to subsequent iterations. To gauge the amplification of error, we are interested in the situation that the relative residual of the inner solves with M is bounded by a small tolerance ϵ. That is, if we use an underline to denote the actual iterates with errors, we assume that

$$\|\underline{p_j} - M\underline{\tilde{p}_j}\| \le \epsilon\|\underline{p_j}\| \quad \text{and} \quad \|\underline{s_j} - M\underline{\tilde{s}_j}\| \le \epsilon\|\underline{s_j}\|. \tag{1}$$

The following subsection characterizes the relative difference between r_{j+1} and $\underline{r_{j+1}}$ under condition (1).

2.1 Analysis

The analysis is based on the fact that the coefficient iterates α_j, β_j, and ω_j are computed such that the inaccuracy incurred in the preconditioning solves is not “adversely” accumulated to affect outer iterations. For this, we need the following observations. They are trivially correct in the fixed preconditioned case, but they are also true when a flexible preconditioner is used. The proof is simple and thus omitted.

Proposition 1 *The iterates in Algorithm 1 have the following properties:*

1. *The vector r_{j+1} is the residual, that is, $r_{j+1} = b - Ax_{j+1}$.*
2. *Consider that s_j is a function of α_j; then the definition of α_j in line 5 makes $s_j \perp \bar{r}_0$.*
3. *Consider that p_{j+1} is a function of β_j; then the definition of β_j in line 11 makes $p_{j+1} \perp \bar{r}_0$.*
4. *Consider that r_{j+1} is a function of ω_j; then the definition of ω_j in line 8 minimizes $\|r_{j+1}\|_2$.*

In light of these observations, we will bound the perturbation to vectors of the form $x - \alpha y$, where the scalar α is used to satisfy some orthogonality or minimization property when x and y are perturbed. We will make heavy use of angles spanned by two $\mathbb{R}^n$ vectors. For this, we use $\angle(x, y)$ to denote the *acute* angle between the two vectors, that is, $|(x, y)| = \|x\|\|y\| \cos\angle(x, y)$. Hence, $\cos\angle(x, y)$ is always nonnegative. The following lemma establishes the basic fact, generalizing from the intuitive $\mathbb{R}^3$ case, that for three vectors spanning three angles, one of the angles is bounded by the sum and the difference of the other two. Then, two lemmas follow, stating that the perturbation to $x - \alpha y$ is in the same order as that to x and y.

Lemma 1 *If all pairwise angles among three vectors x, y, and z are acute, then*

$$|\angle(x, y) - \angle(y, z)| \ \le\ \angle(x, z) \ \le\ \angle(x, y) + \angle(y, z).$$

Proof Without loss of generality, we assume that $\|x\| = \|y\| = \|z\| = 1$. It suffices to prove, for the case $\angle(x, y) + \angle(y, z)$ is acute, that

$$\begin{aligned}\cos\angle(x, y)\cos\angle(y, z) - \sin\angle(x, y)\sin\angle(y, z) \ &\le\ \cos\angle(x, z)\\ \le\ \cos\angle(x, y)\cos\angle(y, z) + \sin\angle(x, y)\sin\angle(y, z),&\end{aligned}$$

which is equivalent to

$$|(x, z) - (x, y)(y, z)| \le \sqrt{1 - (x, y)^2}\sqrt{1 - (y, z)^2}. \tag{2}$$

Note that

$$|(x, z) - (x, y)(y, z)| = |x^T(I - yy^T)z|$$

and that

$$\sqrt{1 - (x, y)^2}\sqrt{1 - (y, z)^2} = \sqrt{x^T(I - yy^T)x}\sqrt{z^T(I - yy^T)z}.$$

Thus, Cauchy’s inequality for vector semi-inner products with respect to the symmetric positive semi-definite matrix $I - yy^T$ proves (2). □

Lemma 2 *Given a vector r, let $z = x - \alpha y$ and $\underline{z} = \underline{x} - \underline{\alpha}\underline{y}$, where $\alpha = (x, r)/(y, r)$ and $\underline{\alpha} = (\underline{x}, r)/(\underline{y}, r)$, and let $\gamma = \operatorname{sgn}[(x, y)(x, r)(y, r)]$. If there exist ϵ_x, ϵ_y such that $\epsilon_x < \cos\angle(x, r)$, $\epsilon_y < \cos\angle(y, r)$ and that*

$$\|x - \underline{x}\| \le \epsilon_x\|x\| \quad \text{and} \quad \|y - \underline{y}\| \le \epsilon_y\|y\|,$$

then $\|z - \underline{z}\| \le \epsilon_z\|z\|$ with

$$\epsilon_z = \frac{\epsilon_x + \dfrac{\cos\angle(x,r)}{\cos\angle(y,r)}\Big[2\epsilon_y + (1+\epsilon_x)(1+B)(1+D) - 1\Big]}{\sqrt{1 + \dfrac{\cos^2\angle(x,r)}{\cos^2\angle(y,r)} - 2\gamma\cos\angle(x,y)\dfrac{\cos\angle(x,r)}{\cos\angle(y,r)}}}, \tag{3}$$

$$B = \epsilon_x\left(\frac{\epsilon_x}{2\sqrt{1-\epsilon_x^2}} + \tan\angle(x,r)\right), \qquad D = \frac{2\epsilon_y\sqrt{1-\epsilon_y^2}\tan\angle(y,r) + \epsilon_y^2}{-2\epsilon_y\sqrt{1-\epsilon_y^2}\tan\angle(y,r) + 2(1-\epsilon_y^2)}.$$

In other words, denoting by $\epsilon = \max\{\epsilon_x, \epsilon_y\}$, we have $\epsilon_z \le C\epsilon$ where the prefactor C has a finite limit

$$\lim_{\epsilon\to 0} C = \frac{\cos\angle(y,r) + \cos\angle(x,r)[3 + \tan\angle(x,r) + \tan\angle(y,r)]}{\sqrt{\cos^2\angle(y,r) + \cos^2\angle(x,r) - 2\gamma\cos\angle(x,y)\cos\angle(x,r)\cos\angle(y,r)}}.$$

Proof First we have

$$\begin{aligned}\|z - \underline{z}\| &\le \|x - \underline{x}\| + \|\alpha y - \underline{\alpha}\underline{y}\| \\ &\le \|x - \underline{x}\| + \|\alpha(y - \underline{y})\| + \|(\alpha - \underline{\alpha})\underline{y}\| \\ &\le \epsilon_x\|x\| + \epsilon_y\|\alpha y\| + \|(\alpha - \underline{\alpha})\underline{y}\|.\end{aligned}$$

Based on Lemma 1, when $\epsilon_x < \cos\angle(x, r)$, the angle between x and r and that between $\underline{x}$ and r will be acute, or obtuse, at the same time. This means that the inner products (x, r) and $(\underline{x}, r)$ have the same sign. Similarly it holds for (y, r) and $(\underline{y}, r)$. Thus, α and $\underline{\alpha}$ have the same sign, and hence

$$\frac{\|(\alpha - \underline{\alpha})\underline{y}\|}{\|\alpha y\|} = \left|\frac{\|\underline{y}\|}{\|y\|} - \frac{\|\underline{\alpha}\underline{y}\|}{\|\alpha y\|}\right|.$$

Using the fact that $|\|\underline{y}\|/\|y\| - 1| \le \epsilon_y$, we get

$$\|(\alpha - \underline{\alpha})\underline{y}\| \le \|\alpha y\|\left(\epsilon_y + \left|1 - \frac{\|\underline{\alpha}\underline{y}\|}{\|\alpha y\|}\right|\right). \tag{4}$$

With $\|\alpha y\| = \|x\|\cos\angle(x, r)/\cos\angle(y, r)$ and

$$\|z\|^2 = \|x\|^2\left(1 + \frac{\cos^2\angle(x,r)}{\cos^2\angle(y,r)} - 2\gamma\cos\angle(x,y)\frac{\cos\angle(x,r)}{\cos\angle(y,r)}\right),$$

we thus obtain

$$\begin{aligned}\|z - \underline{z}\| &\le \epsilon_x\|x\| + \|\alpha y\|\left(2\epsilon_y + \left|1 - \frac{\|\underline{\alpha}\underline{y}\|}{\|\alpha y\|}\right|\right) \\ &= \|z\|\frac{\epsilon_x + \frac{\cos\angle(x,r)}{\cos\angle(y,r)}\left(2\epsilon_y + \left|1 - \frac{\|\underline{\alpha}\underline{y}\|}{\|\alpha y\|}\right|\right)}{\sqrt{1 + \frac{\cos^2\angle(x,r)}{\cos^2\angle(y,r)} - 2\gamma\cos\angle(x,y)\frac{\cos\angle(x,r)}{\cos\angle(y,r)}}}.\end{aligned} \tag{5}$$

Therefore, we proceed to bound

$$\left|1 - \frac{\|\underline{\alpha y}\|}{\|\alpha y\|}\right| = \left|1 - \frac{\|\underline{x}\|}{\|x\|}\frac{\cos\angle(\underline{x}, r)}{\cos\angle(x, r)}\frac{\cos\angle(y, r)}{\cos\angle(\underline{y}, r)}\right|.$$

The strategy of bounding this term is to find $A, B, D > 0$ such that

$$\left|1 - \frac{\|\underline{x}\|}{\|x\|}\right| \le A, \qquad \left|1 - \frac{\cos\angle(\underline{x}, r)}{\cos\angle(x, r)}\right| \le B, \qquad \left|1 - \frac{\cos\angle(y, r)}{\cos\angle(\underline{y}, r)}\right| \le D;$$

then,

$$\left|1 - \frac{\|\underline{\alpha y}\|}{\|\alpha y\|}\right| \le (1 + A)(1 + B)(1 + D) - 1. \tag{6}$$

Clearly, we can let $A = \epsilon_x$.

To simplify notation, let $\angle(x, r) = \beta$ and $\angle(\underline{x}, r) = \beta + \delta$ for some δ. Then,

$$\begin{aligned}\left|1 - \frac{\cos(\beta + \delta)}{\cos\beta}\right| &= |(1 - \cos\delta) + \sin\delta\tan\beta| \\ &= \left|(\sin\delta)\left(\tan\frac{\delta}{2} + \tan\beta\right)\right| \le |\sin\delta|\left(\frac{1}{2}|\tan\delta| + \tan\beta\right).\end{aligned} \tag{7}$$

Because $|\sin\delta| \le \epsilon_x$,

$$\left|1 - \frac{\cos(\beta + \delta)}{\cos\beta}\right| \le \epsilon_x\left(\frac{\epsilon_x}{2\sqrt{1 - \epsilon_x^2}} + \tan\angle(x, r)\right) =: B. \tag{8}$$

Similarly, let $\angle(y, r) = \eta$ and $\angle(\underline{y}, r) = \eta + \tau$ for some τ. Note that based on Lemma 1, $|\tau| \le \pi/2 - \eta$. Using

$$\left|\tan\frac{\tau}{2}\right| \le \frac{1}{2}|\tan\tau| \le \frac{\epsilon_y}{2\sqrt{1 - \epsilon_y^2}},$$

we have

$$\left|1 - \frac{\cos\eta}{\cos(\eta + \tau)}\right| = \left|\frac{\tan\eta + \tan(\tau/2)}{\tan\eta - \cot\tau}\right| \le \frac{\tan\angle(y, r) + \frac{\epsilon_y}{2\sqrt{1-\epsilon_y^2}}}{-\tan\angle(y, r) + \frac{\sqrt{1-\epsilon_y^2}}{\epsilon_y}} =: D. \tag{9}$$

With (8) and (9), we establish (6). Then, together with (5), we have proved (3). □

Lemma 3 *Let $z = x - \alpha y$ and $\underline{z} = \underline{x} - \underline{\alpha}\underline{y}$, where $\alpha = (x, y)/\|y\|^2$ and $\underline{\alpha} = (\underline{x}, \underline{y})/\|\underline{y}\|^2$. If (x, y) and $(\underline{x}, \underline{y})$ have the same sign and there exist ϵ_x, ϵ_y such that $\epsilon_x + \epsilon_y < 1$ and that*

$$\|x - \underline{x}\| \le \epsilon_x\|x\| \quad \textit{and} \quad \|y - \underline{y}\| \le \epsilon_y\|y\|,$$

then $|\alpha - \underline{\alpha}| \le \epsilon_\alpha|\alpha|$ and $\|z - \underline{z}\| \le \epsilon_z\|z\|$ with

$$\epsilon_\alpha = (\epsilon_x + \epsilon_y)(1 + \epsilon_y)\left\{1 + (1 + \epsilon_x)\left[\frac{\epsilon_x + \epsilon_y}{2\sqrt{1 - (\epsilon_x + \epsilon_y)^2}} + \tan\angle(x, y)\right]\right\} \tag{10}$$

and

$$\epsilon_z = (\epsilon_x + \epsilon_y)(1+\epsilon_x) + \frac{\epsilon_x}{\sin\angle(x,y)} + \frac{\epsilon_y}{\tan\angle(x,y)} + \frac{\epsilon_x+\epsilon_y}{\tan\angle(x,y)} \times \left[1 + \frac{(1+\epsilon_x)(\epsilon_x+\epsilon_y)}{2\sqrt{1-(\epsilon_x+\epsilon_y)^2}}\right]. \tag{11}$$

In other words, denoting by $\epsilon = \max\{\epsilon_x, \epsilon_y\}$, *we have* $\epsilon_\alpha \le C_\alpha \epsilon$ *and* $\epsilon_z \le C_z \epsilon$, *where the prefactors* C_α *and* C_z *have finite limits*

$$\lim_{\epsilon\to 0} C_\alpha = 2 + 2\tan\angle(x,y), \qquad \lim_{\epsilon\to 0} C_z = \left(2 + \frac{1+3\cos\angle(x,y)}{\sin\angle(x,y)}\right).$$

Proof Using a similar argument as in the proof of the preceding lemma, we reach

$$\begin{aligned}\|z-\underline{z}\| &\le \epsilon_x\|x\| + \epsilon_y\|\alpha y\| + \|(\alpha-\underline{\alpha})\underline{y}\| \\ &\le \|z\|\left[\frac{\epsilon_x}{\sin\angle(x,y)} + \cot\angle(x,y)\left(2\epsilon_y + \left|1 - \frac{\|\underline{\alpha}\underline{y}\|}{\|\alpha y\|}\right|\right)\right],\end{aligned} \tag{12}$$

by noting that $\|\alpha y\| = \|x\|\cos\angle(x,y)$, that $\|z\| = \|x\|\sin\angle(x,y)$, and that α and $\underline{\alpha}$ have the same sign by the condition of this lemma. Furthermore, with (4), which clearly also holds here, we have

$$\frac{|\alpha-\underline{\alpha}|}{|\alpha|} \le \frac{\|y\|}{\|\underline{y}\|}\left(\epsilon_y + \left|1 - \frac{\|\underline{\alpha}\underline{y}\|}{\|\alpha y\|}\right|\right) \le (1+\epsilon_y)\left(\epsilon_y + \left|1 - \frac{\|\underline{\alpha}\underline{y}\|}{\|\alpha y\|}\right|\right). \tag{13}$$

Therefore, we proceed to bound

$$\left|1 - \frac{\|\underline{\alpha}\underline{y}\|}{\|\alpha y\|}\right| = \left|1 - \frac{\|\underline{x}\|}{\|x\|}\frac{\cos\angle(\underline{x},\underline{y})}{\cos\angle(x,y)}\right|.$$

The strategy of bounding this term is to find $A, B > 0$ such that

$$\left|1 - \frac{\|\underline{x}\|}{\|x\|}\right| \le A, \qquad \left|1 - \frac{\cos\angle(\underline{x},\underline{y})}{\cos\angle(x,y)}\right| \le B;$$

then,

$$\left|1 - \frac{\|\underline{\alpha}\underline{y}\|}{\|\alpha y\|}\right| \le (1+A)(1+B) - 1. \tag{14}$$

Clearly, we can let $A = \epsilon_x$.

To simplify notation, let $\angle(x,y) = \beta$ and $\angle(\underline{x},\underline{y}) = \beta + \delta$ for some δ. Then, the same as (7), we have

$$\left|1 - \frac{\cos(\beta+\delta)}{\cos\beta}\right| \le |\sin\delta|\left(\frac{1}{2}|\tan\delta| + \tan\beta\right).$$

Let θ_x be the angle between x and $\underline{x}$, and similarly for θ_y. Because $\|x-\underline{x}\| \le \epsilon_x\|x\|$ with $\epsilon_x < 1$, θ_x is acute. Similarly, so is θ_y. Note that $\|x\|\sin\theta_x \le \|x-\underline{x}\| \le \epsilon_x\|x\|$; therefore, $\sin\theta_x \le \epsilon_x$, and similarly $\sin\theta_y \le \epsilon_y$. Then, $\sin\theta_x \le \epsilon_x < \sqrt{1-\epsilon_y^2} \le \cos\theta_y$, which indicates that $\theta_x + \theta_y$ is also acute. Thus, the fact that $|\delta| \le \theta_x + \theta_y$ leads to $\sin|\delta| \le$

$\sin\theta_x + \sin\theta_y \le \epsilon_x + \epsilon_y$. Therefore

$$\left|1 - \frac{\cos(\beta+\delta)}{\cos\beta}\right| \le (\epsilon_x + \epsilon_y)\left[\frac{\epsilon_x + \epsilon_y}{2\sqrt{1-(\epsilon_x+\epsilon_y)^2}} + \tan\angle(x,y)\right] =: B. \tag{15}$$

With (15), we establish (14). Then, together with (13) and (12), we have proved (10) and (11). □

Using the above two lemmas, we have the following result. It states that the relative perturbation to the outer residual norm is in the same order as the relative residual norm in the inner solves.

Theorem 1 *If Algorithm* 1 (*with flexible preconditioning*) *is run with a finite number of iterations where neither breakdown nor stagnation occurs, then under condition* (1), *for all* j,

$$\frac{\|r_j - \underline{r_j}\|}{\|r_j\|} = O(\epsilon). \tag{16}$$

Proof To facilitate presentation, we define $\mathrm{err}(a) := \|a - \underline{a}\|/\|a\|$ for any vector or scalar $a \neq 0$. We first observe that

$$\begin{aligned}\frac{\|A\tilde{p}_j - A\underline{\tilde{p}_j}\|}{\|A\tilde{p}_j\|} &= \frac{\|AM^{-1}(p_j - M\underline{\tilde{p}_j})\|}{\|AM^{-1}p_j\|} \le \kappa\frac{\|p_j - M\underline{\tilde{p}_j}\|}{\|p_j\|} \\ &\le \kappa\left(\frac{\|p_j - \underline{p_j}\|}{\|p_j\|} + \frac{\|\underline{p_j} - M\underline{\tilde{p}_j}\|}{\|p_j\|}\right) \le \kappa\left(\frac{\|p_j - \underline{p_j}\|}{\|p_j\|} + \epsilon\frac{\|\underline{p_j}\|}{\|p_j\|}\right),\end{aligned}$$

where κ denotes the condition number of AM^{-1}. Since the big-O notation is used for sufficiently small ϵ, the above observation means that if $\mathrm{err}(p_j) = O(\epsilon)$, then $\mathrm{err}(A\tilde{p}_j) = O(\epsilon)$. Similarly, if $\mathrm{err}(s_j) = O(\epsilon)$, then $\mathrm{err}(A\tilde{s}_j) = O(\epsilon)$.

We now show the theorem by induction on $\mathrm{err}(r_j)$ and $\mathrm{err}(p_j)$. The conditions in the theorem (no breakdown and stagnation) are used to ensure the applicability of the lemmas. At $j = 0$, r_0 and p_0 are unchanged under variable preconditioning. If $\mathrm{err}(r_j) = O(\epsilon)$ and $\mathrm{err}(p_j) = O(\epsilon)$, then because $\mathrm{err}(A\tilde{p}_j) = O(\epsilon)$, we have $\mathrm{err}(s_j) = O(\epsilon)$ by Lemma 2. Consequently, $\mathrm{err}(A\tilde{s}_j) = O(\epsilon)$, and thus $\mathrm{err}(r_{j+1}) = O(\epsilon)$ by Lemma 3.

Now consider $p_{j+1} = r_{j+1} + \beta_j z_j$ where $z_j = p_j - \omega_j A\tilde{p}_j$. Both $\mathrm{err}(p_j)$ and $\mathrm{err}(A\tilde{p}_j)$ are $O(\epsilon)$. On the other hand, $\mathrm{err}(\omega_j)$ is also $O(\epsilon)$ according to Lemma 3, because we have shown that both $\mathrm{err}(s_j)$ and $\mathrm{err}(A\tilde{s}_j)$ are $O(\epsilon)$. Hence, $\mathrm{err}(z_j) = O(\epsilon)$. Then by invoking Lemma 2 again we have $\mathrm{err}(p_{j+1}) = O(\epsilon)$, which completes the induction. □

Remark 1 Equation (16) concerns only the order with respect to ϵ. However, the result is derived based on the condition of a finite number iterations, so that factors other than ϵ are suppressed in the big-O notation as a constant. This hidden constant may be j-dependent and may grow with respect to j; see Fig. 1b in the next subsection.

Theorem 1 considers the perturbation of the residual in the relative sense. As a corollary, a result for the absolute perturbation is given next. Instead of a fixed tolerance ϵ for all the inner solves, we allow the tolerance, denoted by ϵ_j, to vary in each outer iteration j. The result indicates a reciprocal relationship between the residual norm $\|r_j\|$ and ϵ_j.

Corollary 1 *Let the relative inner tolerance* ϵ *depend on the outer iterations indexed by* j*, that is,*

$$\|\underline{p_j} - M\underline{\tilde{p}_j}\| \le \epsilon_j \|\underline{p_j}\| \quad \text{and} \quad \|\underline{s_j} - M\underline{\tilde{s}_j}\| \le \epsilon_j \|\underline{s_j}\|.$$

Under the condition of Theorem 1, if the residual norm $\|r_j\|$ *is monotonically decreasing, then for any* δ *there exists a constant* C *such that if*

$$\epsilon_j = \frac{C\delta}{\|r_j\|},$$

then $\|r_j - \underline{r_j}\| \le \delta$.

Proof Note that Theorem 1 is proved by induction on j. When ϵ_j is monotonically increasing but is still sufficiently small (guaranteed by a finite number of iterations), a stronger conclusion is that there exists a C' that is independent of j such that

$$\mathrm{err}(r_j) \le C'\epsilon_j, \tag{17}$$

because in the right-hand side ϵ_j can always be relaxed later by changing it to ϵ_{j+1}. Rewriting (17), we have $\|r_j - \underline{r_j}\| \le C'\epsilon_j\|r_j\|$. Therefore, if we let $\epsilon_j = C\delta/\|r_j\|$ by using some C such that $C'C \le 1$ and that $C\delta/\|r_j\|$ is sufficiently small to trigger the validity of (17), we immediately have $\|r_j - \underline{r_j}\| \le \delta$. □

2.2 Interpretation and illustrative examples

Roughly speaking, the above results state that if the perturbation to the preconditioning step is not relatively large, then the convergence history looks similar to that of the case without perturbation. Whereas this gives a qualitative description of the behavior of BiCGStab under variable preconditioning, one must be cautious in interpreting the theoretical results and not overly emphasize their predictive power in a quantitative manner. First and most important, these results are not convergence assertions. Since BiCGStab itself is not guaranteed to converge for general unsymmetric matrices, it does not seem reasonable to expect that a flexible version magically proves the opposite.

Second, the perturbation result is built based on a small-ϵ regime. Clearly, when $\epsilon \to 0$, such as when ϵ is the machine precision, the term "flexible" gradually loses its meaning since the convergence history will be almost identical to that of the fixed preconditioning case. A tricky question is determining when the behavior of flexible BiCGStab starts to diverge from that of standard BiCGStab, since it depends on when the accumulated prefactor in front of ϵ, as in (16), grows to an unacceptable level. It is unclear how to characterize this prefactor, but the prefactor is certainly connected to the bound ϵ_z in Lemma 2 and the bounds ϵ_α and ϵ_z in Lemma 3. Empirically we see that the perturbation in the range of $O(10^{-4})$–$O(10^{-2})$ gives reasonably similar convergence behavior to the case of no perturbation ($\epsilon = 0$).

Figure 1 shows the convergence results of solving a model problem that is introduced in more detail in Sect. 3. Here, the discretization of the problem in 2D yields a matrix of size $65,536 \times 65,536$, and a choice of the parameters $\gamma = 4/h$ and $\beta = -0.2/h^2$ renders the matrix positive definite but unsymmetric. With block Jacobi/ILU(0) preconditioning of block size 64×64 and a zero initial vector, the residuals monotonically decrease to machine precision in 40 iterations [see the red curve without markers in plot (a)].

We artificially perturbed the vectors p_j and s_j [see (1)] by a Gaussian random vector normalized to a norm of $\epsilon\|p_j\|$ and $\epsilon\|s_j\|$, respectively, before computing the preconditioned

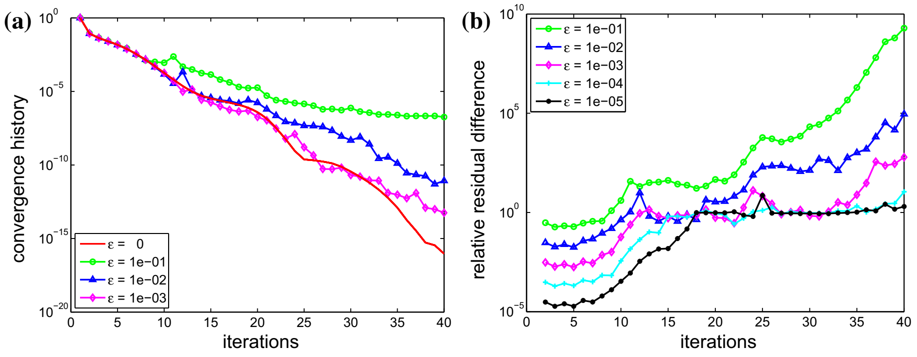

Fig. 1 **a** Convergence history and **b** relative residual difference $\|r_j - \underline{r_j}\|/\|r_j\|$ under a fixed level of perturbation ϵ in the preconditioning step. In **a** a few curves are omitted to avoid cluttering

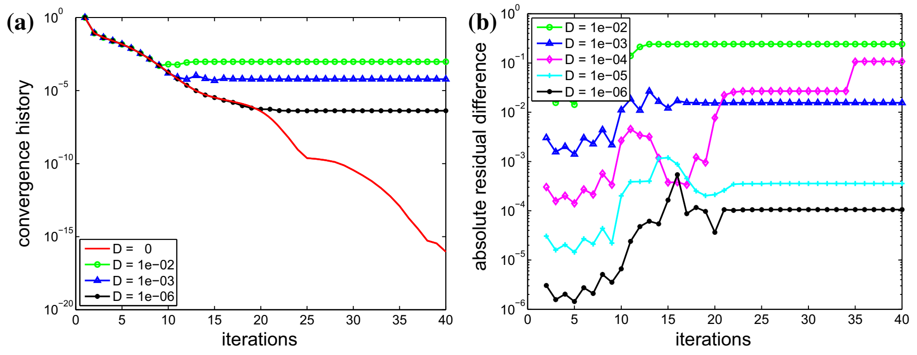

Fig. 2 **a** Convergence history and **b** absolute residual difference $\|r_j - \underline{r_j}\|$ under a dynamically adjusted perturbation $\epsilon_j = D/\|r_j\|$ in the preconditioning step. In **a** a few curves are omitted to avoid cluttering

vectors $\tilde{p}_j$ and $\tilde{s}_j$ by block Jacobi/ILU(0). This mimics the situation of an inexact preconditioning solve with relative residual tolerance ϵ. One sees that in plot (a), as ϵ decreases, the convergence history is closer and closer to the reference red curve. We also ran experiments with smaller ϵ's, but the residual curves were so close to the reference curve that we omit them in the plot to avoid cluttering. In plot (b), we show $\|r_j - \underline{r_j}\|/\|r_j\|$. As predicted by Theorem 1, the curves corresponding to a smaller ϵ tend to be positioned below those corresponding to a larger ϵ.

The absolute difference, $\|r_j - \underline{r_j}\|$, on the other hand, can be made bounded as opposed to the increasing trends of Fig. 1b. One simple way is, in fact, to do nothing, because $\|r_j\|$ itself decreases. More interesting is that we can increase ϵ across iterations, as Corollary 1 implies. In Fig. 2 we define $\epsilon_j = D/\|r_j\|$ for various different choices of D. The iterations converge to a level dependent on D.

One is tempted to infer from Corollary 1 that successively relaxing the perturbation ϵ_j in this manner can perhaps reduce the preconditioning cost as the iterations converge. However, several caveats render this corollary to be less practically useful than Theorem 1. First, when $\|r_j\|$ is large, ϵ_j is typically so small that there is no practical difference with requiring the preconditioning to be exact. Second, we do not know the residuals r_j in variable preconditioning—we have only the perturbed ones $\underline{r_j}$. Third, the corollary is based on the condition that r_j is monotonically decreasing, a too restrictive requirement. Hence we consider Corollary 1 to be mainly of theoretical interest.

3 Numerical examples

To empirically study when flexible preconditioning for BiCGStab is useful and to compare its performance with that of the counterparts of GMRES, we extend the example in Sect. 2 and perform more comprehensive experimentation. Consider the linear system arising from a discretization of the PDE (adopted from [24]):

$$-\Delta u + \gamma \boldsymbol{x} \cdot \nabla u + \beta u = f \tag{18}$$

with zero Dirichlet boundary condition. The linear system can be made indefinite and/or unsymmetric by changing the parameters γ and β. We discretized the 3D domain into a regular grid of size $n_1 \times n_2 \times n_3$ with spacing $h = 1/n_1$ and set $\gamma = 4/h$ to make the problem unsymmetric. We varied the parameter β to include both definite and indefinite cases. The right-hand side was chosen to be the vector of all ones, with the initial x_0 being zero. As a common practice in parallel solvers, if no specific preconditioner is mentioned, an iterative method (including the case of being used for inner iterations) was always preconditioned by block Jacobi/ILU(0), where each block was handled by one processor. The experiments were conducted on the supercomputer Titan (a Cray XK7 system) hosted at the Oak Ridge Leadership Computing Facility [22].

We tested several β values; Table 1 shows the results of three representative cases. As β decreases, the problem becomes harder to solve. In the first case ($\beta = 0.01/h^2$) the system is positive definite, but in the latter two cases ($\beta = -0.4/h^2$ and $-0.6/h^2$) the system is indefinite. Figure 3a shows the convergence history of BiCGStab and GMRES(30). The convergence history for the hardest case $\beta = -0.6/h^2$ is extended in plot 3b. These results were obtained on a $256 \times 256 \times 256$ grid using 16,384 processor cores. The relative residual tolerance and the maximum number of iterations were `1e-8` and 200, respectively.

We compared BiCGStab, FBiCGStab/BiCGStab, GMRES, and FGMRES/GMRES, where FBiCGStab/BiCGStab means FBiCGStab is preconditioned by BiCGStab, and similarly for FGMRES/GMRES. In both of the flexible methods, the stopping criterion for the inner iterations was either a residual tolerance or a fixed number of iterations. The two major comparison metrics are the number of matrix-vector products (MatMult, column 4) and the number of calls to `MPI_Allreduce` (Allreduce, column 5). The number of floating point operations (Flops, column 6) for computing the inner products is listed for reference, but it is not a fair criterion for comparison because (i) the synchronization cost in `MPI_Allreduce` is much higher than that of floating point operations, and (ii) GMRES requires long-term recurrence (orthogonalization) whereas BiCGStab does not. The wallclock time is also listed, but note that the fluctuations in a distributed memory computing environment and the communication latency are factors that affect the actual running time.

Several observations are made according to Table 1 and other experiments with intermediate β values that are not shown in the table. First, for this set of test cases, the GMRES family in general performs better when the system is relatively easy to solve; but as the difficulty level increases, the BiCGStab family outperforms the GMRES family. One sees that in the first case ($\beta = 0.01/h^2$) GMRES is the best solver, whereas in the second case it failed to converge to the required tolerance but BiCGStab did converge. In the second case ($\beta = -0.4/h^2$), however, the fastest solver is still in the GMRES family. Nevertheless,

Table 1 Solution summary of (18) with three choices of β

	Outer	Time	MatMult	Inner product	
$\beta = 0.01/h^2$	iter.	(sec$\times 10^{-1}$)		Allreduce	Flops$\times 10^5$
BiCGStab	21	1.132	42	84	3.010
FBiCGStab, inner `rtol = 1e-3`	2	1.397	58	120	4.246
FBiCGStab, inner `rtol = 1e-2`	2	1.335	44	92	3.233
FBiCGStab, inner `rtol = 1e-1`	4	1.364	54	116	4.037
FBiCGStab, inner `max_it = 5`	2	1.131	44	92	3.233
FBiCGStab, inner `max_it = 10`	1	1.147	40	82	2.910
FBiCGStab, inner `max_it = 20`	1	1.226	46	94	3.334
GMRES	29	0.874	29	58	9.494$\leftarrow$
FGMRES, inner `rtol = 1e-3`	3	1.122	40	83	6.081
FGMRES, inner `rtol = 1e-2`	4	7.200	37	78	4.240
FGMRES, inner `rtol = 1e-1`	7	1.153	38	83	3.292
FGMRES, inner `max_it = 5`	6	1.072	36	78	3.130
FGMRES, inner `max_it = 10`	3	1.066	33	69	4.237
FGMRES, inner `max_it = 20`	2	1.065	40	82	8.720
$\beta = -0.4/h^2$		$\times 10^0$			$\times 10^6$
BiCGStab	105	0.356	210	420	1.505
FBiCGStab, inner `rtol = 1e-3`	2	0.668	438	880	3.144
FBiCGStab, inner `rtol = 1e-2`	2	0.499	270	544	1.948
FBiCGStab, inner `rtol = 1e-1`	6	0.832	586	1184	4.225
FBiCGStab, inner `max_it = 5`	Fail	–	–	–	–
FBiCGStab, inner `max_it = 10`	Fail	–	–	–	–
FBiCGStab, inner `max_it = 20`	25	2.517	2050	4150	14.800
GMRES	Fail	–	–	–	–
FGMRES, inner `rtol = 1e-3`	3	1.556	1165	2298	37.890
FGMRES, inner `rtol = 1e-2`	4	1.062	803	1586	25.550
FGMRES, inner `rtol = 1e-1`	7	0.791	560	1113	17.160
FGMRES, inner `max_it = 5`	26	0.282	156	338	1.893
FGMRES, inner `max_it = 10`	14	0.278	154	322	2.134$\leftarrow$
FGMRES, inner `max_it = 20`	12	0.373	252	516	5.861
$\beta = -0.6/h^2$		$\times 10^1$			$\times 10^7$
BiCGStab	Fail	–	–	–	–
FBiCGStab, inner `rtol = 1e-3`	2	0.951	8314	16632	5.962
FBiCGStab, inner `rtol = 1e-2`	2	0.572	5024	10052	3.605$\leftarrow$
FBiCGStab, inner `rtol = 1e-1`	15	3.564	30120	60270	21.560
FBiCGStab, inner `max_it = 5`	Fail	–	–	–	–
FBiCGStab, inner `max_it = 10`	Fail	–	–	–	–
FBiCGStab, inner `max_it = 20`	Fail	–	–	–	–
GMRES	Fail	–	–	–	–
FGMRES, inner `rtol = 1e-3`	5	5.291	51670	101680	169.600

Table 1 continued

$\beta = -0.6/h^2$		$\times 10^1$			$\times 10^7$
FGMRES, inner `rtol = 1e-2`	5	4.589	41800	82259	136.560
FGMRES, inner `rtol = 1e-1`	7	3.903	37757	74305	123.730
FGMRES, inner `max_it = 5`	Fail	–	–	–	–
FGMRES, inner `max_it = 10`	Fail	–	–	–	–
FGMRES, inner `max_it = 20`	Fail	–	–	–	–

Arrows point to the fastest run

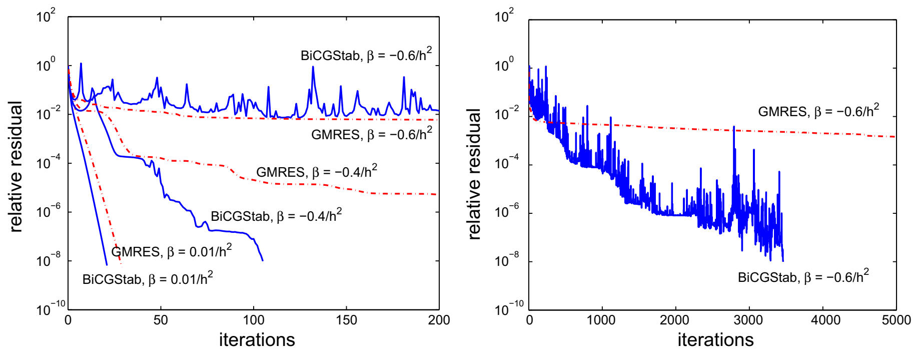

Fig. 3 Convergence of BiCGStab and GMRES for three β's (no inner iterations)

when moving to the third case ($\beta = -0.6/h^2$), FBiCGStab with an inner tolerance stopping criterion clearly wins over all other solvers. In this case, the inner BiCGstab converges much faster than does inner GMRES, as can be seen from the number of matrix-vector products (amortized by the number of outer iterations) and also from the convergence history in Fig. 3b.

Second, for a flexible Krylov method, using a fixed number of inner iterations can sometimes achieve excellent performance; but as the problem becomes harder and harder, it is difficult to specify an appropriate number a priori to ensure the convergence of the outer iterations. One sees that in the first case, using a fixed number of inner iterations as the stopping criterion is in general preferable over using a residual tolerance. This is also true in the second case for the FGMRES solvers; however, for the FBiCGStab solvers the situation is completely opposite. In the third case, none of the solvers using a fixed number of inner iterations converged. In this sense, setting an inner tolerance is a more robust practice.

Third, one can choose an "optimal" inner tolerance for a flexible method. One sees that for FBiCGStab the inner tolerance `1e-2` yields the best results in all the cases, whereas for FGMRES the tolerance is `1e-1`. This is consistent with the observation made in the experiments of flexible QMR [35] and other methods [7,12,29,30], which state that the total solver cost first decreases, then increases as the inner solves are more and more exact. The "optimal" inner tolerance may be related to the convergence behavior of the inner iterations. One sees that in Fig. 3 the relative residual norm of BiCGStab has a steep decrease at the beginning, until between `1e-1` and `1e-2`. This may be the stopping point when BiCGStab becomes the most effective as an inner solve.

Fourth, the inner-outer iterations (that is, FBiCGStab/BiCGStab and FGMRES/GMRES) are often a better alternative to the standard iterations (that is, BiCGStab and GMRES). In harder problems the standard iterations did not converge whereas the inner-outer iterations did. In fact, the outer iterations converge extremely fast when an appropriate inner tolerance is used.

In addition, we vary the grid size and show in Table 2 two indicative factors of the convergence behavior: the number of matrix-vector products and the number of calls to `MPI_Allreduce`. These results confirm again that the problem becomes more difficult to solve as β decreases. Moreover, these results also indicate that the problem becomes more difficult to solve as the grid size increases. While in most of the cases the quickest convergence is obtained by a member of the GMRES family, in the hardest case (lower-right corner of the table) FBiCGStab wins.

4 Application

In this section, we explore the use of flexible BiCGStab to improve the run-time performance of a large-scale application, PFLOTRAN [2,19], where historically standard BiCGStab with block Jacobi/ILU(0) preconditioner has been the preferred linear solver. Based on the analysis in Sect. 2, we compose a multigrid (MG) preconditioner. Its use with BiCGStab yields two to three times improvement in solution time on $O(10^4)$–$O(10^5)$ processor cores. When the coarse grid solver varies slightly (thus becoming a variable preconditioner and having to be used with FBiCGStab), we gain an additional 10–20 % in overall runtime improvement.

PFLOTRAN is a state-of-the-art code for simulating multiscale, multiphase, multicomponent flow and reactive transport in geologic media. It solves a coupled system of mass and energy conservation equations for a number of phases, including air, water, supercritical CO_2 and a number of chemical components. The code utilizes finite volume or mimetic finite difference spatial discretizations and backward-Euler (fully implicit) timestepping. At each time step, Newton–Krylov methods are used for solving the resulting nonlinear algebraic equations. PFLOTRAN is built on the PETSc library [5,6] and makes extensive use of PETSc iterative nonlinear and linear solvers.

The governing equations are described by Richards' equation:

$$\frac{\partial}{\partial t}(\varphi s \rho) + \nabla \cdot \rho \boldsymbol{u} = \mathcal{S},$$

where φ denotes the porosity of the geologic medium, s the saturation (fraction of pore volume filled with liquid water), ρ the fluid density, $\mathcal{S}$ a source/sink term representing water injection/extraction, and $\boldsymbol{u}$ the Darcy velocity defined as

$$\boldsymbol{u} = -\frac{\kappa \kappa_r}{\mu} \nabla (P - \rho g z),$$

where P denotes fluid pressure, μ viscosity, κ the absolute permeability of the medium, κ_r the relative permeability of water to air, g the acceleration of gravity, and z the vertical distance from a datum.

We consider two benchmark problems [18]:

Case 1: Cubic domain with a central injection well. This case models a 100 m × 100 m × 100 m domain with a uniform effective permeability of 1 darcy and an injection well at the exact center.

Case 2: Regional flow without well near river. This case models a 5000 m × 2500 m × 100 m region with a river at the eastern boundary.

Table 2 Solution summary of (18) for varying grid sizes, with the same setting as in Table 1

	Grid 64^3		Grid 128^3		Grid 256^3	
	MatMult	Dot	MatMult	Dot	MatMult	Dot
$\beta = 0.01/h^2$						
BiCGStab	34	68	42	84	42	84
FBiCGStab, inner `rtol = 1e-3`	50	104	60	124	58	120
FBiCGStab, inner `rtol = 1e-2`	36	76	44	92	44	92
FBiCGStab, inner `rtol = 1e-1`	44	96	50	108	54	116
FBiCGStab, inner `max_it = 5`	44	92	44	92	44	92
FBiCGStab, inner `max_it = 10`	36	74	42	86	40	82
FBiCGStab, inner `max_it = 20`	38	78	48	98	46	94
GMRES	24	48	29	58	29	58
FGMRES, inner `rtol = 1e-3`	35	73	40	83	40	83
FGMRES, inner `rtol = 1e-2`	31	66	38	80	37	78
FGMRES, inner `rtol = 1e-1`	34	75	39	85	38	83
FGMRES, inner `max_it = 5`	30	65	36	78	36	78
FGMRES, inner `max_it = 10`	33	69	33	69	33	69
FGMRES, inner `max_it = 20`	34	70	41	84	40	82
$\beta = -0.4/h^2$						
BiCGStab	46	92	76	152	210	420
FBiCGStab, inner `rtol = 1e-3`	66	136	102	208	438	880
FBiCGStab, inner `rtol = 1e-2`	42	88	62	128	270	544
FBiCGStab, inner `rtol = 1e-1`	48	104	90	190	586	1184
FBiCGStab, inner `max_it = 5`	44	92	88	184	Fail	Fail
FBiCGStab, inner `max_it = 10`	42	86	84	172	Fail	Fail
FBiCGStab, inner `max_it = 20`	50	102	80	162	2050	4150
GMRES	33	65	48	95	Fail	Fail
FGMRES, inner `rtol = 1e-3`	42	87	61	125	1165	2298
FGMRES, inner `rtol = 1e-2`	41	86	58	120	803	1586
FGMRES, inner `rtol = 1e-1`	43	93	57	121	560	1113
FGMRES, inner `max_it = 5`	42	91	60	130	156	338
FGMRES, inner `max_it = 10`	44	92	55	115	154	322
FGMRES, inner `max_it = 20`	41	84	63	129	252	516
$\beta = -0.6/h^2$						
BiCGStab	58	116	178	356	Fail	Fail
FBiCGStab, inner `rtol = 1e-3`	84	172	334	672	8314	16632
FBiCGStab, inner `rtol = 1e-2`	56	116	214	432	5024	10052
FBiCGStab, inner `rtol = 1e-1`	96	204	490	992	30120	60270
FBiCGStab, inner `max_it = 5`	66	138	Fail	Fail	Fail	Fail
FBiCGStab, inner `max_it = 10`	84	172	4368	8944	Fail	Fail
FBiCGStab, inner `max_it = 20`	64	130	328	664	Fail	Fail
GMRES	41	81	Fail	Fail	Fail	Fail
FGMRES, inner `rtol = 1e-3`	52	107	364	721	51670	101680
FGMRES, inner `rtol = 1e-2`	49	102	483	957	41800	82259

Table 2 continued

	Grid 64^3		Grid 128^3		Grid 256^3	
	MatMult	Dot	MatMult	Dot	MatMult	Dot
FGMRES, inner `rtol = 1e-1`	51	109	298	599	37757	74305
FGMRES, inner `max_it = 5`	48	104	126	273	Fail	Fail
FGMRES, inner `max_it = 10`	55	115	132	276	Fail	Fail
FGMRES, inner `max_it = 20`	42	86	168	344	Fail	Fail

"Dot" means the number of Allreduces for computing inner products

In our numerical experiments, we ran each test case with a minimum number of time steps—six for case 1 and two for case 2—at which the number of linear iterations per Newton step became stabilized.

BiCGStab preconditioned by block Jacobi/ILU(0) has been the preferred linear solver for PFLOTRAN because of its small memory consumption, compared with that of GMRES, and the empirically fast convergence. Similar to other Krylov methods, however, as the size of the application and the number of processors increase, BiCGStab encounters a well-known scaling difficulty for over 10,000 processor cores because of the bottleneck in vector inner-product calculations. The synchronization cost in `MPI_Allreduce` for computing inner products can constitute more than half of the solution time. In order to overcome the scaling difficulty, two algorithms were explored to successfully reduce the synchronization cost [18]. One was the use of flexible GMRES with a hierarchical GMRES preconditioner, where inner GMRES iterations are run on diagonal submatrices over subgroups of processor cores only. On each submatrix, inner GMRES is further preconditioned by block Jacobi/ILU(0). We denote this approach FGMRES/h-GMRES. The second algorithm was to use the IBiCGStab algorithm [41] with a Chebyshev preconditioner. We denote this approach to be IBiCGStab/Chebyshev. IBiCGStab is mathematically equivalent to BiCGStab, but the iterates are reorganized so that several inner products are computed together to reduce the number of synchronizations. The Chebyshev iterations may not be as effective in reducing the condition number as other Krylov iterations, but an advantage is that it does not require any inner product calculations and thus can be used as a fixed preconditioner. The two algorithms are based on the idea of reducing expensive global inner products across the entire system by using cheaper local inner products (e.g., FGMRES/h-GMRES) or inner iterations that do not compute inner products at all (e.g., IBiCGStab/Chebychev).

The success in [18] prompted us to explore similar ideas for BiCGStab. The first attempt was to use BiCGStab in place of GMRES in FGMRES/h-GMRES, such as FBiCGStab/h-GMRES and FBiCGStab/h-BiCGStab. The use of a small constant number of inner iterations generally failed for convergence; the failure is not surprising because a few iterations make the preconditioner vary too much, and FBiCGStab does not guarantee any form of outer convergence as does FGMRES (monotonic decrease of residuals regardless of the varying of preconditioner). On the other hand, when we used an inner tolerance as the stopping criterion, convergence is seen for a reasonably chosen value (see Fig. 4). In fact, the general trend for the same tolerance is similar for whichever preconditioner is used, either h-GMRES or h-BiCGStab, and the smaller the tolerance, the closer the residual history is to a fixed preconditioner case. However, this convergence was obtained at the cost of a large number of inner iterations. The overall run time cannot compete with the simple strategy of using BiCGStab preconditioned by block Jacobi/ILU(0).

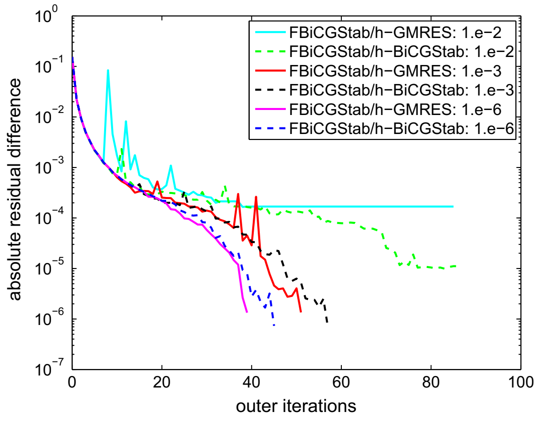

Fig. 4 Convergence history of FBiCGStab/h-GMRES and FBiCGStab/h-BiCGStab for PFLOTRAN case 1 (mesh size $256 \times 256 \times 256$) using 512 processor cores

We therefore searched for a preconditioner that converges faster to the level of `1e-2` to `1e-3`. Among all preconditioners examined, we settled on multigrid with a particular choice of the smoothers and the coarse-grid solver. Multigrid (MG) is generally applicable to steady-state or close to steady-state problems, such as the two cases of PFLOTRAN we consider here. In the standard setting, multigrid is a fixed preconditioner, because the smoothers (such as SOR or Chebyshev) require no inner-product calculations and the coarsest-grid problem is solved exactly by using a direct linear solver. In other words, one can, in principle, write down the linear operator for one cycle of multigrid and see that it does not vary.

For PFLOTRAN applications, the MG preconditioner was known to work well on a small number of processors, but the performance started tailing off at around 1000 processor cores. The difficulty to scale up the number of processors is that in the coarsest level, the problem is so small that the communication cost outweighs the computational cost. We cope with this difficulty by limiting the number of levels (effectively, three), such that the coarsest grid is not too small, and employing an iterative method to approximately solve the coarsest-grid problem. Two coarse grid solvers are (1) 100 Chebyshev iterations, and (2) 5 IBiCGStab iterations, each preconditioned by 20 Chebyshev iterations. For the former, even though the coarsest-grid problem is not solved to full accuracy, multigrid is still considered a fixed preconditioner because there are no inner product calculations. For the latter, clearly, multigrid is a variable preconditioner.

We conducted experiments with the above ideas on two computer systems: Intrepid, an IBM Blue Gene/P supercomputer located at the Argonne Leadership Computing Facility [1], and Titan, a Cray XK7 system located at the Oak Ridge Leadership Computing Facility [22]. Intrepid has 40,960 nodes, each consisting of one 850 MHz quad-core processor and 2GB RAM, resulting in a total of 163,840 cores, 80TB of memory, and a peak performance of 557 TFlops. Titan has 18,688 compute nodes, each consisting of one AMD 16-core Opteron 6274 processor running at 2.2GHz and 32GB of memory, giving a total of 299,008 cores.

Tables 3, 4, 5, and 6 compare the performance of (1) IBiCGStab/Chebyshev, (2) BiCGStab with the fixed MG preconditioner, and (3) FBiCGStab with the variable MG preconditioner on the two benchmark cases. The results of IBiCGStab/Chebyshev (columns 2–3) have been reported in [18] and are used here as the baseline of comparison. For MG preconditioners (columns 4–8), V-cycle was used; the smoothers were 2 steps of Chebyshev iterations because we found that Chebyshev outperformed other choices of smoothers. For both smoothers and coarse grid solvers, block Jacobi/ILU(0) was used as the innermost preconditioner.

Table 3 BiCGStab and FBiCGStab for PFLOTRAN on Intrepid (IBM Blue Gene/P), Case 1

Number of cores (Mesh size)	IBiCGS/cheby		BiCGStab/MG Smoother: cheby CSolve: cheby		FBiCGStab/variable MG Smoother: cheby CSolve: IBiCGS/cheby	
	Iter.	Time	Iter.	Time	Iter.	Time (% Reduction)
512 (256×256×256)	547	212.8	29	97.2	23	86.4 (11)
4096 (512×512×512)	1006	365.1	43	121.3	33	106.1 (12)
32,768 (1024×1024×1024)	1886	654.3	62	153.7	37	119.1 (23)
163,840 (1600 × 1600 × 640)	2843	308.3	88	81.8	53	65.7 (20)

Table 4 BiCGStab and FBiCGStab for PFLOTRAN on Intrepid (IBM Blue Gene/P), Case 2

Number of cores (Mesh size)	IBiCGS/cheby		BiCGStab/MG Smoother: cheby CSolve: cheby		FBiCGStab/variable MG Smoother: cheby CSolve: IBiCGS/cheby	
	Iter.	Time	Iter.	Time	Iter.	Time (% Reduction)
16,384 (1600 × 816 × 320)	844	231.1	33	64.3	22	52.6 (18)
98,304 (1600 × 1632 × 640)	1520	270.5	61	70.0	39	54.7 (22)
163,840 (1600 × 1632 × 640)	1499	169.3	62	52.0	36	40.2 (23)

Table 5 BiCGStab and FBiCGStab for PFLOTRAN on Titan (Cray XK7), Case 1

Number of cores (Mesh size)	IBiCGS/cheby		BiCGStab/MG Smoother: cheby CSolve: cheby		FBiCGStab/variable MG Smoother: cheby CSolve: IBiCGS/cheby	
	Iter.	Time	Iter.	Time	Iter.	Time (% Reduction)
512 (256×256×256)	546	24.2	29	13.5	23	12.5 (7)
4096 (512×512×512)	1033	44.1	43	18.1	33	16.6 (8)
32,768 (1024×1024×1024)	2073	89.0	62	27.4	37	24.3 (11)
160,000 (1600 × 1600 × 640)	2407	52.0	91	24.9	55	22.5 (10)

Overall, FBiCGStab with an MG preconditioner is two to three times faster than the baseline IBiCGStab/Chebyshev on a large number of processor cores. Hence, the focus of comparison here is how much reduction in execution time one can achieve by using a

Table 6 BiCGStab and FBiCGStab for PFLOTRAN on Titan (Cray XK7), Case 2

Number of cores (Mesh size)	IBiCGS/cheby		BiCGStab/MG Smoother: cheby CSolve: cheby		FBiCGStab/variable MG Smoother: cheby CSolve: IBiCGS/cheby	
	Iter.	Time	Iter.	Time	Iter.	Time (% Reduction)
1600 (800×408×160)	411	18.0	20	8.8	20	8.8 (0)
16,000 (1600 × 816 × 320)	829	29.6	30	11.3	22	10.9 (4)
80,000 (1600 × 1632 × 640)	1578	53.1	60	16.7	38	15.0 (10)
224,000 (1600 × 1632 × 640)	1501	20.9	66	16.6	37	14.8 (11)

Table 7 Comparison of FBiCGStab and FGMRES for PFLOTRAN on Titan (Cray XK7), Case 2

Number of cores (Mesh size)	FBiCGStab/variable MG Smoother: cheby CSolve: IBiCGS/cheby			FGMRES/variable MG Smoother: cheby CSolve: IBiCGS/cheby		
	Iter.	Time	Mem. (GB)	Iter.	Time	Mem. (GB)
600 (800×408×160)	20	8.8	3	37	8.5	27
16,000 (1600 × 816 × 320)	22	10.9	25	38	10.2	215
80,000 (1600 × 1632 × 640)	38	15.0	100	61	14.3	859
224,000 (1600 × 1632 × 640)	37	14.8	100	60	13.6	859

variable preconditioner compared with using a fixed one. The percentage of reduction in execution time relative to the fixed MG preconditioner is listed in column 8 (% Reduction). As the size of the problem (hence coarsest grid) increases, the fixed number of iterations used for the coarsest-grid solver weakens the MG preconditioner, resulting in an increased number of outer iterations. Such an increase is less significant for the variable MG preconditioner because IBiCGStab/Chebyshev is more effective as a preconditioner than Chebyshev alone is. When the number of cores becomes larger than 30,000 (see the last two rows of Tables 3 through 6), the number of outer iterations for using the variable MG preconditioner (column 6) is almost half that of using the fixed MG preconditioner (column 4), leading to a reduction of 10 and 20 % in overall execution time, on Titan and Intrepid, respectively.

Comparing the results obtained on the two machines, one sees a consistent iteration number (for some grid sizes, a slightly different number of processor cores was used across machines; this affected the innermost block Jacobi/ILU(0), thus making the iteration numbers slightly different). However, the time improvement of using a variable MG preconditioner is very different across the two machines. Because the clock rate of Intrepid is much lower than that of Titan, the solution time on Intrepid is longer. Comparing the individual operations, the performance gains on Intrepid for the variable preconditioner are primarily due to fewer matrix-vector products and triangular solves in block ILU(0). The increased global synchronization cost of `MPI_Allreduce` by IBCGS on the coarse grid is insignificant because the communication network of Intrepid has a lower latency. Titan has larger memory per core than Intrepid, making the local triangular solves a negligible portion of the entire computation. Thus, the performance gains on Titan for the variable preconditioner arise mainly due to fewer matrix-vector products. In addition, the increased global synchronization cost on Titan becomes more noticeable. This phenomenon is not rare in practice. It showcases that for a solver, not only the theoretical convergence matters, but also the machine architecture plays an important role.

By using the variable MG preconditioner, we change the outer iteration from FBiCGStab to FGMRES and compare their performance; see Table 7. In general, FGMRES outperforms FBiCGStab by a slight margin, due to the fact that FGMRES requires one matrix-vector product and application of the preconditioner per iteration, whereas FBiCGStab requires two. The total number of outer FBiCGStab iterations is slightly more than half of that of FGMRES; hence, the former takes slightly more time. The key to the success of both flexible methods is good use of the MG preconditioner, which lands on the sweet spot of the tradeoff between inner tolerance and outer convergence.

In the table we also list the memory consumption of the outer iterations (since the preconditioners are the same). An advantage of FBiCGStab is that it consumes much less memory than does FGMRES, whose memory footprint is nearly proportional to the restart size, which often needs to be large. Although there is no shortage of computing nodes in extreme-scale computer systems, the cost considerations may limit memory capacity, which improves 10$\times$ slower than does the system peak floating point rate [28]. In addition, memory bandwidth remains challenging for large applications; a light-weight solver enables more efficient data access and can make room for other program components, such as preconditioning.

5 Concluding remarks

BiCGStab has been the de facto method of choice in many application domains for solving linear systems. Motivated by the challenges in large-scale scientific applications and extreme-scale computer architectures that encourage the use of variable preconditioners, we analyzed flexible BiCGStab and showed that the change of the convergence behavior with respect to standard BiCGStab is in accordance with the inaccuracy of the preconditioning solves. Thus, often a stopping criterion with a moderate tolerance, say `1e-4` to `1e-2`, for the preconditioning solves is favored in order both to maintain convergence and to reduce overall computation time. To this end, we demonstrated through numerical experiments, including the PFLOTRAN reacting flow application, that FBiCGStab with variable preconditioning yields comparable performance on extreme-scale computers to FGMRES, yet requires a smaller memory footprint. This work provides insight on the practical use of FBiCGStab for large-scale applications.

Acknowledgments We thank Satish Balay, Jed Brown and Barry Smith for insightful discussions and assistance with experiments. The authors were supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. Part of Jie Chen's work was conducted when he was with Argonne National Laboratory.

References

1. ALCF: Intrepid supercomputer. http://www.alcf.anl.gov/intrepid
2. Andre, B., Bisht, G., Collier, N., Hammond, G., Karra, S., Kumar, J., Lichtner, P., Mills, R.: PFLOTRAN project. http://pflotran.org/
3. Ang, J., Evans, K., Geist, A., Heroux, M., Hovland, P., Marques, O., McInnes, L., Ng, E., Wild, S.: Report on the workshop on extreme-scale solvers: Transitions to future architectures. Office of Advanced Scientific Computing Research, U.S. Department of Energy (2012). URL http://science.energy.gov/~/media/ascr/pdf/program-documents/docs/reportExtremeScaleSolvers2012.pdf Washington, DC, March 8-9, 2012
4. Axelsson, O., Vassilevski, P.S.: A black box generalized conjugate gradient solver with inner iterations and variable-step preconditioning. SIAM J. Matrix Anal. Appl. **12**(4), 625–644 (1991)
5. Balay, S., Abhyankar, S., Adams, M.F., Brown, J., Brune, P., Buschelman, K., Eijkhout, V., Gropp, W.D., Kaushik, D., Knepley, M.G., McInnes, L.C., Rupp, K., Smith, B.F., Zhang, H.: PETSc users manual. Tech. Rep. ANL-95/11-Revision 3.5, Argonne National Laboratory (2014) URL http://www.mcs.anl.gov/petsc
6. Balay, S., Gropp, W.D., McInnes, L.C., Smith, B.F.: Efficient management of parallelism in object oriented numerical software libraries. In: Arge, E., Bruaset, A.M., Langtangen, H.P. (eds.) Modern Software Tools in Scientific Computing, pp. 163–202. Birkhauser Press (1997). URL ftp://info.mcs.anl.gov/pub/tech_reports/reports/P634.ps.Z
7. Bouras, A., Frayssé, V.: Inexact matrix-vector products in Krylov methods for solving linear systems: a relaxation strategy. SIAM J. Matrix Anal. Appl. **26**(3), 660–678 (2005)
8. Bridges, P.G., Ferreira, K.B., Heroux, M.A., Hoemmen, M.: Fault-tolerant linear solvers via selective reliability. CoRR arXiv:1206.1390 (2012)
9. Brown, J., Knepley, M.G., May, D.A., McInnes, L.C., Smith, B.F.: Composable linear solvers for multiphysics. In: Proceeedings of the 11th international symposium on parallel and distributed computing (ISPDC 2012), pp. 55–62. IEEE Computer Society (2012). URL http://doi.ieeecomputersociety.org/10.1109/ISPDC.2012.16
10. Chronopoulos, A., Gear, C.W.: S-step iterative methods for symmetric linear systems. J. Comput. Appl. Math. **25**, 153–168 (1989)
11. El maliki, A., Guenette, R., Fortin, M.: An efficient hierarchical preconditioner for quadratic discretizations of finite element problems. Numer. Linear Algebra Appl. **18**(5), 789–803 (2011). doi:10.1002/nla.757
12. Eshof, Jv, Sleijpen, G.L.G.: Inexact Krylov subspace methods for linear systems. SIAM J. Matrix Anal. Appl. **26**(1), 125–153 (2004)
13. Fletcher, R.: Conjugate gradient methods for indefinite systems. Lect. Notes Math. **506**, 73–89 (1976)
14. Ghysels, P., Ashby, T., Meerbergen, K., Vanroose, W.: Hiding global communication latency in the GMRES algorithm on massively parallel machines. Tech. report 04.2012.1, Intel Exascience Lab, Leuven, Belgium (2012). URL http://twna.ua.ac.be/sites/twna.ua.ac.be/files/latency_gmres.pdf
15. Giladi, E., Golub, G.H., Keller, J.B.: Inner and outer iterations for the Chebyshev algorithm. SIAM J. Numer. Anal. **35**, 300–319 (1995)
16. Golub, G.H., Ye, Q.: Inexact preconditioned conjugate gradient method with inner-outer iteration. SIAM J. Sci. Comput. **21**(4), 1305–1320 (1999)
17. Keyes, D.E., McInnes, L.C., Woodward, C., Gropp, W., Myra, E., Pernice, M., Bell, J., Brown, J., Clo, A., Connors, J., Constantinescu, E., Estep, D., Evans, K., Farhat, C., Hakim, A., Hammond, G., Hansen, G., Hill, J., Isaac, T., Jiao, X., Jordan, K., Kaushik, D., Kaxiras, E., Koniges, A., Lee, K., Lott, A., Lu, Q., Magerlein, J., Maxwell, R., McCourt, M., Mehl, M., Pawlowski, R., Randles, A.P., Reynolds, D., Rivière, B., Rüde, U., Scheibe, T., Shadid, J., Sheehan, B., Shephard, M., Siegel, A., Smith, B., Tang, X., Wilson, C., Wohlmuth, B.: Multiphysics simulations: challenges and opportunities. Int. J. High Perform. Comput. Appl. **27**(1), 4–83 (2013). URL http://www.ipd.anl.gov/anlpubs/2012/01/72183.pdf
18. McInnes, L.C., Smith, B., Zhang, H., Mills, R.T.: Hierarchical Krylov and nested Krylov methods for extreme-scale computing. Parallel Comput. **40**, 17–31 (2014). doi:10.1016/j.parco.2013.10.001
19. Mills, R.T., Sripathi, V., Mahinthakumar, G., Hammond, G., Lichtner, P.C., Smith, B.F.: Engineering PFLOTRAN for scalable performance on Cray XT and IBM BlueGene architectures. In: Proceedings of SciDAC 2010 Annual Meeting (2010)
20. Mohiyuddin, M., Hoemmen, M., Demmel, J., Yelick, K.: Minimizing communication in sparse matrix solvers. In: Proceedings of SC09. ACM (2009). doi:10.1145/1654059.1654096
21. Notay, Y.: Flexible conjugate gradients. SIAM J. Sci. Comput. **22**(4), 1444–1460 (2000)
22. OLCF: Jaguar supercomputer. https://www.olcf.ornl.gov/computing-resources/jaguar/
23. van Rosendale, J.: Minimizing inner product data dependencies in conjugate gradient iteration. In: Proceedings of the IEEE international conference on parallel processing. IEEE computer society (1983)

24. Saad, Y.: A flexible inner-outer preconditioned GMRES algorithm. SIAM J. Sci. Comput. **14**(2), 461–469 (1993). doi:10.1137/0914028
25. Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd edn. SIAM, Philadelpha (2003)
26. Saad, Y., Schultz, M.H.: GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**, 856–869 (1986)
27. Saad, Y., Sosonkina, M.: pARMS: a package for the parallel iterative solution of general large sparse linear systems user's guide. Tech. Rep. UMSI2004-8, Minnesota Supercomputer Institute, University of Minnesota (2004)
28. Shalf, J., Dosanjh, S., Morrison, J.: Exascale computing technology challenges. In: Palma, J.M.L.M., et al. (eds.) VECPAR 2010, LNCS 6449, pp. 1–25 (2010)
29. Simoncini, V., Szyld, D.: Flexible inner-outer Krylov subspace methods. SIAM J. Numer. Anal. **40**(6), 2219–2239 (2003)
30. Simoncini, V., Szyld, D.B.: Theory of inexact Krylov subspace methods and applications to scientific computing. SIAM J. Sci. Comput. **25**(2), 454–477 (2003)
31. Sleijpen, G.L., van Gijzen, M.B.: Exploiting BiCGstab(ℓ) strategies to induce dimension reduction. SIAM J. Sci. Comput. **32**(5), 2687–2709 (2010)
32. Sleijpen, G.L., Sonneveld, P., van Gijzen, M.B.: Bi-CGSTAB as an induced dimension reduction method. Appl. Numer. Math. **60**, 1100–1114 (2010)
33. Sonneveld, P., van Gijzen, M.B.: IDR(s): a family of simple and fast algorithms for solving large non-symmetric systems of linear equations. SIAM J. Sci. Comput. **31**(2), 1035–1062 (2008)
34. Sturler, E.D., van der Vorst, H.A.: Reducing the effect of global communication in GMRES(m) and CG on parallel distributed memory computers. Appl. Numer. Math. **18**, 441–459 (1995)
35. Szyld, D.B., Vogel, J.A.: FQMR: a flexible quasi-minimal residual method with inexact preconditioning. SIAM J. Sci. Comput. **23**(2), 363–380 (2001)
36. van der Vorst, H.: BiCGSTAB: a fast and smoothly converging variant of BiCG for the solution of nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **13**, 631–644 (1992)
37. Van der Vorst, H.A., Vuik, C.: GMRESR: a family of nested GMRES methods. Numer. Linear Algebra Appl. **1**(4), 369–386 (1994)
38. van Gijzen, M.B., Sleijpen, G.L., Zemke, J.P.M.: Flexible and multi-shift induced dimension reduction algorithms for solving large sparse linear systems. Tech. Rep. 11–06, Delft University of Technology (2011)
39. Vogel, J.A.: Flexible BiCG and flexible Bi-CGSTAB for nonsymmetric linear systems. Appl. Math. Comput. **188**(1), 226–233 (2007)
40. Vuduc, R.: Quantitative performance modeling of scientific computations and creating locality in numerical algorithms. Ph.D. thesis, Massachusetts Institute of Technology (1995)
41. Yang, L.T., Brent, R.: The improved BiCGStab method for large and sparse unsymmetric linear systems on parallel distributed memory architectures. In: Proceedings of the Fifth international conference on algorithms and architectures for parallel processing. IEEE (2002)