SIAM J. MATRIX ANAL. APPL.
Vol. 39, No. 3, pp. 1318–1338

THE ADAPTIVE s-STEP CONJUGATE GRADIENT METHOD*

ERIN C. CARSON†

Abstract. The performance of Krylov subspace methods on large-scale parallel computers is often limited by communication, or movement of data. This has inspired the development of s-step (or communication-avoiding) variants, in which iterations are computed in blocks of s. This reformulation can reduce the number of global synchronizations per iteration by a factor of $O(s)$ and has been shown to produce speedups in certain practical settings. Although the s-step variants are mathematically equivalent to their classical counterparts, they can behave quite differently in finite precision depending on the parameter s. If s is too large, the s-step method can suffer convergence delay and decreased attainable accuracy relative to the classical method. As a potential remedy to the loss of accuracy in s-step Krylov subspace methods, we introduce and analyze a variable s-step conjugate gradient method in which s may vary between blocks. From a bound on the growth of the residual gap in block k containing s_k iterations we derive a constraint on the condition numbers of the computed $O(s_k)$-dimensional Krylov subspace bases such that a desired accuracy is attainable. The analysis suggests that s_k can be allowed to grow at a rate inversely proportional to the size of the computed residuals without affecting accuracy. We then introduce the *adaptive s-step conjugate gradient method*, a variable s-step approach which uses inexpensive estimates and a user-specified constraint on accuracy to automatically select an appropriate s_k in each block. Our numerical experiments demonstrate that the adaptive s-step conjugate gradient method is able to attain up to the same accuracy as the classical conjugate gradient method in cases where the fixed s-step conjugate gradient method fails.

Key words. Krylov subspace methods, conjugate gradient, communication-avoiding, iterative methods

AMS subject classifications. 65F10, 65F50, 65G50, 65Y05, 65Y20

DOI. 10.1137/16M1107942

1. Introduction.

In this paper, we study the numerical behavior of variants of the conjugate gradient (CG) method [18] for solving linear systems $Ax = b$ where $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite. Mathematically, given an initial approximate solution x_1 and corresponding residual $r_1 = b - Ax_1$, the CG method iteratively updates the approximate solution using a sequence of nested Krylov subspaces $\mathcal{K}_1(A, r_1) \subset \mathcal{K}_2(A, r_1) \subset \cdots$, where

$$\mathcal{K}_i(A, r_1) = \text{span}(r_1, Ar_1, \ldots, A^{i-1}r_1)$$

denotes the i-dimensional Krylov subspace with matrix A and starting vector r_1. In iteration i, the updated approximate solution $x_{i+1} \in x_1 + \mathcal{K}_i(A, r_1)$ is chosen by imposing the Galerkin condition $r_{i+1} = b - Ax_{i+1} \perp \mathcal{K}_i$.

Thus each iteration of CG requires a matrix-vector product with A in order to expand the dimension of the Krylov subspace, and a number of inner products to perform the orthogonalization. On modern computer architectures, the speed with which the matrix-vector products and inner products can be computed may be limited by communication (i.e., the movement of data between parallel processors or levels of the memory hierarchy). This limits the potential speed of individual iterations attainable by an implementation of CG. To perform a sparse matrix-vector product

*Received by the editors December 14, 2016; accepted for publication (in revised form) by V. Simoncini June 12, 2018; published electronically August 23, 2018.
http://www.siam.org/journals/simax/39-3/M110794.html
†Courant Institute of Mathematical Sciences, New York University, New York, NY 10012 (erinc@cims.nyu.edu, http://math.nyu.edu/~erinc/).

in parallel, each processor must communicate to neighboring processors entries of the source vector and/or the destination vector that it owns. Inner products require a global synchronization; i.e., the computation cannot proceed until all processors have finished their local computation and communicated the result to other processors. For large-scale sparse problems on large-scale machines, the cost of synchronization between parallel processors can dominate the run-time (see, e.g., the exascale computing report [11, p. 28]).

Efforts toward removing the performance bottlenecks caused by communication in CG and other Krylov subspace methods have produced various approaches. One such approach are the s-step Krylov subspace methods (also called "communication-avoiding" Krylov subspace methods); for a thorough treatment of background, related work, and performance experiments, see, e.g., the theses [4, 19]. In s-step Krylov subspace methods, iterations are performed in blocks of s; i.e., in each iteration, the Krylov subspace is expanded by $O(s)$ dimensions by computing $O(s)$ new basis vectors, and then inner products between basis vectors are computed in one block operation. Although not the optimal strategy for every application, this approach has been shown to lead to significant speedups for a number of problems and real-world applications (see, e.g., [22, 29]). In the remainder of the paper, we will refer to the matrices whose columns consist of the $O(s)$ basis vectors computed in each block as *s-step basis matrices*. Further details of the s-step CG method are discussed in section 2.

Although in exact arithmetic the s-step CG method produces the exact same iterates as the classical CG method, in finite precision their behavior can differ significantly. In both s-step and classical CG, rounding errors due to finite precision arithmetic have two basic effects: a decrease in attainable accuracy and a delay of convergence. In the present work, we focus on the *maximum attainable accuracy*, i.e., the accuracy with which a method can solve $Ax = b$ on a computer with unit roundoff ε. The maximum attainable accuracy of CG and other classical Krylov subspace methods has been well studied; see, e.g., work by Greenbaum [16], Van der Vorst and Ye [28], Sleijpen, Van der Vorst, and Fokkema [25], Sleijpen, Van der Vorst, and Modersitzki [26], Björck, Elfving, and Strakoš [1], and Gutknecht and Strakoš [17].

It is well known that for s-step CG, as s is increased, the ill effects of finite precision roundoff error are amplified; see, e.g., [7]. In the extreme case, if the parameter s is chosen to be too large, the s-step method can fail to converge to the same level of accuracy as the classical method. This sensitive numerical behavior can pose an obstacle to the practical use of s-step CG.

As a potential remedy to numerical issues in the s-step CG method, in this work we propose a variable s-step CG method in which the value of s can vary between blocks. We bound the loss of accuracy in each block of the variable s-step CG method, which is determined by the growth of the gap between the true residual $b - Ax_i$ and the updated residual r_i due to finite precision roundoff error. Our results show that in order to guarantee that a specified accuracy is attainable, the condition number of the s-step basis matrix in each block should be no larger than a quantity related to the inverse of the 2-norms of the residuals computed within that block. In other words, as the method converges, more ill-conditioned bases (i.e., larger s values) can be used without causing an unacceptable loss of accuracy.

Our work is closely related to the theory of "inexact Krylov subspace methods," which says that the error in computing matrix-vector products can be allowed to grow at a rate inversely proportional to the norm of the updated residual without adversely affecting the attainable accuracy; see, e.g., the work of Simoncini and Szyld [24], Van

de Eschof and Sleijpen [27], Bouras and Frayssé [2], Bouras, Frayssé, and Giraud [3], and Giraud, Gratton, and Langou [14]. Our analysis results in a somewhat analogous "relaxation strategy," where instead of relaxing the accuracy of the matrix-vector products we instead relax a constraint on the condition numbers of the computed s-step basis matrices.

Based on our analysis of the variable s-step CG method, we then propose the *adaptive s-step CG method*, in which the variable block sizes are automatically determined such that a user-specified accuracy is attained. Whereas the fixed s-step CG method may not attain the same accuracy as classical CG depending on the choice of s, our numerical experiments on a set of example matrices demonstrate that the adaptive s-step CG method can achieve the same accuracy as classical CG. Thus, the benefit of the adaptive approach over the fixed s-step approach is that it is a more reliable method. At one extreme, if high accuracy is required, the adaptive approach may use $s = 1$ in each block, behaving like the classical method. At the other extreme, the adaptive approach may use the specified maximum s value in each block, behaving like the fixed s-step method.

There are previous uses of variable s-step Krylov subspace methods in the literature. Williams et al. [29] use a variable s-step BICGSTAB as the coarse grid solver in a multigrid method, although their motivation is improving performance rather than improving accuracy. Imberti and Erhel [20] have recently developed an s-step GMRES method that allows variable s sizes. They recommend using a predetermined sequence of $\{s_i\}$ values based on the Fibonacci sequence. In contrast, in our approach, the sequence $\{s_i\}$ is dynamically chosen based on the 2-norms of the updated residuals (which are not necessarily monotonically decreasing in CG). In addition, our method is designed such that a user-specified accuracy ε^* can be attained when $\varepsilon^* \geq \varepsilon_{\text{CG}}$, where ε_{CG} is the accuracy attained by classical CG for the same problem.

In section 2 we give a brief overview of the s-step CG method. Sections 3 and 4 present our main contribution: in section 3 we derive a constraint on the s-step basis matrix condition number in terms of the norms of the updated residuals and the desired accuracy in a variable s-step CG method, and in section 4 we detail the adaptive s-step CG algorithm that automatically determines the number of iterations to perform in each block. Numerical experiments are presented in section 5, and we discuss possible extensions and future work in section 6.

2. Background: *s*-step Krylov subspace methods. The growing cost of communication in large-scale sparse problems has created a recent resurgence of interest in the implementation, optimization, and development of s-step Krylov subspace methods; see, e.g., the recent works [10, 19, 22, 33, 21, 30, 32, 29]. The basic idea behind s-step Krylov subspace methods is to grow the underlying Krylov subspace $O(s)$ dimensions at a time and perform the necessary orthogonalization with a single global synchronization. Within a block of s iterations, the vector updates are performed implicitly by updating the coordinates of the iteration vectors in the $O(s)$-dimensional basis spanned by the columns of the s-step basis matrix. The algorithmic details vary depending on the particular Krylov subspace method considered, but the general approach is the same. There is a wealth of literature related to s-step Krylov subspace methods. A thorough bibliography of early work in this area can be found in [19, Table 1.1].

Although we restrict the present paper to s-step Krylov subspace methods, there are other variants designed with the goal of reducing synchronization cost, namely the so-called "pipelined" Krylov subspace methods; see [12, 13] as well as the recent

work [31] which combines s-step and pipelined approaches. These methods also suffer from a potential decrease in attainable accuracy, although the mechanism by which accuracy is lost is quite different. The recent work of Cools et al. [8] analyzes the attainable accuracy of pipelined CG and develops a residual replacement strategy which can improve the accuracy without sacrificing performance in many cases; such strategies have also been developed for s-step variants [5]. Residual replacement strategies are similar to the adaptive technique developed here in that both derive from a bound on the size of the residual gap. They differ, however, in their approach to ensuring that this quantity remains sufficiently small. We suspect that the ideas behind the adaptive s-step approach developed in the present work could also be useful for improving accuracy in the pipelined methods; this is worthy of further investigation.

In this paper, we focus on the s-step conjugate gradient method (Algorithm 2), which is equivalent to the classical CG method (Algorithm 1) in exact arithmetic. In the remainder of this section, we give a brief explanation of the s-step CG method and introduce notation used in the remainder of the paper; further details on the derivation, implementation, and performance of the s-step CG method can be found in, e.g., [4, 19].

Algorithm 1 Conjugate gradient (CG).

Input: $n \times n$ symmetric positive definite matrix A, length-n vector b, and initial approximation x_1 to $Ax = b$

Output: Approximate solution x_{i+1} to $Ax = b$ with updated residual r_{i+1}

1: $r_1 = b - Ax_1$, $p_1 = r_1$
2: **for** $i = 1, 2, \ldots,$ until convergence **do**
3: $\quad \alpha_i = r_i^T r_i / p_i^T A p_i$
4: $\quad q_i = \alpha_i p_i$
5: $\quad x_{i+1} = x_i + q_i$
6: $\quad r_{i+1} = r_i - A q_i$
7: $\quad \beta_i = r_{i+1}^T r_{i+1} / r_i^T r_i$
8: $\quad p_{i+1} = r_{i+1} + \beta_i p_i$
9: **end for**

The s-step CG method consists of an outer loop, indexed by k, which iterates over the blocks of iterations, and an inner loop, which iterates over iterations $j \in \{1, \ldots, s\}$ within a block. For clarity, we globally index iterations by $i \equiv sk + j$. It follows from the properties of CG that for $j \in \{0, \ldots, s\}$,

$$\begin{aligned} &p_{sk+j+1}, r_{sk+j+1} \in \mathcal{K}_{s+1}(A, p_{sk+1}) + \mathcal{K}_s(A, r_{sk+1}), \\ &x_{sk+j+1} - x_{sk+1} \in \mathcal{K}_s(A, p_{sk+1}) + \mathcal{K}_{s-1}(A, r_{sk+1}). \end{aligned} \tag{1}$$

Then the CG vectors for the next s iterations lie in the sum of the column spaces of the matrices

$$\begin{aligned} &\mathcal{P}_{k,s} = [\rho_0(A)p_{sk+1}, \ldots, \rho_s(A)p_{sk+1}], \text{ with } \operatorname{span}(\mathcal{P}_{k,s}) = \mathcal{K}_{s+1}(A, p_{sk+1}), \\ &\mathcal{R}_{k,s} = [\rho_0(A)r_{sk+1}, \ldots, \rho_{s-1}(A)r_{sk+1}], \text{ with } \operatorname{span}(\mathcal{R}_{k,s}) = \mathcal{K}_s(A, r_{sk+1}), \end{aligned} \tag{2}$$

where $\rho_j(z)$ is a polynomial of degree j satisfying the three-term recurrence

$$\begin{aligned} &\rho_0(z) = 1, \qquad \rho_1(z) = (z - \theta_0)\rho_0(z)/\gamma_0, \\ &\rho_j(z) = \left((z - \theta_{j-1})\rho_{j-1}(z) - \mu_{j-2}\rho_{j-2}(z)\right)/\gamma_{j-1}. \end{aligned} \tag{3}$$

For the monomial basis, $\theta_i = 0$, $\gamma_i = 1$, and $\mu_i = 0$ for all i. The Newton basis is another common choice for s-step Krylov subspace methods, since the coefficients can be determined using information about the spectrum of A. In this case, $\mu_i = 0$, $\gamma_i = 1$, and the θ_i's are approximate eigenvalues of A in a Leja ordering; see, e.g., [23] for further details.

With the $n \times (s+1)$ matrix $\mathcal{P}_{k,s}$ and the $n \times s$ matrix $\mathcal{R}_{k,s}$, we define the s-step basis matrix $\mathcal{Y}_{k,s} = [\mathcal{P}_{k,s}, \mathcal{R}_{k,s}]$ and define $\underline{\mathcal{Y}}_{k,s}$ to be the same as $\mathcal{Y}_{k,s}$ except with all zeros in columns $s+1$ and $2s+1$. Note that both $\mathcal{Y}_{k,s}$ and $\underline{\mathcal{Y}}_{k,s}$ are of size $n \times (2s+1)$. Note also that the s's in the subscripts are unnecessary here, but will be useful in describing the variable s-step CG method later on. Under certain assumptions on the nonzero structure of A, $\mathcal{Y}_{k,s}$ can be computed in parallel in each outer loop for the same asymptotic latency cost as a single matrix-vector product using the so-called matrix powers kernel (see [10] for details). Since the columns of $\mathcal{Y}_{k,s}$ are computed according to (3), we can write

$$A\underline{\mathcal{Y}}_{k,s} = \mathcal{Y}_{k,s}\mathcal{B}_{k,s}, \tag{4}$$

where $\mathcal{B}_{k,s}$ is a $(2s+1) \times (2s+1)$ block tridiagonal matrix defined by the three-term recurrence coefficients in (3).

By (1), there exist length-$(2s+1)$ vectors $p'_{k,j+1}$, $r'_{k,j+1}$, and $x'_{k,j+1}$ that represent the coordinates of the CG iterates (length-n vectors) p_{sk+j+1}, r_{sk+j+1}, and $x_{sk+j+1} - x_{sk+1}$, respectively, in $\mathcal{Y}_{k,s}$ for $j \in \{0, \ldots, s\}$. That is,

$$\begin{aligned} p_{sk+j+1} &= \mathcal{Y}_{k,s} p'_{k,j+1}, \\ r_{sk+j+1} &= \mathcal{Y}_{k,s} r'_{k,j+1}, \\ x_{sk+j+1} - x_{sk+1} &= \mathcal{Y}_{k,s} x'_{k,j+1}. \end{aligned} \tag{5}$$

Therefore, in the inner loop of s-step CG, rather than update the CG vectors explicitly, we instead update their coordinates in $\mathcal{Y}_{k,s}$, i.e., for $j \in \{1, \ldots, s\}$,

$$\begin{gathered} x'_{k,j+1} = x'_{k,j} + \alpha_{sk+j} p'_{k,j}, \quad r'_{k,j+1} = r'_{k,j} - \alpha_{sk+j} \mathcal{B}_{k,s} p'_{k,j}, \\ p'_{k,j+1} = r'_{k,j+1} + \beta_{sk+j} p'_{k,j}. \end{gathered}$$

Thus the matrix-vector product with A in each iteration becomes a matrix-vector product with the much smaller matrix $\mathcal{B}_{k,s}$. This along with the length-$(2s+1)$ vector updates can be performed locally by each processor in each inner loop iteration.

The largest source of potential performance improvement is the reduction in global synchronization that comes from computing the inner products in a single block. Letting $G_{k,s} = \mathcal{Y}_{k,s}^T \mathcal{Y}_{k,s}$, and using (5) and (4), α_{sk+j} and β_{sk+j} can be computed by

$$\begin{aligned} \alpha_{sk+j} &= \left(r'^T_{k,j} G_{k,s} r'_{k,j}\right) / \left(p'^T_{k,j} G_{k,s} \mathcal{B}_{k,s} p'_{k,j}\right), \\ \beta_{sk+j} &= \left(r'^T_{k,j+1} G_{k,s} r'_{k,j+1}\right) / \left(r'^T_{k,j} G_{k,s} r'_{k,j}\right). \end{aligned}$$

The matrix $G_{k,s}$ can be computed with one global synchronization per outer loop iteration, which, in terms of asymptotic parallel latency, costs the same as a single inner product. As $\mathcal{B}_{k,s}$ and $G_{k,s}$ are dimension $(2s+1) \times (2s+1)$, α_{sk+j} and β_{sk+j} can be computed locally by each processor within the inner loop. The s-step method thus reduces the number of global synchronizations per iteration by a factor of $O(s)$ versus the classical CG method.

Algorithm 2 s-step conjugate gradient.

Input: $n \times n$ symmetric positive definite matrix A, length-n vector b, and initial approximation x_1 to $Ax = b$

Output: Approximate solution x_{sk+s+1} to $Ax = b$ with updated residual r_{sk+s+1}

$r_1 = b - Ax_1$, $p_1 = r_1$
for $k = 0, 1, \ldots,$ until convergence **do**
  Compute $\mathcal{Y}_{k,s} = [\mathcal{P}_{k,s}, \mathcal{R}_{k,s}]$ according to (2).
  Compute $G_{k,s} = \mathcal{Y}_{k,s}^T \mathcal{Y}_{k,s}$.
  Assemble $\mathcal{B}_{k,s}$ such that (4) holds.
  $p'_{k,1} = [1, 0_{1,2s}]^T$, $r'_{k,1} = [0_{1,s+1}, 1, 0_{1,s-1}]^T$, $x'_{k,1} = [0_{1,2s+1}]^T$
  for $j = 1$ to s **do**
    $\alpha_{sk+j} = (r'^T_{k,j} G_{k,s} r'_{k,j}) / (p'^T_{k,j} G_{k,s} \mathcal{B}_{k,s} p'_{k,j})$
    $q'_{k,j} = \alpha_{sk+j} p'_{k,j}$
    $x'_{k,j+1} = x'_{k,j} + q'_{k,j}$
    $r'_{k,j+1} = r'_{k,j} - \mathcal{B}_{k,s} q'_{k,j}$
    $\beta_{sk+j} = (r'^T_{k,j+1} G_{k,s} r'_{k,j+1}) / (r'^T_{k,j} G_{k,s} r'_{k,j})$
    $p'_{k,j+1} = r'_{k,j+1} + \beta_{sk+j} p'_{k,j}$
  end for
  Recover iterates $\{p_{sk+s+1}, r_{sk+s+1}, x_{sk+s+1}\}$ according to (5).
end for

3. Attainable accuracy in variable s-step conjugate gradient. In the s-step CG method (Algorithm 2), the Krylov subspace basis is computed in blocks of size s in each outer loop. This method can be generalized to allow the blocks to be of varying size; in other words, a different s value can be used in each outer loop iteration. We denote the block size in outer iteration k by s_k, where $1 \le s_k \le \sigma$ for some maximum value σ. We call this a *variable s-step Krylov subspace method*. A generic variable s-step CG method is shown in Algorithm 3.

Our goal is to determine a suitable sequence of s_k values such that a user-specified accuracy is attainable. Writing the true residual $b - A\widehat{x}_i = b - A\widehat{x}_i - \widehat{r}_i + \widehat{r}_i$, we can write the bound

$$\|b - A\widehat{x}_i\| \le \|b - A\widehat{x}_i - \widehat{r}_i\| + \|\widehat{r}_i\|.$$

Then if $\|\widehat{r}_i\| \to 0$ (i.e., as the method converges), we have $\|b - A\widehat{x}_i\| \approx \|b - A\widehat{x}_i - \widehat{r}_i\|$. The size of the *residual gap*, i.e., $\|b - A\widehat{x}_i - \widehat{r}_i\|$, thus determines the attainable accuracy of the method.

We therefore seek to bound the growth of the residual gap in each outer loop iteration k of the variable s-step CG method (Algorithm 3). Our analysis is based on the maximum attainable accuracy analysis for fixed s-step CG in [5], which shows that the attainable accuracy of fixed s-step CG depends on the condition numbers of the s-step basis matrices $\mathcal{Y}_{k,s}$ computed in each outer loop. We begin by considering the rounding errors made in the variable s-step CG method (Algorithm 3).

In the remainder of the paper, hats denote quantities computed in finite precision, ε denotes the unit roundoff, and $\|\cdot\|$ denotes the 2-norm. We let N_A denote the maximum number of nonzeros per row in A, and define $\nu = \||A|\|/\|A\|$, $t_k = \sqrt{2s_k + 1}$, and $m = \sum_{i=0}^{k-1} s_i$. The quantity $\kappa(Y)$ denotes the condition number $\|Y^+\|\|Y\|$, where $Y^+ = (Y^T Y)^{-1} Y^T$ is the Moore–Penrose pseudoinverse. Note that in the case that Y is square and invertible, $\kappa(Y) = \|Y^{-1}\|\|Y\|$. To simplify the analysis, we drop all $O(\varepsilon^2)$ terms (and higher order terms in ε).

Algorithm 3 Variable s-step conjugate gradient.

Input: $n \times n$ symmetric positive definite matrix A, length-n vector b, initial approximation x_1 to $Ax = b$, sequence of s values $(s_0, s_1, \ldots)$

Output: Approximate solution x_{m+s_k+1} to $Ax = b$ with updated residual r_{m+s_k+1}

$r_1 = b - Ax_1$, $p_1 = r_1$
$m = 0$
for $k = 0, 1, \ldots,$ until convergence **do**
  Compute $\mathcal{Y}_{k,s_k} = [\mathcal{P}_{k,s_k}, \mathcal{R}_{k,s_k}]$ according to (2).
  Compute $G_{k,s_k} = \mathcal{Y}_{k,s_k}^T \mathcal{Y}_{k,s_k}$.
  Assemble $\mathcal{B}_{k,s_k}$ such that (4) holds.
  $p'_{k,1} = [1, 0_{1,2s_k}]^T$, $r'_{k,1} = [0_{1,s_k+1}, 1, 0_{1,s_k-1}]^T$, $x'_{k,1} = [0_{1,2s_k+1}]^T$
  for $j = 1$ to s_k **do**
    $\alpha_{m+j} = (r'^T_{k,j} G_{k,s_k} r'_{k,j})/(p'^T_{k,j} G_{k,s_k} \mathcal{B}_{k,s_k} p'_{k,j})$
    $q'_{k,j} = \alpha_{m+j} p'_{k,j}$
    $x'_{k,j+1} = x'_{k,j} + q'_{k,j}$
    $r'_{k,j+1} = r'_{k,j} - \mathcal{B}_{k,s_k} q'_{k,j}$
    $\beta_{m+j} = (r'^T_{k,j+1} G_{k,s_k} r'_{k,j+1})/(r'^T_{k,j} G_{k,s_k} r'_{k,j})$
    $p'_{k,j+1} = r'_{k,j+1} + \beta_{m+j} p'_{k,j}$
  end for
  Recover iterates $\{p_{m+s_k+1}, r_{m+s_k+1}, x_{m+s_k+1}\}$ according to (5).
  $m = m + s_k$
end for

In outer loop k and inner loops $j \in \{1, \ldots, s_k\}$ of finite precision variable s-step CG, using standard models of floating point error (e.g., [15, sect. 2.4]) we have

$$\text{(6)} \qquad A\underline{\hat{\mathcal{Y}}}_{k,s_k} = \widehat{\mathcal{Y}}_{k,s_k} \mathcal{B}_{k,s_k} - E_k,$$
$$\text{where} \quad \|E_k\| \le \varepsilon \left((3 + N_A)\nu t_k \|A\| \|\underline{\hat{\mathcal{Y}}}_{k,s_k}\| + 4t_k \|\widehat{\mathcal{Y}}_{k,s_k}\| \||\mathcal{B}_{k,s_k}|\| \right),$$

$$\text{(7)} \qquad \widehat{x}'_{k,j+1} = \widehat{x}'_{k,j} + \widehat{q}'_{k,j} + \xi_{k,j+1},$$
$$\text{where} \quad \|\xi_{k,j+1}\| \le \varepsilon \|\widehat{x}'_{k,j+1}\|,$$

$$\text{(8)} \qquad \widehat{r}'_{k,j+1} = \widehat{r}'_{k,j} - \mathcal{B}_{k,s_k} \widehat{q}'_{k,j} + \eta_{k,j+1},$$
$$\text{where} \quad \|\eta_{k,j+1}\| \le \varepsilon (\|\widehat{r}'_{k,j+1}\| + 3\||\mathcal{B}_{k,s_k}|\| \|\widehat{q}'_{k,j}\|),$$

$$\text{(9)} \qquad \widehat{x}_{m+j+1} = \widehat{\mathcal{Y}}_{k,s_k} \widehat{x}'_{k,j+1} + \widehat{x}_{m+1} + \phi_{m+j+1},$$
$$\text{where} \quad \|\phi_{m+j+1}\| \le \varepsilon (\|\widehat{x}_{m+j+1}\| + t_k^3 \|\widehat{\mathcal{Y}}_{k,s_k}\| \|\widehat{x}'_{k,j+1}\|), \text{ and}$$

$$\text{(10)} \qquad \widehat{r}_{m+j+1} = \widehat{\mathcal{Y}}_{k,s_k} \widehat{r}'_{k,j+1} + \psi_{m+j+1},$$
$$\text{where} \quad \|\psi_{m+j+1}\| \le t_k^3 \varepsilon \|\widehat{\mathcal{Y}}_{k,s_k}\| \|\widehat{r}'_{k,j+1}\|.$$

Let $\delta_i \equiv b - A\widehat{x}_i - \widehat{r}_i$ denote the residual gap in iteration i. Similar to the analysis in [5], for variable s-step CG we can write the growth of the residual gap in iteration $m + j + 1$ in outer loop k as

(11)
$$\delta_{m+j+1} - \delta_{m+1} = -\sum_{i=1}^{j} \left(A\underline{\hat{\mathcal{Y}}}_{k,s_k} \xi_{k,i+1} + \widehat{\mathcal{Y}}_{k,s_k} \eta_{k,i+1} - E_k \widehat{q}'_{k,i} \right) - A\phi_{m+j+1} - \psi_{m+j+1}.$$

Our goal is now to manipulate a bound on this quantity in order to derive a

method for determining the largest s_k value allowable such that the desired accuracy remains attainable. Using (6)–(10), we have the normwise bound

$$\begin{aligned}\|\delta_{m+j+1}-\delta_{m+1}\| &\le \varepsilon\|A\|\|\underline{\widehat{\mathcal{Y}}}_{k,s_k}\|\sum_{i=1}^{j}\|\widehat{x}'_{k,i+1}\| + \varepsilon\|\widehat{\mathcal{Y}}_{k,s_k}\|\sum_{i=1}^{j}\|\widehat{r}'_{k,i+1}\| \\ &\quad + \varepsilon\left[(3+N_A)\nu t_k\|A\|\|\underline{\widehat{\mathcal{Y}}}_{k,s_k}\| + (3+4t_k)\|\widehat{\mathcal{Y}}_{k,s_k}\|\||\mathcal{B}_{k,s_k}|\|\right]\sum_{i=1}^{j}\|\widehat{q}'_{k,i}\| \\ &\quad + \varepsilon\|A\|\|\widehat{x}_{m+j+1}\| + \varepsilon t_k^3\|A\|\|\widehat{\mathcal{Y}}_{k,s_k}\|\|\widehat{x}'_{k,j+1}\| + \varepsilon t_k^3\|\widehat{\mathcal{Y}}_{k,s_k}\|\|\widehat{r}'_{k,j+1}\|.\end{aligned}$$

To simplify the notation, we let $\tau_k = \||\mathcal{B}_{k,s_k}|\|/\|A\|$. We note that $\||\mathcal{B}_{k,s_k}|\|$ depends on the polynomial basis used and is not expected to be too large; e.g., for the monomial basis, $\||\mathcal{B}_{k,s_k}|\| = 1$. Using $\sum_{i=1}^{j}\|\widehat{q}'_{k,i}\| \le (2+\varepsilon)\sum_{i=1}^{j}\|\widehat{x}'_{k,i+1}\|$, the above bound can be written as

$$\begin{aligned}\|\delta_{m+j+1}-\delta_{m+1}\| &\le \varepsilon\left[2(3+N_A)\nu t_k + (6+8t_k)\tau_k + 1\right]\|A\|\|\widehat{\mathcal{Y}}_{k,s_k}\|\sum_{i=1}^{j}\|\widehat{x}'_{k,i+1}\| \\ &\quad + \varepsilon\|\widehat{\mathcal{Y}}_{k,s_k}\|\sum_{i=1}^{j}\|\widehat{r}'_{k,i+1}\| + \varepsilon t_k^3\|A\|\|\widehat{\mathcal{Y}}_{k,s_k}\|\|\widehat{x}'_{k,j+1}\| \\ &\quad + \varepsilon t_k^3\|\widehat{\mathcal{Y}}_{k,s_k}\|\|\widehat{r}'_{k,j+1}\| + \varepsilon\|A\|\|\widehat{x}_{m+j+1}\| \\ &\le \varepsilon\left[2(3+N_A)\nu t_k + (6+8t_k)\tau_k + t_k^3 + 1\right]\|A\|\|\widehat{\mathcal{Y}}_{k,s_k}\|\sum_{i=1}^{j}\|\widehat{x}'_{k,i+1}\| \\ &\quad + \varepsilon(1+t_k^3)\|\widehat{\mathcal{Y}}_{k,s_k}\|\sum_{i=1}^{j}\|\widehat{r}'_{k,i+1}\| + \varepsilon\|A\|\|\widehat{x}_{m+j+1}\|. \qquad (12)\end{aligned}$$

Now, assuming $\widehat{\mathcal{Y}}_{k,s_k}$ is full rank, we have

$$\|\widehat{\mathcal{Y}}_{k,s_k}\|\|\widehat{r}'_{k,i+1}\| = \|\widehat{\mathcal{Y}}_{k,s_k}\|\|\widehat{\mathcal{Y}}^{+}_{k,s_k}\widehat{\mathcal{Y}}_{k,s_k}\widehat{r}'_{k,i+1}\| \le \kappa(\widehat{\mathcal{Y}}_{k,s_k})\|\widehat{r}_{m+i+1}\| + O(\varepsilon), \qquad (13)$$

$$\|\widehat{\mathcal{Y}}_{k,s_k}\|\|\widehat{x}'_{k,i+1}\| = \|\widehat{\mathcal{Y}}_{k,s_k}\|\|\widehat{\mathcal{Y}}^{+}_{k,s_k}\widehat{\mathcal{Y}}_{k,s_k}\widehat{x}'_{k,i+1}\| \le \kappa(\widehat{\mathcal{Y}}_{k,s_k})\|\widehat{\mathcal{Y}}_{k,s_k}\widehat{x}'_{k,i+1}\|. \qquad (14)$$

To bound $\|\widehat{\mathcal{Y}}_{k,s_k}\widehat{x}'_{k,i+1}\|$ in terms of the size of the residuals, notice that by (9),

$$\begin{aligned}\widehat{\mathcal{Y}}_{k,s_k}\widehat{x}'_{k,i+1} &= \widehat{x}_{m+i+1} - \widehat{x}_{m+1} - \phi_{m+i+1} \\ &= (x-\widehat{x}_{m+1}) - (x-\widehat{x}_{m+i+1}) - \phi_{m+i+1} \\ &= A^{-1}(b-A\widehat{x}_{m+1}) - A^{-1}(b-A\widehat{x}_{m+i+1}) - \phi_{m+i+1} \\ &= A^{-1}(\widehat{r}_{m+1}+\delta_{m+1}) - A^{-1}(\widehat{r}_{m+i+1}+\delta_{m+i+1}) - \phi_{m+i+1} \\ &= A^{-1}(\widehat{r}_{m+1}-\widehat{r}_{m+i+1}) + A^{-1}(\delta_{m+1}-\delta_{m+i+1}) - \phi_{m+i+1},\end{aligned}$$

so

$$\|\widehat{\mathcal{Y}}_{k,s_k}\widehat{x}'_{k,i+1}\| \le \|A^{-1}\|(\|\widehat{r}_{m+1}\| + \|\widehat{r}_{m+i+1}\|) + O(\varepsilon).$$

Thus together with (14) we can write

$$\|\widehat{\mathcal{Y}}_{k,s_k}\|\|\widehat{x}'_{k,i+1}\| \le \kappa(\widehat{\mathcal{Y}}_{k,s_k})\|A^{-1}\|(\|\widehat{r}_{m+1}\| + \|\widehat{r}_{m+i+1}\|) + O(\varepsilon). \qquad (15)$$

Using $x - \widehat{x}_{m+j+1} = A^{-1}\widehat{r}_{m+j+1} + \delta_{m+j+1}$, we have

$$\|\widehat{x}_{m+j+1} - x\| \leq \|A^{-1}\|\|\widehat{r}_{m+j+1}\| + O(\varepsilon).$$

Then writing $\widehat{x}_{m+j+1} = \widehat{x}_{m+j+1} - x + x$ and using (13) and (15), we can write the bound (12) in the form

$$\|\delta_{m+j+1} - \delta_{m+1}\| \leq c_k \varepsilon \kappa(\widehat{\mathcal{Y}}_{k,s_k}) \left(\max_{\ell \in \{0,\ldots,j\}} \|\widehat{r}_{m+\ell+1}\| \right) + \varepsilon \|A\|\|x\|, \tag{16}$$

where $c_k = 2t[2(3+N_A)\nu t_k + (6+8t_k)\tau_k + 2t_k^3 + 3]\kappa(A)$. This gives a normwise bound on the growth of the residual gap due to finite precision errors made in outer iteration k.

Letting $m_\ell = \sum_{i=0}^{\ell-1} s_i$, notice that we have

$$\|\delta_{m_k+s_k+1}\| \leq \|\delta_1\| + \sum_{\ell=0}^{k} \|\delta_{m_\ell+s_\ell+1} - \delta_{m_\ell+1}\|.$$

Suppose that at the end of $k_{\max}$ outer iterations we want the size of the residual gap to be on the order of $k_{\max}\varepsilon^*$ (where $\varepsilon^* \gtrsim O(\varepsilon)\|A\|\|x\|$, as this is the best we could expect). This can be accomplished by requiring that in each outer loop k, for $j \in \{1, \ldots, s_k\}$,

$$\|\delta_{m_k+j+1} - \delta_{m_k+1}\| \leq \varepsilon^*.$$

From (16), this means that we must have, for all inner iterations $j \in \{1, \ldots, s_k\}$,

$$\kappa(\widehat{\mathcal{Y}}_{k,s_k}) \leq \frac{\varepsilon^*}{c_k \varepsilon \max_{\ell \in \{0,\ldots,j\}} \|\widehat{r}_{m+\ell+1}\|}. \tag{17}$$

The bound (17) tells us that as the 2-norm of the residual decreases, we can tolerate a more ill-conditioned basis matrix $\widehat{\mathcal{Y}}_{k,s_k}$ without detriment to the attainable accuracy. Since $\kappa(\widehat{\mathcal{Y}}_{k,s_k})$ grows with increasing s_k, this suggests that s_k can be allowed to increase as the method converges at a rate proportional to the inverse of the residual 2-norm. This naturally gives a relaxation strategy that is somewhat analogous to those derived for inexact Krylov subspace methods (see references in section 1).

4. The adaptive *s*-step conjugate gradient method. In this section, we describe a variant of the variable s-step CG method that makes use of the constraint (17) in determining s_k in each outer loop k, which we call the *adaptive* s-step CG method. In each outer loop k, our method aims to select the largest s_k value possible such that (17) is satisfied, up to some maximum value σ.

The bound (17) indicates that when $\widehat{\mathcal{Y}}_{k,s_k}$ is constructed at the beginning of outer loop k, we should enforce

$$\kappa(\widehat{\mathcal{Y}}_{k,s_k}) \leq \frac{\varepsilon^*}{c_k \varepsilon \|\widehat{r}_{m+1}\|}. \tag{18}$$

Therefore we first construct an $\bar{s}_k$-step basis matrix $\widehat{\mathcal{Y}}_{k,\bar{s}_k}$ using some $\bar{s}_k \leq \sigma$, compute the Gram matrix $\widehat{G}_{k,\bar{s}_k} = \widehat{\mathcal{Y}}_{k,\bar{s}_k}^T \widehat{\mathcal{Y}}_{k,\bar{s}_k}$, and then use the condition numbers of the leading principal submatrices of the blocks of $\widehat{G}_{k,\bar{s}_k}$ to estimate the condition numbers

of the leading columns of the blocks of $\widehat{\mathcal{Y}}_{k,\bar{s}_k}$. To be concrete, for $i \in \{1, 2, \ldots, \bar{s}_k\}$, we have (using MATLAB notation)

$$\widehat{\mathcal{Y}}_{k,i} = \left[\widehat{\mathcal{Y}}_{k,\bar{s}_k}(:, 1{:}i{+}1), \widehat{\mathcal{Y}}_{k,\bar{s}_k}(:, i_s{:}i_s{+}i{-}1)\right]$$

and

$$\widehat{G}_{k,i} = \begin{bmatrix} \widehat{G}_{k,\bar{s}_k}(1{:}i{+}1, 1{:}i{+}1) & \widehat{G}_{k,\bar{s}_k}(1{:}i{+}1, i_s{:}i_s{+}i{-}1) \\ \widehat{G}_{k,\bar{s}_k}(i_s{:}i_s{+}i{-}1, 1{:}i{+}1) & \widehat{G}_{k,\bar{s}_k}(i_s{:}i_s{+}i{-}1, i_s{:}i_s{+}i{-}1) \end{bmatrix},$$

where $i_s = \bar{s}_k{+}2$. Then since $\sqrt{\kappa(\widehat{G}_{k,i})} \approx \kappa(\widehat{\mathcal{Y}}_{k,i})$, we can easily estimate the condition number of the i-step basis matrix $\widehat{\mathcal{Y}}_{k,i}$, i.e., the basis matrix needed to perform i inner loop iterations, for $i \in \{1, \ldots, \bar{s}_k\}$. We therefore let $\tilde{s}_k \in \{1, \ldots, \bar{s}_k\}$ be the largest value possible such that

$$\sqrt{\kappa(\widehat{G}_{k,\tilde{s}_k})} \leq \frac{\varepsilon^*}{c_k \varepsilon \|\widehat{r}_{m+1}\|} \tag{19}$$

holds.

A complication is that in CG, it is the A-norm of the error rather than the 2-norm of the residual that is minimized. So in some inner loop iteration j, we may have $\|\widehat{r}_{m+j+1}\| > \|\widehat{r}_{m+1}\|$. Since $\|\widehat{r}_{m+i+1}\|$ for $i > 1$ are not known at the start of the outer loop iteration, we must determine s_k online, i.e., during the iterations. We therefore begin each outer loop under the assumption that $\|\widehat{r}_{m+i+1}\| \leq \|\widehat{r}_{m+1}\|$ for $i \in \{1, \ldots, \tilde{s}_k\}$. To handle the case where $\|\widehat{r}_{m+j+1}\| > \|\widehat{r}_{m+1}\|$ for some inner iteration j, we add a check within the inner loop iterations. After computing $\widehat{r}'_{k,j+1}$, we check whether

$$\kappa(\widehat{\mathcal{Y}}_{k,\tilde{s}_k}) \leq \frac{\varepsilon^*}{c_k \varepsilon \|\widehat{r}_{m+j+1}\|} \approx \frac{\varepsilon^*}{c_k \varepsilon \left(\widehat{r}'^T_{k,j+1} \widehat{G}_{k,\tilde{s}_k} \widehat{r}'_{k,j+1}\right)^{1/2}} \tag{20}$$

is still satisfied. Note that since $\|\widehat{r}_{m+j+1}\| = (\widehat{r}'^T_{k,j+1} \widehat{G}_{k,\tilde{s}_k} \widehat{r}'_{k,j+1})^{1/2} + O(\varepsilon)$, performing this check can be done inexpensively. If (20) is satisfied for all $j \in \{1, \ldots, \tilde{s}_k\}$, we will perform all $s_k = \tilde{s}_k$ inner loop iterations. If this is not satisfied for some j, we break from the current inner loop iteration and begin the next outer loop; i.e., we perform only $s_k = j$ inner loop iterations in outer loop k.

We note that since a variable s-step CG method with $s_k = 1$ for all outer loops k will produce the same iterates as classical CG, we expect that our adaptive s-step CG method can attain accuracy ε^* when $\varepsilon^* \geq \varepsilon_{\text{CG}}$, where ε_{CG} is the attainable accuracy of classical CG. When the right-hand side of (17) is less than one (due to, e.g., ε^* being set too small), the inequality (17) cannot be satisfied. In this case, we default to using $s_k = 1$ (i.e., we perform an iteration of classical CG).

Although performing the check for determining whether (20) is satisfied incurs minimal additional cost, there is potential wasted effort if $\bar{s}_k > \tilde{s}_k$ and/or $\tilde{s}_k > s_k$, since in these cases we have computed more basis vectors than we will use. One way to mitigate this in an implementation is to only allow $\bar{s}_k$ to grow by at most f vectors in each outer loop, i.e., $\bar{s}_k \leq s_{k-1} + f \leq \sigma$. The best choice for the parameter f will depend on the particular problem.

The described adaptive s-step CG method is shown in Algorithm 4. Note that in Algorithm 4, we have specified that the function c_k can be input by the user. Because our bounds are not tight, in many cases using c_k from the bound (17) overestimates the error, which results in the use of smaller s_k values than necessary.

Algorithm 4 Adaptive s-step conjugate gradient.

Input: $n \times n$ symmetric positive definite matrix A, length-n vector b, initial approximation x_1 to $Ax = b$, maximum s value σ, initial s value $\bar{s}_0$, maximum basis growth factor f, desired convergence tolerance ε^*, function c_k

Output: Approximate solution x_{m+s_k+1} to $Ax = b$ with updated residual r_{m+s_k+1}

$r_1 = b - Ax_1$, $p_1 = r_1$
$m = 0$
for $k = 0, 1, \ldots,$ until convergence **do**
  if $k \neq 0$ **then**
    $\bar{s}_k = \min(s_{k-1} + f, \sigma)$
  end if
  Compute $\bar{s}_k$-step basis matrix $\mathcal{Y}_{k,\bar{s}_k} = [\mathcal{P}_{k,\bar{s}_k}, \mathcal{R}_{k,\bar{s}_k}]$ according to (2).
  Compute $G_{k,\bar{s}_k} = \mathcal{Y}_{k,\bar{s}_k}^T \mathcal{Y}_{k,\bar{s}_k}$.
  Determine $\tilde{s}_k$ by (19); assemble $\mathcal{Y}_{k,\tilde{s}_k}$ and $G_{k,\tilde{s}_k}$.
  Store estimate $\gamma \approx \kappa(\widehat{\mathcal{Y}}_{k,\tilde{s}_k})$.
  Assemble $\mathcal{B}_{k,\tilde{s}_k}$ such that (4) holds.
  $p'_{k,1} = [1, 0_{1,2\tilde{s}_k}]^T$, $r'_{k,1} = [0_{1,\tilde{s}_k+1}, 1, 0_{1,\tilde{s}_k-1}]^T$, $x'_{k,1} = [0_{1,2\tilde{s}_k+1}]^T$
  for $j = 1$ to $\tilde{s}_k$ **do**
    $s_k = j$
    $\alpha_{m+j} = (r'^T_{k,j} G_{k,\tilde{s}_k} r'_{k,j}) / (p'^T_{k,j} G_{k,\tilde{s}_k} \mathcal{B}_{k,\tilde{s}_k} p'_{k,j})$
    $q'_{k,j} = \alpha_{m+j} p'_{k,j}$
    $x'_{k,j+1} = x'_{k,j} + q'_{k,j}$
    $r'_{k,j+1} = r'_{k,j} - \mathcal{B}_{k,\tilde{s}_k} q'_{k,j}$
    $\beta_{m+j} = (r'^T_{k,j+1} G_{k,\tilde{s}_k} r'_{k,j+1}) / (r'^T_{k,j} G_{k,\tilde{s}_k} r'_{k,j})$
    $p'_{k,j+1} = r'_{k,j+1} + \beta_{m+j} p'_{k,j}$
    if $\gamma \geq \frac{\varepsilon^*}{c_k \varepsilon \left(\widehat{r}'^T_{k,j+1} G_{k,\tilde{s}_k} \widehat{r}'_{k,j+1}\right)^{1/2}}$ **then**
      Break from inner loop.
    end if
  end for
  Recover iterates $\{p_{m+s_k+1}, r_{m+s_k+1}, x_{m+s_k+1}\}$ according to (5).
  $m = m + s_k$
end for

5. Numerical experiments. In this section, we test the numerical behavior of the adaptive s-step CG method (Algorithm 4) on a number of small examples from the University of Florida Sparse Matrix Collection [9] (now known as the SuiteSparse Matrix Collection) using MATLAB. The test examples were chosen to exemplify various convergence trajectories of CG, which allows us to demonstrate the validity of our adaptive approach in which s_k depends on the rate of convergence. The test problems used here are small, so we would not use (fixed or variable) s-step methods here in practice; nevertheless, they still serve to exhibit the numerical behavior of the methods. An optimized high-performance implementation is outside the scope of the present work.

All tests were run in double precision, i.e., $\varepsilon \approx 2^{-53}$, and use a right-hand side b with entries $1/\sqrt{n}$, where n is the dimension of A. We use matrix equilibration, i.e., $A \leftarrow D^{-1/2} A D^{-1/2}$, where D is a diagonal matrix of the largest entries in each row of A. Properties of the test matrices (postequilibration) are shown in Table 1.

TABLE 1
Test matrix properties.

Matrix	n	nnz	$\|A\|$	$\kappa(A)$
gr_30_30	900	7744	1.5	1.9e2
mesh3e1	289	1377	1.8	8.6e0
nos6	675	3255	2.0	3.5e6
bcsstk09	1083	18437	2.0	1.0e4
ex5	27	279	3.8	5.7e7

Our results are plotted in Figures 1−5. For each test matrix, we ran classical CG (Algorithm 1), as well as both fixed s-step CG (Algorithm 2) and adaptive s-step CG (Algorithm 4) for σ values 4, 8, and 10 (for fixed s-step CG, $s = \sigma$). For each matrix and σ value, we tested two different ε^* values: $\varepsilon^* = \varepsilon_{\text{CG}}$, the maximum attainable accuracy of classical CG in double precision, and $\varepsilon^* = 10^{-6}$. The monomial basis was used for both variable and fixed s-step methods in all tests, although there is no technical restriction to the monomial basis; the analysis used in deriving the adaptive s-step CG method still applies regardless of the polynomials used in constructing $\widehat{\mathcal{Y}}_{k,\bar{s}_k}$. In all tests for adaptive s-step CG, we set $\bar{s}_0 = f = \sigma$ (see Algorithm 4), as this gives the optimal choice of s_k in each iteration. We also tried using smaller values of f, e.g., $f = 1$ and $f = 2$; this did not significantly change the total number of outer loop iterations performed. We used $c_k = 1$ in our condition for determining the number of inner loop iterations to perform in all experiments except for tests with matrices bcsstk09 (Figure 4) and ex5 (Figure 5); for these problems, convergence is more irregular, and thus the bound (17) is tighter.

Each figure has a corresponding table (Tables 2−6) which lists the number of outer loop iterations required for convergence of the true residual 2-norm to level ε^*. For classical CG, the number reported gives the total number of iterations. Dashes in the tables indicate that the true residual diverged or stagnated before reaching level ε^*.

Note that our method of adaptively selecting s_k is based solely on the goal of meeting some specified accuracy requirement, and not on improving the convergence rate. Therefore there is no expectation that the adaptive s-step method will require fewer outer iterations than the fixed s-step approach or that the convergence rate will be close to that of the classical CG method.

5.1. Observations. In all experiments, the adaptive s-step CG method converged to the desired tolerance ε^*. This is not the case for the fixed s-step CG method, which failed to converge in some cases with larger s and $\varepsilon^* = \varepsilon_{\text{CG}}$. We point out that even for very well-conditioned matrices, choosing s too large can very negatively affect the convergence rate of fixed s-step CG; see, e.g., the case with $s = 10$ and $\varepsilon^* = 10^{-6}$ in Figure 2, where s-step CG requires more than eight times the number of outer loops iterations than classical CG.

In tests where both fixed s-step CG and adaptive s-step CG converge, adaptive s-step CG takes at most two additional outer iterations versus fixed s-step CG, although in most cases it takes as many or fewer outer iterations. We note that in all cases, variable s-step CG uses fewer outer loop iterations than classical CG.

The experiments using $\varepsilon^* = 10^{-6}$ demonstrate how the adaptive s-step CG method is able to automatically take the required accuracy into account. For example, in Figure 1 for gr_30_30, for all tests using $\varepsilon^* = 10^{-6}$, adaptive s-step CG uses $s_k = \sigma$ in each outer iteration. In other words, the fixed s-step CG method will produce a sufficiently accurate solution, so the adaptive s-step method behaves like the fixed s-step method.

FIG. 1. *Convergence of the 2-norm of the true residual for the matrix* gr_30_30. *Plots on the left use* $\varepsilon^* = 3.4\text{e-}14$, *and plots on the right use* $\varepsilon^* = 10^{-6}$. *Plots are shown for three* s *(or* σ*) values:* 4 *(top),* 8 *(middle), and* 10 *(bottom). Outer loop iterations are denoted with markers (red circles for s-step CG, blue stars for adaptive s-step CG, and black dots for classical CG). The horizontal dashed line shows the requested accuracy* ε^*. *(Color available online.)*

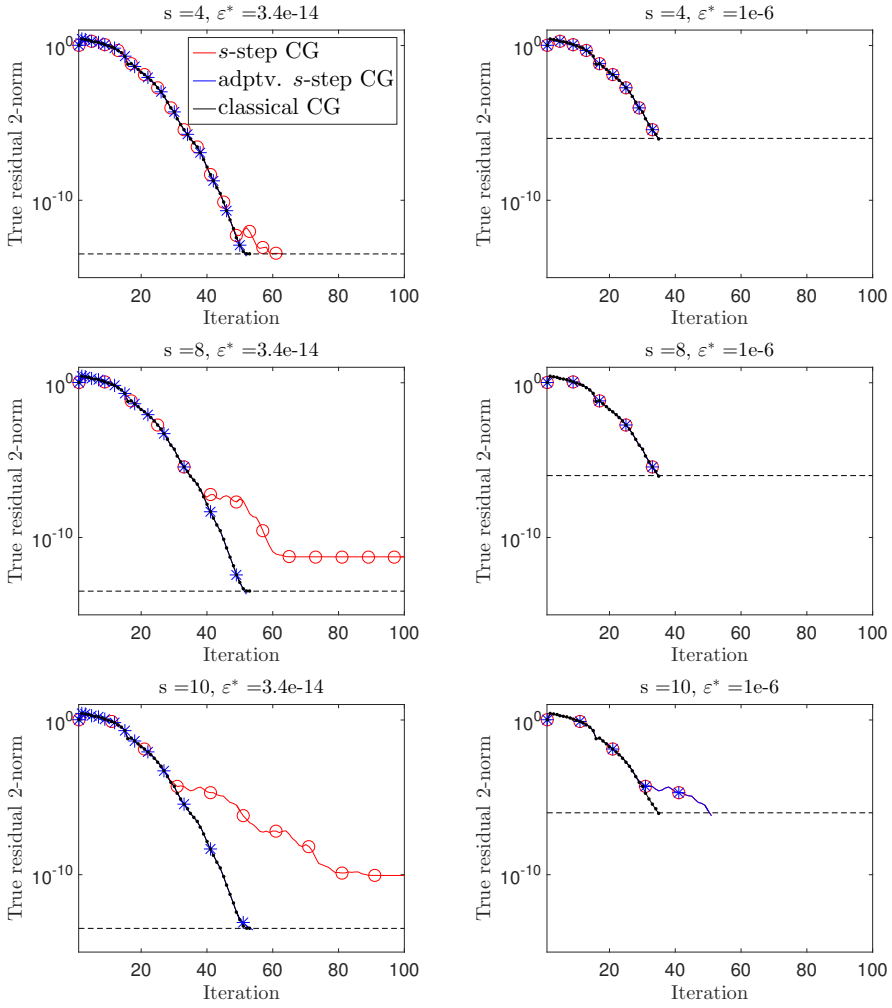

TABLE 2

Number of outer loop iterations in tests for matrix gr_30_30 *(corresponding to Figure* 1*). Dashes (–) in the table indicate that the method failed to converge to the desired level* ε^*.

		Fixed s-step CG	Adaptive s-step CG	Classical CG
$\varepsilon^* = 3.4\text{e-}14$	$s = 4$	16	17	52
	$s = 8$	–	14	
	$s = 10$	–	14	
$\varepsilon^* = 1\text{e-}6$	$s = 4$	9	9	34
	$s = 8$	5	5	
	$s = 10$	5	5	

FIG. 2. *Convergence of the* 2*-norm of the true residual for the matrix* mesh3e1. *Plots on the left use* $\varepsilon^* = 1.0\text{e-}14$, *and plots on the right use* $\varepsilon^* = 10^{-6}$. *Plots are shown for three* s *(or* σ*) values:* 4 *(top),* 8 *(middle), and* 10 *(bottom). Outer loop iterations are denoted with markers (red circles for* s*-step CG, blue stars for adaptive* s*-step CG, and black dots for classical CG). The horizontal dashed line shows the requested accuracy* ε^*. *(Color available online.)*

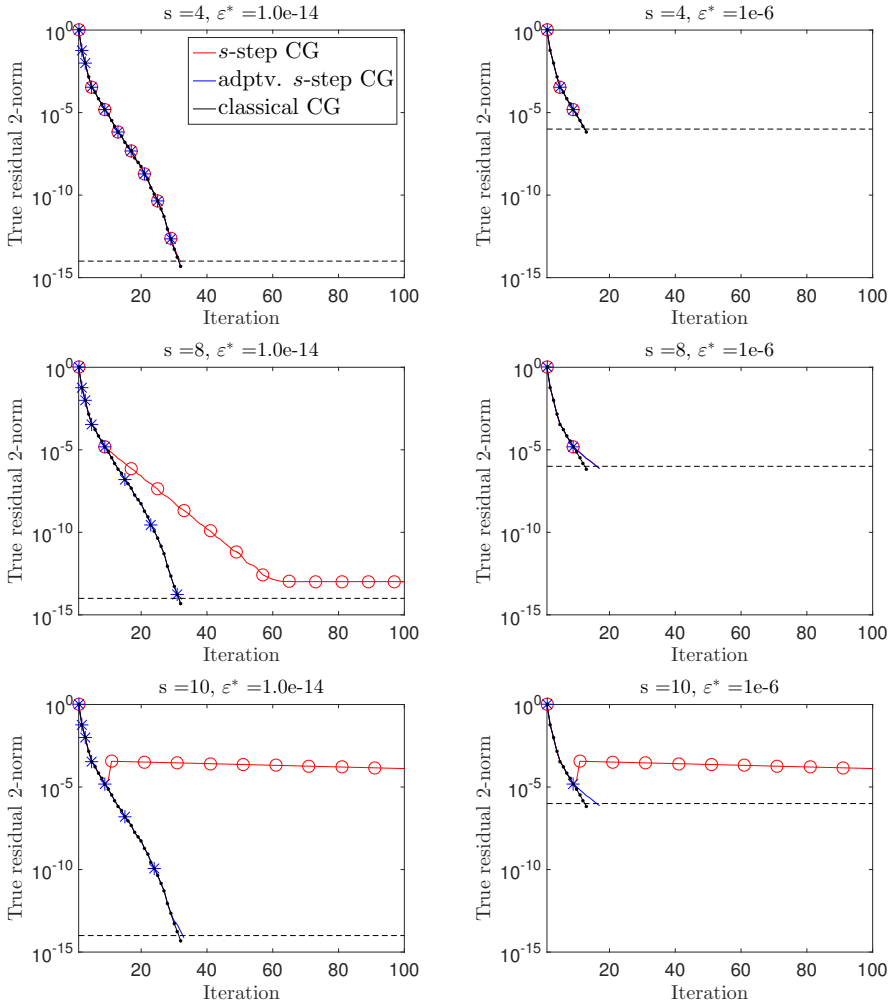

TABLE 3

Number of outer loop iterations in tests for matrix mesh3e1 *(corresponding to Figure* 2*). Dashes (–) in the table indicate that the method failed to converge to the desired level* ε^*.

		Fixed s-step CG	Adaptive s-step CG	Classical CG
$\varepsilon^* = 1\text{e-}14$	$s = 4$	8	10	31
	$s = 8$	–	8	
	$s = 10$	–	7	
$\varepsilon^* = 1\text{e-}6$	$s = 4$	3	3	12
	$s = 8$	2	2	
	$s = 10$	99	2	

FIG. 3. *Convergence of the 2-norm of the true residual for the matrix* nos6. *Plots on the left use* $\varepsilon^* = 5.5\text{e-}10$, *and plots on the right use* $\varepsilon^* = 10^{-6}$. *Plots are shown for three* s *(or* σ*) values:* 4 *(top),* 8 *(middle), and* 10 *(bottom). Outer loop iterations are denoted with markers (red circles for s-step CG, blue stars for adaptive s-step CG, and black dots for classical CG). The horizontal dashed line shows the requested accuracy* ε^*. *(Color available online.)*

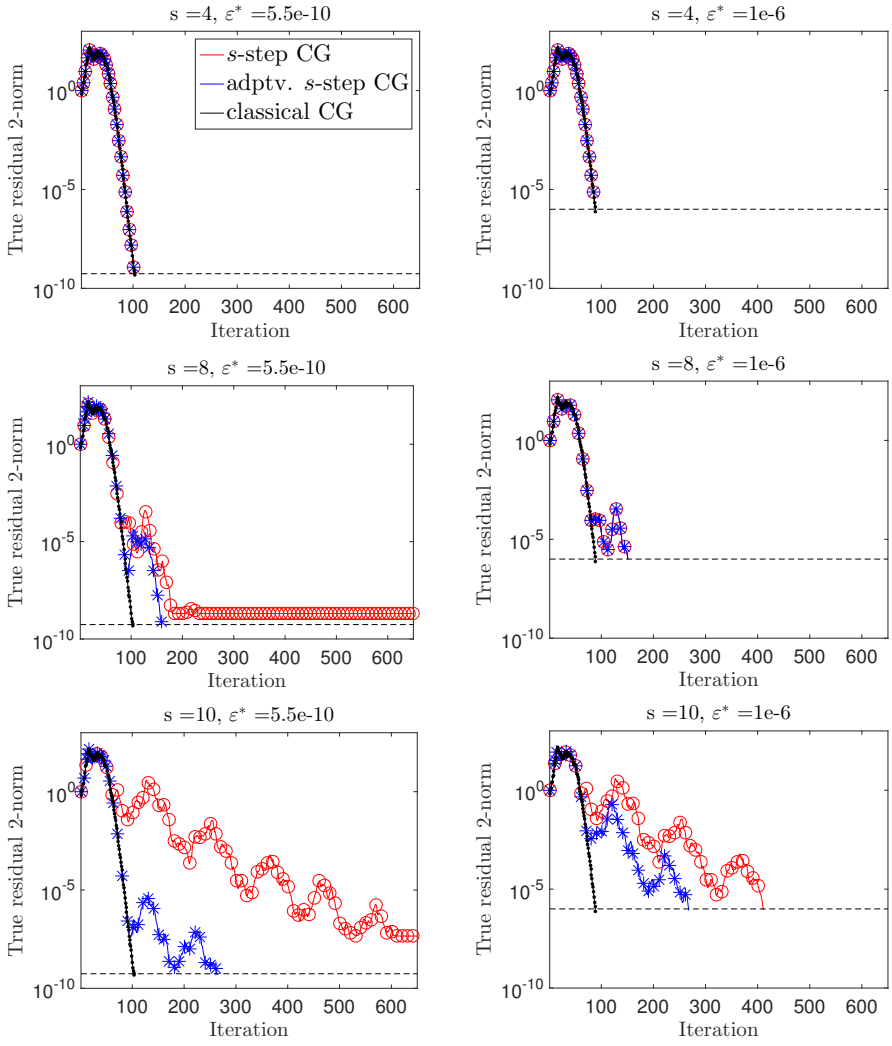

TABLE 4

Number of outer loop iterations in tests for matrix nos6 *(corresponding to Figure* 3*). Dashes (–) in the table indicate that the method failed to converge to the desired level* ε^*.

		Fixed s-step CG	Adaptive s-step CG	Classical CG
$\varepsilon^* = 5.5\text{e-}10$	$s = 4$	26	26	103
	$s = 8$	–	29	
	$s = 10$	–	36	
$\varepsilon^* = 1\text{e-}6$	$s = 4$	22	22	88
	$s = 8$	19	19	
	$s = 10$	41	29	

FIG. 4. *Convergence of the* 2*-norm of the true residual for the matrix* bcsstk09. *Plots on the left use* $\varepsilon^* =$ 1.6e-12, *and plots on the right use* $\varepsilon^* = 10^{-6}$. *Plots are shown for three* s *(or* σ*) values:* 4 *(top),* 8 *(middle), and* 10 *(bottom). Outer loop iterations are denoted with markers (red circles for* s*-step CG, blue stars for adaptive* s*-step CG, and black dots for classical CG). The horizontal dashed line shows the requested accuracy* ε^*. *(Color available online.)*

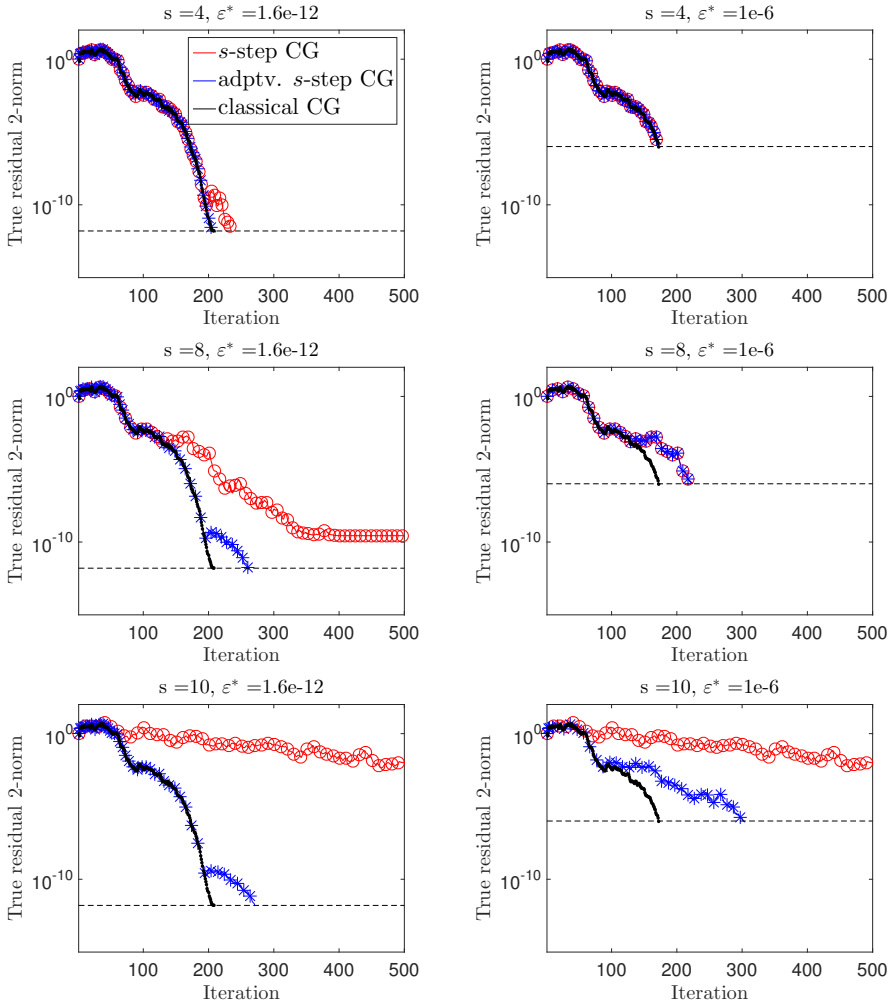

TABLE 5

Number of outer loop iterations in tests for matrix bcsstk09 *(corresponding to Figure* 4*). Dashes (–) in the table indicate that the method failed to converge to the desired level* ε^*. *Tests used* $c_k = 10$.

		Fixed s-step CG	Adaptive s-step CG	Classical CG
$\varepsilon^* =$ 1.6e-12	$s = 4$	59	59	207
	$s = 8$	–	52	
	$s = 10$	–	50	
$\varepsilon^* =$ 1e-6	$s = 4$	43	43	171
	$s = 8$	28	28	
	$s = 10$	122	33	

FIG. 5. *Convergence of the* 2*-norm of the true residual for the matrix* ex5*. Plots on the left use* $\varepsilon^* =$ 1e-8*, and plots on the right use* $\varepsilon^* = 10^{-6}$*. Plots are shown for three* s *(or* σ*) values:* 4 *(top),* 8 *(middle), and* 10 *(bottom). Outer loop iterations are denoted with markers (red circles for* s*-step CG, blue stars for adaptive* s*-step CG, and black dots for classical CG). The horizontal dashed line shows the requested accuracy* ε^**. (Color available online.)*

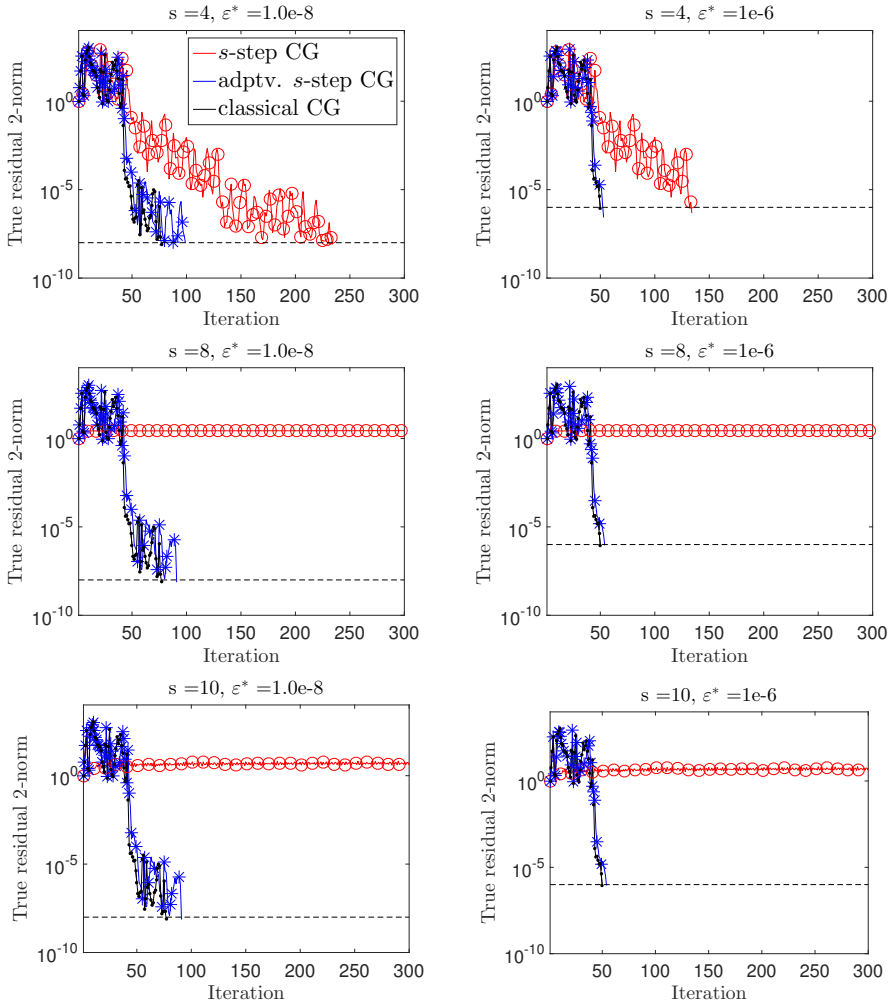

TABLE 6

Number of outer loop iterations in tests for matrix ex5 *(corresponding to Figure* 5*). Dashes (–) in the table indicate that the method failed to converge to the desired level* ε^**. Tests used* $c_k = \sqrt{\kappa(A)}$*.*

		Fixed s-step CG	Adaptive s-step CG	Classical CG
$\varepsilon^* =$ 1e-8	$s = 4$	59	48	76
	$s = 8$	–	45	
	$s = 10$	–	45	
$\varepsilon^* =$ 1e-6	$s = 4$	34	27	49
	$s = 8$	–	27	
	$s = 10$	–	27	

In contrast, in plots on the left using $\varepsilon^* = \varepsilon_{\text{CG}}$ we see that the adaptive s-step CG uses smaller s_k values at the beginning and then increases up to $s_k = \sigma$, which allows the method to attain accuracy ε_{CG}. For this problem the fixed s-step CG fails for $s = 8$ and $s = 10$.

Below we list the sequence of s_k values chosen in adaptive s-step CG for tests with $\varepsilon^* = \varepsilon_{\text{CG}}$ and $\sigma = 10$ (plots in the lower left in each figure):

- gr_30_30: 1, 1, 2, 2, 2, 3, 3, 3, 4, 5, 6, 8, 10, 10;
- mesh3e1: 1, 1, 2, 4, 6, 9, 10;
- nos6: 6, 1, 2, 3, 4, 5, 4, 4, 1, 4, 4, 5, 6, 7, 8, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10;
- bcsstk09: 2, 2, 3, 3, 3, 3, 3, 1, 2, 3, 3, 3, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 6, 7, 1, 6, 6, 7, 7, 7, 7, 8, 8, 8, 9, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10;
- ex5: 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 2, 1, 1, 1, 3, 3, 2, 1, 1, 1, 2, 1, 1, 2, 4, 6, 2, 4, 5, 7, 2, 6, 1, 7, 6.

There are a few things to notice. First, as expected, the sequence of s_k values increases monotonically when the residual 2-norm decreases monotonically or close to it (matrices gr_30_30 and mesh3e1), but when the residual 2-norm oscillates, we no longer have monotonicity of the s_k values (matrices nos6, bcsstk09, and ex5). Second, it may be that $s_k = \sigma$ is never reached, as is the case for matrix ex5. It is also worth pointing out that the sequence of s_k values need not start with $s_0 = 1$; for nos6, $s_0 = 6$, and for bcsstk09, $s_0 = 2$.

Looking at the number of outer loop iterations in Tables 2–6, we see that in our tests for adaptive s-step CG it is often the case that increasing the σ value past a certain point gives diminishing returns in terms of reducing the number of outer loop iterations. This may represent a fundamental limit of the monomial basis for these problems. It may be that higher s_k values are only feasible when better-conditioned polynomial bases are used for constructing $\widehat{\mathcal{Y}}_{k,\bar{s}_k}$ (e.g., Newton or Chebyshev bases).

6. Conclusions and future work. In this paper, we studied the attainable accuracy of a variable s-step CG method. By bounding the growth of the residual gap in outer iteration k, we derived a constraint on the condition number of the s-step basis matrix generated in outer loop k which must hold in order for a specified accuracy to be attained. In short, in outer iteration k, the condition number of the s-step basis matrix must be inversely proportional to the largest residual 2-norm encountered within that outer iteration. Based on our analysis, we propose the adaptive s-step CG method, a variable s-step CG method where the parameter s_k is determined automatically in each outer loop k such that a user-specified accuracy is attainable. Our numerical experiments demonstrate that the adaptive s-step CG method is able to attain up to the same accuracy as classical CG while still significantly reducing the number of outer loop iterations performed. In contrast, the fixed s-step CG method can fail to converge to this accuracy with fixed $s > 4$.

Although outside the scope of this work, a high-performance implementation and performance study is necessary in the future in order to compare the relative suitability of classical CG, fixed s-step CG, and adaptive s-step CG on large-scale problems. Other potential future work includes obtaining a tighter bound to replace (16), which could avoid the requirement of adjusting the function c_k (which represents a multiplicative factor in the bounds (16) and (17)) in Algorithm 4. As discussed in section 5, the adaptive s-step CG method is designed to maximize s_k subject to a constraint on accuracy and makes no guarantees about the resulting rate of convergence. Clearly, it would also be useful to optimize s_k to achieve the *overall* lowest cost. This would

require a better understanding of the finite precision convergence rate of both s-step and classical Krylov subspace methods, which is an active area of research.

For simplicity, we have restricted the analysis here to the unpreconditioned case. However, as preconditioning is considered essential in most practical applications of Krylov subspace methods, we briefly note that the presented bounds can be extended to the preconditioned case with a few modifications. We expect that the resulting constraint on $\kappa(\widehat{\mathcal{Y}}_{k,s_k})$ can be written in a form similar to (17), but note that in the preconditioned case the columns of $\widehat{\mathcal{Y}}_{k,s_k}$ are computed bases for Krylov subspaces with the preconditioned operator rather than A.

We believe that our adaptive approach can be extended to other s-step Krylov subspace methods as well. The same approach outlined here can be used in s-step BICG (see, e.g., [6]), which uses the same recurrences for the solution and updated residual. We conjecture that a similar approach can also be used in s-step variants of other recursively computed residual methods (see [16]), like s-step BICGSTAB [6], as well as methods like s-step GMRES. In the case of s-step GMRES, it may be possible to simplify the adaptive algorithm; because the residual norm is monotonically decreasing, the sequence of s_k values should monotonically increase. Such a sequence of s_k values was used in the variable s-step GMRES method of Imberti and Erhel [20].

As a final comment, we note that in our analysis in section 3, we used (16) to determine a bound on $\kappa(\widehat{\mathcal{Y}}_{k,s_k})$ that guarantees accuracy on the level ε^*. We could have just as easily used (16) to develop a mixed precision s-step CG method where s is fixed, but the precision used in computing in outer loop k varies. In other words, rearranging the inequality (16) we see that, given that we've computed an s-step basis with condition number $\kappa(\widehat{\mathcal{Y}}_{k,s})$, if we want to achieve accuracy ε^*, we must have that

$$\varepsilon_k \leq \frac{\varepsilon^*}{c_k \kappa(\widehat{\mathcal{Y}}_{k,s}) \max_{\ell \in \{0,\ldots,j\}} \|\widehat{r}_{m+\ell+1}\|}.$$

Investigating the potential for mixed precision s-step Krylov subspace methods based on this analysis is left to future work.

Acknowledgment. The author would like to thank the anonymous reviewers, whose suggestions and constructive feedback led to significant improvements in the manuscript.

REFERENCES

[1] Å. Björck, T. Elfving, and Z. Strakoš, *Stability of conjugate gradient and Lanczos methods for linear least squares problems*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 720–736, https://doi.org/10.1137/S089547989631202X.

[2] A. Bouras and V. Frayssé, *Inexact matrix-vector products in Krylov methods for solving linear systems: A relaxation strategy*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 660–678, https://doi.org/10.1137/S0895479801384743.

[3] A. Bouras, V. Frayssé, and L. Giraud, *A Relaxation Strategy for Inner-Outer Linear Solvers in Domain Decomposition Methods*, Technical Report CERFACS TR/PA/00/17, European Center for Research and Advanced Training in Scientific Computing, Toulouse, France, 2000.

[4] E. Carson, *Communication-Avoiding Krylov Subspace Methods in Theory and Practice*, Ph.D. thesis, Electrical Engineering and Computer Sciences Department, University of California at Berkeley, Berkeley, CA, 2015.

[5] E. Carson and J. Demmel, *A residual replacement strategy for improving the maximum attainable accuracy of s-step Krylov subspace methods*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 22–43, https://doi.org/10.1137/120893057.

[6] E. Carson, N. Knight, and J. Demmel, *Avoiding communication in nonsymmetric Lanczos-based Krylov subspace methods*, SIAM J. Sci. Comput., 35 (2013), pp. S42–S61, https://doi.org/10.1137/120881191.

[7] A. Chronopoulos and C. Gear, *s-step iterative methods for symmetric linear systems*, J. Comput. Appl. Math., 25 (1989), pp. 153–168.

[8] S. Cools, E. F. Yetkin, E. Agullo, L. Giraud, and W. Vanroose, *Analyzing the effect of local rounding error propagation on the maximal attainable accuracy of the pipelined conjugate gradient method*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 426–450, https://doi.org/10.1137/17M1117872.

[9] T. Davis and Y. Hu, *The University of Florida Sparse Matrix Collection*, ACM Trans. Math. Software, 38 (2011), pp. 1–25.

[10] J. Demmel, M. Hoemmen, M. Mohiyuddin, and K. Yelick, *Avoiding Communication in Computing Krylov Subspaces*, Technical Report UCB/EECS-2007-123, Electrical Engineering and Computer Sciences Department, University of California at Berkeley, Berkeley, CA, 2007.

[11] J. Dongarra, P. Beckman, T. Moore, P. Aerts, G. Aloisio, J.-C. Andre, D. Barkai, et al., *The International Exascale Software Project roadmap*, Int. J. High Perform. Comput. Appl., 25 (2011), pp. 3–60.

[12] P. Ghysels, T. J. Ashby, K. Meerbergen, and W. Vanroose, *Hiding global communication latency in the GMRES algorithm on massively parallel machines*, SIAM J. Sci. Comput., 35 (2013), pp. C48–C71, https://doi.org/10.1137/12086563X.

[13] P. Ghysels and W. Vanroose, *Hiding global synchronization latency in the preconditioned conjugate gradient algorithm*, Parallel Comput., 40 (2014), pp. 224–238.

[14] L. Giraud, S. Gratton, and J. Langou, *Convergence in backward error of relaxed GMRES*, SIAM J. Sci. Comput., 29 (2007), pp. 710–728, https://doi.org/10.1137/040608416.

[15] G. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[16] A. Greenbaum, *Estimating the attainable accuracy of recursively computed residual methods*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 535–551, https://doi.org/10.1137/S0895479895284944.

[17] M. H. Gutknecht and Z. Strakoš, *Accuracy of two three-term and three two-term recurrences for Krylov space solvers*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 213–229, https://doi.org/10.1137/S0895479897331862.

[18] M. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.

[19] M. Hoemmen, *Communication-Avoiding Krylov Subspace Methods*, Ph.D. thesis, Electrical Engineering and Computer Sciences Department, University of California at Berkeley, Berkeley, CA, 2010.

[20] D. Imberti and J. Erhel, *Varying the s in your s-step GMRES*, Electron. Trans. Numer. Anal., 47 (2017), pp. 206–230.

[21] M. Mehri Dehnavi, Y. El-Kurdi, J. Demmel, and D. Giannacopoulos, *Communication-avoiding Krylov techniques on graphic processing units*, IEEE Trans. Magn., 49 (2013), pp. 1749–1752.

[22] M. Mohiyuddin, M. Hoemmen, J. Demmel, and K. Yelick, *Minimizing communication in sparse matrix solvers*, in SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, ACM, New York, 2009, 36.

[23] B. Philippe and L. Reichel, *On the generation of Krylov subspace bases*, Appl. Numer. Math., 62 (2012), pp. 1171–1186.

[24] V. Simoncini and D. B. Szyld, *Theory of inexact Krylov subspace methods and applications to scientific computing*, SIAM J. Sci. Comput., 25 (2003), pp. 454–477, https://doi.org/10.1137/S1064827502406415.

[25] G. L. G. Sleijpen, H. A. van der Vorst, and D. R. Fokkema, *BiCGstab(l) and other hybrid Bi-CG methods*, Numer. Algorithms, 7 (1994), pp. 75–109.

[26] G. L. G. Sleijpen, H. A. van der Vorst, and J. Modersitzki, *Differences in the effects of rounding errors in Krylov solvers for symmetric indefinite linear systems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 726–751, https://doi.org/10.1137/S0895479897323087.

[27] J. van den Eshof and G. L. G. Sleijpen, *Inexact Krylov subspace methods for linear systems*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 125–153, https://doi.org/10.1137/S0895479802403459.

[28] H. A. van der Vorst and Q. Ye, *Residual replacement strategies for Krylov subspace iterative methods for the convergence of true residuals*, SIAM J. Sci. Comput., 22 (2000), pp. 835–852, https://doi.org/10.1137/S1064827599353865.

[29] S. Williams, M. Lijewski, A. Almgren, B. Van Straalen, E. Carson, N. Knight, and J. Demmel, *s-step Krylov subspace methods as bottom solvers for geometric multigrid*, in Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium, 2014, pp. 1149–1158.

[30] I. Yamazaki, H. Anzt, S. Tomov, M. Hoemmen, and J. Dongarra, *Improving the performance of CA-GMRES on multicores with multiple GPUs*, in Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium, 2014, pp. 382–391.

[31] I. Yamazaki, M. Hoemmen, P. Luszczek, and J. Dongarra, *Improving performance of GMRES by reducing communication and pipelining global collectives*, in Proceedings of the 2017 IEEE International Parallel and Distributed Processing Symposium Workshops, 2017, pp. 1118–1127.

[32] I. Yamazaki, S. Rajamanickam, E. Boman, M. Hoemmen, M. Heroux, and S. Tomov, *Domain decomposition preconditioners for communication-avoiding Krylov methods on a hybrid CPU/GPU cluster*, in SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE Press, Piscataway, NJ, 2014, pp. 933–944.

[33] I. Yamazaki and K. Wu, *A communication-avoiding thick-restart Lanczos method on a distributed-memory system*, in Euro-Par 2011: Parallel Processing Workshops, Springer, Berlin, Heidelberg, 2012, pp. 345–354.