Nečas Center for Mathematical Modeling

On the numerical stability analysis of pipelined Krylov subspace methods

E. C. Carson, M. Rozložník, Z. Strakoš, P. Tichý, M. Tůma

Preprint no. 2016-08

http://ncmm.karlin.mff.cuni.cz/

ON THE NUMERICAL STABILITY ANALYSIS OF PIPELINED KRYLOV SUBSPACE METHODS

ERIN C. CARSON *, MIROSLAV ROZLOŽNÍK†, ZDENĚK STRAKOŠ ‡, PETR TICHÝ§, AND MIROSLAV TŮMA¶

Abstract. Algebraic solvers based on preconditioned Krylov subspace methods are among the most powerful tools for large scale numerical computations in applied mathematics, sciences, technology, as well as in emerging applications in social sciences. The study of mathematical properties of Krylov subspace methods, in both the cases of exact and inexact computations, is a very active area of research and many issues in the analytic theory of Krylov subspace methods remain open. Numerical stability issues have been studied since the formulation of the conjugate gradient method in the middle of the last century, with many remarkable results achieved in the years since.

Inexact computations in Krylov subspace methods, either due to floating point roundoff error or intentional action motivated by savings in computing time or energy consumption, have two basic effects, namely, slowing down convergence and limiting attainable accuracy. Although the methodologies for their investigation are different, these phenomena are closely related and cannot be separated from one another.

As the name suggests, Krylov subspace methods can be viewed as a sequence of projections onto nested subspaces of increasing dimension. They are therefore by their nature implemented as synchronized recurrences. This is the fundamental obstacle to efficient parallel implementation. Standard approaches to overcoming this obstacle described in the literature involve reducing the number of global synchronization points and increasing parallelism in performing arithmetic operations within individual iterations. One such approach, employed by the so-called pipelined Krylov subspace methods, involves overlapping the global communication needed for computing inner products with local arithmetic computations.

Recently, the issues of attainable accuracy and delayed convergence caused by inexact computations became of interest in relation to pipelined Krylov subspace methods. In this contribution we recall the related early results and developments in synchronization-reducing Krylov subspace methods, identify the main factors determining possible numerical instabilities, and outline approaches needed for the analysis and understanding of pipelined Krylov subspace methods. We demonstrate the discussed issues numerically using several algorithmic variants of the conjugate gradient method. The paper concludes with a brief perspective on Krylov subspace methods in the forthcoming exascale era.

Key words. Krylov subspace methods, the conjugate gradient method, numerical stability, inexact computations, delay of convergence, maximal attainable accuracy, pipelined Krylov subspace methods, exascale computations.

1. Introduction. This paper considers the problem of using Krylov subspace methods in solving linear algebraic systems $Ax = b$, where A is a real $N \times N$ nonsingular matrix and b is a real vector of length N. The restriction to real problems is for simplicity of notation and has no other role with respect to the studied phenomena. Given an initial approximation x_0 to x and initial residual $r_0 = b - Ax_0$, with $v_1 = r_0/\|r_0\|$, Krylov subspace methods construct *mathematically*, i.e., assuming exact computation, a sequence of approximate solutions using the nested Krylov subspaces

$$\mathcal{K}_1(A, v_1) \subset \mathcal{K}_2(A, v_1) \subset \cdots \subset \mathcal{K}_n(A, v_1) \subset \cdots$$

defined as

$$\mathcal{K}_i(A, v_1) = \mathcal{K}_i(A, r_0) = \text{span}\{v_1, Av_1, \ldots, A^{i-1}v_1\}, \quad i = 1, 2, \ldots .$$

The nth approximation $x_n \in x_0 + \mathcal{K}_n(A, r_0)$ to the solution x is constructed using the orthogonality of the nth residual to the n-dimensional constraint space $\mathcal{C}_n$, i.e., $r_n = b - Ax_n \perp \mathcal{C}_n$, where the choice

* Courant Institute of Mathematical Sciences, New York University erinc@cims.nyu.edu.

† Institute of Computer Science, Academy of Sciences of the Czech Republic, (miro@cs.cas.cz). Partially supported by the Grant Agency of the Czech Republic Project GA13-06684S

‡ Department of Numerical Mathematics, Faculty of Mathematics and Physics, Charles University strakos@karlin.mff.cuni.cz. Supported by the ERC project MORE LL1202 financed by the MŠMT of the Czech Republic

§ Department of Numerical Mathematics, Faculty of Mathematics and Physics, Charles University, (ptichy@karlin.mff.cuni.cz). Partially supported by the Grant Agency of the Czech Republic Project GA13-06684S

¶ Department of Numerical Mathematics, Faculty of Mathematics and Physics, Charles University, (mirektuma@karlin.mff.cuni.cz). Partially supported by the Grant Agency of the Czech Republic Project GA13-06684S and by the ERC project MORE LL1202 financed by the MŠMT of the Czech Republic

of $\mathcal{C}_n$ along with properties of A distinguish the various Krylov subspace methods. As an example, for symmetric positive definite (SPD) matrices and $\mathcal{C}_n = \mathcal{K}_n(A, r_0)$ we get the conjugate gradient method (CG), where the orthogonality constraint $r_n \perp \mathcal{K}_n(A, r_0)$ is equivalent to the minimization of the energy norm of the error over the Krylov subspaces, i.e.,

$$\|x - x_n\|_A = \min_{z \in x_0 + \mathcal{K}_n(A, r_0)} \|x - z\|_A, \tag{1.1}$$

where $\|u\|_A = (u, u)_A^{1/2} = (Au, u)^{1/2}$ is induced by the energy inner product defined by the SPD matrix A. For symmetric nonsingular matrices and $\mathcal{C}_n = A\mathcal{K}_n(A, r_0)$ we get the minimal residual method (MINRES), and for general nonsingular matrices and $\mathcal{C}_n = A\mathcal{K}_n(A, r_0)$ we get the generalized minimal residual method (GMRES). Both MINRES and GMRES minimize the Euclidean norm (denoted here and elsewhere in this work by $\|\cdot\|$) of the residual over the Krylov subspaces, i.e.,

$$\|b - Ax_n\| = \min_{z \in x_0 + \mathcal{K}_n(A, r_0)} \|b - Az\|.$$

Within the present work we restrict the discussion mostly to CG.

From the given orthogonality conditions it is clear that Krylov subspace methods are based on *linear projections onto highly nonlinear nested subspaces*. They are therefore highly nonlinear in the input data A, b defining the problem. This nonlinearity is what allows Krylov subspace methods to adapt to the problem as the iteration proceeds, but makes analysis of the methods a challenge; see, e.g., [30] and the references therein. In particular, such analyses must respect the relationships between the studied phenomena, making them highly complex and unamenable to easy simplification. We will explain the difficulty of the analysis of Krylov subspace methods in more detail in the rest of this introductory section using the example of CG.

Consider the Lanczos process for computing an orthonormal basis of the sequence of Krylov subspaces for a matrix A and starting vector $v_1 = r_0/\|r_0\|$, which is described by the matrix equations

$$A\,V_n \;=\; V_n\,T_n \;+\; \delta_{n+1} v_{n+1} e_n^T\,, \quad T_n \;=\; V_n^*\,A\,V_n\,, \tag{1.2}$$

where $V_n = [v_1, \ldots, v_n]$ is the $N \times n$ matrix storing the orthonormal basis vectors as its columns, V_n^* denotes the transpose of the matrix V_n, and T_n denotes the Jacobi matrix storing in its columns the orthogonalization and normalization coefficients. The Jacobi matrix T_n depends on the $2n-1$ moments $(A^i v_1, v_1)$, $i = 1, 2, \ldots, 2n-1$. The CG approximation is then given by solving the projected problem with the $n \times n$ symmetric positive definite tridiagonal matrix T_n, which determines the coefficients of the linear combination of the Lanczos basis functions, i.e.,

$$T_n\,y_n \;=\; \|r_0\| e_1\,, \quad x_n \;=\; x_0 \,+\, V_n\,y_n\,. \tag{1.3}$$

Summarizing, given an initial approximation x_0, the sequence of subsequent CG approximations $x_1, x_2, \ldots, x_n$ to the solution of the linear algebraic system $Ax = b$ that represents the original model is obtained by solving the sequence of reduced models (1.3) of increasing dimensionality. Again, we emphasize that the described model reduction is highly nonlinear in the data A, b. The Jacobi matrix T_n is related to the short recurrences characteristic of the Lanczos method, and also has connections with orthogonal polynomials, the Stieltjes problem of moments, Gauss-Cristoffel quadrature, continued fractions, etc.; see [13], [30, Chapter 3].

As for practical computations, Krylov subspace methods such as MINRES and GMRES are indeed commonly implemented in a way that closely resembles the idea of a sequence of projection processes; see, e.g., [30, Section 2.5.5] and the references therein. This is not the case for CG. The standard CG implementation described by Hestenes and Stiefel [25] (see also [30]) uses three two-term recurrences for updating the approximate solution x_i, the direction vector p_i, and the residual vector r_i (which is mathematically related to the Lanczos basis vectors by $r_i = (-1)^i \|r_i\| v_{i+1}$), as follows:

ALGORITHM 1.1. **HS conjugate gradient method**
Input: *SPD matrix $A \in R^{N\times N}$, right-hand side vector $b \in R^N$, initial approximation $x_0 \in R^N$, maximum number of iterations nmax.*
Output: *Approximate solution x_n after the algorithm has been stopped.*

0. **Initialization:** $r_0 = b - Ax_0$, $p_0 = r_0$
1. **for** $i = 1 : imax$ **do**
2. $\quad \alpha_{i-1} = \dfrac{(r_{i-1}, r_{i-1})}{(p_{i-1}, Ap_{i-1})}$
3. $\quad x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
4. $\quad r_i = r_{i-1} - \alpha_{i-1} Ap_{i-1}$
5. $\quad$ *evaluate the stopping criterion*
6. $\quad \beta_i = \dfrac{(r_i, r_i)}{(r_{i-1}, r_{i-1})}$
7. $\quad p_i = r_i + \beta_i p_{i-1}$
8. **end do**

Here the choice of α_{i-1} ensures the minimization of $\|x - x_i\|_A$ *along the line*

$$z(\alpha) = x_{i-1} + \alpha p_{i-1}. \tag{1.4}$$

The mathematical elegance and power of CG is given by the following crucial condition. Provided that the direction vectors are mutually orthogonal with respect to the energy inner product defined by the SPD matrix A, i.e.,

$$p_i \perp_A p_j \quad \text{for} \quad i \neq j, \tag{1.5}$$

the one-dimensional line minimizations at the individual steps 1 to i result in the i-dimensional minimization over the *whole* shifted Krylov subspace

$$x_0 + \mathcal{K}_i(A, r_0) \;=\; x_0 + \mathrm{span}\{p_0, p_1, \ldots, p_{i-1}\}\,.$$

Analogous to the link between the Jacobi matrix and the short recurrences mentioned above, the line search (1.4) and the orthogonality condition (1.5) lead to short recurrences due to the symmetry of the matrix A, or, equivalently, due to the relationship with the orthogonal polynomials that define the algebraic residuals and search vectors.

In each iteration of HS CG (Algorithm 1.1), updating the recurrences for x_i, r_i, and p_i requires computing the inner-products (p_{i-1}, Ap_{i-1}) and (r_i, r_i) in order to form the scalars α_{i-1} and β_i. When Algorithm 1.1 is implemented on a parallel machine, computing each inner-product requires a global synchronization point; i.e., the computation can not proceed until all processors have finished their local computation and communicated the result to other processors. It is well-known that 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 [10, pp. 28]). This limits the potential speed in performing individual iterations attainable by an implementation of CG. Note that in this work, we will use the term *synchronization* to refer to a global synchronization required to compute an inner product; i.e., all processors must exchange data and wait for all communication to finish before proceeding with the computation. In MPI terminology, our use of the term synchronization is synonymous with blocking collective communication.

The goal of removing performance bottlenecks caused by synchronization has motivated work on pipelined Krylov subspace methods (see, e.g, [16], [17]). The key feature of pipelined CG [17] (and other pipelined Krylov subspace methods) is that computing the necessary inner products only requires a single non-blocking communication per iteration. Furthermore, the computation of the inner products can be overlapped with the matrix-vector product and other local computations, effectively hiding the latency of

global communication. This overlapping is enabled by the introduction of one or more auxiliary vectors which are updated via additional recurrences in each iteration.

Other Krylov subspace method variants have been developed for the case where the matrix-vector products required in each iteration dominate the run-time. One example are the so-called "inexact Krylov subspace methods" (see, e.g., [2], [47], [48], [58], [59]) in which the accuracy of the matrix-vector product is intentionally relaxed throughout the iterations according to a function of the residual norm. We note that, in contrast to the terminology introduced in the works mentioned above, in this work we use the term "inexact" in a more general sense, referring to methods in which computed quantities can deviate from their exact mathematical counterparts due to intentional approximation or relaxation of accuracy, or simply due to floating point roundoff error.

In exact arithmetic, the pipelined CG method produces the same sequence of approximate solutions and residual vectors as the standard CG method. In finite precision, however, the introduction of additional recurrences causes the numerical behavior of the standard and pipelined CG and of their implementation variants to differ. It has indeed been observed (see, e.g., [17]) that in finite precision computations, the convergence rate and attainable accuracy of pipelined CG can be worse than in standard CG, with the attainable accuracy reduced by orders of magnitude in some cases. When pipelined Krylov subspace methods are used in practical settings, these differences in numerical behavior must then be taken into account, ideally balancing the application-specific accuracy requirements with potential performance improvements. Analyses which provide insight into the numerical behavior of pipelined Krylov subspace methods are thus vital to their use in practice.

In general, analyses of inexact CG computations (again, either due to roundoff error or an intentional relaxation of accuracy in performing the most costly operations) must address the question of *how the previously-discussed relationships that hold for exact standard CG are changed in the presence of inaccuracies.* After deriving recurrences for the *computed quantities*, i.e., the mathematical equalities analogous to (1.2) and to the lines in Algorithm 1.1, the inaccuracies of individual local computations may be accounted for by adding local error terms to the lines 2.-4., 6.-7. of Algorithm 1.1 (in the Lanczos process, to each column of (1.2)); see [37, Section 4, Theorem 4.1, and Section 5, relation (5.1)]. These local error terms and other subsequent local inaccuracies at individual iteration steps (such as the loss of orthogonality among the consecutive residuals and direction vectors) can be small. Their effect, however, can be drastically amplified within few iterations and can cause, e.g., a significant loss of global orthogonality or even numerical linear independence among the computed vectors generating the associated subspaces. The Jacobi matrices resulting from the computed data can quickly become very different from their exact arithmetic counterparts.

Much work has been done in the area of analyzing standard CG in finite precision arithmetic. In finite-precision CG (as well as in general inexact CG methods), the CG optimality property (1.1) does not have a clear meaning with respect to the subspaces generated by the computed residual (or direction) vectors. Greenbaum showed in 1989 [18] that for the finite precision Lanczos process it does have, however, a rigorously defined meaning with respect to some particular distribution functions defined by the original data and the rounding errors in iterations 1 to n. Greenbaum proved, using the results of Paige on the numerical behavior of the Lanczos method published within 1971-80 [39, 40, 41, 42], that the Jacobi matrix *computed in iteration n in finite precision arithmetic* can be considered a left principal submatrix of a certain larger Jacobi matrix having all its eigenvalues close to the eigenvalues of the original matrix A; for a summary and exposition on the consequences of this fundamental result we refer to [37, Section 5]. Let us assume, for a moment, that the effects of rounding errors in computing y_n (see (1.3)) and subsequently forming x_n are negligible in comparison to the effects of rounding errors in computing the Lanczos vectors $v_1, \ldots, v_n$. Then the first n iterations of the given finite precision CG computation using (1.3) are mathematically equivalent to evaluating *exact* Gauss quadrature for a certain distribution function (that depends on n) that has tight clusters of points of increase around the original eigenvalues of A; see, e.g., [18], [37, Section 5.2], [38], [30, Section 5.9.1]. It is clear that determining how the convergence of the practically computed approximations to the solution x is slowed down in comparison

to the mathematical formulation of CG indeed represents a challenge.

Another question is related to the maximal accuracy that can be attained in practical computations, which becomes important in solving problems for which high accuracy of the computed approximation is required. While in exact arithmetic the exact solution x must be reached by CG at or before step N, with inexact computations this is not the case. The basic methodology for analyzing the attainable accuracy involves comparing the residuals $\bar{r}_i$, iteratively computed using, e.g. Algorithm 1.1, with the residuals mathematically defined as $b - A\bar{x}_i$, where $\bar{x}_i$ denotes the computed quantity. The *mathematical* difference

$$\bar{r}_i - (b - A\bar{x}_i) \tag{1.6}$$

is then evaluated as a function of the increasing index i and is subsequently bounded in an appropriate norm. Using the argument that the iteratively computed residual $\bar{r}_i$ eventually becomes negligible (even in finite precision computations), this gives for i large enough an estimate of the maximal attainable accuracy. This methodology is rigorously supported by formal proofs for some special CG implementations (see, e.g., [19] and the surveys in [26, Section 17.1], [37, Section 5.4]). Otherwise it is based on justified heuristics and observations. In order to avoid confusion that has appeared in literature, it should be stressed that (1.6) represents a *mathematical term* where $\bar{r}_i$ and $\bar{x}_i$ are the outputs of inexact computations. For details of the rounding error analysis of the classical variant HS CG we refer to the survey [37] and to the update in [30, Chapter 5].

From the previous argument it follows that any analysis of inexact Krylov subspace methods, including pipelined variants, as well as any potential technique for counteracting the numerical effects of inexact computations, must very carefully consider the links between the studied phenomena. The assumptions and simplifications used must be justified. A clear, thorough understanding of how inexact computations affect numerical behavior is imperative in balancing the tradeoffs between accuracy and speed in high-performance implementations; this becomes especially important for computations at the exascale level.

The following section will recall some early developments towards increasing parallel efficiency in CG computations that preceded the formulation of pipelined CG. Section 3 then describes and numerically illustrates how modifications of the CG implementation and the corresponding changes in the individual sources of instabilities contribute to the overall final inaccuracy effects. Section 4 will outline one example of how the individual sources of instabilities influence each other. We conclude with brief remarks on the use of Krylov subspace methods for exascale-level computations.

Preconditioning represents a crucial part of any practical iterative computation, and thus the analysis of applications of Krylov subspace methods must always include consideration of preconditioning. Here, however, we take the luxury of formulating most of the text using unpreconditioned CG; as it turns out, the particular preconditioning approach does not affect the description of the basic stability analysis building blocks. As it will be mentioned in Section 2, preconditioning can easily be included in the described framework.

2. Early developments towards parallel efficiency: History repeats. The term "standard CG" is generally associated with the seminal paper by Hestenes and Stiefel [25] and with the HS CG implementation described in Algorithm 1.1. Nevertheless, it is instructive to mention some other early contributions and ideas that lead to this culmination. This approach will enable us to demonstrate that the early development of CG contained many ideas that have been repeatedly reinvented with the emergence of modern computing architectures.

One of the basic driving forces behind the development of CG was a careful investigation of the connection between solving systems of linear algebraic equations and minimizing quadratic functionals. As stated by Hestenes [24], ideas of this type had been pursued previously by many authors. In the middle of the previous century, these ideas where considered in relation to the state-of-the-art relaxation methods of the time. A general framework of the related class of the so-called n-step methods can be found in the paper by Stiefel [51] (they should not be confused with the s-step Krylov subspace methods mentioned below). Investigation of theoretical properties of the n-step methods as well as connections to the steepest gradient method in Stiefel [52] reveals another important concept.

2.1. Three-term recurrence variants. The ST method (or, shortly, ST CG), shown in Algorithm 2.1, is based on *three-term recurrences* for updating the approximate solutions and corresponding residuals. Its derivation was motivated by the relation to three-term recurrences for orthogonal polynomials with a specific choice of density function [57, Chapter 3]. The algorithm is often attributed to Rutishauser [12]; see, e.g., [43]. As mentioned above, it was actually developed much earlier, around the same time as HS CG; see also Rosser [44] that attributes the described results to the joint work with Forsythe, Hestenes, Lanczos, Motzkin and Paige.

ST CG is mathematically equivalent to HS CG given in Algorithm 1.1, but ST CG and HS CG exhibit rather different behavior in finite precision arithmetic. We will return to this point in Section 3.

ALGORITHM 2.1. **ST (three-term) conjugate gradient method**
Input: *SPD matrix $A \in R^{N\times N}$, right-hand side vector $b \in R^N$, initial approximation $x_0 \in R^N$, maximum number of iterations nmax.*
Output: *Approximate solution x_n after the algorithm has been stopped.*

0. **Initialization:** $r_0 = b - Ax_0$, $p_0 = r_0$, $x_{-1} = x_0$, $r_{-1} = r_0$, $e_{-1} = 0$
1. **for** $i = 1 : nmax$ **do**
2. $\quad q_{i-1} = \dfrac{(r_{i-1}, Ar_{i-1})}{(r_{i-1}, r_{i-1})} - e_{i-2}$
3. $\quad x_i = x_{i-1} + \frac{1}{q_{i-1}}[r_{i-1} + e_{i-2}(x_{i-1} - x_{i-2})] \equiv x_{i-1} + \frac{1}{q_{i-1}}[r_{i-1} + e_{i-2}\Delta x_{i-1}]$
4. $\quad r_i = r_{i-1} + \frac{1}{q_{i-1}}[-Ar_{i-1} + e_{i-2}(r_{i-1} - r_{i-2})] = r_{i-1} + \frac{1}{q_{i-1}}[-Ar_{i-1} + e_{i-2}\Delta r_{i-1}]$
5. $\quad$ *evaluate the stopping criterion*
6. $\quad e_{i-1} = q_{i-1}\dfrac{(r_i, r_i)}{(r_{i-1}, r_{i-1})}$
7. **end do**

While the mathematical equivalence of ST CG and HS CG is well-known, we will state this relationship formally to point out the mutually equivalent quantities used in these algorithms and show relations among e_i, q_i, α_i, and β_i.We do this in Observation 2.1.

OBSERVATION 2.1. *Algorithms 1.1 and 2.1 produce the same sequences of iterates x_i and residuals r_i, respectively, for $i = 0, 1, \ldots$. Moreover, we have $q_i = 1/\alpha_i$ and $e_i = q_i\beta_{i+1}$ for $i = 0, 1, \ldots$.*

Proof. Let us first relate the scalar coefficients of the two considered algorithms. Consider q_i and e_i from Algorithm 2.1. Clearly, $q_0 = 1/\alpha_0$ and $e_0 = \beta_1 q_0$. Consider $i > 1$. Using notation from Algorithm 1.1, basic orthogonality relations between the exact quantities and the induction assumption, we get

$$\begin{aligned} q_{i-1} &= \frac{(r_{i-1}, Ar_{i-1})}{(r_{i-1}, r_{i-1})} - e_{i-2} = \frac{(p_{i-1}, Ap_{i-1})}{(r_{i-1}, r_{i-1})} + \frac{(\beta_{i-1}p_{i-2}, \beta_{i-1}Ap_{i-2})}{(r_{i-1}, r_{i-1})} - q_{i-2}\beta_{i-1} \\ &= \frac{1}{\alpha_{i-1}} + \frac{\beta_{i-1}^2}{\beta_{i-1}\alpha_{i-2}} - q_{i-2}\beta_{i-1} = \frac{1}{\alpha_{i-1}}. \end{aligned}$$

The three-term recurrences from Algorithm 2.1 then easily follow. Consider, for example, the recursion for approximate residuals. From Algorithm 1.1 we have $r_i = r_{i-1} - \alpha_{i-1}Ap_{i-1}$. Observing that the direction vector update provides $Ap_i = Ar_i + \beta_i Ap_{i-1}$, clearly,

$$\begin{aligned} r_i &= r_{i-1} - \alpha_{i-1}Ap_{i-1} \\ &= r_{i-1} - \alpha_{i-1}Ar_{i-1} - \alpha_{i-1}\beta_{i-1}Ap_{i-2} \\ &= r_{i-1} - \alpha_{i-1}Ar_{i-1} + \alpha_{i-1}\beta_{i-1}/\alpha_{i-2}(r_{i-1} - r_{i-2}) \\ &= r_{i-1} - 1/q_{i-1}Ar_{i-1} + 1/q_{i-1}e_{i-2}(r_{i-1} - r_{i-2}). \end{aligned}$$

The three-term update for the approximate solution can be obtained analogously. □

The fact that a method using three-term recurrences can behave in finite precision arithmetic very differently from Algorithm 1.1 has been shown by Gutknecht and Strakoš in [22], see also [35, Chapters 6 and 7]. Nevertheless, it should be noted that the authors analyzed a slightly different algorithm and thus the detailed floating-point analysis of Algorithm 2.1 is still to be done. The same is true for other possible mathematically equivalent approaches.

2.2. Early approaches to reducing synchronization in Krylov subspace methods. A renewed interest in the numerical properties of different variants of CG and other Krylov subspace methods came with the advent of modern massively parallel computer architectures. Several modifications of the basic algorithms have been proposed in order to deal with the problem of communication cost. Chronopoulos and Gear proposed in [7], [6] the so-called *s-step* extensions of Krylov subspace methods. A couple of substantial further developments along this line followed; see, e.g., the Ph.D. Theses by Hoemmen [27], Ballard [1], and Carson [4]. The main idea of the s-step methods is to enable more parallelism by relaxing the sequential construction of the basis of the Krylov subspace in Algorithm 1.1 and computing more direction vectors simultaneously. For further information on the state-of-the-art and open questions we refer to [4].

As mentioned in the Introduction, the effort toward improving parallel efficiency in individual CG iterations by reducing the synchronization cost has recently led to the creation of pipelined Krylov subspace methods [16], [17]. Analogous ideas were presented in numerous early works. The recent approaches to reducing the number of synchronization points in fact represent a variation of the three-term recurrences given in Algorithm 2.1 that dates back to the 1950's. Before returning to more recent work, we recall some other early approaches.

An idea independently given by Johnson [28], [29], van Rosendale [63] (see also [14]) and Saad [45] is based on the identity that in the notation of Algorithm 1.1 reads

$$||r_i||^2 + ||r_{i-1}||^2 = \alpha_{i-1}^2 ||Ap_{i-1}||^2. \tag{2.1}$$

This enables the computation of β_i from α_{i-1} and Ap_{i-1} as well as the computation of the vectors x_i, r_i and p_i in one block. As demonstrated by Saad in [46], this approach is unstable and cannot be used in a straightforward way. Despite this, based on the proposal of Meurant (see [34], [33], [5]), it is possible to use the value of $||r_i||^2$ computed via (2.1) in updating the direction vector. The quantity β_i may be recomputed once the corresponding residual vector r_i is available at the expense of an additional inner product. Meurant claims in [34] that this worked well in all his test examples. This idea might deserve further investigation. A similar approach has been proposed in [32] where the computation is based on the formula

$$||r_i||^2 = ||r_{i-1}||^2 + \alpha_{i-1}^2 ||Ap_{i-1}||^2 + 2\alpha_{i-1} r_{i-1} Ap_{i-1}, \tag{2.2}$$

although this work lacks a supporting numerical stability analysis.

Another strategy for computing the inner products using only one synchronization point is to derive α_i from β_{i-1} using the relation

$$\alpha_i = \frac{(r_i, r_i)}{(Ar_i, r_i) - (\beta_{i-1}/\alpha_{i-1})(r_i, r_i)}. \tag{2.3}$$

Observation 2.1 implies that Algorithm 2.1 does exactly this. Probably the first attempt to use Algorithm 2.1 to reduce the number of synchronization points in CG implemented on parallel computers has been published in [55], see also [54]. Since then, the idea to compute α_i based on β_{i-1} has been reinvented a couple of times in parallel computing, sometimes cited as an alternative approach, and often complemented by additional modifications motivated by particular computer architectures. Many later variants slightly differ in the details of the computation of both the scalar quantities and the vector updates, which may influence their actual numerical behavior; see Section 3. In order to make precise our terminology, we introduce the following definition.

DEFINITION 2.1. *We use* three-term CG *to refer to any CG implementation mathematically equivalent to Algorithm 2.1 that computes α_i based on β_{i-1} using (2.3), where this computation can be accomplished by a single synchronization point per iteration.*

Chronopoulos and Gear in [7] in fact reintroduced in their Algorithm 2.2 a three-term CG very closely related to ST CG (Algorithm 2.1). They in addition use recursive computation of the products Ap_i

according to the formula

$$Ap_i = Ar_i + \beta_{i-1} Ap_{i-1}; \tag{2.4}$$

see the following algorithm.

ALGORITHM 2.2. **ChG conjugate gradient method**
Input: *SPD matrix $A \in R^{N\times N}$, right-hand side vector $b \in R^N$, initial approximation $x_0 \in R^N$, maximum number of iterations nmax.*
Output: *Approximate solution x_n after the algorithm has been stopped.*

0. **Initialization:** $r_0 = b - Ax_0$, $p_0 = r_0$, $s_0 = Ap_0$, $\alpha_0 = \dfrac{(r_0, r_0)}{(p_0, s_0)}$
1. **for** $i = 1 : nmax$ **do**
2. $\quad x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$,
3. $\quad r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$
4. $\quad$ *evaluate the stopping criterion*
5. $\quad w_i = Ar_i$
6. $\quad \beta_i = \dfrac{(r_i, r_i)}{(r_{i-1}, r_{i-1})}$
7. $\quad \alpha_i = \dfrac{(r_i, r_i)}{(w_i, r_i) - (\beta_i/\alpha_{i-1})(r_i, r_i)}$
8. $\quad p_i = r_i + \beta_i p_{i-1}$
9. $\quad s_i = w_i + \beta_i s_{i-1}$
10. **end do**

Although the recurrences in Algorithm 2.2 look quite similar to the two-term recurrences used in Algorithm 1.1, this is only a formal resemblance. Substitution of the rows 8 and 9 into the solution and residual updates on the rows 2 and 3 immediately reveals that Algorithm 2.2 is based on three-term recurrences as Algorithm 2.1. Moreover, using notation in Algorithms 2.1 and 2.2, we have

$$q_i = \frac{1}{\alpha_i} = \frac{(w_i, r_i)}{(r_i, r_i)} - \frac{\beta_i (r_i, r_i)}{\alpha_{i-1}(r_i, r_i)} = \frac{(w_i, r_i)}{(r_i, r_i)} - q_{i-1}\beta_i = \frac{(w_i, r_i)}{(r_i, r_i)} - e_{i-1}. \tag{2.5}$$

Algorithm 2.2 uses the additional recurrence (2.4) for the A-multiple of the direction vector, $s_i = Ap_i$. This may bring in an additional source of instability in ChG CG (Algorithm 2.2) in comparison to ST CG (Algorithm 2.1).

There were some other works related to the original contribution by Stiefel [51], [52]. The idea of computing α_i based on β_{i-1} has been used by D'Azevedo and Romine [8]; see also [9] (the connection to the work of Stiefel has not been noticed).

2.3. Recent work in pipelined Krylov subspace methods. An algorithm that combines reducing the number of synchronization points and overlapping inner products with other computations has been developed by Ghysels and Vanroose [17] under the name pipelined conjugate gradient. We present here its unpreconditioned version.

ALGORITHM 2.3. **GV (pipelined) conjugate gradient method**
Input: *SPD matrix $A \in R^{N\times N}$, right-hand side vector $b \in R^N$, initial approximation $x_0 \in R^N$, maximum number of iterations nmax.*
Output: *Approximate solution x_n after the algorithm has been stopped.*

0. **Initialization:** $r_0 = b - Ax_0$, $p_0 = r_0$, $s_0 = Ap_0$, $w_0 = Ar_0$, $z_0 = Aw_0$, $\alpha_0 = \dfrac{(r_0, r_0)}{(p_0, s_0)}$
1. **for** $i = 1 : nmax$ **do**
2. $\quad x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$,
3. $\quad r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$

4. $w_i = w_{i-1} - \alpha_{i-1} z_{i-1}$
5. *evaluate the stopping criterion*
6. $q_i = A w_i$
7. $\beta_i = \dfrac{(r_i, r_i)}{(r_{i-1}, r_{i-1})}$
8. $\alpha_i = \dfrac{(r_i, r_i)}{(w_i, r_i) - (\beta_i/\alpha_{i-1})(r_i, r_i)}$
9. $p_i = r_i + \beta_i p_{i-1}$
10. $s_i = w_i + \beta_i s_{i-1}$
11. $z_i = q_i + \beta_i z_{i-1}$
12. **end do**

Comparing pipelined CG, denoted here as GV CG (Algorithm 2.3), with ChG CG (Algorithm 2.2), we can see that both use the same computation of scalars α_i and the same computation of Ap_i. In addition, GV CG introduces a recurrence for computing Ar_i iteratively. The auxiliary recurrences that enable overlapping the inner product with the matrix-vector multiplication do not explicitly enforce close local orthogonality among the computed vectors by recomputing the associated recurrence coefficients. Consequently, a substantial source of potential numerical instability is introduced.

Another recent approach called asynchronous CG has been proposed by Gropp [21]; see also [17]. In its preconditioned form, asynchronous CG keeps the two synchronization points as in HS CG, but uses additional recurrences to compute the preconditioned residual $M^{-1}r_i$ and the vector Ap_i. Here substantial numerical instabilities may again arise from the additional recursively computed quantities.

Let us emphasize here the role of preconditioning mentioned at the end of the previous section. Some approaches mentioned above that use overlapping of operations conveniently involve preconditioning in balancing the associated computation load. In pipelined CG, preconditioning is implemented at the expense of an additional recursion. This inevitably influences finite precision computations. From a methodological point of view, preconditioning presents no obstacle to performing numerical stability analysis.

The idea of overlapping the computation of inner products with other local computations at the price of auxiliary vector updates (ideas that were used in earlier Krylov subspace method implementations) has not been, to our knowledge, widely publicized prior to [16], [17]. In [55], overlapping was not used because of computer architecture limitations. An example of such overlapping can be found, e.g., in the preconditioned HS algorithm proposed by van der Vorst [60]. Here the preconditioning operation is split into forward and backward solves using the incomplete Cholesky factorization combined with delaying the update of the approximate solution.

The pipelined CG method and other pipelined Krylov subspace methods have attracted recent attention for their potential use in high-performance computing. A large number of subsequent papers and reports emphasize various features of these methods with respect to parallel performance. It is not our task to list them here. Our goal is to point out the need for a thorough rigorous numerical stability analysis and to outline the way towards doing this.

3. Pipelined Krylov subspace methods and individual sources of instabilities. In this section we identify individual sources of instability and use simple examples to demonstrate their effect on numerical behavior. Using the notation standard in numerical stability analysis, the quantities computed in finite precision arithmetic will be decorated with bars. The residual vectors computed using the recurrence $r_i = r_{i-1} - \alpha_{i-1} A p_{i-1}$ satisfy

$$\bar{r}_i = \bar{r}_{i-1} - \bar{\alpha}_{i-1} A \bar{p}_{i-1} + \delta r_i,$$

where $\bar{r}_{i-1}$, $\bar{r}_i$ and $\bar{p}_{i-1}$ are the computed vectors, $\bar{\alpha}_{i-1}$ is the computed coefficient, and δr_i is a vector that accounts for local roundoff errors. We will now recall the basic methodology for numerical stability analysis of CG through the example of investigating the maximal attainable accuracy.

It is well known that there exists a limit on the accuracy of approximate solutions $\bar{x}_i$ computed in finite precision arithmetic. There is always a gap (1.6) between the true residual $b - A\bar{x}_i$ and the recursively updated vectors $\bar{r}_i$ (often called updated residuals). We denote this gap by f_i, i.e., $f_i \equiv \bar{r}_i - (b - A\bar{x}_i)$. While the norm of the recursively updated residual $\bar{r}_i$ eventually decreases beyond the level of unit roundoff (see the Introduction for more details and references), the norm of the true residual $b - A\bar{x}_i$ for i sufficiently large stagnates, and the size of f_i determines the *maximal attainable accuracy* (measured by the norm of the true residual).

It is important to investigate the maximal attainable accuracy of iterative methods and to use some measure for improving it, if needed. The effects of rounding errors, however, go beyond affecting only the maximal attainable accuracy. In particular, rounding errors can cause a significant *delay of convergence* that is of *primary importance* in challenging real world problems, where computations are in most cases stopped much before the maximal attainable accuracy is reached. Moreover, measures taken for improving maximal attainable accuracy can negatively affect delay of convergence; this is discussed further in Section 3.5.

3.1. Coupled two-term recurrences. Let us demonstrate how $\|f_i\|$ can be bounded in the HS CG (Algorithm 1.1). Given an initial approximation $\bar{x}_0$, the computed vectors satisfy:

$$\bar{p}_0 = \bar{r}_0 = b - A\bar{x}_0 + f_0,$$

and

$$\bar{x}_i = \bar{x}_{i-1} + \bar{\alpha}_{i-1}\bar{p}_{i-1} + \delta x_i, \tag{3.1}$$
$$\bar{r}_i = \bar{r}_{i-1} - \bar{\alpha}_{i-1}A\bar{p}_{i-1} + \delta r_i, \tag{3.2}$$
$$\bar{p}_i = \bar{r}_i + \bar{\beta}_i\bar{p}_{i-1} + \delta p_i, \tag{3.3}$$

where the individual δ-terms account for the local roundoff errors. Assume the standard model of floating point arithmetic with machine precision ϵ, see, e.g. [26, relations (2.4)]. Then the vectors of local roundoff errors can be bounded in terms of the dimension N of the problem, machine precision ϵ, the norm of A, and the norms of the associated vectors. To simplify the notation we give these bounds in the form

$$\begin{aligned}
\|\delta x_i\| &\leq \mathcal{O}(\epsilon)\max\{\|\bar{x}_{i-1}\|, \|\bar{x}_i\|\},\\
\|\delta r_i\| &\leq \mathcal{O}(\epsilon)\max\{\|\bar{r}_{i-1}\|, \|\bar{\alpha}_{i-1}A\bar{p}_{i-1}\|\},\\
\|\delta p_i\| &\leq \mathcal{O}(\epsilon)\max\{\|\bar{r}_i\|, \|\bar{\beta}_i\bar{p}_{i-1}\|\},
\end{aligned}$$

where $\mathcal{O}(\epsilon)$ denotes a term that can be bounded by ϵ and a low degree polynomial in N. The details can be found, e.g., in [53].

Combining (3.1) and (3.2) we can express the gap f_i in terms of the gap f_{i-1} and the local roundoff errors:

$$\begin{aligned}
f_i &= \bar{r}_i - (b - A\bar{x}_i)\\
&= \bar{r}_{i-1} - \bar{\alpha}_{i-1}A\bar{p}_{i-1} + \delta r_i - (b - A(\bar{x}_{i-1} + \bar{\alpha}_{i-1}\bar{p}_{i-1} + \delta x_i))\\
&= f_{i-1} + \delta r_i + A\delta x_i.
\end{aligned} \tag{3.4}$$

Hence by induction it can be shown that

$$f_n = f_0 + \sum_{i=1}^{n}\delta r_i + A\sum_{i=1}^{n}\delta x_i. \tag{3.5}$$

The maximal attainable accuracy of the coupled two-term recurrences was analyzed, e.g., by Sleijpen, van der Vorst, and Fokkema [50], and Greenbaum [19]. In these works the gap f_i is analyzed in terms of the local errors associated with the two-term recurrences and it is shown that in this case the local errors simply accumulate. In particular, the authors of [50] prove that

$$\|f_n\| \leq \mathcal{O}(\epsilon)\||A|\|\max_{i=0,\ldots,n}\|x - \bar{x}_i\|, \tag{3.6}$$

and in [19] it is shown that

$$\|f_n\| \leq \mathcal{O}(\epsilon)\|A\| \left(\|x\| + \max_{i=0,\ldots,n} \|\bar{x}_i\| \right). \tag{3.7}$$

The delay of convergence of CG in finite precision arithmetic has been analyzed, e.g., in papers Greenbaum [18], Strakoš [56], Greenbaum and Strakoš [20], Gergelits and Strakoš [15]; for more detailed comments and references see the Introduction.

We now turn our attention to other variants of CG and investigate the individual sources of instabilities. In other words, we are interested in the question of how the attainable accuracy and delay of convergence are affected by various modifications to Algorithm 1.1. To illustrate the numerical effects, throughout this section we use the example of the linear system $Ax = b$ with the matrix `bcsstk03` from set BCSSTRUC1 (BCS Structural Engineering Matrices), from the Harwell-Boeing collection [11], of order $N = 112$. We set x_0 to be the zero vector. The right-hand side b has been chosen such that b has equal components in the eigenvector basis, and such that $\|b\| = 1$. The minimal and maximal eigenvalues of A are $\lambda_{\min} \approx 2.9e+e04$ and $\lambda_{\max} \approx 2.0e+e11$ respectively, so that $\kappa(A) \approx 6.9e+06$. Since large outlying eigenvalues are present in the spectrum, one can expect a significant delay of convergence of CG in finite precision arithmetic; for more details, see, e.g., [56, 53, 15].

Although the investigation of maximal attainable accuracy is formulated in terms of residuals and their norms, in our experiments we purposely plot the A-norm of the error rather than the Euclidean norm of the residuals. The reason is twofold. First, the convergence behavior viewed in terms of the Euclidean norm of the residual can be oscillatory. Second, and even more importantly, as stated clearly in the seminal paper by Hestenes and Stiefel [25], the Euclidean norm of the residual is not an appropriate indicator of convergence for problems where $\kappa(A)$ is substantially larger than one.

3.2. Three-term recurrences. Before considering the implementation using three-term recurrences, we present the following numerical illustration. Consider Algorithm 1.1, and replace the recurrence for updating the direction vector

$$p_i = r_i + \beta_i p_{i-1}, \tag{3.8}$$

by the recurrence

$$p_i = r_i + \frac{\beta_i}{\alpha_{i-1}}(x_i - x_{i-1}). \tag{3.9}$$

At first glance, this looks like a benign change that should not cause any trouble since x_i is computed using the recurrence $x_i = x_{i-1} + \alpha_{i-1}p_{i-1}$. However, as we can observe in Figure 3.1, replacing (3.8) by (3.9) can have a substantial influence on the behavior of the algorithm in finite precision arithmetic, both with respect to the rate of convergence and the attained accuracy. This example shows that a seemingly trivial change can lead to substantially different numerical behavior. We also include in Figure 3.1 the result of exact HS CG computation (simulated via double reorthogonalization of the residual vectors). It shows that in the problem used for our illustration the effect of roundoff error is indeed substantial even for HS CG.

The coupled two-term recurrences used in Algorithm 1.1 can, in mathematical abstraction, be considered to be equivalent to the three-term recurrences (see Section 2)

$$x_i = \alpha_{i-1}r_{i-1} + x_{i-1} + \frac{\alpha_{i-1}\beta_{i-1}}{\alpha_{i-2}}\left(x_{i-1} - x_{i-2}\right), \tag{3.10}$$

$$r_i = -\alpha_{i-1}Ar_{i-1} + r_{i-1} + \frac{\alpha_{i-1}\beta_{i-1}}{\alpha_{i-2}}\left(r_{i-1} - r_{i-2}\right). \tag{3.11}$$

Various three-term CG implementations differ in the way the coefficients are computed in (3.10)-(3.11). The ST CG (see Algorithm 2.1) updates α_{i-1} using the relation,

$$\frac{1}{\alpha_{i-1}} = \frac{r_{i-1}^T A r_{i-1}}{\|r_{i-1}\|^2} - \frac{\beta_{i-1}}{\alpha_{i-2}}. \tag{3.12}$$

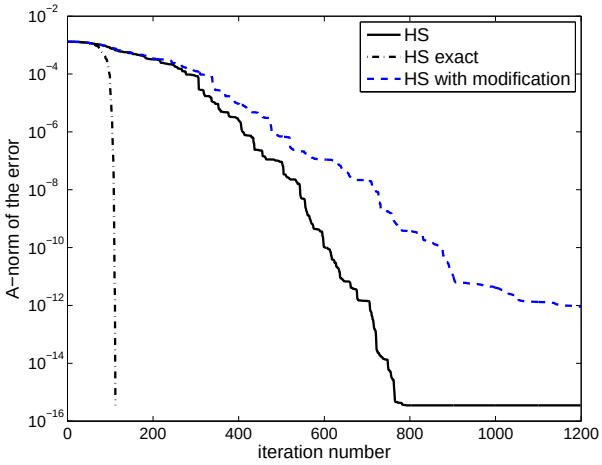

FIG. 3.1. *Replacing* (3.8) *by* (3.9) *in Algorithm 1.1 can significantly change behavior in finite precision arithmetic.*

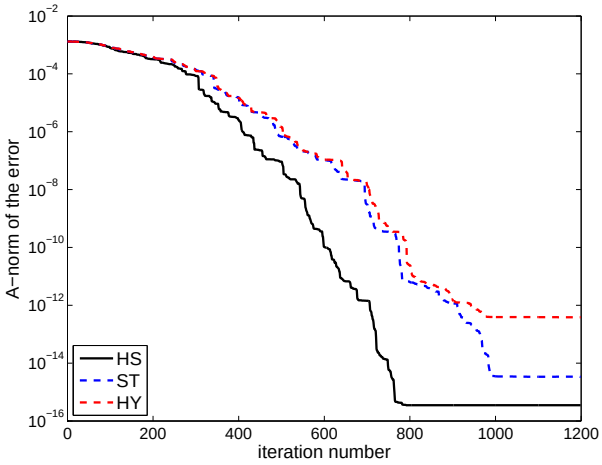

FIG. 3.2. *HS CG (Algorithm 1.1), ST CG (Algorithm 2.1), and Hageman-Young (HY) [23, p. 143] implementations.*

Once α_{i-1} is known, one can compute r_i and evaluate β_i as in Algorithm 1.1. This is probably the most straightforward way of determining the coefficients in (3.10)-(3.11). Note that the implementation by Hageman and Young [23, p. 143] also uses (3.12) (in a scaled form) to derive the updating formula for the coefficient

$$\rho_{i-1} = 1 + \frac{\alpha_{i-1}\beta_{i-1}}{\alpha_{i-2}}.$$

This results in more complicated updating formulas. It appears that the Hageman-Young implementation can be more affected by rounding errors than ST CG; see Figure 3.2.

The gap f_i between the recursively updated residual $\bar{r}_i$ and the true residual $b - A\bar{x}_i$ in three-term recurrence implementations of CG was analyzed by Gutknecht and Strakoš [22]. They showed that the local roundoff errors incurred using (3.10)-(3.11) can be significantly amplified as they propagate through the recurrences. The amplification factors can be essentially expressed in terms of the coefficients in the recurrences (3.10)-(3.11) that can be related to the coefficients in (3.1)-(3.3), which can exhibit, in general, rather irregular behavior. It is therefore shown that using two three-term recurrences can lead to worse maximal attainable accuracy than using three two-term recurrences (Algorithm 1.1). In addition, numerical experiments indicate that using three-term recurrences can also significantly delay convergence in comparison to coupled two-term recurrences; see Figure 3.2.

3.3. Adding auxiliary recurrences. One can also use the updating formula (3.12) to compute the coefficients α_i in HS CG. This allows the computations to be rearranged in a way that reduces the number of synchronization points to one. However, in HS CG, the vectors Ar_{i-1} are not available. So, either we have to perform two matrix-vectors products per iteration, or we need to use the additional recurrence for updating the vectors Ap_i,

$$Ap_i = Ar_i + \beta_i Ap_{i-1}, \quad \text{i.e.} \quad s_i = Ar_i + \beta_i s_{i-1}, \tag{3.13}$$

cf. also Algorithm 2.2. A CG variant that uses the latter option is shown in Algorithm 3.1.

ALGORITHM 3.1. **Conjugate Gradient method with the recursively updated images of direction vectors**
Input: *SPD matrix $A \in R^{N\times N}$, right-hand side vector $b \in R^N$, initial approximation $x_0 \in R^N$, maximum number of iterations nmax.*
Output: *Approximate solution x_n after the algorithm has been stopped.*

0. **Initialization:** $r_0 = b - Ax_0$, $p_0 = r_0$, $s_0 = Ap_0$
1. **for** $i = 1 : imax$ **do**
2. $\quad \alpha_{i-1} = \dfrac{(r_{i-1}, r_{i-1})}{(p_{i-1}, s_{i-1})}$
3. $\quad x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$
4. $\quad r_i = r_{i-1} - \alpha_{i-1} s_{i-1}$
5. $\quad$ *evaluate the stopping criterion*
6. $\quad \beta_i = \dfrac{(r_i, r_i)}{(r_{i-1}, r_{i-1})}$
7. $\quad p_i = r_i + \beta_i p_{i-1}$
8. $\quad s_i = Ar_i + \beta_i s_{i-1}$
9. **end do**

Adding a new recurrence without recomputing the recurrence coefficient β_i so that the computed vectors satisfy local orthogonality relations can negatively affect the numerical behavior of the resulting algorithm. Algorithm 3.1 is closely related to ChG CG (Algorithm 2.2), the only difference being in the way the coefficient α_{i-1} is computed which will be discussed in the next subsection. Note that other CG implementations that allow more parallelism than Algorithm 3.1 like GV CG (Algorithm 2.3) use even more auxiliary recurrences. In particular, Algorithm 2.3 updates the vectors $s_i = Ap_i$, $z_i = A^2 p_i$, $w_i = Ar_i$, and $q_i = A^2 r_i$. In problems sensitive to changes in the underlying distribution function that enlarge its support (see [38]), this can lead to a numerical disaster both with respect to delay of convergence and maximal attainable accuracy; see Figure 3.3. It should be noted that the sensitive problems mentioned above are in particular those with large outlying eigenvalues; for detailed discussion see [15].

3.4. Recurrence coefficients. As mentioned above, the recurrence coefficients in two-term as well as in three-term variants of CG can be computed in many different ways. We will illustrate the effects of changing the computation of the recurrence coefficients using a modified version of Algorithm 1.1 where the formula for updating α_{i-1} is replaced by (3.12) without adding an auxiliary recurrence for Ar_{i-1} (the vectors Ar_{i-1}, Ap_{i-1} are computed by explicit matrix-vector multiplication). This variant is shown in Algorithm 3.2

ALGORITHM 3.2. **Conjugate Gradient method with modified coefficient computation**
Input: *SPD matrix $A \in R^{N\times N}$, right-hand side vector $b \in R^N$, initial approximation $x_0 \in R^N$, maximum number of iterations nmax.*
Output: *Approximate solution x_n after the algorithm has been stopped.*

0. **Initialization:** $r_0 = b - Ax_0$, $p_0 = r_0$
1. **for** $i = 1 : imax$ **do**
2. $\quad \alpha_{i-1} = \left(\dfrac{r_{i-1}^T A r_{i-1}}{\|r_{i-1}\|^2} - \dfrac{\beta_{i-1}}{\alpha_{i-2}} \right)^{-1}$
3. $\quad x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$

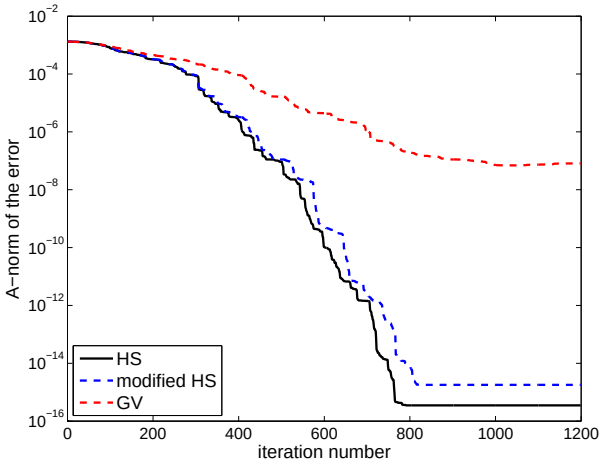

FIG. 3.3. *HS CG (Algorithm 1.1), modified HS CG with the recursive update* (3.13) *(Algorithm 3.1), and GV CG (Algorithm 2.3).*

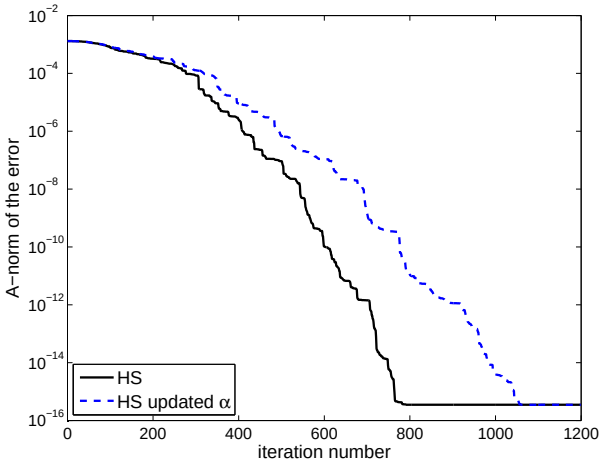

FIG. 3.4. *HS CG (Algorithm 1.1), and the modified HS CG with explicit matrix-vectors multiplications* Ar_{i-1}, Ap_{i-1} *and the formula* (3.12) *for updating* α_{i-1}*; see Algorithm 3.2.*

4. $r_i = r_{i-1} - \alpha_{i-1} A p_{i-1}$
5. *evaluate the stopping criterion*
6. $\beta_i = \dfrac{(r_i, r_i)}{(r_{i-1}, r_{i-1})}$
7. $p_i = r_i + \beta_i p_{i-1}$
9. **end do**

Notice that in this case, (3.4) still holds, so the gap f_i can be bounded analogously to (3.7). In fact, (3.4) will hold regardless of how α_{i-1} is computed as long as the same α_{i-1} is used in updating both the recurrence for x_i and for r_i; this has been pointed out by, e.g., Greenbaum [19]. In Figure 3.4 we observe a significant delay of convergence of Algorithm 3.2 in comparison to HS CG. The maximal attainable accuracy seems to be similar for both algorithms.

3.5. Residual replacement strategy. To attain a higher accuracy in terms of the norm of the residual vector, several authors suggest to synchronize the recursively computed residual $\bar{r}_i$ and the true residual $b - A\bar{x}_i$ by replacing $\bar{r}_i$ by a direct computation of $b - A\bar{x}_i$ in some iterations; see, e.g., [49], [62], [61], [3].

Since the direct computation of $b - A\bar{x}_i$ is rather expensive, the residual replacement should occur

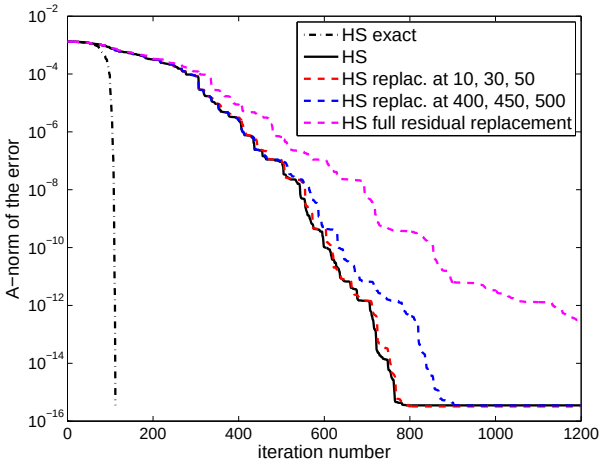

FIG. 3.5. *The HS CG implementation and the modified CG implementations with residual replacement.*

only at a few iteration steps. As also emphasized in [62], these steps should be selected so that the residual replacement does not have an effect on the rate of convergence. Based on that the authors in [62] develop an updating strategy where they set a threshold and carry out the residual replacement when a certain quantity containing the terms $\mathcal{O}(\varepsilon)(\|A\|\|\bar{x}_i\| + \|\bar{r}_i\|)$ reaches the given threshold. The justification of the proposed heuristics assumes that the matrix composed of the computed residual vectors is of full column rank, which may not hold in practical computations.

In Figure 3.5 we plot the A-norm of the error in the exact computation using HS CG simulated as above (see Figure 3.1). The orthogonality and numerical linear independence among the computed residual vectors is lost in HS CG (solid line) very quickly, which results in a significant delay of convergence. Here, just to illustrate the possible effects of the residual replacement on the rate of convergence, we first carry out three residual replacements at steps $n = 10$, $n = 30$, and $n = 50$ before the linear independence of the computed residual vectors is lost. In this case, we indeed do not observe a significant difference in the convergence rate (see the red dashed line in Figure 3.5). However, three replacements after the loss of linear independence (at steps $n = 400$, $n = 450$, and $n = 500$) cause a significant delay of convergence in comparison to HS CG due to the introduction of large perturbations (relative to the residual norm at these steps) in the recurrence relations; see the blue dashed line. In Figure 3.5 we also plot the A-norm of the error in the algorithm with full residual replacement, in which $\bar{r}_i$ is replaced by the direct computation of $b - A\bar{x}_i$ in each iteration step. This costly strategy keeps the gap between the true and updated residuals on the level $\mathcal{O}(\varepsilon)\|A\|(\|\bar{x}_i\| + \|x\|)$. As expected based on the results in [62], this causes a significant deterioration of convergence rate. Similar behavior in terms of the delay of convergence when using various residual replacement strategies has also been observed by Meurant [36, Section 6.4], who suggests an optional residual replacement heuristic. Residual replacement strategies deserve further attention and thorough analysis.

We point out that for this chosen test problem, HS CG already attains an accurate approximate solution, and therefore we do not see in Figure 3.5 further improvement from the use of residual replacement.

4. Example numerical stability analysis. In the following we give an example of a numerical stability analysis for one particular CG implementation. We will focus on the coupled two-term recurrence version extended by the recurrence (3.13) for the vectors $s_i = Ap_i$; see Algorithm 3.1. The purpose is to show through this simple and artificial example the methodology that can be used, with appropriate extensions, to perform full-scale analysis of algorithms like GV CG. Such full-scale analysis is out of the scope of this paper. We will show that the maximal attainable accuracy of Algorithm 3.1 can be significantly worse than in HS CG (Algorithm 1.1). It is worth pointing out that in Figure 3.3, this

difference is not substantial, which underlines the need for extensive experiments in order to gather reliable numerical evidence about the behavior of any implementation. Introducing the notation

$$P_n = (p_0, \ldots, p_{n-1}),\ R_n = (r_0, \ldots, r_{n-1}) \ \text{ and } \ S_n = (s_0, \ldots, s_{n-1}),$$

the recurrences (3.8) and (3.13) (lines 7 and 8 in Algorithm 3.1) for $i = 1, \ldots, n-1$ can be written

$$R_n = P_n U_n, \quad AR_n = S_n U_n, \tag{4.1}$$

where U_n is the n-by-n unit upper bidiagonal matrix of the form

$$U_n = \begin{pmatrix} 1 & -\beta_1 & 0 & 0 \\ 0 & 1 & \ddots & 0 \\ \vdots & \ddots & 1 & -\beta_{n-1} \\ 0 & \ldots & 0 & 1 \end{pmatrix}. \tag{4.2}$$

The quantities computed in finite precision arithmetic (denoted here with bars[1]) satisfy

$$\bar{x}_i = \bar{x}_{i-1} + \bar{\alpha}_{i-1}\bar{p}_{i-1} + \delta x_i, \qquad \|\delta x_i\| \le \mathcal{O}(\epsilon)\max\{\|\bar{x}_{i-1}\|, \|\bar{x}_i\|\}, \tag{4.3}$$

$$\bar{r}_i = \bar{r}_{i-1} - \bar{\alpha}_{i-1}\bar{s}_{i-1} + \delta r_i, \qquad \|\delta r_i\| \le \mathcal{O}(\epsilon)\max\{\|\bar{r}_{i-1}\|, \|\bar{\alpha}_{i-1}\bar{s}_{i-1}\|\}, \tag{4.4}$$

$$\bar{p}_i = \bar{r}_i + \bar{\beta}_i\bar{p}_{i-1} + \delta p_i, \qquad \|\delta p_i\| \le \mathcal{O}(\epsilon)\max\{\|\bar{r}_i\|, \|\bar{\beta}_i\bar{p}_{i-1}\|\}, \tag{4.5}$$

$$\bar{s}_i = A\bar{r}_i + \bar{\beta}_i\bar{s}_{i-1} + \delta s_i, \qquad \|\delta s_i\| \le \mathcal{O}(\epsilon)\max\{\|A\|\|\bar{r}_i\|, \|\bar{\beta}_i\bar{s}_{i-1}\|\}, \tag{4.6}$$

where $i = 1, \ldots, n-1$. Similarly to (4.1), the recurrences (4.5) and (4.6) for $i = 1, \ldots, n-1$ can be rewritten into

$$\bar{R}_n = \bar{P}_n\bar{U}_n - \Delta P_n, \quad A\bar{R}_n = \bar{S}_n\bar{U}_n - \Delta S_n, \tag{4.7}$$

where $\Delta P_n = (\delta p_0, \ldots, \delta p_{n-1})$ and $\Delta S_n = (\delta s_0, \ldots, \delta s_{n-1})$. It is clear from (4.5) and (4.6) that

$$\begin{aligned} \|\delta p_i\| &\le \mathcal{O}(\epsilon)(\|\bar{r}_i\| + \|\bar{\beta}_i\bar{p}_{i-1}\|) \le \mathcal{O}(\epsilon)(\|\bar{R}_n\| + \|\bar{P}_n\|\|\bar{U}_n\|), \\ \|\delta s_i\| &\le \mathcal{O}(\epsilon)(\|A\|\|\bar{r}_i\| + \|\bar{\beta}_i\bar{s}_{i-1}\|) \le \mathcal{O}(\epsilon)(\|A\|\|\bar{R}_n\| + \|\bar{S}_n\|\|\bar{U}_n\|), \end{aligned}$$

for each $i = 0, \ldots, n-1$. Since $\|\Delta P_n\| \le \|\Delta P_n\|_F$ and $\|\Delta S_n\| \le \|\Delta S_n\|_F$, we obtain the bounds

$$\begin{aligned} \|\Delta P_n\| &\le \mathcal{O}(\epsilon)(\|\bar{R}_n\| + \|\bar{P}_n\|\|\bar{U}_n\|), \\ \|\Delta S_n\| &\le \mathcal{O}(\epsilon)(\|A\|\|\bar{R}_n\| + \|\bar{S}_n\|\|\bar{U}_n\|). \end{aligned}$$

The gap f_n between the recursively updated residual $\bar{r}_n$ and the true residual $b - A\bar{x}_n$ can be written in the form

$$\begin{aligned} f_i &= \bar{r}_i - (b - A\bar{x}_i) \\ &= \bar{r}_{i-1} - \bar{\alpha}_{i-1}\bar{s}_{i-1} + \delta r_i - (b - A(\bar{x}_{i-1} + \bar{\alpha}_{i-1}\bar{p}_{i-1} + \delta x_i)) \\ &= f_{i-1} - \bar{\alpha}_{i-1}g_{i-1} + \delta r_i + A\delta x_i, \end{aligned} \tag{4.8}$$

where

$$g_{i-1} = \bar{s}_{i-1} - A\bar{p}_{i-1}$$

[1] Mathematically (assuming exact arithmetic), $s_i = Ap_i$. In finite precision computations, however, $\bar{s}_i$ computed via (3.13) can no longer be identified with the multiplication $A\bar{p}_i$. This point is very important. Examples of confusion mixing mathematically equivalent but computationally different quantities can be seen in recently published literature.

denotes the gap between the recursively computed vector $\bar{s}_{i-1}$ and the computed direction vector $\bar{p}_{i-1}$ multiplied by A. Hence, by induction,

$$f_n = f_0 - \sum_{i=0}^{n-1} \bar{\alpha}_i g_i + \sum_{i=1}^{n} \delta r_i + A \sum_{i=1}^{n} \delta x_i. \tag{4.9}$$

We see from (4.9) that f_n is a superposition of the effects of local errors δx_i and δr_i and the differences g_i multiplied by the coefficients $\bar{\alpha}_i$ throughout all iteration steps $i = 0, \ldots, n-1$. We will show that the sizes of g_i may play a decisive role in the maximum attainable accuracy of Algorithm 3.1. Introducing the notation $G_n = (g_0, \ldots, g_{n-1})$ and $d_n = (\bar{\alpha}_0, \ldots, \bar{\alpha}_{n-1})^T$, the formula (4.9) can be written as

$$f_n = f_0 + \sum_{i=1}^{n} (\delta r_i + A\delta x_i) - G_n d_n. \tag{4.10}$$

We will show that the size of g_i can be significantly larger than the error in the floating point computation $\mathrm{fl}(A\bar{p}_i)$, i.e.,

$$\|\mathrm{fl}(A\bar{p}_i) - A\bar{p}_i\| \leq O(\epsilon)\|A\|\|\bar{p}_i\|.$$

From (4.7) we have

$$G_n = \bar{S}_n - A\bar{P}_n = (\Delta S_n - A\Delta P_n)\bar{U}_n^{-1}.$$

Taking the norms and assuming that the matrix $\bar{U}_n$ is numerically nonsingular with $\mathcal{O}(\epsilon)\kappa(\bar{U}_n) < 1$, we get

$$\begin{aligned}
\|G_n\| &\leq (\|\Delta S_n\| + \|A\|\|\Delta P_n\|)\,\|\bar{U}_n^{-1}\| \\
&\leq \mathcal{O}(\epsilon)\left[(\|\bar{S}_n\| + \|A\|\|\bar{P}_n\|)\|\bar{U}_n\| + \|A\|\|\bar{R}_n\|\right]\|\bar{U}_n^{-1}\| \\
&\leq \mathcal{O}(\epsilon)\left[(\|G_n\| + 2\|A\|\|\bar{P}_n\|)\|\bar{U}_n\| + \|A\|\|\bar{R}_n\|\right]\|\bar{U}_n^{-1}\| \\
&\leq \frac{\mathcal{O}(\epsilon)}{1-\mathcal{O}(\epsilon)}[\kappa(\bar{U}_n)\|A\|\|\bar{P}_n\| + \|A\|\|\bar{R}_n\|\|\bar{U}_n^{-1}\|].
\end{aligned} \tag{4.11}$$

The first term in the bound (4.11) is larger than $\mathcal{O}(\epsilon)\|A\|\|\bar{P}_n\|$ by a factor of $\kappa(\bar{U}_n)$ and it suggests that the local errors at the individual steps contributing to the size of the difference $f_n = \bar{r}_n - (b - A\bar{x}_n)$ in (4.10) can potentially be amplified by the entries in $\bar{U}_n^{-1}$. Since the matrix $\bar{U}_n$ is unit upper bidiagonal, the entries of its inverse can be expressed in terms of the products of the coefficients $\bar{\beta}_i$, for $i = 1, \ldots, n-1$, i.e.,

$$\bar{U}_n^{-1} = \begin{pmatrix}
1 & \bar{\beta}_1 & \ldots & \ldots & \bar{\beta}_1\bar{\beta}_2\ldots\bar{\beta}_{n-1} \\
0 & 1 & \bar{\beta}_2 & \ldots & \bar{\beta}_2\ldots\bar{\beta}_{n-1} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
\vdots & & \ddots & 1 & \bar{\beta}_{n-1} \\
0 & \ldots & \ldots & 0 & 1
\end{pmatrix}. \tag{4.12}$$

In exact arithmetic, the coefficient β_i in Algorithm 3.1 is equal to $\beta_i = \|r_i\|^2/\|r_{i-1}\|^2$ and the multiplicative factors $\beta_i\beta_{i+1}\ldots\beta_j$ are equal to

$$\beta_i\beta_{i+1}\ldots\beta_j = \frac{\|r_j\|^2}{\|r_{i-1}\|^2},\ i < j.$$

Consequently, due to possible oscillations of the CG residual norms, the factors $\|r_j\|^2/\|r_{i-1}\|^2$ can for some $i < j$ be rather large. Similar reasoning can also be applied to the norm of the updated residuals $\|\bar{r}_\ell\|$, leading potentially to large products of the computed coefficients $\bar{\beta}_i\bar{\beta}_{i+1}\ldots\bar{\beta}_j$ in (4.12) and to the

ill-conditioned matrix $\bar{U}_n$. Consequently, a dramatic amplification of the local errors can be expected and the maximum attainable accuracy in computations using Algorithm 3.1 can be significantly worse than when the standard Algorithm 1.1 is used.

This analysis resembles the results for the three-term CG implementations presented in [22]. It demonstrates that a seemingly innocuous change in the CG implementation can turn the two-term recurrence CG into a three-term recurrence CG with substantially different (and often worse) numerical behavior.

5. Conclusions. As mentioned in the Introduction, Krylov subspace methods are highly nonlinear in the input data A,b. This makes them incredibly powerful methods in terms of being efficient solvers, but also makes them inherently difficult to analyze and understand. In this work, we have studied the effects of various modifications to the standard CG method on convergence rate and attainable accuracy for an example problem. The modifications studied include using 3-term recurrences, adding auxiliary recurrences, changing the way in which the recurrence coefficients are computed, and using residual replacement. As an example, we outline a way to perform stability analysis for a particular CG variant that uses an additional auxiliary recurrence for the product Ap_i. This demonstrates how one might perform a numerical stability analysis for more complex pipelined CG variants such as GV CG.

As we as a community push toward the goal of exascale-level computational science, it is important that we not forget the bigger picture of what we want to accomplish: enabling scientific insight, analysis, and discovery through the use of computation. Too often it is the case that new variants of Krylov subspace methods are derived with the goal of optimizing performance, and only speed per iteration is reported as a performance metric. This fails to capture the whole picture of the method's effectiveness within the context of a scientific application. If the modifications to the method cause a convergence delay with a greater effect than the per-iteration performance improvement, the modified method may actually be slower than the standard approach in a practical setting. If the modifications to the method cause a significant decrease in attainable accuracy, then the method may have no use in some scientific applications, making any gains in iteration speed for naught.

It is clear then that in the landscape of high-performance computing, the design and implementation of iterative solvers requires a *holistic* approach. In selecting the right method and parameters to use for a given problem, we must consider the expected time per iteration as well as the numerical stability and convergence properties, which we have seen depend heavily on the particular recurrences used in the method as well as numerical properties of the input data. We must also consider the performance and use of the Krylov subspace method within the context of the overall scientific application.

Related to that, it is very rare that Krylov subspace methods can be used in practice without some transformation of the problem, generally called preconditioning. Technically, this involves introducing additional matrix-vector operations within each iteration step (typically solving a linear algebraic system with the matrix representing the preconditioner). In numerical stability analysis, preconditioning can be included using the standard methodology developed for the unpreconditioned methods. Since preconditioning is closely linked with discretization (see, e.g., [31]), high-performance computing efficiency may stimulate new developments that will consider this link instead of elaborating only on the level of the algebraic system arising from the separated discretization step.

In light of this, we can give a few recommendations on what must be done when deriving a new variant of a Krylov subspace method (or a new technique to be used together with an existing variant, e.g., residual replacement). When deriving a new Krylov subspace method variant, one must analyze the effects on attainable accuracy and convergence rate. Ideally, this should be done both by deriving theoretical results and by performing a thorough experimental evaluation of the method. We stress that theoretical results on convergence rate *assuming exact computation* cannot be applied to inexact computations unless such application is rigorously justified by a thorough analysis (this point is of particular importance for methods using short recurrences). An experimental study should include problems of various sizes and numerical properties, as well as tests for those problems on various computing platforms. It is also worth investigating whether the new variant is fundamentally different than other variants previously studied in

the literature, or whether such differences are only superficial. For example, as we have shown in Section 3.3, certain modified variants of the coupled 2-term recurrence version of CG very closely resemble the 3-term recurrence method ST CG.

We stress that we remain optimistic that pipelined Krylov subspace methods and other synchronization-reducing variants can certainly be useful in achieving practical speedups for certain problems in a number of application domains; there are likely many problems for which the convergence delay in methods like GV CG does not negate the savings from reduced synchronization, and for which the resulting attainable accuracy is acceptable. However, as numerical analysts, it is our task to identify how significant these effects can be in finite precision, to identify classes of problems for which such methods can be used, and to improve our overall understanding of the tradeoffs between reducing synchronization and maintaining the numerical properties of the method. As we work toward enabling computational science at the exascale level, it is our hope that this manuscript helps guide the way toward the future design and analysis of efficient Krylov subspace methods.

REFERENCES

[1] G. M. Ballard, *Avoiding Communication in Dense Linear Algebra*, ProQuest LLC, Ann Arbor, MI, 2013. Thesis (Ph.D.)–University of California, Berkeley.

[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.

[3] 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.

[4] E. C. Carson, *Communication-Avoiding Krylov Subspace Methods in Theory and Practice*, ProQuest LLC, Ann Arbor, MI, 2015. Thesis (Ph.D.)–University of California, Berkeley.

[5] Y. Chauvet and G. Meurant, *Multitasking on the CRAY X-MP*, The Journal of Systems and Software, 2 (1986), pp. 17–20.

[6] A. T. Chronopoulos and C. W. Gear, *On the efficient implementation of preconditioned s-step conjugate gradient methods on multiprocessors with memory hierarchy*, Parallel Comput., 11 (1989), pp. 37–53.

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

[8] E. D'Azevedo, V. Eijkhout, and C. Romine, *Reducing communication costs in the conjugate gradient algorithm on distributed memory multiprocessors*, Technical Report ORNL TM/12192, Oak Ridge National Laboratory, 1992.

[9] ———, *Lapack working note 56: Reducing communication costs in the conjugate gradient algorithm on distributed memory multiprocessors*, Technical Report, University of Tennessee, Knoxville, TN, USA, 1993.

[10] J. Dongarra et al., *The international exascale software project roadmap*, Int. J. High Perform. Comput. Appl., 25 (2011), pp. 3–60.

[11] I. Duff, R. Grimes, and J. Lewis, *Users' guide for the Harwell-Boeing sparse matrix collection (release I)*, 1992.

[12] M. Engeli, Th. Ginsburg, H. Rutishauser, and E. Stiefel, *Refined iterative methods for computation of the solution and the eigenvalues of self-adjoint boundary value problems*, Mitt. Inst. Angew. Math. Zürich. No., 8 (1959), p. 107.

[13] B. Fischer, *Polynomial based iteration methods for symmetric linear systems*, Wiley-Teubner Series Advances in Numerical Mathematics, John Wiley & Sons, Ltd., Chichester; B. G. Teubner, Stuttgart, 1996.

[14] D. Gannon and J. van Rosendale, *Parallel architectures for iterative methods on adaptive, block structured grids*, in Elliptic problem solvers, II (Monterey, Calif., 1983), Academic Press, Orlando, FL, 1984, pp. 93–104.

[15] T. Gergelits and Z. Strakoš, *Composite convergence bounds based on Chebyshev polynomials and finite precision conjugate gradient computations*, Numer. Algorithms, 65 (2014), pp. 759–782.

[16] P. Ghysels, T. 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.

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

[18] A. Greenbaum, *Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences*, Linear Algebra Appl., 113 (1989), pp. 7–63.

[19] ———, *Estimating the attainable accuracy of recursively computed residual methods*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 535–551.

[20] A. Greenbaum and Z. Strakoš, *Predicting the behavior of finite precision Lanczos and conjugate gradient computations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 121–137.

[21] W. Gropp, *Update on libraries for blue waters, a presentation on the third workshop of the INRIA-Illinois joint-laboratory on petascale computing, June 21-24, 2010, Bordeaux, France,* `http://jointlab-pc.ncsa.illinois.edu/events/workshop3/agenda.html`.

[22] 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 (electronic).

[23] L. A. Hageman and D. M. Young, *Applied iterative methods*, Academic Press, Inc. [Harcourt Brace Jovanovich, Publishers], New York-London, 1981. Computer Science and Applied Mathematics.
[24] M. R. Hestenes, *Iterative methods for solving linear equations*, Report 52-9, NAML, 1951. Reprinted in Journal of Opeimization Theory and Applications, Volume 11, pp. 323–334, 1973.
[25] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. of Research of the National Bureau of Standards, 49 (1952), pp. 409–435.
[26] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, second ed., 2002.
[27] M. Hoemmen, *Communication-avoiding Krylov subspace methods*, ProQuest LLC, Ann Arbor, MI, 2010. Thesis (Ph.D.)–University of California, Berkeley.
[28] L. Johnsson, *Highly concurrent algorithms for solving linear systems of equations*, Tech. Report 5079-TR-83, California Institute of Technology, 1983.
[29] ———, *Highly concurrent algorithms for solving linear systems of equations*, in Elliptic problem solvers, II (Monterey, Calif., 1983), Academic Press, Orlando, FL, 1984, pp. 105–126.
[30] J. Liesen and Z. Strakoš, *Krylov Subspace Methods: Principles and Analysis*, Oxford University Press, 2013.
[31] J. Málek and Z. Strakoš, *Preconditioning and the Conjugate Gradient Method in the Context of Solving PDEs*, SIAM Spotlight Series, SIAM, Philadelphia, 2015.
[32] K. McManus, S. Johnson, and M. Cross, *Communication latency hiding in a parallel conjugate gradient method*, in Eleventh International Conference on Domain Decomposition Methods (London, 1998), DDM.org, Augsburg, 1999, pp. 306–313 (electronic).
[33] G. Meurant, *Numerical experiments for the preconditioned conjugate gradient method on the Cray-X-MP/2*, tech. report, University of California, Berkeley, CA, 1984.
[34] ———, *Multitasking the conjugate gradient method on the CRAY X-MP/48*, Parallel Comput., 5 (1987), pp. 267–280.
[35] ———, *Computer Solution of Large Linear Systems*, Elsevier, Amsterdam – Lausanne – New York – Oxford – Shannon – Singapore – Tokyo, 1999.
[36] ———, *The Lanczos and conjugate gradient algorithms*, vol. 19 of Software, Environments, and Tools, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2006. From theory to finite precision computations.
[37] G. Meurant and Z. Strakoš, *The Lanczos and conjugate gradient algorithms in finite precision arithmetic*, Acta Numer., 15 (2006), pp. 471–542.
[38] D. P. O'Leary, Z. Strakoš, and P. Tichý, *On sensitivity of Gauss-Christoffel quadrature*, Numer. Math., 107 (2007), pp. 147–174.
[39] C. Paige, *The computation of eigenvalues and eigenvectors of very large sparse matrices*, PhD thesis, London University, London, UK, 1971.
[40] ———, *Computational variants of the Lanczos method for the eigenproblem*, IMA J. Appl. Math., 10 (1972), pp. 373–381.
[41] ———, *Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix*, IMA J. Appl. Math., 18 (1976), pp. 341–349.
[42] ———, *Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem*, Linear Algebra Appl., 34 (1980), pp. 235–258.
[43] J. K. Reid, *On the method of conjugate gradients for the solution of large sparse systems of linear equations*, in Large sparse sets of linear equations (Proc. Conf., St. Catherine's Coll., Oxford, 1970), Academic Press, London, 1971, pp. 231–254.
[44] J. B. Rosser, *Rapidly converging iterative methods for solving linear equations*, in Simultaneous linear equations and the determination of eigenvalues, National Bureau of Standards Applied Mathematics Series, No. 29, (Proceedings of a symposium held August 23-25, 1951, in Los Angeles, California), U.S. Government Printing Office, Washington, D. C., 1953, pp. 59–64.
[45] Y. Saad, *Practical use of polynomial preconditionings for the conjugate gradient method*, SIAM J. on Scientific and Statistical Computing, 6 (1985), pp. 865–881.
[46] ———, *Krylov subspace methods on supercomputers*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1200–1232. Sparse matrix algorithms on supercomputers.
[47] V. Simoncini and D. Szyld, *Theory of inexact Krylov subspace methods and applications to scientific computing*, SIAM J. Sci. Comput., 25 (2003), pp. 454–477.
[48] ———, *On the occurrence of superlinear convergence of exact and inexact Krylov subspace methods*, SIAM Review, 47 (2005), pp. 247–272.
[49] G. L. G. Sleijpen and H. A. van der Vorst, *Reliable updated residuals in hybrid Bi-CG methods*, Computing, 56 (1996), pp. 141–163.
[50] 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.
[51] E. Stiefel, *Ausgleichung ohne Aufstellung der Gaußschen Normalgleichungen*, Wiss. Z. Technische Hochschule Dresden, 2 (1952/53), pp. 441–442.
[52] ———, *Relaxationsmethoden bester Strategie zur Lösung linearer Gleichungssysteme*, Commentarii Mathematici Helvetici, 29 (1955), pp. 157–179.
[53] Z. Strakoš and P. Tichý, *On error estimation in the conjugate gradient method and why it works in finite precision computations*, Electron. Trans. Numer. Anal., 13 (2002), pp. 56–80 (electronic).

[54] Z. Strakoš, *Performance of the EC 2345 array processor*, Computers and Artificial Intelligence, 4 (3) (1985), pp. 273–284.

[55] ———, *Effectivity and optimizing of algorithms and programs on the host-computer/array-processor system*, Parallel Computing, 4 (2) (1987), pp. 189–207.

[56] ———, *On the real convergence rate of the conjugate gradient method*, Linear Algebra Appl., 154/156 (1991), pp. 535–549.

[57] G. Szegö, *Orthogonal polynomials*, Colloquium Publications Colloquium Publications Amer Mathematical Soc, American Mathematical Society, 4th ed., 1939.

[58] J. van den Eshof, *Nested Iteration methods for Nonlinear Matrix Problems*, PhD thesis, Sept. 2003.

[59] J. van den Eshof and G. Sleijpen, *Inexact Krylov subspace methods for linear systems*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 125–153.

[60] H. van der Vorst, *Parallel iterative solution methods for linear systems arising from discretized PDE's, special course on parallel computing in CFD*, Tech. Report URN:NBN:NL:UI:10-1874-1640, France Workshop Lecture Notes, http://dspace.library.uu.nl/handle/1874/1640, Utrecht University, 2001.

[61] H. A. van der Vorst, *Iterative Krylov methods for large linear systems*, vol. 13 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2003.

[62] 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 (electronic).

[63] J. van Rosendale, *Minimizing inner product data dependencies in conjugate gradient iteration*, Tech. Report 172178, ICASE-NASA, 1983.