PHILOSOPHICAL TRANSACTIONS A

royalsocietypublishing.org/journal/rsta

Opinion piece

Cite this article: Carson E, Strakoš Z. 2020 On the cost of iterative computations. *Phil. Trans. R. Soc. A* **378**: 20190050.
http://dx.doi.org/10.1098/rsta.2019.0050

Accepted: 23 July 2019

One contribution of 15 to a discussion meeting issue 'Numerical algorithms for high-performance computational science'.

Subject Areas:
computational mathematics, mathematical modelling, computer modelling and simulation

Keywords:
computational mathematics, matrix computations, high performance computing, iterative methods

Author for correspondence:
Erin Carson
e-mail: carson@karlin.mff.cuni.cz

On the cost of iterative computations

Erin Carson and Zdeněk Strakoš

Faculty of Mathematics and Physics, Charles University, Prague 18675, Czech Republic

EC, 0000-0001-9469-7467

With exascale-level computation on the horizon, the art of predicting the cost of computations has acquired a renewed focus. This task is especially challenging in the case of iterative methods, for which convergence behaviour often cannot be determined with certainty *a priori* (unless we are satisfied with potentially outrageous overestimates) and which typically suffer from performance bottlenecks at scale due to synchronization cost. Moreover, the amplification of rounding errors can substantially affect the practical performance, in particular for methods with short recurrences. In this article, we focus on what we consider to be key points which are crucial to understanding the cost of iteratively solving linear algebraic systems. This naturally leads us to questions on the place of numerical analysis in relation to mathematics, computer science and sciences, in general.

This article is part of a discussion meeting issue 'Numerical algorithms for high-performance computational science'.

1. Introduction

In this work, we focus on solving linear algebraic systems $Ax = b$. Such problems are ubiquitous throughout the computational and data sciences, and thus designing methods, algorithms and software that aim to solve such problems accurately and efficiently is the focus of much effort in the field of high-performance computing and numerical mathematics. In particular, we are concerned with the case where an approximate solution to $Ax = b$ is computed iteratively, which is the approach generally used when A is large and sparse (it need not even be explicitly available) or when a solution adequate for the application requires less accuracy than offered by the available machine precision.

THE ROYAL SOCIETY PUBLISHING

In the era of supercomputing hardware capable of exascale-level computation, developing effective approaches for the iterative solution of such systems will require the development of new methods, algorithms and implementations capable of exploiting the underlying hardware of the machine in an effort to optimize the computational approach for a given machine and a given instance of data A and b. This invokes the seemingly simple question: *What is the cost of an iterative computation?*

In this work, we aim to illuminate long-standing results that are often missed or misused in efforts to optimize iterative computations. We claim no novel technical contribution; the contribution of this work is rather to explain and clarify important considerations for those working in related areas. Towards this goal, we use the example of the conjugate gradient method (CG). This choice is, in our opinion, appropriate for several reasons. First, the method has a prominent role as a representative of the Krylov subspace methods, which are ubiquitous in computational applications. Second, CG has incredibly deep roots and interconnections, being linked with classical mathematical objects such as continued fractions, orthogonal polynomials and Gauss quadrature. Furthermore, CG is used as an important example in various works aimed at reconsidering the foundations of numerical computations. CG is also now used as a benchmark to access the speed of supercomputers [1].

CG dates back to the work of Hestenes & Stiefel [2] and Lanczos [3]. There has since been a wealth of work towards understanding its mathematical properties and the numerical properties of algorithms which implement it. It is not the purpose of this work to give a historical account of the literature on CG, including seminal works by Axelsson, Brezinski, Golub, Grcar, Greenbaum, Meurant, O'Leary, Paige, Parlett, Saad, Simon, van der Sluis, van der Vorst, and many others. We refer the curious reader to the book [4] for historical notes and relevant background.

2. What is a computation?

In order to discuss the cost of an iterative computation, we must first carefully explain what we mean by a computation. We have deliberately chosen to avoid the word 'algorithm' in the title of this manuscript for reasons that we hope to make clear in this section.

The goal of any computation is to solve some *problem*. In the example, we will use throughout this manuscript, the problem is solving an $N \times N$ linear algebraic system

$$Ax = b. \tag{2.1}$$

The definition of *problem* should, however, be considered in a wider context. The matrix A and vector b typically arise in some intermediate stage of the solution of the original problem. The original problem can come from many vastly different applications; for example, a partial differential equation (an infinite dimensional problem), or some discrete collected data. Regardless of the particular application, it is almost always the case that the origin of the problem imposes certain mathematical structure present in A and b (which may be immediately discernible in some cases and less apparent in others).

A computation involves some particular *method* for solving the problem. There are many possible methods for solving $Ax = b$ to choose from depending on the structure of the matrix A and its numerical properties. In the case that A is of a moderate size and dense (i.e. it is not advantageous to consider zero entries or low-rank structure), a direct method such as LU factorization might be used. In the case we focus on here, where A is a very large, sparse and symmetric positive definite (SPD) matrix, CG is an obvious choice.

CG is in the class of Krylov subspace methods. A Krylov subspace method on the problem $Ax = b$ implicitly constructs in each iteration an approximation of A by restricting and projecting the matrix viewed as the operator A onto the ith Krylov subspace

$$\mathcal{K}_i(A, r_0) \equiv \text{span}\{r_0, Ar_0, \ldots, A^{i-1}r_0\},$$

where $r_0 = b - Ax_0$ is the initial residual given initial approximate solution x_0. The matrix A can then be, on the ith Krylov subspace, represented by the tridiagonal Jacobi matrix T_i; see [4, (ch. 3)].

Algorithm 1 . Conjugate Gradient variant of Hestenes and Stiefel (HSCG).

1: **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*, stopping criterion.
2: **Output:** Approximate solution x_i after the algorithm has been stopped.
3: **Initialization:** $r_0 = b - Ax_0$, $p_0 = r_0$
4: **for** $i = 1 : nmax$ **do**
5: $\alpha_{i-1} = (r_{i-1}^T r_{i-1})/(p_{i-1}^T A p_{i-1})$
6: $x_i = x_{i-1} + \alpha_{i-1} p_{i-1}, \quad r_i = r_{i-1} - \alpha_{i-1} A p_{i-1}$
7: Evaluate the stopping criterion.
8: $\beta_i = (r_i^T r_i)/(r_{i-1}^T r_{i-1})$
9: $p_i = r_i + \beta_i p_{i-1}$
10: **end for**

The CG approximate solution $x_i \in x_0 + \mathcal{K}_i(A, r_0)$ is determined by the optimality condition

$$\|x - x_i\|_A = \min_{z \in x_0 + \mathcal{K}_i(A,r_0)} \|x - z\|_A, \tag{2.2}$$

which can be equivalently written as

$$\|x - x_i\|_A = \|\rho_i^{\mathrm{CG}}(A)(x - x_0)\|_A = \min_{\rho(0)=1, \deg(\rho)\le i} \|\rho(A)(x - x_0)\|_A, \tag{2.3}$$

with the ith CG polynomial $\rho_i^{\mathrm{CG}}(\lambda)$ determined by the eigenvalues $\theta_1^{(i)}, \ldots, \theta_i^{(i)}$ of the Jacobi matrix T_i via

$$\rho_i^{\mathrm{CG}}(\lambda) = \frac{(\lambda - \theta_1^{(i)}) \cdots (\lambda - \theta_i^{(i)})}{(-1)^i \theta_1^{(i)} \cdots \theta_i^{(i)}}. \tag{2.4}$$

Here and in the remainder of the paper, $\|v\|_A = \sqrt{v^{\mathrm{T}} A v}$ denotes the energy norm of a vector v.

In other words, in each iteration i, CG adds to the initial approximate solution the vector in $\mathcal{K}_i(A, r_0)$ that minimizes the energy norm of the error. *The CG polynomial ρ_i^{CG} is the unique polynomial that minimizes the energy norm of the error over the ith Krylov subspace.* CG can equivalently be seen as the Vorobyev method of moments or as a matrix formulation of Gauss–Christoffel quadrature. *These are all mathematically the same method*. For details, see e.g. [4, (ch. 3)]. It is also important to consider the connection of the finite precision CG method with the infinite-dimensional CG method. For details, we refer the reader to [5]; see also [6] for an example of this connection for elliptic boundary value problems. It is common in the literature to use different polynomials, e.g. Chebyshev polynomials, in (2.3) to obtain an upper bound on the error in CG; we elaborate on this in §3.

Another aspect of the computation is the *algorithm* used to realize the particular method. For example, the most well-known algorithm for CG is that of Hestenes and Stiefel (HSCG) [2], displayed in algorithm 1, which uses three coupled two-term recurrences. Another algorithmic variant of CG, first developed by Stiefel [7] (see also [8,9]) uses two three-term recurrences. There are a number of other algorithms for CG (many motivated by reducing the cost per iteration) such as those that use different formulae for the computation of the coefficients (e.g. [10,11]), use auxiliary vectors (e.g. the variant of Chronopoulos & Gear [12] and the pipelined CG method of Ghysels & Vanroose [13]), and block iterations into groups of s (the so-called s-step variants; e.g. [14–16]). All the mentioned algorithms are realizations of CG; they are all mathematically equivalent, i.e. in exact arithmetic they produce exactly the same iterates and thus all exhibit the CG optimality property (2.3). As described in the next section, the cost per iteration cannot be reduced to an operation count, as is done in some influential early textbooks as well as recent publications such as [17,18].

A computation also involves an *implementation*, which involves the rewriting of the algorithm into instructions to be run on a computer. The implementation includes aspects such as the parallelization of the algorithm, the particular programming language, and also hardware aspects such as the machine precision.

Finally, there is the particular *instance of data and machine* involved in the computation. In our example, the data are a particular matrix A, vector b, initial approximate solution vector x_0, and the required accuracy of the approximation to the exact solution. The numerical properties and structure of the data in combination with the particular machine hardware ultimately determine the cost of the iterative computation, and thus ultimately the particular instance of data and machine must play a key role in developing an appropriate method, algorithm and implementation to use to solve the problem.

Achieving good performance at scale will thus require a revolutionary change in the entire computational process. It is not merely a question of using a different algorithm. This was formulated in a visionary way by von Neumann at the time of the dawn of the computer. Quoted by Grcar in his beautiful survey on the origins of modern numerical analysis [19], von Neumann writes in a letter to Maxwell Newman [20, p. 1],

> I am convinced that the methods of 'approximation mathematics' will have to be changed very radically in order to use . . .[a computer] sensibly and effectively - and to get into the position of being able to build and use still faster ones.

Outside of the field of numerical mathematics, there is a tendency to emphasize the importance of the algorithm in defining the cost of a computation, which has led to work in extending the Turing machine model for discrete computations to the realm of numerical computations in order to rectify what some see as a lack of rigour in the field. For example, Blum *et al*. [21, p. 21; 22, p. 23] write

> There is not even a formal definition of algorithm in the subject . . . Thus we view numerical analysis as an eclectic subject with weak foundations . . .

By contrast, Baxter & Iserles conclude in their essay *On the foundations of computational mathematics* [23, p. 27, 30],

> The purpose of computation is not to produce a solution with least error but to produce reliably, robustly and affordably a solution which is within a user-specified tolerance . . . The foundations of computational mathematics are the totality of mathematics.

We concur with the last quote and summarize:

> In the field of computational mathematics (numerical mathematics/numerical analysis) the *transformation of the data* is the *key* concept, consisting of a problem, method, algorithm, implementation and particular instance of data and machine.

Algorithms are merely a part of this, and thus efforts to define the cost of an algorithm in isolation from these other aspects will not deal with the bigger picture. Moreover, in works focused on the algorithm as the primary (and essentially the only) object, the notion of cost or complexity of algorithms is reduced to counting arithmetic operations using specific assumptions on the data, and the characteristically nonlinear convergence behaviour of Krylov subspace methods such as CG and of their various algorithmic realizations is falsely described in terms of linear contractions.

It is also worth noting that the cost of a computation is well defined in terms of energy consumption (e.g. the kilowatt hours required to execute the computation) and/or in terms of runtime (e.g. the number of seconds required for a computation to run on a machine). These are meaningful metrics, important in practice as well as theory, which are measurable even in total absence of a formal model of real computation.

As for extrapolating the role of contraction arguments in the description of the convergence behaviour of Krylov subspace methods such as CG, one can only say that highly nonlinear phenomena can be meaningfully linearized only locally. In his article on describing rates of convergence published in 1977, Pták [24] makes a similar comment, noting that

> . . .a description which fits the whole process, not only an asymptotic one, is only possible by means of suitable functions, not just numbers.

For a recent work which develops these ideas further, see [25]. We elaborate on this in the following section.

3. Describing the cost of an iterative computation

Various performance models have been developed to describe the cost of a computation in terms of the cost of floating point operations and the cost of moving blocks of data of a given size between levels of the sequential memory hierarchy and/or between parallel processors. For an overview of sequential memory models, see [26, Section 2.2.1]. In parallel models (assuming a distributed memory machine), data movement is between different parallel processors. Well-known parallel models include the PRAM model [27], the $n_{1/2}$ model [28], the Bulk Synchronous Parallel model [29], the LogP model [30] and the (α, β, γ)-model [31].

Unlike direct computations which perform a fixed number of floating point operations,[1] in iterative computations one cannot determine *a priori* (except in a few special cases) the number of iterations that will be required to converge to the prescribed accuracy. Assuming a constant cost per iteration, the cost of an iterative computation is then given by

$$\text{Cost} = \big(\text{Cost per iteration}\big) \times \big(\text{Number of iterations}\big), \tag{3.1}$$

where 'Cost per iteration' will depend on the efficiency of the implementation as well as specific structure of the problem and the underlying hardware, and 'Number of iterations' will depend on the accuracy requirements of the application, the numerical properties of the data, the method used and the finite precision behaviour of the particular algorithm chosen for the given method.

The effects of finite precision roundoff error can have a significant influence on the number of iterations required and cannot be ignored; different algorithms for the same method can result in widely varying costs, and in the extreme case, roundoff errors can prevent an algorithm from satisfying the prescribed accuracy requirements at all. In the following subsections, we briefly recall work towards understanding the behaviour of CG (of both the method itself and in finite precision algorithms) and examine various aspects deserving careful consideration.

(a) Bounding the norm of the error

As described in §2, the behaviour of CG is described by the minimization problem (2.3) where the polynomial that achieves this minimization is the CG polynomial ρ_i^{CG} in (2.4). From (2.3), we can write

$$\frac{\|x - x_i\|_A}{\|x - x_0\|_A} \leq \min_{\rho(0)=1, \deg(\rho)\leq i} \max_{1\leq j\leq N} |\rho(\lambda_j)|, \tag{3.2}$$

where λ_j, $j = 1, \ldots, N$, are the eigenvalues of the SPD matrix A. This bound is sharp, meaning that if A has d distinct eigenvalues, then for each $i = 0, \ldots, d$ there is some initial approximation x_0

[1]It should be noted that in the case of sparse matrices and direct methods such as LU factorization involving pivoting for numerical stability, the number of floating point operations cannot be determined *a priori*.

such that the left- and right-hand sides in (3.2) are equal. Further, for each $i = 1, \ldots, d-1$, there exist $i+1$ distinct eigenvalues of A, $\hat{\lambda}_1, \ldots, \hat{\lambda}_{i+1}$, such that [32]

$$\min_{\rho(0)=1, \deg(\rho)\leq i} \max_{1\leq j\leq N} |\rho(\hat{\lambda}_j)| = \left(\sum_{k=1}^{n+1} \prod_{j=1, j\neq k}^{i+1} \frac{|\hat{\lambda}_j|}{|\hat{\lambda}_j - \hat{\lambda}_k|} \right)^{-1}. \tag{3.3}$$

This again demonstrates that the convergence behaviour of CG depends strongly on the distribution of eigenvalues of A and that CG is *highly nonlinear*.

As mentioned in §2, a frequently made oversimplification is to replace the set of increasingly ordered eigenvalues of A by the continuous interval $[\lambda_1, \lambda_N]$ and upper bound the right-hand side of (3.2) using scaled and shifted Chebyshev polynomials. This leads to the often-used bound

$$\frac{\|x - x_i\|_A}{\|x - x_0\|_A} \leq 2 \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^i, \tag{3.4}$$

where $\kappa(A) = \lambda_N/\lambda_1$ is called the condition number of the SPD matrix A. The highly nonlinear convergence behaviour of CG is thus bounded by a linear contraction for *any* distribution of eigenvalues between λ_1 and λ_N and *any* distribution of the components of the initial residuals in the individual invariant subspaces. We further note that due to a result of Greenbaum [33], which we elaborate on in §3b(i), the bound (3.4) approximately applies also to finite precision HSCG. Because the specific distribution of eigenvalues is crucial in determining CG convergence, and, further, because the behaviour of finite-precision HSCG can differ significantly from the exact method, it is clear that (3.4) will often fail to give a good estimate of the CG convergence behaviour; see e.g. [4] and also the numerical experiment in §3c. For a recent demonstration of how inappropriate use of (3.4) can be misleading, see [6], which explains the limitations as well as the non-trivial requirements for the appropriate use of condition number arguments for a class of elliptic boundary value problems.

Despite the obvious shortcomings of the bound (3.4), condition number-based contraction is frequently *identified with* the CG 'convergence rate' in general in the literature. This has been extended to describing the complexity of CG in terms of the condition number (e.g. [17,18,34–36]). The justification for this is frequently misattributed to classical papers, which interpret the result differently. For example, in [17, pp. 8–9] (see also [37]) it is stated that

> Soon after the introduction of $\kappa(A)$ for error analysis, Hestenes and Stiefel [2] showed that this quantity also played a role in complexity analysis. More precisely, they showed that the number of iterations of the conjugate gradient method (assuming infinite precision) needed to ensure that the current approximation to the solution of a linear system attained a given accuracy is proportional to $\sqrt{\kappa(A)}$.

This follows from (3.4), which is not presented by Hestenes and Stiefel; in fact, it was clear through the connections they make to orthogonal polynomials, Gauss quadrature and continued fractions (see §2 and [2, Sections 14–18]) that the nonlinearity of CG was well understood. In the context of solving large-scale linear systems, Chebyshev polynomials were considered in the early works of Young, Lanczos, Rutishauser and others; see [4, Section 5.5.3] for a detailed description. However, these authors never had in mind that the upper bound for CG that can be derived using Chebyshev polynomials should be equated with a linear 'rate of convergence' of CG; the concept of a linear 'rate of convergence' is applicable to stationary iterative methods but makes little sense in the context of Krylov subspace methods. Daniel, who first published the explicit form of the bound (3.4) [38, Theorem 1.2.2], was careful to explain that

> ... assuming only that [the spectrum of the matrix A lies inside the interval $[\lambda_1, \lambda_N]$], we can do no better than [the bound (3.4)]. Under more specialized assumptions, it is possible to deduce more accurate estimates ...

Similar cautions have been repeated by others. In [39, p. 28], Axelsson & Barker state

> It should be kept in mind, however, that depending on the distribution of the interior eigenvalues [the bound (3.4)] may be quite pessimistic, hiding the true rate of convergence.

This is also echoed by van der Vorst [40, p. 50], who writes

> Upperbounds as in [the bound (3.4)] show that we have global convergence behaviour, but they do not help us explain all sorts of local effects in the convergence behaviour of CG.

See also the comments of Hackbusch [41, Remark 9.4.13, p. 272].

Illustrative examples using SPD diagonal matrices A with N real, positive distinct eigenvalues were used in [42,43], published in 1991 and 1992, respectively (see also the more recent monograph [44]). Given the smallest eigenvalue λ_1 and the largest eigenvalue λ_N, the inner eigenvalues are defined by, for $j = 2, \ldots, N-1$

$$\lambda_j = \lambda_1 + \frac{j-1}{N-1}(\lambda_N - \lambda_1)\gamma^{N-j}, \tag{3.5}$$

for a given parameter $0 < \gamma \leq 1$. Varying the value of γ generates matrices with the same condition number $\kappa(A)$ but with different distributions of eigenvalues, which drastically affect CG convergence behaviour (including the effects of rounding errors).

Similarly, it is easy to demonstrate the importance of the right-hand side, which is not captured by the bound (3.4). For further work on the investigation of the role of the right-hand side in the convergence behaviour of CG; see e.g. [45].

Although the experiments using (3.5) can be considered as simple illustrations, they capture the points that are present in practical computations. More details can be found, e.g. in [4, (Section 5.6)] and in the literature quoted therein. Experiments using a discretized boundary value problem together with a detailed explanation of CG convergence and its relationship to the bounds based on the so-called *adapted* condition number can be found in [6]; see also §3c and [4, Section 5.9.2].

(b) Algorithms designed for high-performance computing

On modern computer architectures, the time to perform a floating point operation is much less than the time to move a word of data, both between levels of the memory hierarchy and between parallel processors. Thus in the setting of large-scale computations on large-scale machines, the number of floating point operations performed is often a poor measure of runtime and/or energy cost, which depends much more heavily on the communication (i.e. the data movement). This has motivated the study and development of algorithms that minimize the amount of communication, which becomes especially important at the exascale level (e.g. [46]). The difficulty of obtaining good performance in iterative solvers such as CG is highlighted by the HPCG benchmark [1], on which today's top supercomputer obtains less than 2% of peak performance [47].

There has been a wealth of work, mostly in the context of direct numerical linear algebra, towards proving lower bounds on the amount of data an algorithm must move and in how many messages it must do so (e.g. [26,48–51]). The subsequent task is to find an approach that attains the proved lower bound on communication, potentially at the price of extra floating point operations (which have negligible cost). This line of research has resulted in a number of algorithms and implementations with improved runtimes on a variety of practical problems, many of which have since been implemented in libraries such as LAPACK [52].

The extension of such ideas to iterative methods, however, is tricky. There is a long history of developing algorithms that reduce the number of synchronizations per iteration in Krylov

subspace methods; for an overview, see [53]. As mentioned in §2, two recent approaches are '*s*-step' and 'pipelined' algorithms. The former reduces the number of synchronizations by a factor of s over a fixed number of iterations and the latter hides synchronization cost by overlapping global synchronizations with other useful work. For problems with certain sparsity structure, both approaches can offer faster runtime over a fixed number of steps.

The issue is with the phrase *a fixed number of steps*. As we have outlined, the cost of an iterative computation has two components; both the cost per iteration and the number of iterations to convergence to a prescribed accuracy. Especially in the case of those Krylov subspace methods for which algorithms typically use short recurrences (like CG), rounding errors can have a significant effect on the delay of convergence.

The finite precision effects evident in HSCG can be even worse in algorithms designed to reduce the cost per iteration. Despite being equivalent to HSCG in exact arithmetic, sometimes seemingly innocuous algorithmic modifications can cause finite precision roundoff errors to be amplified in a way that causes the numerical behaviour to differ, potentially significantly, from the classical HSCG algorithm. This means that the *total number of iterations required for convergence to the prescribed accuracy*, which is typically far from the maximum attainable accuracy, can be much greater than in HSCG. If the increase in the required number of iterations outweighs the reduction in cost per iteration, this can potentially negate any benefits of synchronization-reducing and synchronization-overlapping approaches.

Thus consideration of the effects of rounding errors, especially the potential for their amplification, is *inextricable* from the design of iterative algorithms. It makes little sense to declare algorithms to be 'high-performance algorithms' or 'exascale algorithms'; whether or not they are suitable for high-performance computing depends *heavily* on the implementation and on the input data A, b, both in terms of their structure and numerical properties.

In the following subsection, we will summarize existing results on the finite precision behaviour of HSCG and related methods. We then briefly describe the *s*-step and pipelined variants, and for each, describe what is known and what remains open regarding their finite precision behaviour.

(i) HSCG in finite precision

We first review some results regarding the convergence delay and attainable accuracy in finite precision HSCG; for further details on this topic, see e.g. [54]. In the remainder of the paper, we use hats to denote quantities computed in finite precision whenever it is necessary to distinguish them from the exact quantities and ε to denote the machine unit roundoff.

In any iterative approach to solving linear systems, the accuracy $\|x - \widehat{x}_i\|$ is generally not computable since it requires knowing the true solution x. However, since $x - \widehat{x}_i = A^{-1}(b - A\widehat{x}_i)$, it is common to use the size of the *true residual*, $\|b - A\widehat{x}_i\|$, as a computable measure of accuracy.[2] Over the course of the iterations, rounding errors made in updating $\widehat{x}_i$ and $\widehat{r}_i$ cause the true residual $b - A\widehat{x}_i$ to deviate from the recursively updated residual $\widehat{r}_i$ computed in line 6 of algorithm 1. Writing $b - A\widehat{x}_i = \widehat{r}_i + (b - A\widehat{x}_i - \widehat{r}_i)$, we can bound $\|b - A\widehat{x}_i\| \leq \|\widehat{r}_i\| + \|b - A\widehat{x}_i - \widehat{r}_i\|$. If the method is convergent, we eventually have $\|\widehat{r}_i\| \to 0$, and thus the size of the true residual will depend on the size of the *residual gap* $f_i \equiv b - A\widehat{x}_i - \widehat{r}_i$. There is a wealth of work on bounding the attainable accuracy for classical variants of CG, including HSCG (the first of which was due to Greenbaum [55]) and also variants that use two coupled 3-term recurrences, e.g. [56]. In short, if the finite precision iterate updates in HSCG satisfy $\widehat{x}_i = \widehat{x}_{i-1} + \hat{\alpha}_{i-1}\widehat{p}_{i-1} + \delta x_i$ and $\widehat{r}_i = \widehat{r}_{i-1} - \hat{\alpha}_{i-1}A\widehat{p}_{i-1} + \delta r_i$, then using standard floating point rounding error analysis, a bound on the size of the residual gap f_i can be written in various forms [55,57,58], e.g.

$$\|f_i\| \leq \|f_0\| + O(\varepsilon) \sum_{m=0}^{i} (1 + N)\|A\|\|\widehat{x}_m\| + \|\widehat{r}_m\|. \tag{3.6}$$

[2]Note that the residual is not necessarily a reliable measure of the level of accuracy. This was noted already by Hestenes & Stiefel in their original paper [2].

Describing the behaviour of HSCG in finite precision is a more difficult task. Finite precision errors cause orthogonality and eventually linear independence among the computed residual vectors (which are crucial in forming the direction vectors and updating the approximate solution) to be lost. The numerical rank deficiency of the subspace spanned by the residual vectors, which is, strictly speaking, no longer a Krylov subspace, can be related to the delay of convergence. Relatedly, in CG, the coefficients $\alpha_0, \ldots, \alpha_i$ and $\beta_1, \ldots, \beta_i$ determine through an easy calculation the entries of the Jacobi matrix T_i whose eigenvalues (called Ritz values) $\theta_\ell^{(i)}$, $\ell = 1, \ldots, i$, approximate those of A. Paige [59] has shown that rank deficiency (and thus convergence delay) goes hand-in-hand with the appearance of multiple copies of eigenvalues of A appearing as Ritz values in T_i. Paige's results were subsequently used as the basis for the 'backward-like' error analysis of Greenbaum [33], who proved that the finite precision HSCG[3] algorithm for matrix A behaves like the exact CG method for a larger matrix $\tilde{A}$ which has tight clusters of possibly many eigenvalues around the eigenvalues of A. The investigation of backward-like stability and convergence behaviour for even simple algorithms like HSCG remains an active area of research; see the recent works [60,61].

We return briefly to the connection of CG with Gauss–Christoffel quadrature mentioned in §2. The coefficients $\alpha_0, \ldots, \alpha_i$ and $\beta_1, \ldots, \beta_i$ determine distribution functions $\omega^{(i)}(\lambda)$ which approximate, in terms of the ith Gauss–Christoffel quadrature, the distribution function $\omega(\lambda)$ determined by inputs A, b and x_0. The A-norm of the CG error for $f(\lambda) = \lambda^{-1}$ can thus be related to the scaled quadrature error, i.e.

$$\frac{\|x - x_0\|_A^2}{\|r_0\|^2} = \int \lambda^{-1} d\omega(\lambda) = \sum_{\ell=1}^{i} \omega_\ell^{(i)} \{\theta_\ell^{(i)}\}^{-1} + \frac{\|x - x_i\|_A^2}{\|r_0\|^2};$$

(e.g. [4, Section 3.5]). For a particular CG algorithm variant in finite precision, the size of the error can be related to the distribution functions $\hat{\omega}^{(i)}$ determined by the *computed* quantities (more precisely, by the Jacobi matrices determined through the computed coefficients α_j and β_j). In other words, $\widehat{\omega}^{(i)}(\lambda)$ can be associated with some distribution function $\widehat{\omega}(\lambda)$ which is meaningfully related to the original distribution function $\omega(\lambda)$, i.e.

$$\int \lambda^{-1} \mathrm{d}\omega(\lambda) \approx \int \lambda^{-1} \mathrm{d}\widehat{\omega}(\lambda) = \sum_{\ell=1}^{i} \widehat{\omega}_\ell^{(i)} \{\hat{\theta}_\ell^{(i)}\}^{-1} + \frac{\|x - \widehat{x}_i\|_A^2}{\|r_0\|^2} + F_i, \tag{3.7}$$

where F_i is relatively small. For HSCG, this was proved by Greenbaum [33] (see also [43,62]). In this way, the delay of convergence due to roundoff is rigorously mathematically described through the error in the Gauss–Christoffel quadrature.

(ii) *s*-step CG in finite precision

The basic idea behind s-step variants of Krylov subspace methods is to expand the Krylov subspace $O(s)$ dimensions at a time and then perform a block orthogonalization using a single global synchronization. We give a brief description of s-step CG; for a more detailed derivation and a historical overview of related work, see [63,64].

The s-step CG method, shown in algorithm 2, consists of an outer loop indexed by k and an inner loop which iterates over $j \in \{1, \ldots, s\}$. In each outer loop k (each block of s iterations), we construct bases for the Krylov subspaces

$$\mathcal{K}_{s+1}(A, p_{sk}) = \mathrm{span}\{p_{sk}, Ap_{sk}, \ldots, A^s p_{sk}\} \quad \text{and} \quad \mathcal{K}_s(A, r_{sk}) = \mathrm{span}\{r_{sk}, Ar_{sk}, \ldots, A^{s-1} r_{sk}\}. \tag{3.8}$$

We let $\mathcal{P}_k$ and $\mathcal{R}_k$ denote matrices whose columns form bases for subspaces in (3.8) with starting vectors p_{sk} and r_{sk}, respectively, and define the $N \times (2s+1)$ matrix $\mathcal{Y}_k = [\mathcal{P}_k, \mathcal{R}_k]$. Note that (assuming exact arithmetic) any polynomials can be used for constructing these bases. Letting $\underline{\mathcal{Y}}_k$ denote $\mathcal{Y}_k$ with columns $s+1$ and $2s+1$ set to zero, we have the recurrence $A\underline{\mathcal{Y}}_k = \mathcal{Y}_k \mathcal{B}_k$, where $\mathcal{B}_k$ a matrix of size $(2s+1) \times (2s+1)$ (and which is in general upper Hessenberg) containing the

[3]The work of Greenbaum [33] uses a slightly different variant than HSCG; the results hold with minor modification.

Algorithm 2 . *s*-step CG.

Input: SPD matrix $A \in R^{N\times N}$, right-hand side vector $b \in R^N$, initial approximation $x_0 \in R^N$, positive integer s, maximum number of iterations $nmax$, stopping criterion.
Output: Approximate solution x_{sk+j} after the algorithm has been stopped.
Initialization: $r_0 = b - Ax_0$, $p_0 = r_0$
for $k = 0 : nmax/s$ **do**
  Compute $\mathcal{Y}_k$ given $\mathcal{B}_k$ such that $A\underline{\mathcal{Y}_k} = \mathcal{Y}_k\mathcal{B}_k$ and $\mathrm{span}(\mathcal{Y}_k) = \mathcal{K}_{s+1}(A, p_{sk}) + \mathcal{K}_s(A, r_{sk})$.
  $G_k = \mathcal{Y}_k^T\mathcal{Y}_k$
  Initialize $x'_0 = 0, r'_0 = e_{s+2}, p'_0 = e_1$, where e_i denotes the ith column of the identity.
  for $j = 1 : s$ **do**
    $\alpha_{sk+j-1} = (r'^T_{j-1} G_k r'_{j-1})/(p'_{j-1} G_k \mathcal{B}_k p'_{j-1})$
    $x'_j = x'_{j-1} + \alpha_{sk+j-1} p'_{j-1}, \quad r'_j = r'_{j-1} - \alpha_{sk+j-1}\mathcal{B}_k p'_{j-1}$
    Evaluate the stopping criterion and determine approximate solution x_{sk+j}.
    $\beta_{sk+j} = (r'^T_j G_k r'_j)/(r'_{j-1} G_k r'_{j-1})$
    $p'_j = r'_j + \beta_{sk+j} p'_{j-1}$
  end for
  $[x_{s(k+1)} - x_{sk}, r_{s(k+1)}, p_{s(k+1)}] = \mathcal{Y}_k[x'_s, r'_s, p'_s]$
end for

coefficients for the particular polynomial bases used. Subsequently, the inner products between basis vectors are computed in a single global synchronization by constructing the Gram matrix $G_k = \mathcal{Y}_k^T\mathcal{Y}_k$. Within the inner loops iterations, rather than update the length-N vectors x, r and p, we instead update their coordinates in the bases stored in the columns of $\mathcal{Y}_k$, denoted x', r' and p', respectively. In the basis $\mathcal{Y}_k$, the distributed matrix–vector multiplication Ap_{i-1} can be computed via $\mathcal{B}_k p'_{sk+j-1}$. Inner products computed during the iterations can be recovered using, e.g. $r_i^T r_i = r'^{\mathrm{T}}_{sk+j}\mathcal{Y}_k^T\mathcal{Y}_k r'_{sk+j} = r'^{\mathrm{T}}_{sk+j} G_k r'_{sk+j}$. Since s is assumed to be much smaller than N, the matrices $\mathcal{B}_k$ and G_k and the coordinate vectors can be stored on each processor locally, and thus there is no interprocessor communication required within the inner loop.

The attainable accuracy of s-step CG has been studied in [65]; see also [64]. In summary, in iteration $sk + j$, the residual gap can be bounded as

$$\|f_{sk+j}\| \le \|f_0\| + \varepsilon\bar{\Gamma}_k \sum_{\ell=1}^{sk+j} (1+N)\|A\|\|\widehat{x}_\ell\| + \|\widehat{r}_\ell\|, \tag{3.9}$$

where $\bar{\Gamma}_k = \max_{m\le k}\Gamma_m$ with $\Gamma_m = c\|\widehat{\mathcal{Y}}_m^+\|\|\widehat{\mathcal{Y}}_m\|$, where the superscript '+' denotes the Moore–Penrose pseudoinverse and c is a low-degree polynomial in s. We can think of Γ_m as a type of condition number of the computed basis $\widehat{\mathcal{Y}}_m$. The bound (3.9) says that the local rounding errors in each iteration can be amplified by ill-conditioned $\widehat{\mathcal{Y}}_m$, linking the condition numbers of the polynomial bases computed in each outer iteration with the loss of accuracy in finite precision s-step CG.

The convergence delay in s-step CG has not been fully investigated theoretically, although it is clear that the basis condition number plays a large role. In [66], the analysis of Paige [59] for the classical Lanczos algorithm is extended to the s-step Lanczos algorithm, and it is shown that the conclusions of Paige for the classical Lanczos method regarding convergence of Ritz values and Ritz vectors also apply to the s-step Lanczos method as long as

$$\bar{\Gamma}_k^2 < (24\varepsilon(N + 11s + 15))^{-1}. \tag{3.10}$$

The results in [66] may provide a path forward for extending the results of Greenbaum [33] to the s-step CG algorithm. In particular, we conjecture that as long as (3.10) holds, the finite precision s-step CG algorithm on a matrix A can be associated with the exact algorithm on a

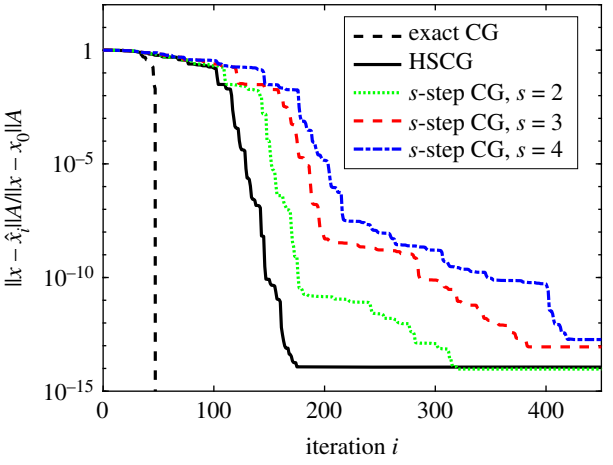

Figure 1. Convergence of HSCG and s-step CG (using monomial basis) with $s = 2, 3, 4$ in double precision and exact CG for matrix `bcsstk01`. (Online version in colour.)

larger matrix with eigenvalues in clusters around those of A, where the diameter of the clusters will depend on $\bar{\Gamma}_k$. We note that such a result would also imply that as long as (3.10) holds, the Chebyshev polynomial-based convergence bound (3.4) approximately holds even for s-step CG in finite precision. Other potential avenues for studying the convergence behaviour of s-step CG in finite precision are described in [53, (Section 4.2)].

In general, we expect the condition number of the computed bases to increase as s increases, the rate of which depends on the particular polynomials used. This presents a trade-off between the cost per iteration (which we generally expect to decrease with s up to some point) and the number of iterations (which we generally expect to increase with s).

In figure 1, we demonstrate the numerical behaviour of s-step CG with $s = 2, 3, 4$ in finite (double) precision using the matrix `bcsstk01` from SuiteSparse [67]. Here, we use the monomial basis for the Krylov subspaces in (3.8). The right-hand side b has equal components in the eigenbasis of A with $\|b\|_2 = 1$ and $x_0 = 0$. For comparison, we also plot HSCG in double precision as well as CG in (simulated) exact arithmetic. It is clear that both the convergence delay and the decrease in attainable accuracy in s-step CG grow worse with s. We note that the effects of roundoff error are visible even for the HSCG method.

It is well known that bounds on the condition number of the monomial basis grow exponentially with s, and thus it is expected that the numerical behaviour will quickly deteriorate with growing s. That the use of monomials leads to numerical instability was recognized even in early works on the s-step CG method (e.g. [12]). This led to the use of more well-conditioned bases, constructed using, e.g. Newton or Chebyshev polynomials, parameters for which can be approximated using rough estimates of the spectrum of A (which can be estimated using Ritz values from the first few iterations; e.g. the works [68–72]). In many cases, this has been shown to improve numerical behaviour, but we stress that this is not a universal rule. For example, for the test problem in figure 1, both Newton and Chebyshev bases constructed using estimates of the extremal eigenvalues of A perform worse than the monomial basis in terms of both convergence and accuracy, despite a smaller $\bar{\Gamma}_k$. The bound (3.9) of course still applies; it is merely that the attainable accuracy using the monomial basis is much better than what is predicted by the right-hand side of (3.9).

(iii) Pipelined CG in finite precision

The idea behind pipelined CG, shown in algorithm 3, is to decouple the sparse matrix–vector multiplication and the inner products so that they can be computed simultaneously using

Algorithm 3 . Pipelined CG.

1: **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 $nmax$, stopping criterion.
2: **Output:** Approximate solution x_i after the algorithm has been stopped.
3: **Initialization:** $r_0 = b - Ax_0$, $p_0 = r_0$, $s_0 = Ap_0$, $w_0 = Ar_0$,$z_0 = Aw_0$, $\alpha_0 = r_0^T r_0 / p_0^T s_0$
4: **for** $i = 1 : nmax$ **do**
5: $\quad x_i = x_{i-1} + \alpha_{i-1} p_{i-1}, \quad r_i = r_{i-1} - \alpha_{i-1} s_{i-1}, \quad w_i = w_{i-1} - \alpha_{i-1} z_{i-1}$
6: $\quad$ Evaluate the stopping criterion.
7: $\quad \beta_i = (r_i^T r_i)/(r_{i-1}^T r_{i-1})$
8: $\quad \alpha_{i-1} = (r_i^T r_i)/(w_i^T r_i - (\beta_i/\alpha_{i-1}) r_i^T r_i)$
9: $\quad p_i = r_i + \beta_i p_{i-1}, \quad s_i = w_i + \beta_i s_{i-1}, \quad z_i = Aw_i + \beta_i z_{i-1}$
10: **end for**

asynchronous communication collectives. This decoupling is accomplished by the introduction of auxiliary recurrences $s_i \equiv Ap_i$, $w_i \equiv Ar_i$ and $z_i \equiv Aw_i \equiv A^2 r_i$, as well as the use of an alternative form for the computation of the scalar α_i. We limit our discussion to the unpreconditioned version of pipelined CG; the preconditioned variant requires the use of additional auxiliary vectors.

The pipelined CG algorithm, introduced by Ghysels & Vanroose [13], builds on a wealth of previous work such as the algorithm of Chronopoulos & Gear [12]. For a more complete description of the history and related methods, see [53]. Other pipelined variants of Krylov subspace methods have been developed, such as pipelined GMRES [73] and pipelined BiCGStab [74], as well as 'deep-pipelined' variants [73,75].

For variants such as pipelined CG that use auxiliary recurrences, the maximum attainable accuracy can be bounded using the methodology outlined in [53]. A detailed analysis of the maximum attainable accuracy in pipelined CG was given by Cools *et al.* [76] which follows this methodology. We briefly summarize the main result of their analysis.

Letting δ_i^p, δ_i^s, δ_i^z, δ_i^x, δ_i^r and δ_i^w denote the local errors due to finite precision computation in the associated vector updates, the size of the residual gap in pipelined CG can be expressed as $\|f_i\| \le \|f_0\| + \sum_{j=1}^{i} (\|A\|\|\delta_j^x\| + \|\delta_j^r\| + |\hat{\alpha}_j|\|g_j\|)$, where

$$g_j = \left(\prod_{k=1}^{j} \hat{\beta}_k\right) g_0 + \sum_{k=1}^{j} \left(\prod_{\ell=k+1}^{j} \hat{\beta}_\ell\right) (A\delta_k^p - \delta_k^s + h_k), \quad h_k = h_0 - \sum_{\ell=0}^{k-1} (-A\delta_\ell^r + \delta_\ell^w + \hat{\alpha}_\ell c_\ell)$$

$$\text{and} \quad c_\ell = \left(\prod_{m=1}^{\ell} \hat{\beta}_m\right) c_0 + \sum_{m=1}^{\ell} \left(\prod_{n=m+1}^{\ell} \hat{\beta}_n\right) (A\delta_m^q - \delta_m^z);$$

see [76, Section 2.3]. It is clear from the above expression that local rounding errors made in each iteration can be *globally amplified* by the values of the computed $\hat{\alpha}$'s and $\hat{\beta}$'s. In exact arithmetic, the β terms give the ratio between subsequent residual norms, which are not monotonically decreasing in CG. Thus, we might expect amplification of errors whenever there is residual oscillation; see [56] for an early description of this phenomenon.

In figure 2, we demonstrate the effect of finite precision errors in pipelined CG in double precision using the same problem as in figure 1. Relative to HSCG, pipelined CG suffers a delay of convergence and exhibits a drastic decrease in attainable accuracy.

We note that figures 1 and 2 are not necessarily representative of the general numerical behaviour of the s-step and pipelined CG algorithms. In figure 3, we plot the convergence of HSCG, s-step CG with $s = 2, 3, 4$, and pipelined CG in double precision for a different problem using the matrix `nos4` from SuiteSparse [67]. We also include CG in (simulated) exact arithmetic for reference. Again, the right-hand side b was set to have equal components in the eigenbasis of A with $\|b\|_2 = 1$ and $x_0 = 0$. For this problem, the s-step and pipelined algorithms all behave similar

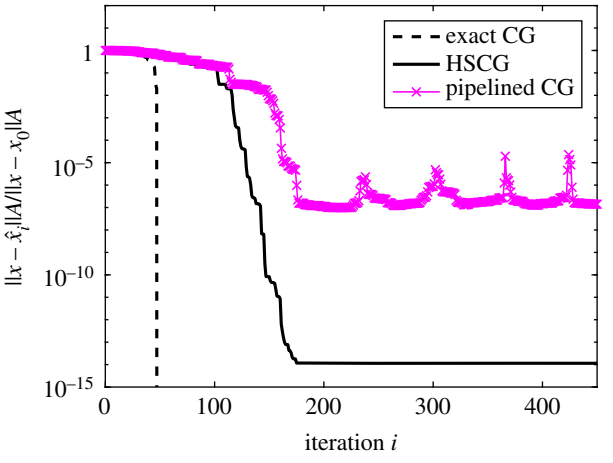

Figure 2. Convergence of HSCG and pipelined CG in double precision and exact CG for matrix `bcsstk01`. (Online version in colour.)

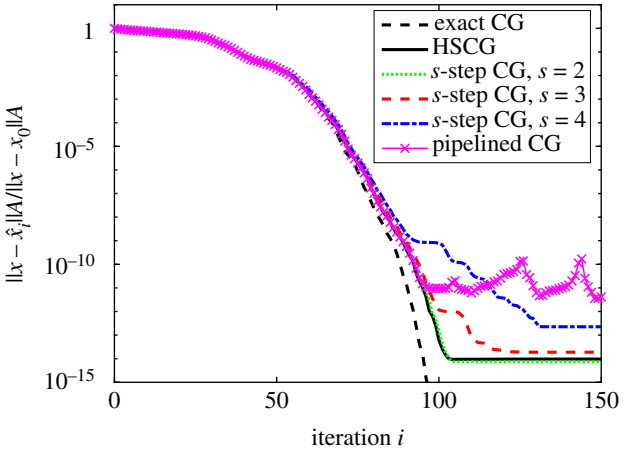

Figure 3. Convergence of HSCG, s-step CG (using monomial basis) with $s = 2, 3, 4$, and Pipelined CG in double precision as well as exact CG for matrix `nos4`. (Online version in colour.)

to HSCG up until a relative accuracy of around 10^{-11}. Depending on the prescribed accuracy requirements of the application, any of the variants of HSCG could be perfectly suitable from a numerical perspective.

We close this subsection with the comment that the focus thus far (in the mentioned variants of Krylov subspace methods and also in e.g. 'inexact Krylov subspace methods' [77]) has been almost exclusively on bounding the maximum attainable accuracy, as this is a much simpler task than determining effects on convergence delay. Although useful for determining the applicability of the method to a given problem, typically in practice the prescribed accuracy requirement is much, much above than the maximum attainable accuracy. The more important question is in determining whether such approaches are beneficial in terms of cost, which requires a study of the convergence behaviour in finite precision. One way forward may be through the mentioned connection to Gauss–Christoffel quadrature [53, Section 4.2].

(c) The effect of preconditioning

Preconditioning is a technique which involves transforming the linear system $Ax = b$ into an equivalent one for which the iterative method will converge more rapidly. Given a preconditioner

M, in the case of left preconditioning for example, the preconditioned system becomes $M^{-1}Ax = M^{-1}b$. There are a few considerations concerning the construction and application of preconditioners that require attention.

The name 'preconditioning' often invokes the argument that the condition number plays the dominant role in determining convergence behaviour, which is certainly not true; see §3a. Even when the distribution of eigenvalues is taken into account, statements regarding the convergence behaviour often rely on arguments about eigenvalue clustering. A frequent assertion is that the number of tight clusters of eigenvalues is equal to the number of iterations in which a Krylov subspace method can be expected to provide an approximation close to the exact solution. *This is not true* in general for any Krylov subspace method we are aware of.

In order to explain the point for CG and SPD matrices, we use the original analysis of Greenbaum [33] mentioned in §3b. Skipping some technical details, she proved that given the number of steps, finite precision CG on a problem with a matrix A (possibly with simple eigenvalues) behaves like CG in exact arithmetic on a problem with a larger matrix $\tilde{A}$ which has tight clusters of eigenvalues around those of A. It is evident that finite precision CG on a problem with the matrix A can behave very differently from exact CG for the same problem. Therefore, it cannot be true that in general exact CG for the problem with the matrix $\tilde{A}$ having tight clusters of possibly many eigenvalues behaves similarly to exact CG for the problem with the matrix A where each cluster is represented by a single eigenvalue. This disproves the generality of the clustering argument.[4] We stress that the consideration of exact CG for the matrices $\tilde{A}$ and A does not include any effects of rounding errors. The *location* as well as the diameter of the clusters is important in determining convergence.

As a demonstration, in figure 4 we show CG convergence in exact arithmetic for diagonal matrices A with eigenvalues described by (3.5) with $N = 25$, $\lambda_1 = 0.1$, $\lambda_N = 100$ and $\gamma = 0.65$. Note that this test case was used in [4, e.g. fig. 5.15]. The dashed line shows convergence for the matrix as described with right-hand side b having identical entries and $\|b\|_2 = 1$. The circles plot convergence for the matrix constructed by taking A and splitting its two smallest eigenvalues into clusters of five eigenvalues each, with the entries in b split accordingly. Similarly, the dotted line shows convergence for the matrix constructed in the same way except with the two largest eigenvalues split into clusters of five each. The dash-dotted line shows convergence for the matrix with every eigenvalue split into a cluster of five eigenvalues (again, with entries of b split accordingly). The distance between neighbouring eigenvalues within a cluster was 10^{-14}.

Each of these matrices has *the same number of clusters of eigenvalues*, and yet the convergence behaviour can be different. Notice that the location of the clusters matters. Clusters of eigenvalues close to the origin do not affect the behaviour of CG, whereas the eigenvalue clusters around the high end of the spectrum visibly cause a convergence delay. We stress that these tests are run in (simulated) exact arithmetic; the differing convergence is not due to roundoff error. A detailed explanation with other experiments can be found in [4, Section 5.9.1]. To demonstrate the theory of Greenbaum [33], we include the solid line which shows convergence of HSCG run in double precision on the original matrix with simple eigenvalues. We can see that the behaviour is qualitatively similar to the behaviour of the exact method with each eigenvalue replaced by a cluster. Fully analogous arguments can be applied and the same conclusions can be drawn for problems with symmetric indefinite matrices solved, e.g. by the MINRES method.

The clustering notion as a general argument fails for non-symmetric problems and other Krylov subspace methods like GMRES as well. Even for normal matrices, many eigenvalues close to the origin are known to cause stagnation of GMRES convergence (e.g. [79] and the examples therein). In the non-normal case, it is even more clear that the clustering argument does not

[4]In relation to this, large outlying eigenvalues are frequently identified with acceleration of CG convergence using the argument that a large outlier is 'eliminated' at the cost of one iteration and CG then proceeds with convergence behaviour governed by the so-called effective condition number. As explained in [78] (see also [6]), this reasoning assumes exact arithmetic and it cannot be applied to practical computations.

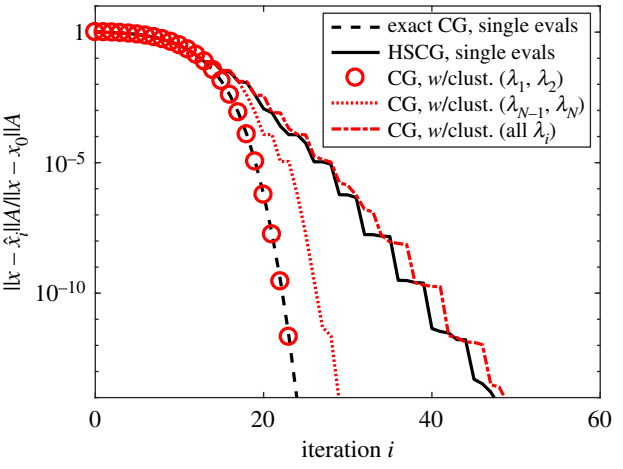

Figure 4. Convergence of CG with clusters of eigenvalues. (Online version in colour.)

hold; it is possible to construct an example problem (or even a parametrized set of problems) with *any* prescribed spectrum such that GMRES exhibits *any* prescribed convergence behaviour, e.g. stagnation, until the final iteration [80–82]. Consequently, any fast convergence behaviour of GMRES is possible with the system matrix having eigenvalues widespread in the complex plane and any poor convergence behaviour is possible with real eigenvalues tightly clustered around one. Therefore, any statement that links convergence of GMRES solely with the distribution of the eigenvalues of the system matrix must be justified in relation to these results proved more than 20 years ago. This is in the literature done very rarely.

The clustering argument is often used in combination with the remarkable fact pointed out in [83] that a certain class of preconditioners leads to preconditioned matrices with a low degree minimal polynomial. Such preconditioners are very important in practice where they are often replaced by some more computationally feasible approximations, which can lead instead of a low degree minimal polynomial to a few tight clusters of eigenvalues. This subtle shift requires a rigorous analysis. In [83], the authors issue a strong warning that while replacing the low degree minimal polynomial argument with the argument that the preconditioned matrix has a few distinct eigenvalues one *must* include a strong assumption that the preconditioned matrix is either diagonalizable or the Jordan blocks have small size.[5]

While replacing single eigenvalues by tight clusters one must admit that a rigorous argument is involved[6] and the behaviour in practice can depend, as pointed out above in the symmetric case, on the location of the clusters. Our point is that presenting rigorously assumptions under which a statement holds is imperative. Linking convergence of GMRES solely with the distribution of eigenvalues without considering the structure of invariant subspaces (a common practice) is not only mathematically incorrect, but closes the door for investigation of potentially important phenomena. It may happen that eigenvalues do determine GMRES convergence despite that the structure of the associated invariant subspaces does not allow the use of any existing bounds. In this case, we must investigate this structure—it almost certainly bears significant information about the problem. This can be a very positive outcome of the results in [80–82].

An unrelated issue is that much work in designing preconditioners focuses only on the convergence of a Krylov subspace method for the resulting system and ignores the effects on attainable accuracy in finite precision. Even if the preconditioned system is well conditioned,

[5]The relationship of GMRES convergence with the minimal polynomial of the system matrix is summarized, e.g. in [82, Section 3].

[6]It must include, in addition to multiplicities of the eigenvalues, also perturbation arguments that in the highly non-normal case becomes rather intriguing. A small perturbation of the matrix can cause a large change in the eigenvalues.

it must be kept in mind that we do not explicitly form this system in practice; the matrix–vector product and preconditioner application are computed separately. We can thus write the computed preconditioned vector as

$$\hat{z} = (M^{-1}A + \Delta)r, \quad \text{where } \|\Delta\|_2 \leq O(\varepsilon)N^{3/2}\,(\kappa(A) + \kappa(M))\,\|M^{-1}A\|_2;$$

c.f. [84, Section 3]. If the original matrix is ill conditioned (which is often the case where we would want to use a preconditioner), any reasonably efficient preconditioner is likely also ill conditioned. Thus, large $\kappa(A)$ and $\kappa(M)$ can potentially introduce significant roundoff errors which can affect the numerical behaviour of the method despite $\kappa(M^{-1}A)$ being small. This was investigated in [84, Section 3] for the case where M consists of the triangular factors from an LU factorization; it was shown there that if A is very ill conditioned (and thus M is also), GMRES provides a backward stable solution to the left-preconditioned system if double the working precision is used in the matrix–vector product and preconditioner application.

Finally, another important consideration is the increase in cost per iteration due to extra computations involved in preconditioning. Even if the preconditioner reduces the number of iterations required for convergence, each iteration may now be much more expensive due to additional communication and/or computation involved in applying the preconditioner, which is of particular importance in highly parallel environments. It is thus not always clear that even with a reduced number of iterations there will be an overall benefit to performance/energy cost. This trade-off must be carefully balanced in minimizing the cost of an iterative method (e.g. [85]) as well as the book of Meurant [86, ch. 8] which nicely describes this and other issues relevant to preconditioning.

4. The importance of data and adaptivity

The importance of considering the source of the problem in developing efficient iterative methods and algorithms has been stressed by many working in the field. In their textbook published in 1981 [9, p. x], Hageman & Young write

> The choice of an effective iterative solution method for a particular problem depends heavily on the details peculiar to the problem and on the particular architecture of the computer to be used.

Along similar lines, in the influential textbook [87, p. xiii], Saad explains that

> . . . it still cannot be stated that an arbitrary sparse linear system can be solved iteratively in an efficient way. If physical information about the problem can be exploited, more effective and robust methods can be tailored for the solutions.

The particular problem instance and the underlying application are also of great importance in consideration of the effects of finite precision computation. As Parlett concisely states in his article on the contributions of James Wilkinson [88, p. 20].

> . . . tiny arithmetic inaccuracies leap sharply from being negligible in one domain to critical in the other.

Thus, the *data* must be a main focus in designing iterative schemes and evaluating the cost of iterative computations; designing iterative algorithms based on general assumptions about the numerical properties and structure of the data, or the statistical distribution of the data, is an effort that does not help in solving particular problems.

As we have seen throughout this article, 'the data' cannot be reduced to something as simple as the matrix condition number. For CG, the nonlinear adaptation to the data is the main principle

behind the method; this is the beauty of Krylov subspace methods. This was appreciated early on by Lanczos and Einstein, who corresponded on the topic of the Lanczos method [4]:[7]

> The reason why I am strongly drawn to such approximation mathematics problems is not the practical applicability of the solution, but rather the fact that a very 'economical' solution is possible only when it is very 'adequate'. To obtain a solution in very few steps means nearly always that one has found a way that does justice to the inner nature of the problem.
>
> — Cornelius Lanczos in a letter to Albert Einstein on 9 March 1947

> Your remark on the importance of adapted approximation methods makes very good sense to me, and I am convinced that this is a fruitful mathematical aspect, and not just a utilitarian one.
>
> — Einstein's reply to Lanczos on 18 March 1947

Equating the (nonlinear) convergence of CG with the (linear) contraction bound based on condition number completely loses any notion of the adaptivity behind the method.[8]

The complex, nonlinear, data-dependent behaviour of Krylov subspace methods, combined with the effect of rounding errors, makes efficient use of such methods seem daunting. Software libraries which provide implementations of algorithms for such iterative methods often provide little or (usually) no documentation describing important aspects of numerical behaviour, leaving the user blindly guessing at the best approach. Baxter & Iserles [23, p. 30] warn that

> ... the impetus towards 'general' methods and 'general' software, which can cater to many different problems in a broad category, might be inimical to progress. Once we classify mathematical problems by their structural features, broad categories are much too coarse.

Given the view of the highly data-dependent nature of both the convergence behaviour and the cost per iteration, it seems infeasible to develop approaches with efficient cost for large classes of problems. There is, however, some hope offered by the possibility of explicitly *adaptive* approaches. Baxter & Iserles [23, p. 27] also write, on the topic of adaptivity,

> ... good computation requires the algorithm to respond to the data it is producing and change the allocation of computing resources (step size, size of the grid, number of iterations, even the discretization method itself) accordingly ... All this emphasis on *adaptivity* is hardly 'foundational' and has been the focus of much effort in numerical analysis, scientific computing and software engineering ... adaptivity is linked to important foundational issues and, ideally, in place of its current niche of half-mathematics, half-art, should be brought into the fully mathematic domain.

Echoing the quote above, there are numerous instances where explicitly controlled adaptivity has been incorporated in iterative algorithms to balance trade-offs between numerical behaviour and cost per iteration, in which decisions are made on-line depending on measured quantities and the particular input data. Some well-established examples in Krylov subspace methods include selective reorthogonalization in the Lanczos process and implicit restarting in the Arnoldi method. Baxter and Iserles are correct in pointing out that such algorithmic innovations are often lacking a full theoretical foundation. But some of them are well rooted in theoretical results, with both selective reorthogonalization and implicit restarting being good examples. With the emergence of multiprecision capabilities in hardware, another current area of interest is in the

[7]We thank the Albert Einstein Archives, The Hebrew University of Jerusalem, Israel, which holds the copyrights on Einstein's letters. The included quotes were translated from German to English and presented in [4].

[8]There certainly are problems where CG converges linearly because no adaptivity can take place. This raises the question of whether CG is the right method to use in such cases.

design of algorithms which adaptively vary the precision used throughout the computation. Such techniques present the opportunity for significant performance improvements and are deserving of rigorous theoretical analysis. We expect this to be a fruitful area of future research.

5. The place of numerical analysis, challenges and opportunities

The example of CG illuminates key aspects involved in evaluating the cost of iterative computations that will inevitably play a role in solving challenging numerical problems, and reveals the potential hazards in approaches that fail to take a holistic view of this cost. The primary theses are summarized below.

We stress that the mathematical transformation of the data represented by the iterative scheme is the key concept; algorithms are only a part of the consideration. The cost of a computation in terms of computational time or consumed energy is a well-defined metric, and does not require a formal model of real computation.

Further, the premise that the linearization of highly nonlinear phenomena can give insights only locally or under very specific circumstances is obviously valid in iterative computations. Thus, in the case of Krylov subspace methods, simplified analyses based on condition number and contraction arguments are in general incapable of giving realistic estimates of practical behaviour. Further, rounding errors can have a significant effect on the delay of convergence to the prescribed accuracy. Thus consideration of the effects of rounding errors, especially the potential for their amplification, is inextricable from the design and analysis of iterative schemes. Declaring roundoff issues to be 'out of the scope' of such work (which usually means 'postponed indefinitely') can only have deleterious effects on the development of efficient iterative computations. Formulae developed under the assumption of exact arithmetic, such as various estimates in quadratic forms (e.g. [44,62,89]) cannot safely be used in computations unless they are justified by an appropriate analysis of the effects of roundoff. This is, in most of the literature, simply neglected.

Finally, iterative computations, and in particular Krylov subspace methods, are used in a number of application areas, and the structure and numerical properties of the particular data A and b will depend on the underlying application problem. Thus, performing analyses of cost or designing algorithms based on abstract and restrictive assumptions solely on the algebraic level can lead to failures in many practical cases. The data must be a primary focus in designing iterative schemes, and this necessitates adaptive approaches.

Computational mathematics and numerical analysis historically lacks a definitive academic home. In some institutions, it lives within mathematics and in others within computer science. Sometimes, it is separated into its own 'interdisciplinary' institute. In recent years, the overlap with informatics and data science is becoming increasingly important. We again evoke the essay of Baxter & Iserles [23, p. 4], where they comment that

> Numerical analysis lies at the meeting point of pure mathematics, computer sciences and application areas. It often attracts some degree of hostility from all three. Yet it is important to note that the complaints of pure mathematicians, computer scientists and the broad constituency of scientists and engineers are very different and often contradictory.

Indeed, in the area of iterative computations, where computations using Krylov subspace methods can serve as an instructive example, it is clear that efforts towards developing abstract complexity theory, developing algorithms designed for high-performance computing and developing preconditioners are all coming from vastly different academic perspectives. As we have attempted to demonstrate here, no approach isolated to one academic perspective can give a truly complete view of the cost in practical iterative computations. *Interdisciplinary research is not only beneficial; it is critical to progress in this area.*

There is an emerging trend connecting data science and data informatics with the realm of computational science. We agree fully with this development. Scientific computing consists

of much more than solving PDEs; it also includes, e.g. tasks such as event identification and correlation in high-energy physics data, climate simulation and its validation using sensor data, determining patterns in astronomical data, analysis of data from social and other complex networks and genetic sequencing. Supercomputer hardware is now being designed with data-driven applications in mind (e.g. including more memory at the node level, the inclusion of accelerators/specialized processors for neural network computations, and the inclusion of reduced-precision capabilities in hardware).

As argued above, numerical analysis (computational mathematics) is by its nature interdisciplinary and so is its mathematical foundation. Although in this work we have pointed out many areas in which domain-specific viewpoints can lead to missing the bigger picture, we see this as a positive opportunity for further research and development. Towards this, it is imperative that assumptions and limitations are clearly stated, experiments are designed in accordance with the principles of the scientific method, and that results are interpreted honestly. In truth, negative results are often the most valuable in terms of furthering scientific research, because they eventually lead to big steps forward or even major breakthroughs. We invite the greater computational mathematics community, including those in computer science, data science, computer engineering and mathematics, to join together and adhere to these tenets in the push towards achieving unprecedented scales of computational power.

Data accessibility. This article has no additional data.

Authors' contributions. E.C. carried out the numerical experiments and drafted the manuscript. Both authors collaborated on the content and revision of the article. Both authors gave final approval for publication and agree to be held accountable for the work performed therein.

Competing interests. We declare we have no competing interests.

Funding. The first author is supported by Charles University Primus program project PRIMUS/19/SCI/11. The second author is partially supported by the Grant Agency of the Czech Republic Project GA18-12719S.

Acknowledgements. The authors thank Jörg Liesen, Volker Mehrmann, Gérard Meurant, Miroslav Tůma and the two anonymous referees for their insightful comments on the manuscript.

References

1. Dongarra J, Heroux M, Luszczek P. 2016 High-performance conjugate-gradient benchmark. *Int. J. High Perform. Comput.* **30**, 3–10. (doi:10.1177/1094342015593158)
2. Hestenes MR, Stiefel E. 1952 Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand.* **49**, 409–436. (doi:10.6028/jres.049.044)
3. Lanczos C. 1952 Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bur. Stand.* **49**, 33–53. (doi:10.6028/jres.049.006)
4. Liesen J, Strakoš Z. 2013 *Krylov subspace methods: principles and analysis*. Numerical Mathematics and Scientific Computation. Oxford, UK: Oxford University Press.
5. Málek J, Strakoš Z. 2015 *Preconditioning and the conjugate gradient method in the context of solving PDEs*, vol. 1. SIAM Spotlights. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM).
6. Gergelits T, Mardal K-A, Nielsen B, Strakoš Z. 2019 Laplacian preconditioning of elliptic PDEs: localization of the eigenvalues of the discretized operator. *SIAM J. Numer. Anal.* **57**, 1369–1395. (doi:10.1137/18M1212458)
7. Stiefel E. 1952/1953 Ausgleichung ohne Aufstellung der Gaußschen Normalgleichungen. *Wiss. Z. Technische Hochschule Dresden* **2**, 441–442.
8. Engeli M, Ginsburg Th., Rutishauser H, Stiefel E. 1959 *Refined iterative methods for computation of the solution and the eigenvalues of self-adjoint boundary value problems*. Basel, Switzerland: Birkhäuser.
9. Hageman LA, Young DM. 1981 *Applied iterative methods*. New York, NY: Academic Press.
10. Johnsson L. 1983 Highly concurrent algorithms for solving linear systems of equations. In *Elliptic problem solvers II* (eds G Birkhoff, A Schoenstadt), pp. 105–126. New York, NY: Academic Press.
11. Saad Y. 1985 Practical use of polynomial preconditionings for the conjugate gradient method. *SIAM J. Sci. Stat. Comput.* **6**, 865–881. (doi:10.1137/0906059)

12. Chronopoulos AT, Gear CW. 1989 On the efficient implementation of preconditioned *s*-step conjugate gradient methods on multiprocessors with memory hierarchy. *Parallel Comput.* **11**, 37–53. (doi:10.1016/0167-8191(89)90062-8)
13. Ghysels P, Vanroose W. 2014 Hiding global synchronization latency in the preconditioned conjugate gradient algorithm. *Parallel Comput.* **40**, 224–238. (doi:10.1016/j.parco.2013.06.001)
14. van Rosendale J. 1983 Minimizing inner product data dependencies in conjugate gradient iteration. Technical Report 172178, ICASE-NASA.
15. Leland RW. 1989 The effectiveness of parallel iterative algorithms for solution of large sparse linear systems. PhD thesis, University of Oxford.
16. Toledo SA. 1995 Quantitative performance modeling of scientific computations and creating locality in numerical algorithms. PhD thesis, Massachusetts Institute of Technology.
17. Cucker F. 2015 A theory of complexity, condition, and roundoff. *Forum Math. Sigma* **3**, e4. (doi:10.1017/fms.2015.2).
18. Bürgisser P, Cucker F. 2013 *Condition: the geometry of numerical algorithms*. Berlin, Germany: Springer.
19. Grcar JF. 2011 John von Neumann's analysis of Gaussian elimination and the origins of modern numerical analysis. *SIAM Rev.* **53**, 607–682. (doi:10.1137/080734716)
20. von Neumann J. 1946 Letter to Maxwell H. A. Newman on 19 March 1946.
21. Blum L, Cucker F, Shub M, Smale S. 1996 Complexity and real computation: a manifesto. *Int. J. Bifurcat. Chaos* **6**, 3–26. (doi:10.1142/S0218127496001818)
22. Blum L, Cucker F, Shub M, Smale S. 1998 *Complexity and real computation*. Berlin, Germany: Springer Science & Business Media.
23. Baxter BJC, Iserles A. 2003 On the foundations of computational mathematics. *Handb. Numer. Anal.* **11**, 3–34.
24. Pták V. 1977 What should be a rate of convergence? *R.A.I.R.O. Analyse Numérique/Numerical Analysis* **11**, 279–286. (doi:10.1051/m2an/1977110302791)
25. Liesen J. 2015 Pták's nondiscrete induction and its application to matrix iterations. *IMA J. Numer. Anal.* **36**, 1242–1260. (doi:10.1093/imanum/drv037)
26. Ballard G. 2013 Avoiding communication in dense linear algebra. PhD thesis, University of California, Berkeley.
27. Fortune S, Wyllie J. 1978 Parallelism in random access machines. In *Proc. of the tenth Annual ACM Symp. on Theory of Computing, San Diego, CA, 1–3 May*, pp. 114–118. New York, NY: ACM.
28. Hockney RW. 1983 Characterizing computers and optimizing the FACR(l) Poisson-solver on parallel unicomputers. *IEEE Trans. Comput.* **C-32**, 933–941. (doi:10.1109/TC.1983.1676137)
29. Valiant LG. 1990 A bridging model for parallel computation. *Commun. ACM* **33**, 103–111. (doi:10.1145/79173.79181)
30. Culler D, Karp R, Patterson D, Sahay A, Schauser KE, Santos E, Subramonian R, von Eicken T. 1993 LogP: towards a realistic model of parallel computation. In *Proc. 4th ACM SIGPLAN Symp. Principles and Practice of Parallel Programming, San Diego, CA, 19–22 May*, pp. 1–12. New York, NY: ACM.
31. Chan E, Heimlich M, Purkayastha A, van de Geijn R. 2007 Collective communication: theory, practice, and experience. *Concurr. Comput. Pract. Exp.* **19**, 1749–1783. (doi:10.1002/cpe.1206)
32. Greenbaum A. 1979 Comparison of splittings used with the conjugate gradient algorithm. *Numer. Math.* **33**, 181–193. (doi:10.1007/BF01399553)
33. Greenbaum A. 1989 Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences. *Linear Algebra Appl.* **113**, 7–63. (doi:10.1016/0024-3795(89)90285-1)
34. Shewchuk JR. 1994 *An introduction to the conjugate gradient method without the agonizing pain*. Report no. CMU-CS-94-125. Pittsburgh, PA: Carnegie Mellon University.
35. Smale S. 1997 Complexity theory and numerical analysis. *Acta Numer.* **6**, 523–551. (doi:10.1017/S0962492900002774)
36. Cucker F, Smale S. 1998 Complexity estimates depending on condition and round-off error. In *European Symp. on Algorithms, Venice, Italy, 24–26 August*, pp. 115–126. Berlin, Germany: Springer.
37. Cucker F. 2016 Probabilistic analyses of condition numbers. *Acta Numer.* **25**, 321–382. (doi:10.1017/S0962492916000027)
38. Daniel JW. 1967 The conjugate gradient method for linear and nonlinear operator equations. *SIAM J. Numer. Anal.* **4**, 10–26. (doi:10.1137/0704002)
39. Axelsson O, Barker VA. 1984 *Finite element solution of boundary value problems: theory and computation*. Computer Science and Applied Mathematics. Orlando, FL: Academic Press, Inc.

40. van der Vorst HA. 2003 *Iterative Krylov methods for large linear systems*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge, UK: Cambridge University Press.
41. Hackbusch W. 1994 *Iterative solution of large sparse systems of equations*, vol. 95. Applied Mathematical Sciences. New York, NY: Springer.
42. Strakoš Z. 1991 On the real convergence rate of the conjugate gradient method. *Linear Algebra Appl.* **154**, 535–549. (doi:10.1016/0024-3795(91)90393-B)
43. Greenbaum A, Strakoš Z. 1992 Predicting the behavior of finite precision Lanczos and conjugate gradient computations. *SIAM J. Matrix Anal. Appl.* **13**, 121–137. (doi:10.1137/0613011)
44. Golub GH, Meurant G. 2010 *Matrices, moments, and quadrature with applications*. Princeton, NJ: Princeton University Press.
45. Beckermann B, Kuijlaars ABJ. 2002 Superlinear CG convergence for special right-hand sides. *Electron. Trans. Numer. Anal.* **14**, 1–19.
46. Dongarra J *et al*. 2014 Applied mathematics research for exascale computing. Technical Report LLNL-TR-651000, Lawrence Livermore National Laboratory.
47. Top500 List HPCG Benchmark, June 2019. See https://www.top500.org/hpcg/lists/\penalty-\@M2019/06/.
48. Hong J-W, Kung HT. 1981 I/O complexity: the red-blue pebble game. In *Proc. 13th Annual ACM Symp. Theory of Computing, Milwaukee, WI, 11–13 May*, pp. 326–333. New York, NY: ACM.
49. Irony D, Toledo S, Tiskin A. 2004 Communication lower bounds for distributed-memory matrix multiplication. *J. Parallel Dist. Comput.* **64**, 1017–1026. (doi:10.1016/j.jpdc.2004.03.021)
50. Ballard G, Demmel J, Holtz O, Schwartz O. 2011 Minimizing communication in numerical linear algebra. *SIAM J. Matrix Anal. Appl.* **32**, 866–901. (doi:10.1137/090769156)
51. Knight NS. 2015 Communication-optimal loop nests. PhD thesis, University of California, Berkeley.
52. Anderson E *et al*. 1999 *LAPACK users' guide*, 3rd edn. Philadelphia, PA: SIAM.
53. Carson EC, Rozložník M, Strakoš Z, Tichý P, Tůma M. 2018 The numerical stability analysis of pipelined conjugate gradient methods: historical context and methodology. *SIAM J. Sci. Comput.* **40**, A3549–A3580. (doi:10.1137/16M1103361)
54. Meurant G, Strakoš Z. 2006 The Lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numer.* **15**, 471–542. (doi:10.1017/S096249290626001X)
55. Greenbaum A. 1997 Estimating the attainable accuracy of recursively computed residual methods. *SIAM J. Matrix Anal. Appl.* **18**, 535–551. (doi:10.1137/S0895479895284944)
56. Gutknecht MH, Strakoš Z. 2000 Accuracy of two three-term and three two-term recurrences for Krylov space solvers. *SIAM J. Matrix Anal. Appl.* **22**, 213–229. (doi:10.1137/S0895479897331862)
57. van der Vorst HA, Ye Q. 2000 Residual replacement strategies for Krylov subspace iterative methods for the convergence of true residuals. *SIAM J. Sci. Comput.* **22**, 835–852. (doi:10.1137/S1064827599353865)
58. Sleijpen GLG, van der Vorst HA. 1996 Reliable updated residuals in hybrid Bi-CG methods. *Computing* **56**, 141–163. (doi:10.1007/BF02309342)
59. Paige CC. 1980 Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. *Linear Algebra Appl.* **34**, 235–258. (doi:10.1016/0024-3795(80)90167-6)
60. Paige CC. 2010 An augmented stability result for the Lanczos Hermitian matrix tridiagonalization process. *SIAM J. Matrix Anal. Appl.* **31**, 2347–2359. (doi:10.1137/090761343)
61. Paige CC. 2019 Accuracy of the Lanczos process for the eigenproblem and solution of equations. (submitted). See https://www.cs.mcgill.ca/chris/pub/Pai19.pdf.
62. Strakoš Z, Tichý P. 2002 On error estimation in the conjugate gradient method and why it works in finite precision computations. *Electron. Trans. Numer. Anal.* **13**, 8.
63. Hoemmen MF. 2010 Communication-avoiding Krylov subspace methods. PhD thesis, University of California, Berkeley.
64. Carson EC. 2015 Communication-avoiding Krylov subspace methods in theory and practice. PhD thesis, University of California, Berkeley.
65. Carson E, Demmel J. 2014 A residual replacement strategy for improving the maximum attainable accuracy of s-step Krylov subspace methods. *SIAM J. Matrix Anal. Appl.* **35**, 22–43. (doi:10.1137/120893057)

66. Carson E, Demmel JW. 2015 Accuracy of the s-step Lanczos method for the symmetric eigenproblem in finite precision. *SIAM J. Matrix Anal. Appl.* **36**, 793–819. (doi:10.1137/140990735)
67. Davis TA, Hu Y. 2011 The University of Florida sparse matrix collection. *ACM Trans. Math. Softw. (TOMS)* **38**, 1:1–1:25. (doi:10.1145/2049662.2049663)
68. Philippe B, Reichel L. 2012 On the generation of Krylov subspace bases. *Appl. Numer. Math.* **62**, 1171–1186. (doi:10.1016/j.apnum.2010.12.009)
69. Joubert WD, Carey GF. 1992 Parallelizable restarted iterative methods for nonsymmetric linear systems. Part I: theory. *Int. J. Comput. Math.* **44**, 243–267. (doi:10.1080/00207169208804107)
70. Erhel J. 1995 A parallel GMRES version for general sparse matrices. *Electron. Trans. Numer. Anal.* **3**, 160–176.
71. de Sturler E, van der Vorst HA. 1995 Reducing the effect of global communication in GMRES(m) and CG on parallel distributed memory computers. *Appl. Numer. Math.* **18**, 441–459. (doi:10.1016/0168-9274(95)00079-A)
72. Bai Z, Hu D, Reichel L. 1994 A Newton basis GMRES implementation. *IMA J. Numer. Anal.* **14**, 563–581. (doi:10.1093/imanum/14.4.563)
73. Ghysels P, Ashby TJ, Meerbergen K, Vanroose W. 2013 Hiding global communication latency in the GMRES algorithm on massively parallel machines. *SIAM J. Sci. Comput.* **35**, C48–C71. (doi:10.1137/12086563X)
74. Cools S, Vanroose W. 2017 The communication-hiding pipelined BiCGStab method for the parallel solution of large unsymmetric linear systems. *Parallel Comput.* **65**, 1–20. (doi:10.1016/j.parco.2017.04.005)
75. Cornelis J, Cools S, Vanroose W. 2018 The communication-hiding conjugate gradient method with deep pipelines. (http://arxiv.org/abs/1801.04728)
76. Cools S, Yetkin EF, Agullo E, Giraud L, Vanroose W. 2018 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**, 426–450. (doi:10.1137/17M1117872)
77. Simoncini V, Szyld DB. 2003 Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.* **25**, 454–477. (doi:10.1137/S1064827502406415)
78. Gergelits T, Strakoš Z. 2014 Composite convergence bounds based on Chebyshev polynomials and finite precision conjugate gradient computations. *Numer. Algs.* **65**, 759–782. (doi:10.1007/s11075-013-9713-z)
79. Liesen J, Tichý P. 2004 The worst-case GMRES for normal matrices. *BIT Numer. Math.* **44**, 79–98. (doi:10.1023/B:BITN.0000025083.59864.bd)
80. Greenbaum A, Strakoš Z. 1996 Matrices that generate the same Krylov residual spaces. In *Recent advances in iterative methods* (eds G Golub, M Luskin, A Greenbaum). IMA Volumes in Mathematics and its Applications, vol. 60, pp. 95–118. New York, NY: Springer.
81. Greenbaum A, Pták V, Strakoš Z. 1996 Any nonincreasing convergence curve is possible for GMRES. *SIAM J. Matrix Anal. Appl.* **17**, 465–469. (doi:10.1137/S0895479894275030)
82. Arioli M, Pták V, Strakoš Z. 1998 Krylov sequences of maximal length and convergence of GMRES. *BIT Numer. Math.* **38**, 636–643. (doi:10.1007/BF02510405)
83. Murphy MF, Golub GH, Wathen AJ. 2000 A note on preconditioning for indefinite linear systems. *SIAM J. Sci. Comput.* **21**, 1969–1972. (doi:10.1137/S1064827599355153)
84. Carson E, Higham NJ. 2017 A new analysis of iterative refinement and its application to accurate solution of ill-conditioned sparse linear systems. *SIAM J. Sci. Comput.* **39**, A2834–A2856. (doi:10.1137/17M1122918)
85. Bauer S, Huber M, Ghelichkhan S, Mohr M, Rüde U, Wohlmuth B. 2019 Large-scale simulation of mantle convection based on a new matrix-free approach. *J. Comput. Sci.* **31**, 60–76. (doi:10.1016/j.jocs.2018.12.006)
86. Meurant G. 1999 *Computer solution of large linear systems*, vol. 28. Studies in Mathematics and Its Applications. Amsterdam, The Netherlands: Elsevier.
87. Saad Y. 2003 *Iterative methods for sparse linear systems*, 2nd edn. Philadelphia, PA: SIAM.
88. Parlett BN. 1990 The contribution of J. H. Wilkinson to numerical analysis. In *A history of scientific computing* (ed. SG Nash), pp. 17–30. New York, NY: ACM.
89. Golub GH, Strakoš Z. 1994 Estimates in quadratic formulas. *Numer. Algs.* **8**, 241–268. (doi:10.1007/BF02142693)