Numerically Stable Recurrence Relations for the Communication Hiding Pipelined Conjugate Gradient Method

Siegfried Cools*, Jeffrey Cornelis*, Wim Vanroose*

Abstract—Pipelined Krylov subspace methods (also referred to as communication-hiding methods) have been proposed in the literature as a scalable alternative to classic Krylov subspace algorithms for iteratively computing the solution to a large linear system in parallel. For symmetric and positive definite system matrices the pipelined Conjugate Gradient method, p(l)-CG, outperforms its classic Conjugate Gradient counterpart on large scale distributed memory hardware by overlapping global communication with essential computations like the matrix-vector product, thus “hiding” global communication. A well-known drawback of the pipelining technique is the (possibly significant) loss of numerical stability. In this work a numerically stable variant of the pipelined Conjugate Gradient algorithm is presented that avoids the propagation of local rounding errors in the finite precision recurrence relations that construct the Krylov subspace basis. The multi-term recurrence relation for the basis vector is replaced by ℓ two-term recurrences, improving stability without increasing the overall computational cost of the algorithm. The proposed modification ensures that the pipelined Conjugate Gradient method is able to attain a highly accurate solution independently of the pipeline length. Numerical experiments demonstrate a combination of excellent parallel performance and improved maximal attainable accuracy for the new pipelined Conjugate Gradient algorithm. This work thus resolves one of the major practical restrictions for the useability of pipelined Krylov subspace methods.

Index Terms—Krylov subspace methods, Pipelining, Parallel performance, Global communication, Latency hiding, Conjugate Gradients, Numerical stability, Inexact computations, Attainable accuracy.

AMS subject classifications: 65F10, 65N12, 65G50, 65Y05, 65N22.

1 INTRODUCTION

The family of iterative solvers known as Krylov subspace methods (KSMs) [28], [34], [42] are among the most efficient present-day methods for solving large scale sparse systems of linear equations. The mother of all Krylov subspace methods is undoubtedly the Conjugate Gradient method (CG) that was derived in 1952 [24] to the aim of solving linear systems $Ax = b$ with a symmetric positive definite and preferably sparse matrix $A$. The CG method is one of the most widely used methods for solving said systems today, which form the basis of a plethora of scientific and industrial applications. However, driven by the essential transition from optimal single node performance towards massively parallel computer hardware over the last decades [36], the bottleneck for fast execution of Krylov subspace methods has shifted. Whereas in the past the application of the sparse matrix-vector product (SPMV) was considered the most time-consuming part of the algorithm, the global synchronizations required in dot product and norm computations form the main bottleneck for efficient execution on present-day distributed memory hardware [14].

Driven by the increasing levels of parallelism in present-day HPC machines, as attested by the current strive towards exascale high-performance computing software [13], research on the elimination of the global communication bottleneck has recently regained significant attention from the international computer science, engineering and numerical mathematics communities. Sprouting largely from pioneering work on reducing communication in Krylov subspace methods from the late 1980’s and 90’s [5], [11], [12], [15], [30], [39], a number of variants of the classic Krylov subspace algorithms have been introduced over the last years. We point out recent work by Chronopoulos et al. [6], Hoemmen [26], Carson et al. [3], MclInnes et al. [29], Grigori et al. [22], Eller et al. [15], Imberti et al. [27] and Zhuang et al. [46].

The contents of the current work are situated in the research branch on so-called “pipelined” Krylov subspace methods [8], [17], [18]. Alternatively called “communication hiding” methods, these algorithmic variations to classic Krylov subspace methods are designed to overlap the time-consuming global communications in each iteration of the algorithm with computational tasks such as calculating SPMVs or AXPYS (vector operations of the form $y \leftarrow \alpha x + y$). Thanks to the reduction/elimination of the synchronization bottleneck, pipelined algorithms have been shown to increase parallel efficiency by allowing the algorithm to continue

1. Note: In the context of communication reduction in Krylov subspace methods, the terminology “pipelined” KSMs that is used throughout the related applied linear algebra and computer science literature refers to software pipelining, i.e. algorithmic reformulations to the KSM procedure in order to reduce communication overhead, and should not be confused with hardware-level instruction pipelining (ILP).
scaling on large numbers of processors [35], [45]. However, the algorithmic reformulations that allows for this efficiency increase come at the cost of reduced numerical stability [4], [18], which presently is one of the main drawbacks of pipelined (as well as other communication reducing) methods. Research on analyzing and improving the numerical stability of pipelined Krylov subspace methods, which is essential both for a proper understanding and the practical usability of the methods, has recently been performed by the authors [7], [9] and others [2], [4].

This work presents a numerically stable variant of the \( l \)-length pipelined Conjugate Gradient method, \( p(l) \)-CG for short. The \( p(l) \)-CG method was presented in [10] and allows to overlap each global reduction phase with the computational work of \( l \) subsequent iterations. The pipeline length \( l \) is a parameter of the method that can be chosen depending on the problem and hardware setup (as a function of the communication-to-computation ratio). As is the case for all communication reducing Krylov subspace methods, the preconditioner choice influences the communication-to-computation ratio and thus affects the performance of the method. The pipeline length hence also depends on the effort invested in the preconditioner. A preconditioner that uses limited global communication (block Jacobi, no-overlap DDM, ... ) is generally preferred in this setting.

The propagation of local rounding errors in the multi-term recurrence relations of the \( p(l) \)-CG algorithm is the primary source of loss of attainable accuracy on the final solution [10]. By introducing intermediate auxiliary basis variables, we derive a new \( p(l) \)-CG algorithm with modified recurrence relations for which no rounding error propagation occurs. It is proven analytically that the resulting recurrence relations are numerically stable for any pipeline length \( l \). The new algorithm is guaranteed to reach the same accuracy as the classic CG method. This work thus resolves some of the major restrictions for the practical use of pipelined Krylov subspace methods. The redesigned algorithm comes at no additional computational cost and has only a minor storage overhead compared to the former \( p(l) \)-CG algorithm, thus effectively replacing the earlier implementation of the method. In addition, it is shown that formulating a preconditioned version of the new algorithm is straightforward.

We conclude this introduction by providing a short overview of the further contents of this manuscript. Section 2 presents a high-level summary of the basic principles behind the \( l \)-length pipelined CG method and formulates the key numerical properties of the method that motivate this work. It familiarizes the reader with the notations and concepts used throughout this paper. In Section 3 the numerical stability analysis of the \( p(l) \)-CG recurrence relations is briefly recapped, as it forms the basis for the analysis of the stable algorithm in Section 4. Section 5 contains the main contributions of this work, presenting the technical details of the stable \( p(l) \)-CG algorithm alongside an overview of its main implementation properties and a numerical analysis of the new rounding error resilient recurrence relations. Numerical experiments that demonstrate both the parallel performance of the \( p(l) \)-CG method and the excellent attainable accuracy in comparison to earlier variants of pipelined Krylov subspace methods are presented in Section 5. The manuscript is concluded in Section 6.

2 Deep pipelined Conjugate Gradients

The deep pipelined Conjugate Gradient method, denoted \( p(l) \)-CG for short, was first presented in [10], where it was derived in analogy to the \( p(l) \)-GMRES method [17]. The parameter \( l \) represents the pipeline length which indicates the number of iterations that are overlapped by each global reduction phase. We summarize the current state-of-the-art deep pipelined \( p(l) \)-CG method below, which forms the starting point for the discussion in this work.

2.1 Basis recurrence relations in exact arithmetic

Let \( V_{i-1,l+1} = [w_0, w_1, \ldots, w_{l-1}] \) be the orthonormal basis for the Krylov subspace \( K_{i-1,l+1} \) in iteration \( i \) of the \( p(l) \)-CG algorithm, consisting of \( i - l + 1 \) vectors. Here \( A \) is a symmetric positive definite matrix. The Krylov subspace basis vectors satisfy the Lanczos relation

\[
AV_j = V_{j+1}T_{j+1,j}, \quad 1 \leq j \leq i - l.
\]

with

\[
T_{j+1,j} = \begin{pmatrix}
\gamma_0 & \delta_0 & \cdots & \\
\delta_0 & \gamma_1 & \cdots & \\
& \cdots & \cdots & \cdots \\
& & \cdots & \gamma_{j-1} \\
& & & \delta_{j-2}
\end{pmatrix}
\]

Let \( \delta_{-1} = 0 \), then the Lanczos relation (1) translates in vector notation to

\[
v_{j+1} = (Av_j - \gamma_jv_j - \delta_{j-1}v_{j-1})/\delta_j, \quad 0 \leq j \leq i - l.
\]

The auxiliary basis \( Z_{i+1} = [z_0, z_1, \ldots, z_l] \) runs \( l \) vectors ahead of the basis \( V_{i-1,l+1} \) and is defined as

\[
z_j = \begin{cases}
  v_0, & j = 0, \\
  P_l(A)v_j, & 0 < j \leq l, \\
  P_l(A)v_{j-l}, & l < j \leq i,
\end{cases}
\]

where the matrix polynomial \( P_l(A) \) is given by

\[
P_k(t) = \prod_{j=0}^{k-1}(t - \sigma_j), \quad k \leq l,
\]

with optional stabilizing shifts \( \sigma_j \in \mathbb{R} \), see [10], [17], [26]. We refer to Section 2.2 for a discussion on the Krylov subspace basis \( Z_{i+1} \), i.e. the choice of the polynomial \( P_l(A) \). Contrary to the basis \( V_{i-1,l+1} \), the auxiliary basis \( Z_{i+1} \) is in general not orthonormal. It is constructed using the recursive definitions

\[
z_{j+1} = \begin{cases}
  (A - \sigma_jI)z_j, & 0 \leq j \leq l, \\
  (Az_j - \gamma_jz_j - \delta_{j-1}z_{j-1})/\delta_j, & l \leq j < i,
\end{cases}
\]

which are obtained by multiplying the Lanczos relation (1) on both sides by \( P_l(A) \). Expression (6) translates into a Lanczos type matrix relation

\[
AZ_j = Z_{j+1}B_{j+1,j}, \quad 1 \leq j \leq i,
\]

where the matrix \( B_{j+1,j} \) contains the matrix \( T_{j+1,l+1,j-l}, \) which is shifted \( l \) places along the main diagonal. The bases \( V_j \) and \( Z_j \) are connected through the basis transformation

\[
Z_j = V_jG_j \quad 1 \leq j \leq i - l + 1, \quad \text{where } G_j \text{ is a banded matrix}.
\]
upper triangular matrix with a bandwidth of $2l + 1$ non-zero diagonals. For a symmetric matrix $A$ the matrix $G_{i+1}$ is symmetric around its $l$-th upper diagonal, since

$$g_{i,i} = (z_i, v_j) = (P_i(A)v_{i-1}, v_j) = (v_{i-1}, P_i(A)v_j) = (v_{i-1}, z_{j+l}).$$

The following recurrence relation for $v_{j+1}$ is derived from the basis transformation (with $0 \leq j < i - l$):

$$v_{j+1} = \left(z_{j+1} - \sum_{k=j-2l+1}^{j} g_{k,j+1} v_k\right) / g_{j+1,j+1}.$$  \hspace{1cm} (9)

A total of $l$ iterations after the dot-products

$$g_{j,i+1} = \begin{cases} (z_{i+1}, v_j); & j = \max(0, i - 2l + 1), \ldots, i - l + 1, \\ (z_{i+1}, z_j); & j = i - l + 2, \ldots, i + 1, \end{cases} \hspace{1cm} (10)$$

have been initiated, the elements $g_{j,i+1}$ with $i - 2l + 2 \leq j \leq i - l + 1$, which were computed as $(z_{i+1}, z_j)$, are corrected as follows:

$$g_{j,i+1} = g_{j,i+1} - \sum_{k=i-2l+1}^{i-1} g_{k,j} g_{k,i+1}, \hspace{1cm} (11)$$

for $j = i - 2l + 2, \ldots, i - l$, and:

$$g_{i-1,l+i-1} = \sqrt{g_{i-1,l+i-1} - \sum_{k=i-2l+1}^{i-1} g_{k,i+1}^2}.$$  \hspace{1cm} (12)

Additionally, in the $i$-th iteration the tridiagonal matrix $T_{i-1,l+i-1}$, see [1], can be updated recursively by adding one column. The diagonal element $\gamma_{i-1}$ is characterized by the expressions:

$$\gamma_{i-1} = \begin{cases} g_{i-1,l+i-1} + \sigma_{i-1} g_{i-1,l+i-1} - g_{i-1,l+i-1} \delta_{i-1} - 1, & l \leq i < 2l, \\ g_{i-1,l+i-1} - \gamma_{i-2l} + g_{i-1,l+i-1} \delta_{i-2l} - g_{i-1,l+i-1} - \delta_{i-1} - 1, & i \geq 2l, \\ g_{i-1,l+i-1}, & i = 2l. \end{cases}$$  \hspace{1cm} (13)

The term $-g_{i-1,l+i-1} \delta_{i-1}$ is considered zero when $i = l$. The update for the off-diagonal element $\delta_{i-1}$ is given by

$$\delta_{i-1} = \begin{cases} g_{i-1,l+i-1} / g_{i-1,l+i-1}, & l \leq i < 2l, \\ g_{i-1,l+i-1} / g_{i-1,l+i-1}, & i \geq 2l. \end{cases}$$  \hspace{1cm} (14)

The element $\delta_{i-1}$ has already been computed in the previous iteration and can thus simply be copied due to the symmetry of $T_{i-1,l+i-1}$.

Once the basis $V_{i-1}$ has been constructed, the solution $x_{i-1}$ can be updated based on a search direction $p_{i-1}$, following the classic derivation of D-Lanczos in [34]. Sec. 6.7.1. The Ritz-Galerkin condition

$$0 = V_m^T r_m = V_m^T (r_0 - AV_m y_m) = V_m^T (V_m r_0^2 e_1) - T_m y_m = r_0^2 e_1 - T_m y_m.$$  \hspace{1cm} (15)

implies $y_m = T_m^{-1} r_0^2 e_1$. The LU-factorization of the tridiagonal matrix $T_{i-1,l+i-1} = L_{i-1,l+i-1}$ is given by

$$\begin{pmatrix} 1 & \lambda_1 & \cdots & 0 \\ \lambda_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{i-1} & \cdots & \cdots & 1 \end{pmatrix} \begin{pmatrix} \eta_0 \\ \delta_0 \\ \eta_1 \\ \cdots \\ \delta_{i-1} \\ \eta_{i-1} \end{pmatrix} = \begin{pmatrix} \eta_0 \\ \delta_0 \\ \eta_1 \\ \cdots \\ \delta_{i-1} \\ \eta_{i-1} \end{pmatrix}.$$  \hspace{1cm} (16)

Note that $\gamma_0 = \eta_0$. It follows from (16) that the elements of the lower/upper triangular matrices $L_{i-1,l+i}$ and $U_{i-1,l+i}$ are given by (with $1 \leq j \leq i - l$)

$$\lambda_j = \delta_j - 1/\eta_j - 1, \quad \eta_j = \gamma_j - \lambda_j \delta_j - 1.$$  \hspace{1cm} (17)

Expression (15) indicates that the approximate solution $x_{i-1}$ equals

$$x_{i-1} = x_0 + V_{i-1} y_{i-1} = x_0 + V_{i-1} L_{i-1,l+i}^{-1} r_0^2 e_1 = x_0 + V_{i-1} U_{i-1,l+i}^{-1} r_0^2 e_1.$$  \hspace{1cm} (18)

where $P_{i-1} = V_{i-1} L_{i-1,l+i}^{-1}$ and $q_{i-1} = L_{i-1,l+i}^{-1} r_0^2 e_1$. Note that $p_{i-1} = v_0 / \eta_0$. The columns $p_j$ (for $1 \leq j \leq i - l$) of $P_{i-1}$ can be computed recursively. From $P_{i-1} U_{i-1,l+i} = V_{i-1,l+i}$ it follows

$$p_j = \eta_j^{-1} (v_j - \delta_j - 1 p_{j-1}), \quad 1 \leq j \leq i - l.$$  \hspace{1cm} (19)

Denoting the vector $q_{i-1}$ by $[\zeta_0, \ldots, \zeta_{i-1}]^T$, it follows from $L_{i-1,l+i} q_{i-1} = ||r_0||_2 e_1$ that $\zeta_0 = ||r_0||_2$ and $\zeta_j = -\lambda_j \zeta_{j-1}$ for $1 \leq j \leq i - l$. Using the search direction $p_{i-1}$ and the scalar $\zeta_{i-1}$, the approximate solution $x_{i-1}$ is updated using the recurrence relation:

$$x_{i-1} = x_{i-1} + \zeta_{i-1} p_{i-1}.$$  \hspace{1cm} (20)

The above expressions are combined in Alg. [1]. Once the initial pipeline for $z_0, \ldots, z_l$ has been filled, the relations (16-19) are used to recursively compute the basis vectors $v_{i-1}$ and $z_{i-1}$ in iterations $i \geq l$ (see lines 15-16). The scalar results of the global reduction phase (line 18) are required $l$ iterations later (line 5-6). In every iteration global communication is thus overlapped with the computational work of $l$ subsequent iterations, forming the heart of the communication hiding p(l)-CG algorithm.

Remark 1. Residual norm in p(l)-CG. Note that the residual $r_j = b - A x_j$ is not computed in Alg. [1] but its norm is characterized by the quantity $|\zeta_j| = ||r_0||_2$ for $0 \leq j \leq i - l$. This quantity can be used to formulate a stopping criterion for the p(l)-CG iteration, see Alg. [1] line 27.

Remark 2. Dot products in p(l)-CG. Although Alg. [1] line 18 indicates that in each iteration $i \geq (2l + 1)$ a total of $(2l + 1)$ dot products need to be computed, the number of dot product computations can be limited to $(l+1)$ by exploiting the symmetry of the matrix $G_{i+1}$, see expression (3). Since $g_{j,i+1} = g_{i-1,j+i}$ for $j \leq i + 1$, only the dot products $(z_j, z_{j+1})$ for $j = i - l + 2, \ldots, i+1$ and the $l$-th upper diagonal element $(v_{i-1}, z_{i+1})$ need to be computed in iteration $i$.

2.2 On the conditioning of the auxiliary basis

As $V_i$ is an orthonormal basis, the transformation $Z_i = V_i G_i$ can be interpreted as a QR factorization of the auxiliary basis $Z_i$. Moreover, it holds that $G_i^T G_i$ is the Cholesky factorization of $Z_i^T Z_i$, since

$$Z_i^T Z_i = G_i^T V_i^T V_i G_i = G_i^T G_i.$$  \hspace{1cm} (21)

The elements of the transformation matrix $G_i$ are computed on lines 5-6 of Alg. [1] precisely by means of this Cholesky factorization. This observation leads to the following two essential insights related to the conditioning of the basis $Z_i$. 
Remark 3. Square root breakdowns in p(l)-CG. The auxiliary basis vectors $z_j$ are defined as $P_l(A)v_{j-1}$, but the basis $Z_l$ is in general not orthogonal. Hence, vectors $z_j \in Z_l$ are not necessarily linearly independent. In particular for longer $l$, different $z_j$ vectors are expected to become more and more aligned. This leads to $Z_l^T Z_l$ being ill-conditioned, closing to singularity as $i$ augments. The effect is the most pronounced when $\sigma_0 = \ldots = \sigma_{l-1} = 0$, in which case $P_l(A) = A^l$. Shifts $\sigma_i$ can be set to improve the conditioning of $Z_l^T Z_l$, see also Remark 4.

When for certain $i$ the matrix $Z_l^T Z_l$ becomes (numerically) singular, the Cholesky factorization procedure in p(l)-CG will fail. This may manifest in the form of a square root breakdown on line 7 in Alg. 1 when the root argument $g_{l+1,i-l+1} - \sum_{j=3+1}^{i-1} g_{l+1,j}^2$ becomes negative. Numerical round-off errors may increase the occurrence of these breakdowns in practice. When a breakdown occurs in p(l)-CG the iteration is restarted, in analogy to the GMRES algorithm, although it should be noted that the nature of the breakdown in both algorithms is quite different. Evidently, the restarting strategy may delay convergence compared to standard CG, where no square root breakdowns occur.

Remark 4. Choice of the auxiliary basis and relation to the shifts. It follows from the Cholesky factorization \[ G_l^{-1} = (Z_l^T Z_l)^{-1} G_l^T, \] that the inverse of the transformation matrix $G_l$ is given by

\[ G_l^{-1} = (Z_l^T Z_l)^{-1} G_l^T. \]

The conditioning of the matrix $G_l^{-1}$ is thus determined by the conditioning of $Z_l^T Z_l$. Furthermore, it holds that

\[ Z_{l+1} = (P_l(A)V_{i-1})^T P_l(A)V_{i-1} = V_{i-1}^T P_l(A)^2 V_{i-1}. \]

where $Z_i$ is a part of the basis $Z_i$ obtained by dropping the first $l$ columns. Hence, the polynomial $P_l(A)$ has a major impact on the conditioning of the matrix $G_l^{-1}$, which in turn plays a crucial role in the propagation of local rounding errors in the p(l)-CG algorithm [10], see Section 3.1 of the current work. This observation indicates intuitively why $P_l(A)$ should preferably be as small as possible, which can be achieved by choosing appropriate values for the shifts $\sigma_i$. Optimal shift values in the sense of minimizing the Euclidean 2-norm of $P_l(A)$ are the Chebyshev shifts [10, 17, 26] (for $i = 0, \ldots, l - 1$):

\[ \sigma_i = \frac{\lambda_{\max} + \lambda_{\min}}{2} + \frac{\lambda_{\max} - \lambda_{\min}}{2} \cos \left( \frac{(2i + 1)\pi}{2l} \right), \]

which are used throughout this work. This requires a notion of the largest (smallest) eigenvalue $\lambda_{\max}$ (resp. $\lambda_{\min}$), which can be estimated a priori, e.g. by a few power method iterations.

2.3 Loss of orthogonality and attainable accuracy

This work is motivated by the observation that two main issues affect the convergence of pipelined CG methods: loss of basis vector orthogonality and inexact Lanczos relations. We comment briefly on both issues from a high-level point of view and clearly mark the scope of this work. Important insights about the similarities and differences between classical CG and p(l)-CG are highlighted before going into more details on the numerics in Sections 3 and 4.

2.3.1 Loss of orthogonality

It is well-known that in finite precision arithmetic the orthogonality of the Krylov subspace basis $V_i$, i.e. $V_i^T V_i = I_i$,
(identity matrix) may not hold exactly. Inexact orthogonality may appear in every variant of the CG algorithm, see [28], in particular in the D-Lanczos algorithm [34], where a new basis vector is constructed by orthogonalizing with respect to the previous two basis vectors, as well as in the related p(l)-CG method, Alg. 2. Loss of orthogonality typically leads to delay of convergence, meaning the residual deviates from the one in the scenario in which orthogonality would not be lost. We use a notation with bars to designate variables that are actually computed in a finite precision implementation of the algorithm. The key relation for the Conjugate Gradient method is the Ritz-Galerkin condition:

$$V_i^T(b - Ax_i) = V_i^T(\bar{r}_0 - AV_i\bar{T}_i^{-1}\bar{r}_0)\leq (1)$$

This equality only holds under the assumption that $$V_i^TAV_i = T_i$$ which requires $$V_i^TV_{i+1} = I_i$$, i.e., the basis vectors are orthogonal. Note that in finite precision arithmetic the convergence delay can be observed in both the actual residual norm $$\|b - Ax_i\|$$ as well as the recursively computed residual norm $$\|\bar{r}_i\|$$, where both quantities are based on the (possibly non-orthogonal) basis $$\bar{V}_{i+1}$$, see Fig. 1a-2a (discussed further in Section 2.3.3).

### 2.3.2 The inexact Lanczos relation

The basis vectors in the pipelined CG algorithm are not computed explicitly using the Lanczos relation (1). Rather, they are computed recursively, see [9], to avoid the computation of additional SPMVs. In finite precision, local rounding errors in the recurrence relation may contaminate the basis $$\bar{V}_i$$, such that the Lanczos relation $$AV_i - \bar{V}_{i+1}\bar{T}_{i+1} = 0$$ is no longer valid. Moreover, due to propagation of local rounding errors, $$\|AV_i - \bar{V}_{i+1}\bar{T}_{i+1}\|$$ may grow dramatically as the iteration proceeds. Using the classic model for floating point arithmetic with machine precision $$\epsilon$$ ([20], [21], [32], [44]), the round-off error on basic computations on the matrix $$A \in \mathbb{R}^{n \times n}$$, vectors $$v$$, $$w$$, and a scalar $$\alpha$$ are bounded by

$$\|\alpha v - \bar{f}(\alpha v)\| \leq \|\alpha v\|\epsilon = |\alpha|\|v\|\epsilon,$$

$$\|v + w - \bar{f}(v + w)\| \leq \|v\| + \|w\|\epsilon,$$

$$\|Av - \bar{f}(Av)\| \leq \mu v\sqrt{n}\|A\|\|v\|\epsilon,$$

Here $$\bar{f}(\cdot)$$ indicates the finite precision floating point representation, $$\mu$$ is the maximum number of nonzeros in any row of $$A$$, and the norm $$\|\cdot\|$$ represents the Euclidean 2-norm. Under this model the recurrence relations for $$\bar{v}_i$$ and $$\bar{p}_i$$ in a finite precision implementation of p(l)-CG are

$$\bar{x}_i = \bar{x}_{i-1} + \bar{z}_{i-1}\bar{p}_{i-1} + \xi^p = \bar{x}_0 + \bar{P}\bar{q}_0 + \Xi^p 1,$$

$$\bar{p}_i = (\bar{v}_i - \delta_{i-1}\bar{p}_{i-1})/\bar{q}_i + \xi^p \Rightarrow \bar{V}_i = \bar{P}\bar{U}_i + \Xi^p,$$

where $$\Xi^p = [\xi^p_1, \ldots, \xi^p_{i-1}$$] and $$\Xi^p = [\bar{p}_0\bar{q}_0, \ldots, \bar{p}_{i-1}\bar{q}_{i-1}]$$ are local rounding errors which are bounded by $$\|\Xi^p\| \leq \|\xi^p\| + 2\|\bar{z}_{i-1}\|\|\bar{p}_{i-1}\|\epsilon$$ and $$\|\xi^p\| \leq \|\xi^p\| + 3\|\bar{z}_{i-1}\|\|\bar{q}_i\|\|\bar{p}_{i-1}\|\epsilon$$, and $$1 = [1, \ldots, 1]^T$$. The actual residual satisfies the following relations in a finite precision setting:

2. Note: The D-Lanczos (short for "direct Lanczos") algorithm is a variant of the CG method that is equivalent to the latter in exact arithmetic, save for the solution of the system $$T_iy_i = \|r_i\|e_1$$ which is computed by using Gaussian elimination in D-Lanczos. The D-Lanczos method is the basic Krylov subspace method from which the p(l)-CG method was derived, see [10], Section 2, for details.

$$b - A\bar{x}_i = \bar{r}_0 - AV_i\bar{T}_i^{-1}\bar{r}_0 + AV_i\bar{T}_i^{-1}\xi^p + \Xi^p 1 + \xi^p_0$$

$$= \bar{r}_0 - \bar{V}_{i+1}\bar{T}_{i+1,1}\bar{U}_i^{-1}\bar{r}_0 - (AV_i - \bar{V}_{i+1}\bar{T}_{i+1,1})\bar{U}_i^{-1}\bar{r}_0 + LRE$$

$$= \bar{r}_i - (AV_i - \bar{V}_{i+1}\bar{T}_{i+1,1})\bar{U}_i^{-1}\bar{r}_i + LRE$$

where $$\xi^p_0 = (b - \bar{r}_0) - \bar{r}_0$$.

The recursively computed residual $$\bar{r}_i = -\delta_{i-1}(\xi^p_0\bar{U}_i^{-1}\bar{q}_i\bar{v}_i = \xi^p_0\bar{v}_i$$ that appears in expression (25) tends to zero. The actual residual norm $$\|b - A\bar{x}_i\|$$ on the other hand, stagnates around $$\|(AV_i - \bar{V}_{i+1}\bar{T}_{i+1,1})\bar{U}_i^{-1}\bar{r}_i\|$$, a quantity referred to as the maximal attainable accuracy of the method. The difference between the norm of the actual residual and the recursively computed residual is illustrated in Fig. 1a. A detailed analysis of the deviation from the Lanczos relation in finite precision ("inexact Lanczos") can be found in Section 3.

### 2.3.3 Scope and limitations of this manuscript

The issue of inexact Lanczos relations in p(l)-CG is timely and deserving of attention. Loss of accuracy resulting from the inexact Lanczos relation has long been a limiting factor in applying p(l)-CG and related algorithms in practice. Fig. 1a illustrates how the norms of the actual residuals $$\|b - A\bar{x}_i\|$$ stagnate while the recursively computed residual norms $$\|\bar{r}_i\|$$ continue to decrease. For p(l)-CG local rounding errors in the recurrence relations are propagated leading to reduced attainable accuracy compared to D-Lanczos. Moreover, while loss of orthogonality also warrants further investigation, this issue is not exclusive to pipelined methods. Delayed convergence is observed in classic Krylov subspace methods also, see Fig. 1b, whereas loss of attainable accuracy is not. The issue could be addressed by e.g. re-orthogonalizing the basis, see [41]. However, communication-reducing methods are not particularly suitable to include re-orthogonalization, since this introduces additional global reduction phases.

Although loss of orthogonality does not originate from applying the pipelining technique, it may be more pronounced for pipelined methods compared to their classic counterparts, see Fig. 1c. However, Fig. 1d indicates that the effect of the inexact Lanczos relation on convergence is much more dramatic than the effect of inexact orthogonality for all pipeline lengths I. This manuscript thus focuses on improving the numerical stability of the p(l)-CG method by neutralizing the propagation of local rounding errors in the recursively computed basis vector updates. As such, this work proposes a key step towards a numerically stable communication hiding variant of the CG method.

### 3 Analyzing rounding error propagation

This section recaps the analysis of local rounding errors that stem from the recurrence relations in the pipelined p(l)-CG method, Alg. 2. It aims to precisely explain the source of the loss of accuracy observed for the p(l)-CG method. The methodology for the analysis is similar to the one used in classic works by Paige [32], [33], Greenbaum [19], [20], [21], Gutknecht [23], Strakos [40], Meurant [31], Sleijpen [37], [38], Van der Vorst [43], Higham [25], and others.

Finite precision variants of the exact scalar and vector variables introduced in Section 2 are denoted by a bar
symbol in this section. We differentiate between “actual”
vector variables, which satisfy the Lanczos relations exactly
but are not computed in the algorithm, and “recursively
computed” variables, which contain machine-precision sized
round-off errors related to finite precision computations.

3.1 Local rounding error behavior in finite precision

For any \( j \geq 0 \) the true basis vector, denoted by \( \vec{v}_{j+1} \), satisfies the Lanczos relation (3) exactly, that is, for \( 0 \leq j < i - l \):

\[
\vec{v}_{j+1} = (A\vec{v}_j - \gamma_j \vec{v}_j - \delta_{j-1} \vec{v}_{j-1})/\delta_j,
\]

without the addition of a local rounding error. This vector is not actually computed in the p(l)-CG algorithm. Instead, the computed basis vector \( \tilde{v}_{j+1} \) is calculated from the finite precision variant of relation (3) for \( 0 \leq j < i - l \), i.e.:

\[
\tilde{v}_{j+1} = \left( \vec{v}_{j+1} - \sum_{k=j-2l+1}^j g_{k,j+1} \vec{v}_k \right)/g_{j+1,j+1} + \xi_{j+1},
\]

where the size of the local rounding errors is bounded in terms of machine precision: \( \|\xi_{j+1}\| \leq (2 \|\tilde{v}_{j+1}\|/g_{j+1,j+1} + 3 \sum_{k=j-2l+1}^j |g_{k,j+1}|/|g_{j+1,j+1}|)\|\vec{v}_k\|\epsilon \). Let \( \tilde{V}_{j+1} = [\tilde{v}_0, \tilde{v}_1, \ldots, \tilde{v}_j] \) and \( \tilde{V}_{j+1} = [V_0, \tilde{v}_1, \ldots, V_j] \). Relation (29) alternatively translates to the following formulation in matrix notation (with \( 1 \leq j \leq i - l \)):

\[
A\tilde{V}_{j+1} = \tilde{V}_{j+1}T_{j+1} + (\tilde{V}_{j+1} - \tilde{V}_j)\Delta_{j+1,j},
\]

where \( \Delta_{j+1,j} \) is a \((j + 1)\)-by-\(j\) rectangular matrix holding the entries \( \{\delta_0, \ldots, \delta_{j-1}\} \) directly below the main diagonal. The matrix \( (\tilde{V}_{j+1} - \tilde{V}_j) = [V_0 - \tilde{v}_0, V_1 - \tilde{v}_1, \ldots, V_j - \tilde{v}_j] \) collects the gaps between the actual and recursively computed basis vectors, which quantify the deviation from the Lanczos relation in the finite precision setting. These gaps are crucial in describing the propagation of local rounding errors throughout the p(l)-CG algorithm and are directly linked to the gap between the actual and recursively computed residuals, see expression (28). From (30) one obtains

\[
\tilde{Z}_j = \tilde{V}_jG_j + \Xi_j^0, \quad 1 \leq j \leq i - l,
\]

where \( \Xi_j^0 = [0, g_{1,j}\xi_1^0, \ldots, g_{j-1,j-1}\xi_{j-1}^0] \) collects the local rounding errors. The computed auxiliary vector \( \tilde{z}_{j+1} \) satisfies a finite precision version of the recurrence relation (6), which summarizes to

\[
\tilde{z}_{j+1} = \begin{cases}
(A - \sigma_j I)\tilde{z}_j + \Xi_j^0, & 0 \leq j < l, \\
A\tilde{z}_j - \gamma_j \tilde{z}_j - \delta_{j-1}\tilde{z}_{j-1} + \xi_{j+1}, & l \leq j < i,
\end{cases}
\]

where \( \xi_{j+1}^0 \) is the local rounding error which can again be bounded in terms of machine precision \( \epsilon \). Expression (33) can be formulated in matrix notation as:

\[
A\tilde{Z}_j = \tilde{Z}_{j+1}B_{j+1,j} + \Xi_j^0; \quad 1 \leq j \leq i.
\]

Furthermore, the following matrix relations hold between the scalar coefficients \( \tilde{\gamma}_j \) and \( \tilde{\delta}_j \) in Alg. 1:

\[
\tilde{G}_{j+1}B_{j+1,j} = T_{j+1,j}\tilde{G}_j; \quad 1 \leq j \leq i - l.
\]

Subsequently, using expressions (31), (32), (34) and (35) and it is derived that the gaps on the basis vectors are given by

\[
\tilde{V}_{j+1} - \tilde{V}_j = (\Xi_j^0 - A\Xi_j^0 + \Xi_j^0B_{j+1,j})G_j^{-1}\Delta_{j+1,j},
\]

where \( \Delta_{j+1,j} = (\Delta_{j+1,j}^{++}\Delta_{j+1,j})^{-1}\Delta_{j+1,j}^{++} \) should be interpreted as a Moore-Penrose (left) pseudo-inverse. Hence, the local rounding errors in this expression are possibly amplified by the entries of the matrix \( G_j^{-1}\Delta_{j+1,j}^{++} \), which may lead to loss of attainable accuracy for the p(l)-CG method. The inexact Lanczos relation may in turn give rise to a growing gap between the computed and actual residual on the solution, see (28). It is clear from expression (36) that the conditioning of the matrix \( G_j^{-1}\Delta_{j+1,j}^{++} \) plays a crucial role in the rounding error propagation in the p(l)-CG algorithm as indicated in Section 2.2 see Remark 4.

Fig. 1: Reduced attainable accuracy and loss of orthogonality in CG variants. (a) Actual residual norm \( \|b - A\tilde{x}\|/\|b\| \) (full line) and recursive residual norm \( \|\tilde{r}_i\|/\|b\| \) (dotted line) for a 2D 100 \times 100 unknowns Laplace problem. Square root breakdown indicated by \( \star \), see Remark 4. (b) Basis orthogonality characterized by \( \|I - V_i^T V_i\|_\infty \). For D-Lanczos (cyan) orthogonality was modified by setting \( \gamma_i = (A\tilde{v}_i, \tilde{v}_i)/(1 + \epsilon) \) with \( \epsilon = 1.0\times10^{-11} \) in each iteration.
3.2 Toward stability by using the Lanczos relation

Section 3.1 shows that the recurrence relation (30) is the main cause for the amplification of local rounding errors throughout the p(l)-CG algorithm. Moreover, the possibly ill-conditioned matrix $G_j$ that is used construct the basis $V_j$, see expression (32), may be detrimental for convergence. A straightforward countermeasure would be to eliminate $G_j$ in the construction of the basis. This can be achieved by simply replacing the recurrence relation (30) by the original Lanczos relation, i.e., for $0 \leq j < i - l$:

$$\bar{v}_{j+1} = (A\bar{v}_j - \bar{γ}_j\bar{v}_j - \bar{δ}_{j-1}\bar{v}_{j-1})/\bar{δ}_j + \psi_0^{\bar{j}+1}. \quad (37)$$

Here $\psi_0^{\bar{j}+1}$ represents a local rounding error which is generally different from the error $\xi_j$ occurring in expression (30). Recurrence relation (37) can alternatively be written as

$$A\bar{V}_j = \bar{V}_{j+1}T_{j+1,j} - \Psi_{j+1,\bar{j}+1}\bar{A}_{j+1,j}, \quad (38)$$

with $1 \leq j \leq i - l$. The gap between the true basis vector $\bar{v}_{j+1}$ and the computed basis vector $\bar{v}_{j+1}$ then reduces to

$$\bar{V}_{j+1} - \bar{V}_j = -\Psi_{j+1,\bar{j}+1}. \quad (39)$$

By using recurrence (37) for $\bar{v}_{j+1}$ instead of relation (30) in Algorithm 1, no amplification of local rounding errors occurs, see (39), and the influence of rounding errors on attainable accuracy remains limited. However, to use the recurrence relation (37) an additional SPMV, i.e. $A\bar{v}_j$, is computed in each iteration of the algorithm, leading to an undesirable increase in computational cost. Although the use of expression (37) would not hinder the ability to overlap the global reduction phase with computations (for pipeline length $l$ the global reduction would simply be overlapped with $2l$ SPMVs), we aim to avoid adding SPMV computations to the algorithm.

The technique proposed by expression (37) shows similarity to the concept of residual replacement, which was suggested by several authors as a countermeasure to local rounding error propagation in various multi-term recurrence variants of CG [3, 9, 43]. While the idea is valuable, it cannot be implemented in the p(l)-CG method in its current form, i.e. using expression (37), due to the significantly augmented computational cost caused by the additional SPMV in each iteration.

4 DERIVING STABLE RECURRENCES RELATIONS

We now present the core technique for obtaining a numerically stable variant of the recurrence relations used in the p(l)-CG algorithm by introducing additional auxiliary bases and corresponding recurrence relations. Sections 4.1, 4.3 are again written in the exact arithmetic framework in order to derive the algorithm, interluded by a short discussion on computational costs and storage requirements in Section 4.2.

We return to the finite precision framework for the analysis of the improved method in Section 4.4.

4.1 Derivation of a stable pipelined CG method

We introduce a total of $l + 1$ bases, denoted by $Z_j^{(l)}$, where the upper index $(l) \prime$ (with $0 \leq k \leq l$) labels the different bases and the lower index 'i' indicates the iteration like before. The basis $Z_i^{(0)}$ will denote the original Krylov subspace basis, that is: $Z_i^{(0)} = V_i$. The auxiliary basis vectors $Z_j^{(l)} = [z_j^{(l)}, z_j^{(l)}, \ldots, z_j^{(l)}]$ are defined similarly to the basis $Z_{i+1}$ in p(l)-CG, cf. (3), i.e.:

$$z_j^{(l)} = \begin{cases} v_0, & j = 0, \\ P_l(A)v_j, & 0 < j < k, \\ P_k(A)v_{j-l}, & l < j \leq i, \end{cases} \quad (40)$$

where the polynomial is defined by (5). In addition, we also define $l - 1$ intermediy bases $Z_j^{(l)}, \ldots, Z_j^{(l)}$ that will enable us to use a variant of the Lanczos relation (32) to recursively update $v_{j+1}$, but without the necessity to compute the SPMV $A\bar{v}_j$. The auxiliary bases are defined as follows:

$$z_j^{(k)} = \begin{cases} v_0, & j = 0, \\ P_l(A)v_j, & 0 < j < k, \\ P_k(A)v_{j-k}, & k < j \leq i, \end{cases} \quad (41)$$

Note that the first $k + 1$ vectors in basis $Z_j^{(k)} (j \geq k)$ are identical to the first $k + 1$ vectors in all bases $Z_j^{(l)}$ with $k' \geq k$. By definition (41) the ‘zero-th’ basis $Z_j^{(0)}$ is simply the original basis $V_j$, whereas the $l$-th basis $Z_j^{(l)}$ is the auxiliary basis $Z_j$ from the p(l)-CG method, see Section 2.

A crucial relation connects each pair of bases $Z_j^{(k)}$ and $Z_j^{(k+1)}$ (for $j > k$). It holds that

$$z_j^{(k+1)} = P_{k+1}(A)v_{j-k} = (A - σ_k)P_k(A)v_{j-k} = (A - σ_k)z_j^{(k)}, \quad j \geq k, \quad 0 \leq k \leq l - 1, \quad (42)$$

which translates into

$$A z_j^{(k+1)} = z_j^{(k+1)} + σ_k z_j^{(k)}, \quad j \geq k, \quad 0 \leq k \leq l - 1. \quad (43)$$

By multiplying the original Lanczos relation (3) for $v_j$ on both sides by the respective polynomial $P_k(A)$ with $1 \leq k \leq l$ and by exploiting the associativity of $A$ and $P_k(A)$, it is straightforward to derive that each auxiliary basis $Z_j^{(k)}$ with $0 \leq k \leq l$ satisfies a Lanczos type recurrence relation:

$$z_j^{(k+1)} = (A z_j^{(k)} - γ_j z_j^{(k)} - δ_{j-1} z_j^{(k)} - δ_j z_j^{(k)} / δ_j), \quad (44)$$

for $j \geq k$ and $0 \leq k \leq l$. Note that when $k = 0$ expression (44) yields the Lanczos relation (3) for $v_{j+1}$, whereas setting $k = l$ results in the recurrence relation (6) for $z_{j+1}$.

The recursive expressions (44) for the bases $Z_j^{(0)}$, ..., $Z_j^{(l)}$ are not particularly useful in practice, since each recurrence relation requires to compute an additional SPMV to form the next basis vector. However, using relation (45) the recurrence relations (44) can alternatively be written as:

$$z_j^{(k+1)} = (z_j^{(k+1)} + (σ_k - γ_j) z_j^{(k)} - δ_{j-1} z_j^{(k)} - δ_j z_j^{(k)} / δ_j), \quad (45)$$

with $j \geq k$ and $0 \leq k < l$. We stress that only for $k = l$, i.e. to compute the vectors in the auxiliary basis $Z_j^{(l)} = Z_j$, we use the recursive update given by expression (44):

$$z_j^{(l)} = (A z_j^{(l)} - γ_j z_j^{(l)} - δ_{j-l} z_j^{(l)} - δ_{j-l-1} z_j^{(l)} / δ_{j-l-1} - δ_j z_j^{(l)} / δ_j), \quad (46)$$

for $j \geq l$, which paves down to the recurrence relation (6). The recurrence relations (45) allow us to compute the vector updates for the bases $Z_j^{(0)}$, ..., $Z_j^{(l)}$ without the need to compute any additional SPMV. Adding the recurrence
Note that all vector updates make use of the relation (45) for the auxiliary bases

$$z^{(l)}_{i+1} = \begin{cases} (A - \sigma_i I)z^{(l)}_i, & i < l \\ A^{(l)} z^{(l)}_i, & i \geq l \end{cases}$$

for the auxiliary bases $Z_j^{(0)}, \ldots, Z_j^{(l-1)}$ to the p(l)-CG method leads to the stable p(l)-CG method, Alg. 2.

Let us expand on Alg. 2 in some more detail. In the $i$-th iteration of Alg. 2 each basis $Z_j^{(0)}, Z_j^{(1)}, \ldots, Z_j^{(l-1)}$ is updated by adding one vector. Thus the algorithm computes a total of $l + 1$ new basis vectors, i.e. $\{v_{i+1} = \mathcal{A}_j^{(0)} z_{i+1}, \ldots, v_{i+1} = Z_j^{(l-1)} z_{i+1}\}$, per iteration. For each basis, the corresponding vector update in iteration $i \geq l$ is computed as follows:

$$
\begin{align*}
\mathcal{A}^{(0)} z_{i+1} &= (z_{i+1}^{(1)} + (\sigma_0 - \gamma_i i)z_{i+1}^{(1)})/\delta_{i-1}, \\
\mathcal{A}^{(2)} z_{i+1} &= (z_{i+1}^{(2)} + (\sigma_1 - \gamma_i i)z_{i+1}^{(1)})/\delta_{i-1}, \\
&\vdots \\
\mathcal{A}^{(l-1)} z_{i+1} &= (z_{i+1}^{(l-1)} + (\sigma_{l-1} - \gamma_i i)z_{i+1}^{(l-1)})/\delta_{i-1},
\end{align*}
$$

Note that all vector updates make use of the same scalar coefficients $\gamma_i, \delta_i$ and $\delta_{i-1}$ that are computed in iteration $i$ of Alg. 2 [lines 11-17]. We also remark that only one SPMV, namely $A^{(k)}$, is required per iteration to compute all basis vector updates, similar to Alg. 1.

The merit of the intermediate basis vector updates is that they allow to replace the recurrence (9) for the original basis vector $v_{j+1}$ by relation (44), which for $k = 0$ yields:

$$v_{j+1} = z_{j+1}^{(0)} = (z_{j+1}^{(1)} + (\sigma_0 - \gamma_j i)z_{j+1}^{(1)})/\delta_j,$$

with $j \geq 0$. Since relation (43) with $k = 0$ states that $v_{j+1}^{(0)} + \sigma_0 v_{j+1} = Av_{j+1}$, expression (47) very closely resembles the finite precision Lanczos recurrence relation (57). In particular, we point out that the matrix $G_{j+1}$ does not occur in the recursive update (47) for $v_{j+1}$. We clarify the difference between the finite precision and exact variant of recurrence relation (47) in Section 4.4.

Example 1. To illustrate the methodology for constructing the basis in the stable p(l)-CG method, consider the case where the pipeline length $l = 3$. We formulate the improved p(3)-CG method following the derivation above. This scenario features the default Krylov subspace basis $Z_j^{(0)} = V_j$ and three auxiliary bases: $Z_j^{(1)}$, $Z_j^{(2)}$ and $Z_j^{(3)} = Z_j$. To compute a new vector for each auxiliary basis, the following recurrence relations are used by Alg. 2 in iteration $i \geq 3$:

$$
\begin{align*}
\mathcal{A}^{(2)} z_{i+1} &= (z_{i+1}^{(2)} + (\sigma_0 - \gamma_i i)z_{i+1}^{(2)})/\delta_{i-1}, \\
\mathcal{A}^{(3)} z_{i+1} &= (z_{i+1}^{(3)} + (\sigma_1 - \gamma_i i)z_{i+1}^{(3)})/\delta_{i-1}, \\
\mathcal{A}^{(4)} z_{i+1} &= (z_{i+1}^{(4)} + (\sigma_2 - \gamma_i i)z_{i+1}^{(4)})/\delta_{i-1},
\end{align*}
$$

The final recurrence relation to update $z_{i+1}^{(3)} = z_{i+1}$ is identical to expression (6) with $l = 3$. The former recurrence relation for $z_{i+1}^{(2)} = v_{i-2}$, expression (9), is replaced by the (stable) relation (47). This update explicitly uses the auxiliary variable $z_{i+1}^{(1)}$ and
(l + 1) local dot products (see Remark 2) to form the $G_j$ matrix and use two additional \texttt{AXPY} operations to update the search direction $p_{l-1}$ and the iterate $x_{l-1}$. In summary, as indicated by the \texttt{Flops} column in Table 1 both p(l)-CG algorithms use a total of $(6l + 10)N$ flops in each iteration. The recurrence relations in the p(l)-CG algorithm can thus be stabilized at no additional computational cost using the framework outlined in Section 4.4.

### 4.2 Computational costs and storage requirements

We give an overview of implementation details of the stable p(l)-CG method, Alg. 2, including global storage requirements and number of \texttt{flops} (floating point operations) per iteration. We compare to the same properties for the former version of p(l)-CG Alg. 1 and Ghysels’ p-CG method [18]. The latter algorithm, although mathematically equivalent to p(1)-CG, was derived in an essentially different way.

#### 4.2.1 Floating point operations per iteration

All Conjugate Gradient variants listed in Table 1 compute a single SPMV in each iteration. However, as indicated by the \texttt{Time} column, time per iteration may be reduced significantly by overlapping the global reduction phase with the computation of one or multiple SPMVs. Time required by the local \texttt{AXPY} and \texttt{DOT-PR} computations is neglected, since these operations are assumed to scale perfectly as a function of the number of workers.

Comparing Alg. 1 to Alg. 2 it is clear that the latter requires an additional $2(l - 1)$ \texttt{AXPY} operations per iteration to update the auxiliary vectors $z_{l-1}, z_{l-2}, \ldots, z_{l-(l-1)}$ using the recurrence relations (45). However, the recurrence relation to update $v_{l-1}, v_{l-2}, \ldots, v_{l-(l-1)}$ only requires 2 \texttt{AXPY} operations, instead of the 2l \texttt{AXPY} operations required to update $v_{l-1}$ in Alg. 1 using expression \texttt{9}. Both algorithms furthermore compute implicitly depends on the other $l - 1 = 2$ auxiliary variables $z_{l-2}^{(2)}$ and $z_{l-2}^{(3)}$ through the respective recurrence relations. All four recurrence relations above make use of the same scalar coefficients $\delta_{l-3}, \gamma_{l-3}, \gamma_{l-4}$ that form the last column of the matrix $T_{l-1,j-2}$. Similar to Alg. 1 these coefficients are computed on line 11–17 in Alg. 2 right before the recursive vector updates.

Example 2. In the case where the pipeline length is one, i.e. $l = 1$, the stable p(l)-CG Alg. 2 formally differs slightly from the original formulation of p(1)-CG, Alg. 1. The improved algorithm uses the following recurrence relations for $v_i$ and $z_i$ in iteration $i \geq 1$:

$$z_i^{(0)} = (z_i^{(1)} + (\sigma_0 - \gamma_{l-1})v_i - \delta_{l-2}v_{i-1})/\delta_{l-1},$$

$$z_{l+1}^{(0)} = (A_{l-2}v_i - \gamma_{l-1}z_{l-2} - \delta_{l-2}z_{l-1})/\delta_{l-1},$$

where $z_i^{(0)} = v_i$ and $z_{l+1}^{(0)} = z_{l+1}$. This implies that the only difference between the improved and original p(1)-CG method is the recurrence relation for $v_i$. The recurrence relation for $v_i$ above is equivalent to the recurrence relation in exact arithmetic, but it is numerically (more) stable as explained in Section 4.4.

### 4.3 A stable preconditioned p(l)-CG algorithm

Preconditioning is indispensable to efficiently solve linear systems in practice. We briefly comment on the straightforward extension of Alg. 2 to include a preconditioner. This section follows the standard methodology that was described for pipelined CG in [18] and for Alg. 1 in [10].

Let the preconditioner be given by the matrix $M^{-1}$. We aim to solve the left-preconditioned linear system $M^{-1}Ax = M^{-1}b$, where $M$ and $A$ are both symmetric positive definite matrices. This assumption does not necessarily imply that $M^{-1}A$ is symmetric. Nonetheless, symmetry can be preserved by observing that $M^{-1}A$ is self-adjoint with respect to the $M$ inner product $(x, y)_M = (Mx, y)$. The basic strategy is thus to replace all Euclidean dot products occurring in Alg. 2 with $M$ inner products.

We cannot simply use the matrix $M$ to calculate these $M$ inner products, since the preconditioner inverse is in general not available. However, by introducing the \textit{unpreconditioned auxiliary variables} $u_i = Mz_i^{(l)}$ and observing that these variables again satisfy a Lanczos type relation:

$$u_{i+1} = \begin{cases} A_{i+1}z_i^{(l)} - \sigma_i u_i, & i < l, \\
(A_{i+1}z_i^{(l)} - \gamma_{i-1}u_i - \delta_{i-2}u_{i-1})/\delta_{i-1}, & i \geq l,
\end{cases}$$

(48)

this obstacle is circumvented. Using these unpreconditioned auxiliary variables $u_{i+1}$, the dot products $g_{j,i+1}$ for $0 \leq j \leq l + 1$ can be computed as follows. For $i < l$ it holds that

$$g_{j,i+1} = (z_{i+1}^{(l)}, z_j^{(l)})_M = (Mz_{i+1}^{(l)}, z_j^{(l)}) = (u_{i+1}, z_j^{(l)}),$$

(49)
with $j = 0, \ldots, i + 1$. For $i \geq l$ we find

$$g_{j,i+1} = \begin{cases} 
    \left( s_{i+1}^{(l)} z_{j}^{(0)} \right)_M = (u_{i+1}, z_{j}^{(0)}), \\
    \text{for } j = \max(0, i-2l+1), \ldots, i-l+1, \\
    \left( s_{i+1}^{(l)} z_{j}^{(0)} \right)_M = (u_{i+1}, z_{j}^{(l)}), \\
    \text{for } j = i-l+2, \ldots, i+1. 
\end{cases}$$

This allows to formulate a preconditioned version of Alg. 2 by adding the recurrence relation (48) for the unpreconditioned auxiliary variables $u_{i_0}$ and replacing the dot products on line 23 and 25 by expressions (49) and (50) respectively. From an implementation point of view the extension to the preconditioned algorithm only requires the application of the preconditioner $M^{-1}$, two additional AXPY operations and storage of three additional vectors in memory.

### 4.4 Rounding error analysis for the improved method

We now consider the finite precision equivalents of the above recurrence relations for all basis vectors in the improved version of p(l)-CG, Alg. 2. The actually computed finite precision variants of the basis vectors and scalar variables are again denoted by bars.

Let the true basis vector $\tilde{v}_j$ satisfy the actual Lanczos relation without local roundoff (for $0 \leq j < i - l$) as

$$\tilde{v}_{j+1} = (A\tilde{v}_j - \gamma_j \tilde{v}_j - \tilde{\delta}_{j-l}\tilde{v}_{j-l})/\tilde{\delta}_j,$$

and let, analogously, the true auxiliary basis vectors be defined (for $j \geq k$ and $0 \leq k \leq l$) as

$$\tilde{z}_{j+1}^{(k)} = (A\tilde{z}_j^{(k)} - \gamma_j \tilde{z}_j^{(k)} - \tilde{\delta}_{j-k}\tilde{z}_{j-k})/\tilde{\delta}_j - \tilde{\delta}_{j-k} - \tilde{\delta}_{j-k-1}\tilde{z}_{j-k-1}^{(k)}/\tilde{\delta}_j - \tilde{\delta}_{j-k}.$$

On the other hand, the finite precision variants of the recurrence relations (44), which are actually computed in Alg. 2 are for $j \geq k$ and $0 \leq k < l$ given by the expressions

$$\tilde{z}_{j+1}^{(k)} = (\tilde{z}_{j+1}^{(k)} + (\sigma_k - \tilde{\gamma}_k - \tilde{\delta}_{j-k})\tilde{z}_{j-k}^{(k)} - \tilde{\delta}_{j-k-1}\tilde{z}_{j-k-1}^{(k)})/\tilde{\delta}_j + \tilde{\xi}^{(k)}_{j+1},$$

whereas for $k = l$ the following finite precision relation holds (with $j \geq l$):

$$\tilde{z}_{j+1}^{(l)} = (A\tilde{z}_j^{(l)} - \gamma_j \tilde{z}_j^{(l)} - \tilde{\delta}_{j-l}\tilde{z}_{j-l})/\tilde{\delta}_j - \tilde{\delta}_{j-l} - \tilde{\delta}_{j-l-1}\tilde{z}_{j-l-1}^{(l)}/\tilde{\delta}_j - \tilde{\delta}_{j-l} + \tilde{\xi}^{(l)}_{j+1},$$

The local rounding errors $\tilde{\xi}^{(k)}_{j+1}$ ($0 \leq k \leq l$) in the above expressions can be bounded by machine precision $\epsilon$ multiplied by the norms of the respective vectors, i.e. $\|\tilde{\xi}^{(k)}_{j+1}\| \leq O(\epsilon)$.

Switching to matrix forms to simplify the notation, we write the Lanczos relations (52) as

$$Z_{j+1}^{(k)} - Z_{j+1}^{(k)} = (AZ_j^{(k)} - Z_j^{(k)} T_{j+1,j}) A_{j+1,j}^{+},$$

for $j > k$ and $0 \leq k \leq l$. Note that this expression neglects the “initial” gaps, i.e. the quantities $Z_j^{(k)} - Z_j^{(k)}$ for $0 \leq k \leq l$, consisting of the local rounding errors from the application of the matrix polynomials $P_j(A)$ to the initial vector $v_0$, which are computed explicitly in the stable p(l)-CG method, Alg. 2 (line 3). The recurrence relations (48) are summarized by the matrix expressions

$$Z_{2,j+1}^{(k+1)} = Z_{j+1}^{(k)} T_{j+1,j} - \sigma_k Z_{j+1}^{(k)} - Z_{j+1}^{(k)} A_{j+1,j},$$

with $j > k$ and $0 \leq k < l$, where $Z_j^{(k)} = [\xi_1^{(k)}, \ldots, \xi_l^{(k)}]$ and $Z_{2,j+1}^{(k+1)} = [\xi_1^{(k+1)}, \xi_2^{(k+1)}, \ldots, \xi_{k+1}^{(k+1)}]$. The recurrence relation (54) for $Z_j^{(k)}$ can be written as

$$A\tilde{Z}_{j+1}^{(l)} = \tilde{Z}_{j+1}^{(l)} T_{j+1,j} - \tilde{Z}_{j+1}^{(l)} A_{j+1,j},$$

with $j > l$. To find an expression for the gap on the basis vectors in $V_j$, i.e. the gap $Z_j^{(0)} - \tilde{Z}_j^{(0)}$, we now progressively compute the gaps on the auxiliary bases. Starting from $Z_j^{(l)} - \tilde{Z}_j^{(l)}$, we compute $Z_j^{(l-1)} - \tilde{Z}_j^{(l-1)}$, in which we substitute the gap on $Z_j^{(l)}$, followed by $Z_j^{(l-2)} - \tilde{Z}_j^{(l-2)}$, etc., until we eventually obtain an expression for the gap $Z_j^{(0)} - \tilde{Z}_j^{(0)}$. For $k = l$ we derive from (55) and (57) that

$$Z_{j+1}^{(k)} - Z_{j+1}^{(l)} = -Z_{j+1}^{(l)}, \quad j > l.$$

As indicated by this expression, the gaps on the basis vectors in the basis $Z_j^{(l)}$ thus consist of local rounding errors only. The following auxiliary lemma can easily be proven by induction.

**Lemma 1.** Let $Z_j^{(k)}$ be defined by (56) and $Z_{j+1}^{(k+1)}$ be defined by (55). Then it holds for $j > k$ and $0 \leq k < l$ that

$$A\tilde{Z}_j^{(k)} - \sigma_k \tilde{Z}_j^{(l)} = Z_{2,j+1}^{(k+1)} + \Theta_{2,j+1}^{(k+1)},$$

where $\Theta_{2,j+1}^{(k+1)} = [\theta_1^{(k+1)}, \theta_2^{(k+1)}, \ldots, \theta_l^{(k+1)}]$ with $\theta_l^{(k+1)}$ a local rounding error that is bounded by $O(\epsilon)$ and $Z_{2,j+1}^{(k+1)} = [\xi_1^{(k+1)}, \xi_2^{(k+1)}, \ldots, \xi_l^{(k+1)}]$.

Next, we combine expressions (55) and (56) and Lemma 1 for the case $k = l - 1$. We obtain (with $j > l - 1$)

$$Z_{j+1}^{(l-1)} - Z_{j+1}^{(l-1)} = (A\tilde{Z}_j^{(l-1)} - \tilde{Z}_j^{(l-1)} - \sigma_{l-1} Z_j^{(l-1)}) A_{j+1,j}^{+} = (Z_{2,j+1}^{(l-1)} + \Theta_{2,j+1}^{(l-1)}) A_{j+1,j}^{+} Z_{j+1}^{(l-1)}.$$

Hence, the gaps on the basis vectors $Z_{j}^{(l-1)}$ are coupled to the gaps on the basis vectors $Z_{j}^{(l)}$. After substitution of expression (58) it is clear that $Z_{j}^{(l-1)} - Z_{j}^{(l-1)}$ consists only of local rounding errors. The above relation can be generalized for any $k \in \{0, 1, \ldots, l - 1\}$ as follows:

$$Z_{j+1}^{(k)} - Z_{j+1}^{(k)} = (A\tilde{Z}_j^{(k)} - Z_j^{(k)} T_{j+1,j}) A_{j+1,j}^{+} = (Z_{2,j+1}^{(k)} + \Theta_{2,j+1}^{(k)}) A_{j+1,j}^{+} Z_{j+1}^{(k)}.$$

where $j > k$. After subsequent substitution of this expression starting from $Z_{0,j+1}^{(0)} - Z_{0,j+1}^{(0)}$ up until the characterization of $Z_{j+1}^{(l)} - Z_{j+1}^{(l)}$, see (58), it appears that the gap $Z_{0,j+1}^{(0)} - Z_{0,j+1}^{(0)} = \tilde{V}_{j+1} + \tilde{V}_{j+1} - \tilde{V}_{j+1}$ on the original Krylov subspace basis is just a sum of local rounding errors. No rounding error propagation takes place in the stable p(l)-CG method, Alg. 2, on any of the auxiliary basis vectors, see expressions (58) - (60). By introducing the intermediate auxiliary bases $Z_{0,j+1}^{(l)}$, $Z_{0,j+1}^{(l)}$, for the recursive computation of the basis $V_j$, the dependency of the basis vector gaps on the possibly ill-conditioned matrix $G_j$, cf. expressions (32) and (36), is thus removed, resulting in a numerically stable algorithm.
5 NUMERICAL RESULTS

We present various numerical experiments to benchmark the stable p(l)-CG method proposed in Section 4. The benchmark problems exemplify both the performance of the improved p(l)-CG method on large scale parallel hardware as well as its error resilience compared to other CG variants. Performance measurements result from a PETSc implementation of the p(l)-CG algorithm on a distributed memory machine using the message passing paradigm (MPI).

5.1 Hardware and software specifications

Parallel performance experiments are performed on up to 128 compute nodes of a cluster consisting of two 14-core Intel E5-2680v4 Broadwell generation CPUs each (28 cores per node). Nodes are connected through an EDR InfiniBand network. We use PETSc version 3.8.3 [1]. The MPI library used for this experiment is Intel MPI 2018v3. The PETSc environment variables `MPICH_ASYNC_PROGRESS=1` and `MPICH_MAX_THREAD_SAFETY=multiple` ensure optimal parallelism by allowing for asynchronous non-blocking global communication. Timing results reported in this manuscript are the most favorable results (smallest overall run-time) over 3 individual runs of each method. Experiments also show results for Ghysels’ p-CG method [18] as a reference. The p-CG method is similar to p(l)-CG in operational cost (see Table 1), but features a significant loss of attainable accuracy due to rounding error propagation in its recurrence relations [9], similar to p(l)-CG, Alg. 1.

5.2 Benchmark (B1): 2D Elliptic Laplacian PDE

Fig 2 is the analogue of the experiment reported in Fig. 4 (2D Laplacian, 100 × 100 unknowns), where the p(l)-CG Alg. 1 is replaced by the improved Alg. 2. In contrast to Fig. 1a, no loss of accuracy due to local rounding error amplification in the recurrence relations is observed in Fig. 2a. The stabilized recurrences (45)-(46) ensure that the quantity \( \| (AV_i - V_{i+1}T_{i+1})U_i^{-1}q_i \| \) is of order machine precision. The true residual norms (full lines) and recursively computed residual norms (dotted lines) coincide up to \( \| b - A\hat{x}_i \| / \| b \| = 1.0 \times 10^{-12} \) (maximal attainable accuracy) for all methods in Fig. 2a. Fig. 2b quantifies the inexact orthogonality for the different methods which is comparable to Fig. 1b. Hence, similarly to Fig. 1a a delay of convergence may be observed in Fig. 2a.

Fig. 3 shows a performance experiment on the hardware and software setup specified above. A 2D Laplace type linear system with exact solution \( \hat{x}_j = 1 \), right-hand side \( b = A\hat{x} \) and initial guess \( x_0 = 0 \) is solved. This problem is available as example 2 in the PETSc Krylov subspace solvers (KSP) folder. The simulation domain is discretized using a 1,750 × 1,750 uniform finite difference mesh (3, 062, 500 unknowns). No preconditioner is applied. For p(l)-CG Chebyshev shifts \( \{ \sigma_0, \ldots, \sigma_{l-1} \} \) are used based on the interval \( \lambda_{\min}, \lambda_{\max} = [0, 8] \), see (24).

Fig 3a shows the speedup achieved by different CG methods over single-node classic CG for various pipeline lengths and node setups. Classic CG scales poorly for this model problem; no speedup is achieved beyond 8 nodes. The pipelined methods scale well. The length one p-CG and p(1)-CG method achieve a relative speedup of approximately 5× compared to classic CG when both are executed on 16 nodes. The longer pipelined p(2)-CG and p(3)-CG methods out-scale the latter method, with p(2)-CG achieving a 7× speedup relative to classic CG on 32 nodes. When \( l = 1 \) the global communication phase in each iteration is only partially ‘hidden’ behind the SPMV computation, whereas overlapping with more than two SPMVs by using pipelines length \( l \geq 3 \) does not seem to improve performance further. Pipeline length \( l = 2 \) is optimal for this problem, striking a good balance between overlapping communication and introducing additional auxiliary vectors.

Fig 2b plots the relative residual norms as a function of the total time spent (in s.) by various CG algorithms (on 500 iteration intervals). The p-CG method stagnates around \( \| b - A\hat{x}_i \| / \| b \| = 1.0 \times 10^{-8} \) and is unable to attain

![Fig. 2: (B1)](image-url)
a better accuracy regardless of computational effort. The stable p(l)-CG methods all are able to attain a much higher accuracy, stagnating around \(\|b - Ax_i\|_2 / \|b\|_2 = 2.7e-13\) for \(l \in \{1, 2, 3, 4, 5\}\). Maximal attainable accuracy is reached in 2.01 s. for p(1)-CG, 1.28 s. for p(2)-CG, 1.34 s. for p(3)-CG, 1.74 s. for p(4)-CG, 2.73 s. for p(5)-CG. Note that classic CG attains \(\|b - Ax_i\|_2 / \|b\|_2 = 2.5e-13\) in 13.9 s. (outside the graph). These results are consistent with Fig. 3a, indicating that a pipeline length of \(l = 2\) suffices to hide the global communication phase for this problem.

### 5.3 Benchmark (B2): 3D Hydrostatic Ice Sheet Flow

Fig. 4a shows the result of two strong scaling experiment for the 3D Hydrostatic Ice Sheet Flow problem, available as example 48 is the PETSc Scalable Nonlinear Equations Solvers (SNES) folder. The Blatter/Pattyn equations are discretized using \(100 \times 100 \times 20\) (Fig. 4a) and \(150 \times 150 \times 100\) (Fig. 4b) finite elements respectively. Hard- and software specifications are presented in Section 5.1. A Newton-Krylov outer-inner iteration is used to solve the non-linear problem. The CG methods used as inner solver are combined with a block Jacobi preconditioner (one block Jacobi step per CG iteration; one block per processor). The relative tolerance of the inner solvers is set to \(\|b - Ax_i\|_2 / \|b\|_2 = 1.0e-10\) (Fig. 4a) and \(\|b - Ax_i\|_2 / \|b\|_2 = 1.0e-5\) (Fig. 4b), while the outer relative tolerance is chosen to be \(1.0e-8\). Seven Newton iterations are needed to reach the outer tolerance.

In Fig. 4b the scalability of p(l)-CG for \(l \in \{1, 2, 3\}\) is comparable to that of the p-CG method. On 128 nodes a speedup of approximately 10x is measured for the pipelined methods compared to classic CG on 128 nodes. There is no gain in using longer pipeline lengths for this problem and hardware setup, indicating that the amount of computational work of a single iteration (SPMV + PREC) suffices to hide the communication in the global reduction in each iteration. It is expected that longer pipeline lengths out-scale the methods with shorter pipelines on very large numbers of nodes, since communication costs would increase accordingly. This is illustrated to some extend by the smaller problem reported in Fig. 4a which shows that for heavily communication bound problems the use of longer pipelines is beneficial for this benchmark. Fig. 4a indicates that depending on the amount of hardware parallelism (number of nodes) p(1)-CG (4-24 nodes), p(2)-CG (32-40 nodes) or p(3)-CG (48+ nodes) respectively display the biggest speedup.

An accuracy experiment for the \(150 \times 150 \times 100\) FE benchmark problem is shown in Fig. 4b. The experiment is run on 8 nodes and the relative tolerance of the inner solver is \(\|b - Ax_i\|_2 / \|b\|_2 = 1.0e-10\). A relatively small delay of convergence is observed for the p(l)-CG method, with the effect worsening for longer pipelines. The convergence delay is due to restarts (caused by a square root breakdown) which come forth from the ill-conditioned auxiliary basis \(\bar{Z_j}\). The effect is negligible since the maximal total delay after seven Newton iterations is 174 iterations (comparing p(3)-CG to CG), relative to a total of 4,500 Krylov iterations. The p-CG method fails to reach the inner tolerance \(1.0e-10\). As shown by the analysis in Section 4.4 no rounding error propagation occurs in the stable p(l)-CG algorithm and the method is able to attain a highly accurate solution.

### 5.4 Benchmark (B3): 2D Bratu Solid Fuel Ignition

The 2D Bratu Solid Fuel Ignition problem results from the nonlinear PDE that can be found in the PETSc SNES folder as example 5. The two-dimensional domain is discretized with \(1,000 \times 1,000\) uniformly spaced grid points. The Bratu parameter \(\text{par}\) is set to 6.0, implying significant non-linearity. Different CG methods are used as the inner solver in the Newton-Krylov scheme. The preconditioner is a SOR-preconditioned Chebyshev iteration (three SOR
steps per Chebyshev iteration; one Chebyshev step per CG iteration). Outer solver relative tolerance is set to 1.0e-8, which is attained by all methods after three Newton steps. For p(l)-CG Chebyshev shifts based on the interval $[\lambda_{\text{min}}, \lambda_{\text{max}}] = [0, 1.2]$ are used. A relative tolerance of $\|b - A\bar{x}_i\|_2/\|b\|_2 = 1.0e-10$ is imposed on the inner solvers.

Fig. 5a presents the relative speedup of the p(l)-CG methods for $l \in \{1, 2, 3\}$ compared to classic CG on one node. Timing data for the p-CG method is not included in the figure since it is unable to achieve the desired accuracy; the maximal accuracy attainable by p-CG is $\|b - A\bar{x}_i\|_2/\|b\|_2 = 1.3e-9/4.2e-8/6.4e-8$ for outer SNES iteration 1/2/3 respectively. On 32 nodes the speedup achieved by the pipelined methods over classic CG is approximately 4x. For this small sized problem the p(l)-CG method does not scale beyond 32 nodes since CPU times are dominated by the $O(\log_2(\#\text{nodes}))$ behavior of the global reduction phases in this regime. Longer pipelines do not yield a higher speedup compared to pipeline length 1 due to the cost of the preconditioner, which requires two SPMVs each iteration.

Fig. 5b shows the relative residual norm as a function of the total time spent for solving the three Newton iterations on a 24 node setup. The timings and residual norms are taken from the same experiment as Fig. 5a. The total time-to-solution is divided by the respective number of iterations for the different CG methods to accurately compare them for accurate comparison on a single graph. The p(l)-CG methods outperform the classic CG method despite a small increase in total number of iterations for longer pipeline lengths. The stable p(l)-CG algorithm reaches the relative tolerance 1.0e-10 for any choice of the parameter $l$.

6 CONCLUSIONS

This work presents a redesigned algorithmic variant of the $l$-length pipelined Conjugate Gradient method, p(l)-CG for short. The main improvement over former pipelined CG variants is the significantly improved maximal attainable accuracy that is attained by the new algorithm. More specifically, it is shown analytically and verified experimentally that the stable p(l)-CG method attains the same precision on the solution that is attainable by the classic CG method. By introducing intermediate auxiliary bases the propagation of local rounding errors in the recurrence relations is eliminated, allowing for high-precision solution independently of the choice of the pipeline length $l$. The new p(l)-CG
algorithm is elegant in the sense that it has no additional computational overhead and only minor additional storage requirements compared to previous versions of the p(l)-CG algorithm. The increased stability thus comes without the cost of increased complexity. The improved algorithm effectively replaces former (less stable) pipelined CG variants. As such, this work resolves one of the major numerical issues that has restricted the practical usability of pipelined Krylov subspace methods since their initial development.

Generalizing the stabilization technique proposed in this work to other pipelined methods is non-trivial. A similar methodology could be applied to the existing pipelined GMRES method, p(l)-GMRES [17]. However, practical restrictions limit the viability of this approach. Notably, the full storage of an additional \((l - 1)\) auxiliary bases would be required, which can be assumed to be unfeasible for large-scale applications. An \(l\)-length variant of the pipelined BiCGStab method [8] is currently not available, but the proposed technique could be promising for direct application in the context of Bi-Conjugate Gradient methods.

The research presented in this manuscript provides vital advancements towards establishing a numerically stable communication hiding variant of the Conjugate Gradient method. However, it is well-known that Krylov subspace methods may also suffer from delay of convergence due to loss of basis orthogonality. Our experiments indicate that this effect is typically enlarged as a function of pipeline length. Analyzing the stability issues related to loss of orthogonality deserves to be treated as part of future work.

**Acknowledgments**

The authors are grateful for the funding received that supported this work. In particular, J.C. acknowledges funding by the University of Antwerp Research Council under the University Research Fund (BOF) and S.C. is funded by the Flemish Research Foundation (FWO Flanders) under grant 12H4617N. We also thank Pieter Ghysels for providing useful comments on an earlier version of this manuscript.

**References**

[1] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L. Curfman McInnes, K. Rupp, B.F. Smith, S. Zampini, and H. Zhang. PETSc Web page. www.mcs.anl.gov/petsc. 2015.

[2] E. Carson and J. Demmel. A residual replacement strategy for improving the maximum attainable accuracy of s-step Krylov subspace methods. *SIAM Journal on Matrix Analysis and Applications*, 35(1):22–43, 2014.

[3] E. Carson, N. Knight, and J. Demmel. Avoiding communication in nonsymmetric Lanczos-based Krylov subspace methods. *SIAM Journal on Scientific Computing*, 35(5):S42–S61, 2013.

[4] E. Carson, M. Rozloznik, Z. Strakoš, P. Tichý, and M. Tůma. The numerical stability analysis of pipelined Conjugate Gradient methods: Historical context and methodology. *SIAM Journal on Scientific Computing*, 40(5):A3549–A3580, 2018.

[5] A.T. Chronopoulos and C.W. Gear. s-Step iterative methods for symmetric linear systems. *Journal of Computational and Applied Mathematics*, 25(2):153–168, 1989.

[6] A.T. Chronopoulos and A.B. Kucherov. Block s-step Krylov iterative methods. *Numerical Linear Algebra with Applications*, 17(1):3–15, 2010.

[7] S. Cools. Analyzing and improving maximal attainable accuracy in the communication hiding pipelined BiCGStab method. *Parallel Computing*, PMAA’18 Special Issue, submitted, 2018.

[8] S. Cools and W. Vanroose. The communication-hiding pipelined BiCGStab method for the parallel solution of large unsymmetric linear systems. *Parallel Computing*, 65:1–20, 2017.

[9] S. Cools, E.F. Yetkin, E. Agullo, L. Giraud, and W. Vanroose. Analyzing the effect of local rounding error propagation on the maximal attainable accuracy of the pipelined Conjugate Gradient method. *SIAM Journal on Matrix Analysis and Applications*, 39(1):426–450, 2018.

[10] J. Cornelis, S. Cools, and W. Vanroose. The communication-hiding Conjugate Gradient method with deep pipelines. *Submitted,* SIAM Journal on Scientific Computing, preprint: arXiv:1801.04728, 2017.

[11] E.F. D’Azevedo, V. Eijkhout, and C.H. Romine. Reducing communication costs in the Conjugate Gradient algorithm on distributed memory multiprocessors. Technical report, Technical report, Oak Ridge National Lab, TN/12929, TN, US, 1992.

[12] J.W. Demmel, M.T. Heath, and H.A. Van der Vorst. Parallel Numerical Linear Algebra. *Acta Numerica*, 2:111–197, 1993.

[13] J. Dongarra, P. Beckman, T. Moore, P. Aerts, G. Allosio, J. Andre, D. Barkai, J. Berthou, T. Boku, B. Braunschweig, et al. The international exascale software project roadmap. *International Journal of High Performance Computing Applications*, 25(1):3–60, 2011.
[14] J. Dongarra, M.A. Heroux, and P. Luszczek. HPCG benchmark: a new metric for ranking high performance computing systems. University of Tennessee, Electrical Engineering and Computer Science Department, Technical Report UT-EECS-15-736, 2015.

[15] P. Ghysels and W. Gropp. Scalable non-blocking preconditioned Conjugate Gradient methods. In SC16: International Conference for High Performance Computing, Networking, Storage and Analysis, pages 204–215. IEEE, 2016.

[16] J. Erhel. A parallel GMRES version for general sparse matrices. Electronic Transactions on Numerical Analysis, 3(12):160–176, 1995.

[17] P. Ghysels, T.J. Ashby, K. Meerbergen, and W. Vanroose. Hiding global communication latency in the GMRES algorithm on massively parallel machines. SIAM Journal on Scientific Computing, 35(1):C48–C71, 2013.

[18] A. Greenbaum. Behavior of slightly perturbed Lanczos and Conjugate-Gradient recurrences. Linear Algebra and its Applications, 113:7–63, 1989.

[19] A. Greenbaum and Z. Strakoš. Predicting the behavior of finite precision Lanczos and Conjugate Gradient computations. SIAM Journal on Matrix Analysis and Applications, 13(1):121–137, 1992.

[20] L. Grigori, S. Moufawad, and F. Nataf. Enlarged Krylov subspace Conjugate Gradient methods for reducing communication. SIAM Journal on Matrix Analysis and Applications, 37(2):744–773, 2016.

[21] M.H. Gutknecht and Z. Strakoš. Accuracy of two three-term and three two-term recurrences for Krylov space solvers. SIAM Journal on Matrix Analysis and Applications, 22(1):213–229, 2000.

[22] L.C. McInnes, B. Smith, H. Zhang, and R.T. Mills. Hierarchical algorithms in finite precision arithmetic. Computers, 35(1):C48–C71, 2013.

[23] K. Swirydowicz, J. Langou, S. Ananthan, U. Yang, and S. Thomas. Low synchronization GMRES algorithms. submitted to IEEE 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, preprint arXiv:1809.05805, 2018.

[24] H.A. Van der Vorst. Iterative Krylov methods for large linear systems, volume 13. Cambridge University Press, 2003.

[25] J.H. Wilkinson. Rounding errors in algebraic processes. Courier Corporation, 1994.

[26] I. Yamazaki, M. Hoemmen, P. Luszczek, and J. Dongarra. Improving performance of GMRES by reducing communication and pipelining global collectives. In Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017 IEEE International, pages 1118–1127. IEEE, 2017.

[27] S. Zhuang and M. Casas. Iteration-Fusing Conjugate Gradient. In Proceedings of the International Conference on Supercomputing, pages 21–30. ACM, 2017.

Siegfried Cools is a post-doctoral Research Foundation Flanders (FWO) fellow hosted by the Department of Mathematics and Computer Science, University of Antwerp, Belgium, from which he obtained his PhD of Science. Applied Mathematics in May 2015. In 2013 he was awarded a PhD Student Paper Award at the 2013 16th Copper Mountain Conference on Multigrid Methods for the SIAM SISC publication “An Efficient Multigrid Calculation of the Far Field map for Helmholtz equations”. In 2015-2016 he was involved in the H2020 FP7-ICT European HPC EXA2CT HPC project (WP1: Solvers). He has contributed various open source parallel algorithms to the PETSc library. Siegfried Cools has co-organized several mini-symposia on the topic of “Communication Reducing Algorithms for Exascale Computing” at leading international conferences such as PMAA’18 and SIAM PP’18, and presented his research at high performance computing meetings (PARCO’17, SIAM ALA’18, SIAM CSE’19).

Jeffrey Cornelis is a junior scientist and PhD student at the Department of Mathematics and Computer Science, University of Antwerp, Belgium. He obtained his Master’s Degree in Science: Applied Mathematics in July 2017. His Master’s thesis“Construction, numerical properties and parallel performance of communication reducing pipelined Krylov subspace methods” was critically acclaimed by the international jury and has led to the publication “The Communication Hiding Conjugate Gradient Method with Deep Pipelines” which will be published in SIAM SISC in 2019 and is expected to have a high future impact. Jeffrey Cornelis has recently presented his research on “Communication Reducing Iterative Methods for Large Scale Linear Systems” to the international Numerical Mathematics, Engineering and Computer Science communities at leading international meetings such as PMAA’18 and SIAM CSE’19.

Wim Vanroose is a senior scientist, Applied Mathematics group lead and chair of the Mathematics and Computer Science Department at UAntwerp, where he has held a permanent position since 2006. He obtained his PhD in Computational Physics at UAntwerp in 2001. Between 2001 and 2004 he was a post-doctoral scientist at Lawrence Berkeley National Lab, followed by a post-doc at KULeuven (2004-2006). He is author of over 70 peer reviewed publications on the analysis, development and parallel implementation of mathematical methods for large scale complex systems. The 2005 Science publication by Vanroose et al. has been cited 100+ times. From 2013 to 2016 Wim Vanroose was group lead in the H2020 FP7-ICT European EXA2CT HPC project, a joint collaboration between academic and industrial partners among which USI, IMEC, INRIA, Intel, Fraunhofer, T-Systems and IT4I. He has been invited as plenary speaker at several leading international conferences, most recently at SIAM PP’18 in Tokyo. He co-founded two industrial companies outside his academic appointment: Motulus CVBA (2015) and Polygonal BVBA (2016).