Analysis of rounding error accumulation in Conjugate Gradients to improve the maximal attainable accuracy of pipelined CG

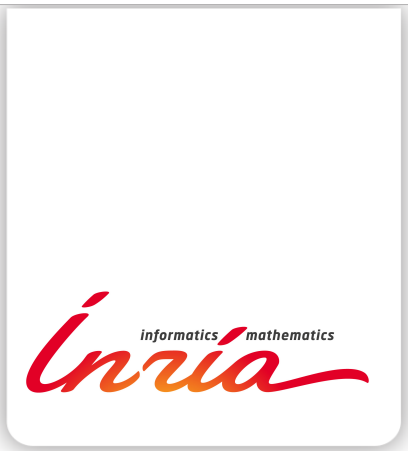
Siegfried Cools, Emrullah Fatih Yetkin, Emmanuel Agullo, Luc Giraud, Wim Vanroose

Analysis of rounding error accumulation in Conjugate Gradients to improve the maximal attainable accuracy of pipelined CG

Siegfried Cools, Emrullah Fatih Yetkin, Emmanuel Agullo, Luc Giraud, Wim Vanroose

RESEARCH REPORT
N° 8849
January 2016
Project-Teams HiePACS

ISSN 0249-6399 ISRN INRIA/RR--8849--FR+ENG



Analysis of rounding error accumulation in Conjugate Gradients to improve the maximal attainable accuracy of pipelined CG

Siegfried Cools*, Emrullah Fatih Yetkin†, Emmanuel Agullo†,
Luc Giraud†, Wim Vanroose*

Project-Teams HiePACS

Research Report n° 8849 — January 2016 — 31 pages

Abstract: Pipelined Krylov solvers typically offer better scalability in the strong scaling limit compared to standard Krylov methods. The synchronization bottleneck is mitigated by overlapping time-consuming global communications with useful computations in the algorithm. However, to achieve this communication hiding strategy, pipelined methods feature multiple recurrence relations on additional auxiliary variables to update the guess for the solution. This paper aims at studying the influence of rounding errors on the convergence of the pipelined Conjugate Gradient method. It is analyzed why rounding effects have a significantly larger impact on the maximal attainable accuracy of the pipelined CG algorithm compared to the traditional CG method. Furthermore, an algebraic model for the accumulation of rounding errors throughout the (pipelined) CG algorithm is derived. Based on this rounding error model, we then propose an automated residual replacement strategy to reduce the effect of rounding errors on the final iterative solution. The resulting pipelined CG method with automated residual replacement improves the maximal attainable accuracy of pipelined CG to a precision comparable to that of standard CG, while maintaining the efficient parallel performance of the pipelined method.

Key-words: Conjugate gradients, Parallelization, Latency hiding, Pipelining, Rounding errors, Maximal attainable accuracy, Global communication

* Applied Mathematics Group, University of Antwerp, Middelheimlaan 1, 2020 Antwerp, BE
† HiePACS, Inria Bordeaux - Sud-Ouest, 200 Avenue de la Vieille Tour, 33405 Talence, FR

RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200, Avenue de la vieille tour
33405 Talence Cedex

Analyse de l'accumulation d'erreurs d'arrondis dans la Gradient COnjugué pour améliorer la précision attaignable de sa version pipelinée

Résumé : Les méthodes de Krylov pipelinées offrent généralement de meilleures performance en terme de passage à l'échelle parallèle. Nous analysons dans ce travail l'effet de l'arithmétique finie sur la précision atteinte par la variante pipelinée du Gradient conjugué et proposons une alternative pour l'améliorer.

Mots-clés : Gradient conjugué, parallélisation, masquer la latence, erreurs d'arrondis, meilleure précision atteignable, communication globale

Contents

1 Introduction

Krylov subspace methods form the basis linear algebra solvers for many contemporary high-performance computing applications. The Conjugate Gradient (CG) method, dating back to the 1952 paper by Hestenes and Stiefel [16], can be considered as the first of these Krylov methods. Although more than 60 years old, the CG method is still the work horse method for the solution of linear systems with symmetric positive definite (SPD) matrices due to its numerical simplicity and easy implementation. These SPD systems originate from a variety of applications, such as the simulation of problems modeled by a partial differential equation (PDE).

Due to the transition of hardware towards the exascale regime in the coming years, research on the scalability of Krylov methods on massively parallel architectures has become increasingly prominent. Moreover, since the system matrix is often sparse, see e.g. the simulation of PDE type problems, the main bottleneck for efficient parallelization is typically not the sparse matrix-vector product (SPMV), but the communication overhead (bandwidth saturation) caused by global reductions required in the computation of dot-products. This is reflected in the new High Performance Conjugate Gradients (HPCG) benchmark for ranking HPC systems introduced by Dongarra et al. in 2013 [9, 10], which is based on sparse matrix-vector computations and data access patterns, rather than the dense matrix algebra used in the traditional High Performance LINPACK (HPL) benchmark.

Recently significant research has been devoted to the reduction and/or elimination of the synchronization bottleneck in Krylov methods. The idea of reducing the number of global communication points in Krylov methods on parallel computer architectures was first presented by Barrett et al. in [2], and was elaborated further by the work on s-step methods, see among others Chronopoulos and Gear [5] and more recently Carson et al. [4, 3]. In addition to communication avoiding methods, research on hiding global communication by overlapping communication with computations was performed by a variety of authors over the last decades, see De Sturler et al. [6], Demmel et al. [8] and Ghysels et al. [12].

The pipelined CG (p-CG) method proposed in [13] aims at hiding the global synchronization latency of standard preconditioned CG by reorganizing the algorithm and removing the multiple costly global synchronization points by only performing a single global reduction per iteration. Moreover, the global communication can be overlapped by the sparse matrix-vector product (SPMV), which requires only local communication. In this way, idle core time is minimized by performing useful computations simultaneously to the expensive global communication phase, cf. [11]. The current paper focuses on analyzing the rounding error propagation in different variants of the CG algorithm, and uses the performed analysis to improve the maximal attainable accuracy of the pipelined CG method.

To achieve the overlap of communication with computations in the CG algorithm, the pipelined CG method employs several additional AXPY ($y \leftarrow \alpha x + y$) operations to compute auxiliary variables, cf. Algorithm 3. Vector operations such as an AXPY are typically computed locally, and thus do not require communication between nodes. Dot-products of two vectors, on the other hand, involve global communication between all processes, and are grouped together in the pipelined CG method to reduce global communication to only one memory bottleneck per iteration. In exact arithmetic the resulting pipelined CG algorithm is numerically equivalent to standard CG. However, when switching to finite precision, each of the additional recurrences accumulate rounding errors, and the extra recursions typically reduce the asymptotic accuracy of the solution, as shown further on in this manuscript.

The paper is structured as follows. In Section 2 an analysis of the accumulation of rounding errors in standard preconditioned CG, Chronopoulos/Gear CG, and the pipelined CG algorithm is presented. Based on an estimate for the so-called non-commutativity errors, an error model is

presented that allows to predict the residual stagnation point for the different methods. Section 3 discusses the incorporation of a residual replacement strategy in the pipelined CG method. A criterion for automated residual replacement based on the rounding error model from Section 2 is suggested. Extensive numerical tests showing the validity of the error model and the improvement in maximal attainable accuracy for the pipelined CG method are reported on in Section 4. Furthermore, we illustrate that the proposed residual replacement strategy does not affect the parallel scalability of the pipelined CG method. Indeed, it is demonstrated that the pipelined CG algorithm with automated residual replacement is both accurate and computationally efficient. Finally, along with a brief discussion on the subject, conclusions are drawn in Section 5.

2 Analysis of rounding error propagation in CG and pipelined CG

It is a well-known observation that the recursive residual r calculated by the Conjugate Gradient (CG) algorithm and the true residual, defined as $b - Ax$, start to deviate at some point during the CG iteration. The reason for this phenomenon is that, with the exception of the initial residual $r_0 = b - Ax_0$, the true residual is never explicitly calculated by the CG algorithm. Instead, the residual is computed using a recurrence relation to avoid the additional SPMV that would be required to compute the true residual in each iteration of the algorithm. However, in finite precision arithmetic, rounding errors occur in each of these recurrence relations (AXPYs). These rounding errors in the recurrences accumulate over the iteration, resulting in a significant difference between the true and recursive residual near the end of the algorithm.

2.1 Benchmark problem: the two-dimensional Poisson equation

Krylov subspace methods are often used as high-performance solvers in the context of scientific applications simulating a problem modeled by a partial differential equation (PDE). The primary PDE model problem used for illustration purposes throughout Sections 2 and 3 of this work is the well-known two-dimensional Poisson problem on the unit square with homogeneous Dirichlet boundary conditions,

$$\begin{aligned} \Delta u(x,y) = &\quad b(x,y), && (x,y) \in \Omega = [0,1]^2, \\ u(x,y) = &\quad 0, && (x,y) \in \partial\Omega, \end{aligned} \tag{1}$$

where Δ is the 2D Laplacian, u is the Poisson solution, and b is the known source function. A linear system can be derived from the continuous equation (1) using any discretization technique such as finite differences, finite volumes or finite elements in which neighboring variables are related by a stencil, typically leading to a sparse linear operator. In this work, equation (1) is discretized using a second order finite difference method with uniform, equidistant mesh sizes on an $N = n \times n$ points internal grid $\Omega^N \subset \Omega$, yielding the linear system

$$Ax^N = b^N, \quad A \in \mathcal{M}^{N \times N}. \tag{2}$$

Here A is the sparse matrix representation of the Laplacian Δ, which is symmetric positive definite (SPD) and has the well-known five-band structure, and $x^N = (x_1^N, \ldots, x_N^N)^T$ is the discrete representation of the solution function u on Ω^N. The current work focuses primarily on this type of sparse, well-structured matrices, or, alternatively, on matrix-free linear operators such as stencil applications.

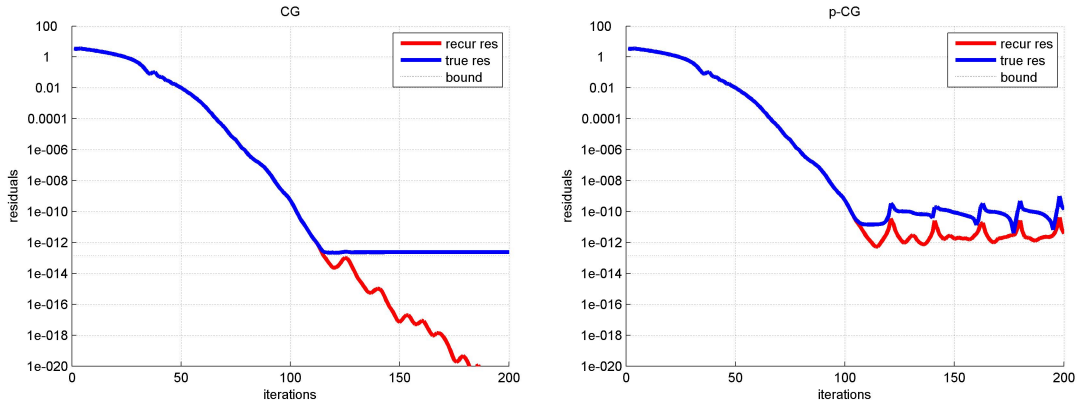


Figure 1: *Comparison of residual convergence for CG and pipelined CG. Left: residual history for standard CG. The recursive residuals (red) continue to converge, while the true residuals (blue) stagnate close to the bound* $\|A\|\psi$ *(dotted line). Right: residual history for p-CG. The true residual stagnates several orders of magnitude above the CG bound; a stagnation in the recursive residual is also observed.*

2.2 Difference between true and recursive residual in CG and pipelined CG

We first demonstrate the deviation between the true and recursive residual in both the CG and pipelined CG method. As illustrated in Figure 1 (left), where the CG residual history for the 2D Poisson model problem (2) with $n = 50$ is shown, the recursive residual computed by the CG method, Algorithm 1, typically keeps decreasing, while the true residual $r = b - Ax$ stagnates upon reaching the lower bound $\|A\|\psi$, where ψ denotes the machine precision.[1] In the pipelined CG method, Algorithm 3, an additional number of auxilary vectors defined as $s := Ap$, $w := Ar$ and $z := As = A^2p$ are computed in each iteration. These vectors are again not calculated explicitly, but in turn fulfill a recurrence relation. Each of these recurrences, however, accumulates rounding errors over the course of the algorithm, resulting in a deviation from the true vector value.

Figure 1 compares the convergence of the standard CG and pipelined CG algorithms. The left panel shows the convergence history for standard CG, Algorithm 1, for which the final true residual lies in the order of $\|A\|\psi$. The right panel shows the residuals corresponding to pipelined CG, Algorithm 3. Note how the residuals stagnate several order of magnitude above the CG bound. The final solution is significantly less accurate compared to the standard CG solution, which is reflected in the norm of the true residual upon stagnation. As reported by Ghysels et al. in [13], this loss of maximal attainable accuracy is typical for pipelined CG.

Summarizing, the accuracy loss in pipelined CG is due to the accumulation of rounding errors in the additional recursions of the algorithm that were introduced to avoid additional global reductions. In the remainder of this section we argue that these rounding errors originate primarily from the non-commutativity of the SPMV and a scalar value in finite precision arithmetic. Indeed, it will be shown that given a matrix A, a vector p and a scalar β, the commutativity

[1]Note: true residual stagnation around $\|A\|\psi$ was observed for CG when applied to the Poisson benchmark problem (1), cf. Figure 1. However, this lower bound is problem-dependent, and although the residual is bound to stagnate at some point, the observed stagnation near $\|A\|\psi$ does not necessarily generalize to the broader class of all SPD matrices.

Algorithm 1 Preconditioned CG

1: **procedure** PCG(A, M^{-1}, b, x_0)
2: $r_0 := b - Ax_0$; $u_0 := M^{-1}r_0$; $p_0 = u_0$
3: **for** $i = 0, \ldots$ **do**
4: $s_i := Ap_i$
5: $\alpha_i := (r_i, u_i) \,/\, (s_i, p_i)$
6: $x_{i+1} := x_i + \alpha_i p_i$
7: $r_{i+1} := r_i - \alpha_i s_i$
8: $u_{i+1} := M^{-1} r_{i+1}$
9: $\beta_{i+1} := (r_{i+1}, u_{i+1}) \,/\, (r_i, u_i)$
10: $p_{i+1} := u_{i+1} + \beta_{i+1} p_i$
11: **end for**
12: **end procedure**

expression $\beta(Ap) = A(\beta p)$ generally does not hold in finite-precision arithmetic, due to rounding errors on the addition and multiplication operations. We analyze the propagation of rounding errors in the CG, Chronopoulos/Gear CG and pipelined CG algorithms, and propose a model for rounding error propagation based on non-commutativity observations. This error model will in turn allow us to propose countermeasures to the accuracy loss in pipelined CG using a residual replacement strategy, presented in Section 3.

2.3 Accumulation of rounding errors in CG

The approach to analyzing the propagation of rounding errors in CG and related algorithms presented in this section has been discussed in papers by Greenbaum [14], Gutknecht et al. [15], Strakos et al. [19], Vandervorst et al. [21] and Tong et al. [20]. We recall some of these results below, which will form the basis for the analysis and improvement of the pipelined CG algorithm.

In traditional preconditioned Conjugate Gradients, Algorithm 1, the solution x_{i+1} and residual r_{i+1} are updated respectively as

$$\begin{cases} x_{i+1} = x_i + \alpha_i p_i + \delta_i^x, \\ r_{i+1} = r_i - \alpha_i s_i + \delta_i^r, \end{cases} \tag{3}$$

where $s_i := Ap_i$. Here δ_i^x and δ_i^r denote vectors representing the local rounding errors made in the current calculation. Each element in these vectors is assumed to be at most of the order of machine precision, meaning $|(\delta_i^x)_j| \leq \psi|(x_i + \alpha_i p_i)_j|$ and $|(\delta_i^r)_j| \leq \psi|(r_i - \alpha_i s_i)_j|$ for all $j = 1, \ldots, N$.

We consider Δ_{i+1}^r as the difference between the true residual $b - Ax_{i+1}$ and the residual r_{i+1} from the CG recurrence. It holds that

$$\begin{aligned} \Delta_{i+1}^r &:= r_{i+1} - (b - Ax_{i+1}) = r_i - \alpha_i s_i + \delta_i^r - b + A(x_i + \alpha_i p_i + \delta_i^x) \\ &= r_i - (b - Ax_i) - \alpha_i(Ap_i) + A(\alpha_i p_i) + A\delta_i^x + \delta_i^r \\ &= \Delta_i^r + \underbrace{A(\alpha_i p_i) - \alpha_i(Ap_i)}_{\text{non-commutativity error}} + \underbrace{A\delta_i^x + \delta_i^r}_{\text{local rounding error}}, \end{aligned} \tag{4}$$

where the equation on the second line holds since $s_i := Ap_i$ is calculated exactly in Algorithm 1. The error term that is added by the current update consists of two components. The first term allows to focus on the non-commutativity of the floating point representation in the SPMV $\alpha_i Ap_i$.

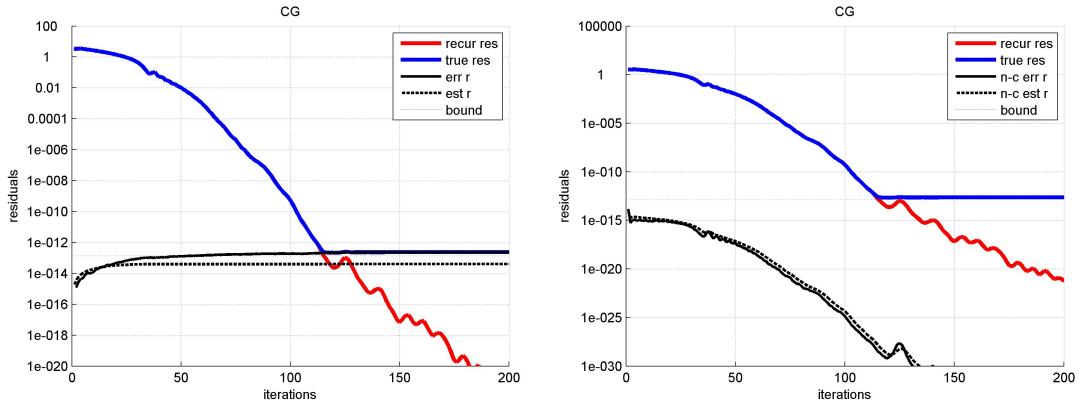


Figure 2: *Propagation of rounding errors for standard CG. Left: residual rounding error* $\|r_i - (b - Ax_i)\|$ *(solid black) and estimate* $\|\Delta_i^r\|$ *computed using* (17) *(dashed black). The true and recursive residual deviate around iteration* $i = 110$*, where rounding errors start to dominate convergence. The main accumulation of rounding errors is observed to take place in the first few iterations of CG. Right: non-commutativity error* $\|A(\alpha_i p_i) - \alpha_i(Ap_i)\|$*, see* (6) *(solid black) and estimate* $2\alpha_i\|s_i\|\psi$ (16) *(dashed black). Rounding errors are largest in the initial phase of the algorithm, where the update* $\alpha_i p_i$ *is large.*

The second term is due to the local rounding errors of the recurrence relations (3) in the i-th iteration. We briefly expound on the significance of both terms in the propagation of rounding errors in the CG algorithm.

From (4) it follows that the accumulated rounding error after k iterations is

$$\Delta_k^r = \sum_{i=0}^{k} \left(A(\alpha_i p_i) - \alpha_i(Ap_i)\right) + \sum_{i=0}^{k} \left(A\delta_i^x + \delta_i^r\right). \tag{5}$$

The initial error is zero, since r_0 is computed as the true residual. In the initial steps of the algorithm the norm of the search direction update $\alpha_i p_i = x_{i+1} - x_i$ is typically large, since large updates of the solution take place at this stage in the numerical scheme. Hence, because of these large updates, the non-commutativity errors, $\|A(\alpha_i p_i) - \alpha_i(Ap_i)\|$, are large, and vastly dominate the local rounding errors $\|A\delta_i^x + \delta_i^r\|$. As a result the error Δ_i^r grows rapidly in the first few steps of the CG algorithm. This is illustrated in Figure 2 (left panel).

In this paper we assume that δ_i^x and δ_i^r feature no errors larger than machine precision.[2] In this case, the accumulated rounding error that can be observed in Figure 2 at the stagnation point is entirely due to the non-commutativity error $\|A(\alpha_i p_i) - \alpha_i(Ap_i)\|$. Hence, when modeling the propagation of rounding errors in CG and related algorithms, the contributions of the local rounding error terms $A\delta_i^x + \delta_i^r$ are, at this moment, neglected.

We obtain the following rounding error propagation model for the residual in CG:

$$\|\Delta_{i+1}^r\| = \|\Delta_i^r\| + \|A(\alpha_i p_i) - \alpha_i(Ap_i)\|. \tag{6}$$

[2]Note: An error component that is significantly larger than machine precision in δ_i^x or δ_i^r (e.g. due to hardware failure, a so-called 'soft error') unavoidably has a long-lasting effect on CG convergence. This will be permanently reflected in the difference between the true and recursive residual, since all errors as summed, cf. (5). Although the elementary insights proposed here may possibly also prove useful in the context of soft error analysis, the focus of this work is on rounding error propagation.

The norm $\|\cdot\|$ indicates the standard 2-norm throughout this work. The above expression in principle gives an upper bound on the error; however, numerical results indicate that in practice this bound is tight, and the worst-case error estimate (6) is hence accurate. The model (6) implies that the rounding error on the residual increases in function of the iterations, as shown in Figure 2 (left). Around iteration $i = 110$, the rounding errors attain the same order of magnitude as the residuals, and the true residuals stagnate. The largest accumulation of rounding errors takes place in the initial iterations of the CG algorithm, where the non-commutativity error, $\|A(\alpha_i p_i) - \alpha_i(Ap_i)\|$, is the largest, see Figure 2 (right). This is due to the fact that the update $\alpha_i p_i$ is the largest in the initial phase of the algorithm, as commented on above.

The error model (6) accurately predicts the rounding errors, and can hence be used to monitor the magnitude of the propagated rounding errors on run time level while executing CG. This might allow to stop the algorithm at the stagnation point, i.e. $\|\Delta_i^r\| = \|r_i\|$, where performing additional iterations no longer improves the true residual. However, the computation of the non-commutativity error $\|A(\alpha_i p_i) - \alpha_i(Ap_i)\|$ would require an additional two SPMV operations in each step of the algorithm, which is an unacceptable extra computationally cost. We therefore propose to estimate the non-commutativity error in (6) using the following lemma.

Lemma 2.1 *The non-commutativity error $\|A(\alpha_i p_i) - \alpha_i(Ap_i)\|$ in the rounding error model* (6) *can be bounded from above by*

$$\|A(\alpha_i p_i) - \alpha_i(Ap_i)\| \leq 2(m+1)\alpha_i \|s_i\| \psi, \tag{7}$$

where m is the average number of nonzero elements per row of the matrix A.

Proof: We adopt the notation used by Demmel in [7], Section 1.5.1, i.e.

$$\mathrm{fl}(a \odot b) = (a \odot b)(1 + \delta^{\odot}) \tag{8}$$

where $|\delta^{\odot}| < \psi$. Here $\odot$ stands for any of the operations $+$, $-$, $*$ or $/$. Note that in the remainder of this proof we drop the iteration index in (7) for notational convenience. The calculation of the i-th element of the vector $A(\alpha p)$, where A is the system matrix, α is a scalar and $p = (p_1, \ldots, p_N)^T$ is a column vector, features the following rounding errors:

$$\begin{aligned} \mathrm{fl}\left((A(\alpha p))_i\right) &= \mathrm{fl}\left(\sum_{j=1}^{N} A_{ij}(\alpha p_j)\right) \\ &= \left(\sum_{j=1}^{N} A_{ij}(\alpha p_j)\right)(1+\delta_2^*)(1+\delta_1^*)\prod_{j=1}^{m-1}(1+\delta_j^+), \end{aligned} \tag{9}$$

where $|\delta_1^*| < \psi$ summarizes the rounding errors on the products αp_j, $|\delta_2^*| < \psi$ represents the rounding errors on the products $A_{ij}(\alpha p_j)$, and $|\delta_j^+| < \psi$ are the $m-1$ rounding errors on the m-term sum, with m the number of non-zero elements in the i-th row of the sparse matrix A. Note that we have

$$(1-\psi)^{m+1} \leq \prod_{j=1}^{m+1}(1+\delta_j^{\odot}) \leq (1+\psi)^{m+1}. \tag{10}$$

Since for any sparse matrix A it holds that $(m+1)\psi < 1$, we can use the first order approximations to the bounds to obtain

$$1-(m+1)\psi \leq \prod_{j=1}^{m+1}(1+\delta_j^{\odot}) \leq 1+(m+1)\psi, \tag{11}$$

cf. the work by Paige [18]. Hence, we obtain

$$\mathrm{fl}\left((A(\alpha p))_i\right) = \left(\sum_{j=1}^{N} A_{ij}(\alpha p_j)\right)(1+\bar{\delta}_i), \qquad i = 1,\ldots,N, \tag{12}$$

with $|\bar{\delta}_i| \leq (m+1)\psi$. Analogously, the rounding error on the i-th element of $\alpha(Ap)$ is

$$\mathrm{fl}\left((\alpha(Ap))_i\right) = \left(\sum_{j=1}^{N} \alpha(A_{ij}p_j)\right)(1+\bar{\bar{\delta}}_i), \qquad i = 1,\ldots,N, \tag{13}$$

with $|\bar{\bar{\delta}}_i| \leq (m+1)\psi$. The element-wise non-commutativity error can thus be estimated as

$$\begin{aligned} \left|\mathrm{fl}\left((A(\alpha p))_i\right) - \mathrm{fl}\left((\alpha(Ap))_i\right)\right| &= \left|\left(\sum_{j=1}^{N} A_{ij}(\alpha p_j)\right)\bar{\delta}_i - \left(\sum_{j=1}^{N} \alpha(A_{ij}p_j)\right)\bar{\bar{\delta}}_i\right| \\ &\leq 2(m+1)\alpha\left|\sum_{j=1}^{N} A_{ij}p_j\right|\psi. \end{aligned} \tag{14}$$

Using simple properties of the 2-norm and the fact that $s = Ap$, the inequality (14) then readily yields the following upper bound

$$\|\mathrm{fl}(A(\alpha p)) - \mathrm{fl}(\alpha(Ap))\| \leq 2(m+1)\alpha\|s\|\psi. \tag{15}$$

□

Lemma 2.1 provides an absolute upper bound on the non-commutativity error. Indeed, the sum $\sum_{j=1}^{N}|A_{ij}p_j|$ bounds the largest possible value that one could compute without incorporating error cancellation caused by adding positive and negative numbers. In practice, however, cancellation of rounding error does occur, most notably in the m-term row-sum in (9), such that it holds $|\bar{\delta}_i| \leq \psi$ and $|\bar{\bar{\delta}}_i| \leq \psi$. Hence, the worst-case upper bound (7) often largely overestimates the actual non-commutativity error. We therefore propose to use the following estimate for the error

$$\|A(\alpha_i p_i) - \alpha_i(Ap_i)\| \approx 2\alpha_i\|s_i\|\psi. \tag{16}$$

Using this estimate, the non-commutativity error, and hence the rounding error in the i-th step of the CG algorithm, can be approximately computed as

$$\|\Delta_{i+1}^r\| = \|\Delta_i^r\| + 2\alpha_i\|s_i\|\psi. \tag{17}$$

The above estimates are introduced in a somewhat *ad hoc* fashion here. However, numerical experiments performed in Section 4 of this article on the Poisson model and a variety of SPD matrices from Matrix Market support the error estimate (17).

Note that the incorporation of the error estimate (17) in the algorithm implies only very limited additional computational overhead, since only the vector norm $\|s_i\|$ has to be computed. This dot-product computation requires global communication, but can be combined with the existing global reduction required to compute α_i in Algorithm 1, line 5, such that the number of global reductions remains identical. Hence, the estimated rounding error can be monitored by keeping track of (17) in the CG algorithm.

The dashed lines in Figure 2 show the rounding error estimates for the per-iteration non-commutativity error (16) (right) and the rounding error (17) (left). Moreover, the error estimate

Algorithm 2 Preconditioned Chronopoulos/Gear CG

1: **procedure** PCGCG(A, M^{-1}, b, x_0)
2: $r_0 := b - Ax_0$; $u_0 := M^{-1}r_0$; $w_0 := Au_0$
3: $\alpha_0 := (r_0, u_0)/(w_0, u_0)$; $\beta := 0$; $\gamma_0 := (r_0, u_0)$
4: **for** $i = 0, \ldots$ **do**
5: $p_i := u_i + \beta_i p_{i-1}$
6: $s_i := w_i + \beta_i s_{i-1}$
7: $x_{i+1} := x_i + \alpha_i p_i$
8: $r_{i+1} := r_i - \alpha_i s_i$
9: $u_{i+1} := M^{-1} r_{i+1}$
10: $w_{i+1} := Au_{i+1}$
11: $\gamma_{i+1} := (r_{i+1}, u_{i+1})$
12: $\delta := (w_{i+1}, u_{i+1})$
13: $\beta_{i+1} := \gamma_{i+1}/\gamma_i$
14: $\alpha_{i+1} := (\delta/\gamma_{i+1} - \beta_{i+1}/\alpha_i)^{-1}$
15: **end for**
16: **end procedure**

$\|\Delta_i^r\|$ could be used in a stopping criterion to predict the residual stagnation point, i.e. when $\|r_i\| \leq \|\Delta_i^r\|$, see Figure 2 (left). Given that the error estimate is sufficiently accurate, this stopping criterion could act as a valuable alternative to traditional stopping criteria based on backward error computation

$$\frac{\|x - x_i\|}{\|x\|} \leq \kappa(A) \frac{\|r_i\|}{\|b\|}, \qquad r_i = b - Ax_i, \tag{18}$$

see Liesen and Strakos [17] (p. 317). Indeed, the true residual r_i is generally not computed in the CG algorithm and the recursive residual may be polluted by rounding errors as outlined above, rendering (18) possibly inconvenient to use. The error estimate $\|\Delta_i^r\|$ may provide valuable information to complement the backward error stopping criterion in this context.

2.4 Accumulation of rounding errors in Chronopoulos/Gear CG

The Chronopoulos/Gear CG algorithm, Algorithm 2, is an alternative formulation of CG proposed in [5], which reduces the number of global synchronization points from two (Algorithm 1, lines 5 and 9) to just one (Algorithm 2, lines 11-12). This reformulation is obtained at the cost of one additional AXPY to update the auxiliary variable $s_i = Ap_i$, which is now also computed recursively using the exactly computed $w_i := Au_i$. Note that in exact arithmetic, Algorithm 2 is mathematically equivalent to standard preconditioned CG, Algorithm 1.

In Algorithm 2 the solution x_{i+1}, the residual r_{i+1}, the search direction p_i and the auxiliary variable s_i are updated recursively as

$$\begin{cases} p_i = r_i + \beta_i p_{i-1}, \\ s_i = w_i + \beta_i s_{i-1}, \end{cases} \qquad \begin{cases} x_{i+1} = x_i + \alpha_i p_i, \\ r_{i+1} = r_i - \alpha_i s_i. \end{cases} \tag{19}$$

The propagation of rounding errors in Chronopoulos/Gear CG is however different from the traditional CG algorithm, since in each step rounding errors are now introduced in both the

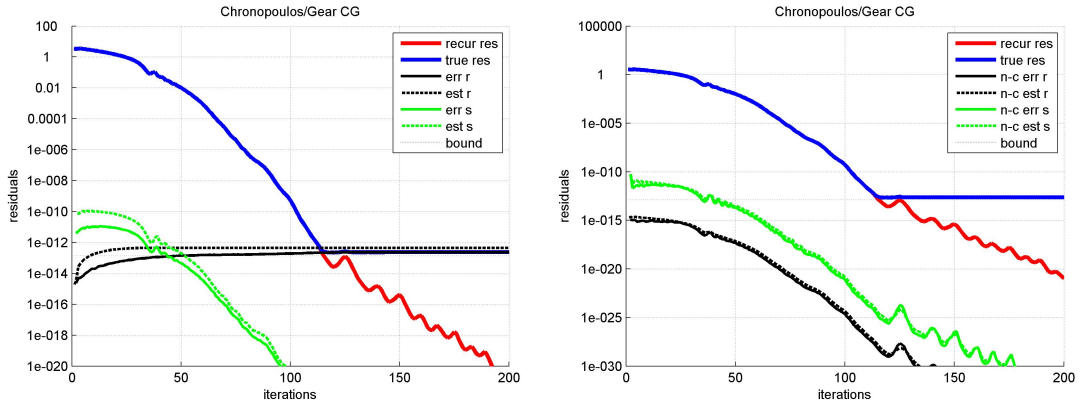


Figure 3: *Propagation of rounding errors for Chronopoulos/Gear CG. Left: residual rounding error* $\|r_i - (b - Ax_i)\|$ *(solid black) and auxiliary rounding error* $\|s_i - Ap_i\|$ *(solid green), and their respective estimates* $\|\Delta_i^r\|$ *and* $\|\Delta_i^s\|$*, see (23) (dashed lines). Note how the residual rounding errors* $\|\Delta_i^r\|$ *stagnate, while the rounding errors* $\|\Delta_i^s\|$ *decay. Right: non-commutativity errors* $\|A\alpha_i p_i - \alpha_i A p_i\|$ *(solid black) and* $\|A\beta_{i+1}p_i - \beta_{i+1}Ap_i\|$ *(solid green), and their estimates* $2\alpha_i\|s_i\|\psi$ *(dashed black) and* $2\beta_{i+1}\|s_i\|\psi$ *(dashed green).*

recursions for r_i and s_i. The rounding error on the residual r_{i+1} is

$$\begin{aligned} \Delta_{i+1}^r &:= r_{i+1} - (b - Ax_{i+1}) = r_i - \alpha_i s_i - b + A(x_i + \alpha_i p_i) \\ &= (r_i - (b - Ax_i)) - \alpha_i(s_i - Ap_i) + A\alpha_i p_i - \alpha_i A p_i \\ &= \Delta_i^r - \alpha_i \Delta_i^s + A\alpha_i p_i - \alpha_i A p_i, \end{aligned} \tag{20}$$

where again the local rounding errors are neglected to simplify the notation. Note that in Algorithm 2 s_i, defined as Ap_i, is also not explicitly calculated but rather derived from a recurrence relation. Hence, like the error on the residual, the rounding error $\Delta_i^s = s_i - Ap_i$ also grows in function of i. We find in a similar way to the above

$$\begin{aligned} \Delta_{i+1}^s &:= s_{i+1} - Ap_{i+1} = w_{i+1} + \beta_{i+1}s_i - A(r_{i+1} + \beta_{i+1}p_i) \\ &= (w_{i+1} - Ar_{i+1}) + \beta_{i+1}(s_i - Ap_i) - A\beta_{i+1}p_i + \beta_{i+1}Ap_i \\ &= \beta_{i+1}\Delta_i^s - A\beta_{i+1}p_i + \beta_{i+1}Ap_i, \end{aligned} \tag{21}$$

where the final equation holds since $w_{i+1} := Ar_{i+1}$ is computed exactly in Algorithm 2, line 10.

Combining (20) and (21) we obtain the following rounding error propagation model for Chronopoulos/Gear CG:

$$\begin{pmatrix} \|\Delta_{i+1}^r\| \\ \|\Delta_{i+1}^s\| \end{pmatrix} = \begin{pmatrix} 1 & \alpha_i \\ 0 & \beta_{i+1} \end{pmatrix} \begin{pmatrix} \|\Delta_i^r\| \\ \|\Delta_i^s\| \end{pmatrix} + \begin{pmatrix} \|\alpha_i A p_i - A\alpha_i p_i\| \\ \|\beta_{i+1}Ap_i - A\beta_{i+1}p_i\| \end{pmatrix}. \tag{22}$$

Note that the scalars α_i and β_i are positive per definition. The above error propagation model is the analogon of the standard CG model (6). However, in Chronopoulos/Gear CG, the residual rounding errors are propagated by contributions from non-commutativity errors in both r_i and s_i. We point out that, similar to (6), the model (22) in fact simulates a worst-case scenario for rounding error propagation, due to the summation of the right-hand side norms in (20)-(21).

Algorithm 3 Pipelined Chronopoulos/Gear CG

procedure PIPE-CG(A, b, x_0)
  $r_0 := b - Ax_0$; $w_0 := Ar_0$
  for $i = 0, \ldots$ **do**
    $\gamma_i := (r_i, r_i)$
    $\delta := (w_i, r_i)$
    $q_i := Aw_i$
    if $i > 0$ **then**
      $\beta_i := \gamma_i/\gamma_{i-1}$; $\alpha_i := (\delta/\gamma_i - \beta_i/\alpha_{i-1})^{-1}$
    else
      $\beta_i := 0$; $\alpha := \gamma_i/\delta$
    end if
    $z_i := q_i + \beta_i z_{i-1}$
    $s_i := w_i + \beta_i s_{i-1}$
    $p_i := r_i + \beta_i p_{i-1}$
    $x_{i+1} := x_i + \alpha_i p_i$
    $r_{i+1} := r_i - \alpha_i s_i$
    $w_{i+1} := w_i - \alpha_i z_i$
  end for
end procedure

To obtain more insight into the propagation of the individual errors on r and s, we briefly suppose that the scalars α_i and β_i are iteration independent constants, such that the small 2-by-2 matrix in (22) does not change over the course of the iteration. The eigenvalues of this matrix then predict the asymptotic, long-term growth of the error components. Its eigenvalues are $\lambda_1 = 1$ and $\lambda_2 = \beta_{i+1}$, corresponding to the residual rounding error $\|\Delta_i^r\|$ and the auxiliary rounding error $\|\Delta_i^s\|$ respectively. Given the assumption that $\beta_i < 1$ becomes small near the end of the algorithm, and given that β_i remains approximately constant over the course of the algorithm, the residual rounding error is hence expected to stagnate, while the error on s decays.

This is illustrated on Figure 3 (left panel), where the effective rounding errors $\|r_i - (b - Ax_i)\|$ and $\|s_i - Ap_i\|$ are displayed by solid lines. Compared to standard CG, the additional rounding errors caused by the recursion for the auxiliary variable s_i has little effect on the true residual at the stagnation point.

In analogy to (17) and applying Lemma 2.1, the propagation of rounding errors in Chronopoulos/Gear CG can now be estimated by approximating the non-commutativity errors in (22) as follows

$$\begin{pmatrix} \|\Delta_{i+1}^r\| \\ \|\Delta_{i+1}^s\| \end{pmatrix} = \begin{pmatrix} 1 & \alpha_i \\ 0 & \beta_{i+1} \end{pmatrix} \begin{pmatrix} \|\Delta_i^r\| \\ \|\Delta_i^s\| \end{pmatrix} + \begin{pmatrix} 2\alpha_i \|s_i\| \psi \\ 2\beta_{i+1} \|s_i\| \psi \end{pmatrix}. \tag{23}$$

Figure 3 (right) illustrates the effective non-commutativity errors (solid lines) and their respective estimates (dashed lines). The dashed lines on the right panel show the estimated accumulated rounding errors $\|\Delta_i^r\|$ and $\|\Delta_i^s\|$, which are computed from the non-commutativity error estimates using error model (23).

2.5 Accumulation of rounding errors in pipelined CG

A similar rounding error analysis can be performed for pipelined Krylov algorithms. We discuss the pipelined Chronopoulous/Gear CG shown in Algorithm 3. In addition to the solution x_i and the residual $r_i = b - Ax_i$, Algorithm 3 uses three auxiliary vectors, defined as $s_i := Ap_i$,

$w_i := Ar_i$ and $z_i := As_i = A^2 p_i$. This implies the following recursive updates are computed in each step of the algorithm:

$$\begin{cases} z_i = q_i + \beta_i z_{i-1}, \\ s_i = w_i + \beta_i s_{i-1}, \\ p_i = r_i + \beta_i p_{i-1}, \end{cases} \qquad \begin{cases} x_{i+1} = x_i + \alpha_i p_i, \\ r_{i+1} = r_i - \alpha_i s_i, \\ w_{i+1} = w_i - \alpha_i z_i. \end{cases} \tag{24}$$

Lemma 2.2 *Let r_i, s_i, w_i, z_i, p_i and x_i as defined above, be the vectors generated by the pipelined Chronopoulos/Gear CG Algorithm 3, and let the respective rounding errors Δ_i^r, Δ_i^s, Δ_i^w and Δ_i^z be defined as follows*

$$\begin{cases} \Delta_i^r := r_i - (b - Ax_i), \\ \Delta_i^s := s_i - Ap_i, \\ \Delta_i^w := w_i - Ar_i, \\ \Delta_i^z := z_i - As_i. \end{cases} \tag{25}$$

Then the rounding errors between the true and recursive values satisfy the relation

$$\begin{pmatrix} \|\Delta_{i+1}^r\| \\ \|\Delta_{i+1}^s\| \\ \|\Delta_{i+1}^w\| \\ \|\Delta_{i+1}^z\| \end{pmatrix} = \begin{pmatrix} 1 & \alpha_i & 0 & 0 \\ 0 & \beta_{i+1} & 1 & \alpha_i \\ 0 & 0 & 1 & \alpha_i \\ 0 & 0 & 0 & \beta_{i+1} \end{pmatrix} \begin{pmatrix} \|\Delta_i^r\| \\ \|\Delta_i^s\| \\ \|\Delta_i^w\| \\ \|\Delta_i^z\| \end{pmatrix} + \begin{pmatrix} e_i^r \\ e_i^s \\ e_i^w \\ e_i^z \end{pmatrix}, \tag{26}$$

where the non-commutativity errors e_i^r, e_i^s, e_i^w and e_i^z are defined as

$$\begin{pmatrix} e_i^r \\ e_i^s \\ e_i^w \\ e_i^z \end{pmatrix} := \begin{pmatrix} \|\alpha_i Ap_i - A\alpha_i p_i\| \\ \|\beta_{i+1} Ap_i - A\beta_{i+1} p_i\| + \|\alpha_i As_i - A\alpha_i s_i\| \\ \|\alpha_i As_i - A\alpha_i s_i\| \\ \|\beta_{i+1} As_i - A\beta_{i+1} s_i\| \end{pmatrix}. \tag{27}$$

Proof: For the difference between the true residual and the recursive residual we find the recurrence relation:

$$\begin{aligned} \Delta_{i+1}^r &:= r_{i+1} - (b - Ax_{i+1}) = r_i - \alpha_i s_i - b + A(x_i + \alpha_i p_i) \\ &= (r_i - b + Ax_i) - \alpha_i (s_i - Ap_i) + A\alpha_i p_i - \alpha_i Ap_i \\ &= \Delta_i^r - \alpha_i \Delta_i^s + A\alpha_i p_i - \alpha_i Ap_i, \end{aligned} \tag{28}$$

where we added and subtracted $\alpha_i Ap_i$ identically to (20). Note that s_i is not explicitly calculated but rather derived recursively. The error Δ_i^s is given by

$$\begin{aligned} \Delta_{i+1}^s &:= s_{i+1} - Ap_{i+1} = w_{i+1} + \beta_{i+1} s_i - A(r_{i+1} + \beta_{i+1} p_i) \\ &= \beta_{i+1}(s_i - Ap_i) + (w_{i+1} - Ar_{i+1}) - A\beta_{i+1} p_i + \beta_{i+1} Ap_i \\ &= \beta_{i+1} \Delta_i^s + \Delta_{i+1}^w - A\beta_{i+1} p_i + \beta_{i+1} Ap_i. \end{aligned} \tag{29}$$

Similarly, the auxiliary variable w_i is not explicitly calculated and thus diverges from its exact value Ar_i. We compute the error Δ_{i+1}^w as

$$\begin{aligned} \Delta_{i+1}^w &:= w_{i+1} - Ar_{i+1} = w_i - \alpha_i z_i - A(r_i - \alpha_i s_i) \\ &= (w_i - Ar_i) - \alpha_i (z_i - As_i) + A\alpha_i s_i - \alpha_i As_i \\ &= \Delta_i^w - \alpha_i \Delta_i^z + A\alpha_i s_i - \alpha_i As_i. \end{aligned} \tag{30}$$

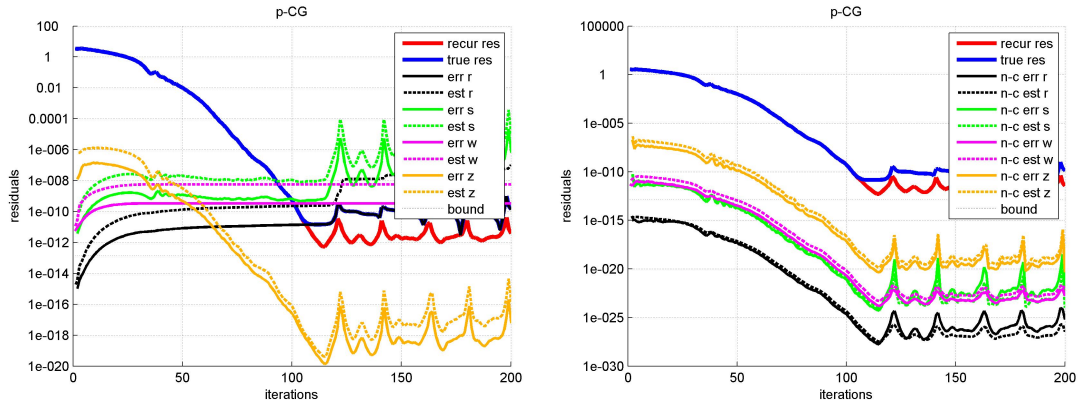


Figure 4: *Propagation of rounding errors for pipelined Chronopoulos/Gear CG. Left: residual rounding error* $\|r_i - (b - Ax_i)\|$ *(solid black) and auxiliary rounding errors* $\|s_i - Ap_i\|$ *(solid green),* $\|w_i - Ar_i\|$ *(solid magenta), and* $\|z_i - As_i\|$ *(solid yellow). Respective error estimates* $\|\Delta_i^r\|$, $\|\Delta_i^s\|$, $\|\Delta_i^w\|$ *and* $\|\Delta_i^z\|$ *(dashed lines) are computed using* (26) *and* (32)*. Note how the rounding errors* $\|\Delta_i^r\|$, $\|\Delta_i^s\|$ *and* $\|\Delta_i^w\|$ *stagnate, while the rounding errors* $\|\Delta_i^z\|$ *decay. The true residual stagnates several orders of magnitude above the bound* $\|A\|\psi$*. Right: non-commutativity errors* e_i^r *(solid black),* e_i^s *(solid green),* e_i^w *(solid magenta) and* e_i^z *(solid yellow), see* (27)*, and their respective estimates (dashed lines), see* (32) *.*

Here we observe that the auxiliary variable z_i is derived recursively, such that its recursive and true values diverge. Thus

$$\begin{aligned} \Delta_{i+1}^z &:= z_{i+1} - As_{i+1} = q_{i+1} + \beta_{i+1} z_i - A(w_{i+1} + \beta_{i+1} s_i) \\ &= (q_{i+1} - Aw_{i+1}) + \beta_{i+1} z_i - \beta_{i+1} As_i + \beta_{i+1} As_i - A\beta_{i+1} s_i \\ &= \beta_{i+1}\Delta_i^z - A\beta_{i+1} s_i + \beta_{i+1} As_i, \end{aligned} \tag{31}$$

where the final equation holds since $q_{i+1} := Aw_{i+1}$ is computed exactly in Algorithm 3, line 6. □

To incorporate the rounding error model into the pipelined CG Algorithm 3, we again propose to estimate the non-commutativity errors, (27), as follows

$$\begin{pmatrix} e_i^r \\ e_i^s \\ e_i^w \\ e_i^z \end{pmatrix} \approx \begin{pmatrix} 2\alpha_i \|s_i\| \psi \\ 2\beta_{i+1}\|s_i\|\psi + 2\alpha_i \|z_i\|\psi \\ 2\alpha_i \|z_i\|\psi \\ 2\beta_{i+1}\|z_i\|\psi \end{pmatrix}, \tag{32}$$

which only requires the vector norms $\|s_i\|$ and $\|z_i\|$ to be computed. These computations, which are computationally inexpensive but require additional global communication, can be combined with the existing global reduction in Algorithm 3, line 4-5, and hence do not reduce the overall efficiency of the algorithm.

Figure 4 shows the propagated rounding errors (left) and the per-iteration non-commutativity errors (right) for pipelined CG. The estimated rounding errors are computed by replacing the non-commutativity error estimates (32) into the error model (26). Although the non-commutativity estimates are sharp, the rounding errors appear to be slightly overestimated. Nonetheless, the error prediction model (26) allows for a fairly accurate prediction of the stagnation point. Note how the residual rounding error stagnates several orders of magnitude above the bound $\|A\|\psi$

Algorithm 4 Preconditioned pipelined CG

procedure PPIPE-CG(A, M^{-1}, b, x_0)
  $r_0 := b - Ax_0$; $u_0 := M^{-1}r_0$; $w_0 := Au_0$
  for $i = 0, \ldots$ **do**
    $\gamma_i := (r_i, u_i)$
    $\delta := (w_i, u_i)$
    $m_i := M^{-1}w_i$
    $n_i := Am_i$
    if $i > 0$ **then**
      $\beta_i := \gamma_i/\gamma_{i-1}$; $\alpha_i := (\delta/\gamma_i - \beta_i/\alpha_{i-1})^{-1}$
    else
      $\beta_i := 0; \alpha_i = \gamma_i/\delta$
    end if
    $z_i := n_i + \beta_i z_{i-1}$
    $q_i := m_i + \beta_i q_{i-1}$
    $s_i := w_i + \beta_i s_{i-1}$
    $p_i := u_i + \beta_i p_{i-1}$
    $x_{i+1} := x_i + \alpha_i p_i$
    $r_{i+1} := r_i - \alpha_i s_i$
    $u_{i+1} := u_i - \alpha_i q_i$
    $w_{i+1} := w_i - \alpha_i z_i$
  end for
end procedure

(dotted line), as was observed in Figure 1. Indeed, the additional rounding errors by the recursions for the three auxiliary variables s_i, w_i and z_i in pipelined CG have a significant impact on the value of the true residual at the stagnation point. After the true residual stagnates (around iteration $i = 110$ in the current 2D Poisson example), the updates to α_i and β_i become corrupt, leading to irregular behavior, see Section 3.1.

To gain an insight into the long-term growth of the error components, we consider the eigenvalues of the 4-by-4 propagation matrix in (26). Since the propagation matrix is upper triangular, it is easy to derive that they are given by $\lambda_1 = \lambda_3 = 1$ and $\lambda_2 = \lambda_4 = \beta_{i+1}$. We again suppose that close to convergence β_i is approximately constant and $\beta_i < 1$ is small. The rounding errors $\|\Delta_i^r\|$ and $\|\Delta_i^w\|$ correspond to the eigenvalues λ_1 and λ_3, and are hence expected to stagnate. However, since the propagation matrix additionally features a strong dependence of $\|\Delta_{i+1}^s\|$ on $\|\Delta_i^w\|$, the stagnation of $\|\Delta_i^w\|$ implies that the error $\|\Delta_i^s\|$ will also stagnate after sufficiently many iterations. The error $\|\Delta_i^z\|$ corresponds to the eigenvalue β_{i+1}, and is thus expected to decay in function of the iterations. This is observed in Figure 4 (left), where the three rounding errors on r, s and w effectively stagnate, and only the error on the auxiliary variable z decays.

2.6 Accumulation of rounding errors in preconditioned pipelined CG

In this section we extend the discussion of the growth of the rounding errors to preconditioned pipelined CG, Algorithm 4. Compared to unpreconditioned pipelined CG, Algorithm 4 features six recursions for x_i, p_i, $r_i := b - Ax_i$, $s_i := Ap_i$, $w_i := Au_i$ and $z_i := Aq_i$ as before. However, two additional recursions are computed for the preconditioned residual $u_i := M^{-1}r_i$ and the

auxiliary variable $q_i := M^{-1}s_i$, leading to a total of eight recursive updates:

$$\begin{cases} z_i = n_i + \beta_i z_{i-1}, \\ q_i = m_i + \beta_i q_{i-1}, \\ s_i = w_i + \beta_i s_{i-1}, \\ p_i = u_i + \beta_i p_{i-1}, \end{cases} \qquad \begin{cases} x_{i+1} = x_i + \alpha_i p_i, \\ r_{i+1} = r_i - \alpha_i s_i, \\ u_{i+1} = u_i - \alpha_i q_i, \\ w_{i+1} = w_i - \alpha_i z_i. \end{cases} \tag{33}$$

The extra recursions for u_i and q_i additionally lead to rounding errors. However, their influence on the algorithm appears to be very limited, as shown by the following lemma.

Lemma 2.3 *Let r_i, s_i, w_i, z_i, p_i, u_i, q_i, and x_i as defined above, be the vectors generated by the preconditioned pipelined Chronopoulos/Gear CG Algorithm 4, and let the respective rounding errors Δ_i^r, Δ_i^s, Δ_i^w, Δ_i^z, Δ_i^u and Δ_i^q be defined as follows*

$$\begin{cases} \Delta_i^r := r_i - (b - Ax_i), \\ \Delta_i^s := s_i - Ap_i, \\ \Delta_i^w := w_i - Au_i, \\ \Delta_i^z := z_i - Aq_i, \\ \Delta_i^u := u_i - M^{-1}r_i, \\ \Delta_i^q := q_i - M^{-1}s_i. \end{cases} \tag{34}$$

Then the rounding errors between the true and recursive values satisfy the relation

$$\begin{pmatrix} \|\Delta_{i+1}^r\| \\ \|\Delta_{i+1}^s\| \\ \|\Delta_{i+1}^w\| \\ \|\Delta_{i+1}^z\| \\ \|\Delta_{i+1}^u\| \\ \|\Delta_{i+1}^q\| \end{pmatrix} = \begin{pmatrix} 1 & \alpha_i & 0 & 0 & 0 & 0 \\ 0 & \beta_{i+1} & 1 & \alpha_i & 0 & 0 \\ 0 & 0 & 1 & \alpha_i & 0 & 0 \\ 0 & 0 & 0 & \beta_{i+1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \alpha_i \\ 0 & 0 & 0 & 0 & 0 & \beta_{i+1} \end{pmatrix} \begin{pmatrix} \|\Delta_i^r\| \\ \|\Delta_i^s\| \\ \|\Delta_i^w\| \\ \|\Delta_i^z\| \\ \|\Delta_i^u\| \\ \|\Delta_i^q\| \end{pmatrix} + \begin{pmatrix} e_i^r \\ e_i^s \\ e_i^w \\ e_i^z \\ e_i^u \\ e_i^q \end{pmatrix}, \tag{35}$$

where the non-commutativity errors e_i^r, e_i^s, e_i^w, e_i^z, e_i^u and e_i^q are defined as

$$\begin{pmatrix} e_i^r \\ e_i^s \\ e_i^w \\ e_i^z \\ e_i^u \\ e_i^q \end{pmatrix} := \begin{pmatrix} \|\alpha_i Ap_i - A\alpha_i p_i\| \\ \|\beta_{i+1}Ap_i - A\beta_{i+1}p_i\| + \|\alpha_i Aq_i - A\alpha_i q_i\| \\ \|\alpha_i Aq_i - A\alpha_i q_i\| \\ \|\beta_{i+1}Aq_i - A\beta_{i+1}q_i\| \\ \|\alpha_i M^{-1}s_i - M^{-1}\alpha_i s_i\| \\ \|\beta_{i+1}M^{-1}s_i - M^{-1}\beta_{i+1}s_i\| \end{pmatrix}. \tag{36}$$

Proof: The derivations of the rounding error relations for Δ_{i+1}^r, Δ_{i+1}^s, Δ_{i+1}^w and Δ_{i+1}^z are completely analogue to the proof of Lemma 2.3, and are omitted here to avoid redundancy. The rounding error on the preconditioned residual u_i is computed as follows

$$\begin{aligned} \Delta_{i+1}^u &:= u_{i+1} - M^{-1}r_{i+1} = u_i - \alpha_i q_i - M^{-1}(r_i - \alpha_i s_i) \\ &= (u_i - M^{-1}r_i) - \alpha_i(q_i - M^{-1}s_i) + M^{-1}\alpha_i s_i - \alpha_i M^{-1}s_i \\ &= \Delta_i^u - \alpha_i \Delta_i^q + M^{-1}\alpha_i s_i - \alpha_i M^{-1}s_i. \end{aligned} \tag{37}$$

The auxiliary variable q_i is in turn computed from a recurrence relation, and hence deviates from its exact value. This deviation is characterized as

$$\begin{aligned}\Delta_{i+1}^q &:= q_{i+1} - M^{-1}s_{i+1} = m_{i+1} + \beta_{i+1}q_i - M^{-1}(w_{i+1} + \beta_{i+1}s_i) \\ &= (m_{i+1} - M^{-1}w_{i+1}) + \beta_{i+1}(q_i - M^{-1}s_i) - M^{-1}\beta_{i+1}s_i + \beta_{i+1}M^{-1}s_i \\ &= \beta_{i+1}\Delta_i^q - M^{-1}\beta_{i+1}s_i + \beta_{i+1}M^{-1}s_i,\end{aligned} \tag{38}$$

where the final equation holds since $m_{i+1} := M^{-1}w_{i+1}$ is calculated exactly in each step of Algorithm 4, line 6. □

The non-commutativity errors (36) can be approximately calculated in each step of the preconditioned pipelined CG Algorithm 4 as

$$\begin{pmatrix} e_i^r \\ e_i^s \\ e_i^w \\ e_i^z \\ e_i^u \\ e_i^q \end{pmatrix} := \begin{pmatrix} 2\alpha_i\|s_i\|\psi \\ 2\beta_{i+1}\|s_i\|\psi + 2\alpha_i\|z_i\|\psi \\ 2\alpha_i\|z_i\|\psi \\ 2\beta_{i+1}\|z_i\|\psi \\ 2\alpha_i\|q_i\|\psi \\ 2\beta_{i+1}\|q_i\|\psi \end{pmatrix}. \tag{39}$$

Equation (35) shows that for preconditioned pipelined CG, a total of six recursions induce rounding errors in the algorithm. Note, however, that the system of error relations is in fact decoupled, since no coupling exists between the Δ_i^u and Δ_i^q errors and the remaining four errors components. Hence, in the case of preconditioned pipelined CG, two independent blocks of coupled errors exist, i.e.

$$\begin{pmatrix} \|\Delta_{i+1}^r\| \\ \|\Delta_{i+1}^s\| \\ \|\Delta_{i+1}^w\| \\ \|\Delta_{i+1}^z\| \end{pmatrix} = \begin{pmatrix} 1 & \alpha_i & 0 & 0 \\ 0 & \beta_{i+1} & 1 & \alpha_i \\ 0 & 0 & 1 & \alpha_i \\ 0 & 0 & 0 & \beta_{i+1} \end{pmatrix} \begin{pmatrix} \|\Delta_i^r\| \\ \|\Delta_i^s\| \\ \|\Delta_i^w\| \\ \|\Delta_i^z\| \end{pmatrix} + \begin{pmatrix} e_i^r \\ e_i^s \\ e_i^w \\ e_i^z \end{pmatrix} \tag{40}$$

and

$$\begin{pmatrix} \|\Delta_{i+1}^u\| \\ \|\Delta_{i+1}^q\| \end{pmatrix} = \begin{pmatrix} 1 & \alpha_i \\ 0 & \beta_{i+1} \end{pmatrix} \begin{pmatrix} \|\Delta_i^u\| \\ \|\Delta_i^q\| \end{pmatrix} + \begin{pmatrix} e_i^u \\ e_i^q \end{pmatrix}. \tag{41}$$

The 4-by-4 block is identical to the error propagation model in unpreconditioned pipelined CG, see (26), and predicts the rounding error on the residual r_i. The small 2-by-2 block models the rounding error on the preconditioned residual u_i. This implies that, when incorporating the rounding error model in Algorithm 4, in principle both the rounding error on the residual and preconditioned residual must be monitored separately to predict the stagnation point of the true residual. In practice, however, the accumulation of rounding error in the preconditioned residual u_i is negligible compared to the rounding error accumulation on the residual r_i. This is due to the fact that, given M is an accurate and easily invertible preconditioner to A, the non-commutativity errors e_i^u and e_i^q are significantly smaller compared to the non-commutativity errors e_i^r, e_i^s, e_i^w and e_i^z. Indeed, if M and A are chosen such that $\|M^{-1}\| \ll \|A\|$, it follows that $\|\alpha M^{-1}s - M^{-1}\alpha s\| \ll \|\alpha As - A\alpha s\|$ for any given scalar α and vector $s = (s_1, \ldots, s_N)^T$. Hence, the accumulation of rounding errors on the preconditioned residual u_i described by (41) is much smaller than the accumulation on the unpreconditioned residual r_i given by (40), and can thus be neglected when aiming to predict the stagnation point in Algorithm 4. This observation is confirmed by the numerical results in Section 4 of this paper.

3 Pipelined CG with residual replacement to improve accuracy

The aim of this section is to suggest a residual replacement strategy for pipelined CG methods to improve the maximal attainable accuracy of pipelined CG, Algorithms 3 and 4. As shown above, the cause of this reduced accuracy can be found in the preceding analysis of rounding error propagation in pipelined CG. We therefore propose robust countermeasures to reduce the propagation of rounding errors.

3.1 Residual stagnation

Experiments with pipelined CG show that the true residual stagnates several order of magnitude above the maximal attainable accuracy of the true residual in standard CG. This effect was briefly discussed by Ghysels et al. in [13], and is clearly visible from Figure 4. The cause of this early stagnation was traced back to the accumulation of rounding errors by the analysis in the previous section. Furthermore, the residual history shows irregular peaks in the true residual beyond the stagnation point, see Fig. 1 (right) and Fig. 4.

The origin of the irregular behavior can also be found in the rounding errors for the recursive updates on r_i, and its effect on the recurrence for the scalars α_i and β_i in Algorithm 3 and 4. Algorithm 3 uses the subsequent $\gamma_i := (r_i, r_i)$ to calculate an update for α_i and β_i. Here β_i is traditionally defined as $\gamma_i/\gamma_{i-1} = (r_i, r_i)/(r_{i-1}, r_{i-1})$. Furthermore, $\alpha_i := \gamma_i/(p_i, Ap_i)$, such that we can write $p_i^T A p_i = \gamma_i/\alpha_i$. This yields

$$\delta := r_i^T A r_i = (p_i - \beta_i p_{i-1})^T A (p_i - \beta_i p_{i-1}) = p_i^T A p_i + \beta_i^2 p_{i-1}^T A p_{i-1}, \tag{42}$$

where we used that the search directions p_i are A-orthogonal to eliminate the term $p_i^T A p_{i-1}$. We obtain

$$\delta = \frac{\gamma_i}{\alpha_i} + \beta_i^2 \frac{\gamma_{i-1}}{\alpha_{i-1}} = \frac{\gamma_i}{\alpha_i} + \beta_i \frac{\gamma_i}{\alpha_{i-1}} \tag{43}$$

Reorganizing the above directly leads to the relation for α_i

$$\alpha_i = (\delta/\gamma_i - \beta_i/\alpha_{i-1})^{-1}. \tag{44}$$

When the pipelined CG algorithm is close to convergence, the rounding errors dominate the update for r_i as described by the error model (26). This causes the (orthogonality) relations that typically hold between subsequent residuals to become invalid, leading to significant errors in α_i and β_i. As a result, inaccuracies and even divergence from the solution may arise, leading to the peaks observed on Figure 4 after the stagnation point. However, the inevitable movement away from the solution results in a growth of (r_i, r_i). Consequently, after a small number of iterations the scalars α_i and β_i can again be accurately calculated, leading to renewed convergence and a reduction in the residuals. When the residual rounding errors again become dominant, the cycle restarts and a new peak in the residual history is formed.

3.2 Residual replacement strategy

In this section we propose a residual replacement strategy to improve the maximal attainable accuracy of pipelined CG. Furthermore, the proposed method will alleviate the residual irregularities after the stagnation point that were described above. The idea of residual replacement in Krylov subspace methods was discussed by Van der Vorst and Ye [21].

The main idea in using residual replaced for pipelined CG is to replace the vectors r, s, w and z, which are computed using recurrence relations, see Algorithm 3, line 12-13 and 16-17, and

are, hence, contaminated by rounding errors in each step of the algorithm, by their definition values at a given iteration i in the algorithm, i.e.

$$\begin{cases} s_i = Ap_i, \\ z_i = As_i, \\ r_{i+1} = b - Ax_{i+1}, \\ w_{i+1} = Ar_{i+1}. \end{cases} \tag{45}$$

Note how the current solution guess x_{i+1} and the search direction p_i are not replaced, since the exact values of these vectors are unknown. In the preconditioned pipelined Algorithm 4, we additionally replace the vectors u and q by their exact values

$$\begin{cases} q_i = M^{-1}s_i, \\ u_{i+1} = M^{-1}r_{i+1}. \end{cases} \tag{46}$$

However, two important questions directly arise when incorporating a residual replacement in pipelined CG. First, one could inquire if such a drastic replacement strategy does not destroy the established Krylov convergence. The analysis performed in [21] substantiates that residual replacement effectively maintains convergence when it is applied at the right moment in the iteration. A second, related question concerns the iteration in which replacements should take place. Since each residual replacement step comes at the cost of computing additional SPMV's, an accurate criterion to determine the need for residual replacement, i.e. the iteration i, that does not overestimate the total number of replacements, is essential.

We briefly recapitulate the main results from [21] and [20] below, which allow to formulate a robust residual replacement strategy. The recurrences for r and p in pipelined CG, Algorithm 3, are

$$\begin{cases} r_i = r_{i-1} - \alpha_{i-1}Ap_{i-1} + \Delta_i^r, \\ p_i = r_i + \beta_i p_{i-1} + \Delta_i^p. \end{cases} \tag{47}$$

The rounding errors Δ_i^r and Δ_i^p include the errors from calculating Ap_{i-1} through s_i and using recurrences for all auxiliary variables, see (26). These rounding errors can typically be bounded in terms of ψ. Combining the above recursions yields the perturbed Lanczos relation

$$AZ_i = Z_iT_i - \frac{\|r_0 - \alpha_0 Ap_0\|}{\alpha_i\|r_1\|\|r_i\|} r_{i+1}e_i^T + F_i \quad \text{with} \quad Z_i = \left[\frac{r_1}{\|r_1\|}, \ldots, \frac{r_i}{\|r_i\|}\right], \tag{48}$$

see [21], where T_i is a tridiagonal matrix and $F_i = [f_1, \ldots, f_i]$ with

$$f_i = \frac{A\Delta_i^p}{\|r_i\|} + \frac{1}{\alpha_i}\frac{\Delta_{i+1}^r}{\|r_i\|} - \frac{\beta_i}{\alpha_{i-1}}\frac{\Delta_i^r}{\|r_i\|}. \tag{49}$$

Equation (48) is the Lanczos equation satisfied by the exact CG iteration in infinite precision, perturbed by the matrix F_i which depends on the rounding errors Δ_j^r and Δ_j^p for all $j = 1, \ldots, i$.

A key result from [20] states that if r_i satisfies the perturbed Lanczos relation (48), then

$$\|r_{i+1}\| \leq C_i \min_{p\in\mathcal{P}_i, p(0)=1} \|p(A + \Delta A_i)r_1\|, \tag{50}$$

where $C_i > 0$ is an iteration-dependent constant and $\Delta A_i = -F_i Z_i^+$. This implies that even if the perturbation F_i is significantly larger than ψ, which is the case after residual replacement in step i, the norm of the residual $\|r_{i+1}\|$ is not significantly affected, and convergence remains

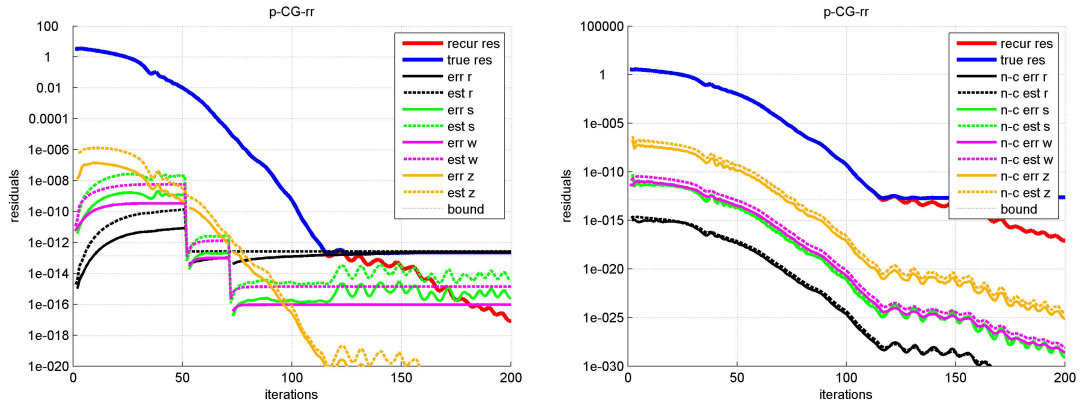


Figure 5: *Propagation of rounding errors for pipelined CG with residual replacement. Left: residual rounding error* $\|r_i - (b - Ax_i)\|$ *(solid black) and auxiliary rounding errors* $\|s_i - Ap_i\|$ *(solid green),* $\|w_i - Ar_i\|$ *(solid magenta), and* $\|z_i - As_i\|$ *(solid yellow). Respective error estimates* $\|\Delta_i^r\|$, $\|\Delta_i^s\|$, $\|\Delta_i^w\|$ *and* $\|\Delta_i^z\|$ *(dashed lines) are computed using* (26) *and* (32)*. Note how the residual replacement strategy* (51) *improves the true residual norm at the stagnation point (around iteration* $i = 110$*) compared to standard pipelined CG, see Figure 4, leading to a final residual norm which is comparable to standard CG, see Figure 2. Right: non-commutativity errors* e_i^r *(solid black),* e_i^s *(solid green),* e_i^w *(solid magenta) and* e_i^z *(solid yellow), see* (27)*, and their respective estimates (dashed lines), see* (32) *.*

stable. Based on the bound (50), Van der Vorst et al. propose to update the residuals and other vectors to their true values only when the residual vector norm $\|r_i\|$ is sufficiently large compared to the rounding error $\|\Delta_i^r\|$. Convergence is then expected to resume in a similar fashion after the residual replacement step. Performing replacements when $\|r_i\|$ is small is not recommended, since this could affect the CG convergence [21].

As the iteration proceeds, $\|\Delta_i^r\|$ typically increases, while the residual norm $\|r_i\|$ is expected to decrease, see Figure 4 (left). The replacement strategy reduces the rounding error Δ_i^r, but, according to (50), should be carried out before $\|\Delta_i^r\|$ becomes too large relative to $\|r_i\|$. Hence, in analogy to [21], we use a threshold τ, typically chosen as $\tau = \sqrt{\psi}$, and introduce the residual replacement in step i of the pipelined CG Algorithm 3 or the preconditioned Algorithm 4 if

$$\|\Delta_{i-1}^r\| \leq \tau \|r_{i-1}\| \quad \text{and} \quad \|\Delta_i^r\| > \tau \|r_i\|. \tag{51}$$

The above criterion ensures that convergence is maintained when residual replacement takes places, since replacements are only allowed when $\|r_i\|$ is sufficiently large. Furthermore, it ensures that no excess residual replacement steps are applied, keeping the total cost of the algorithm as low as possible. The rounding error model (26), which was incorporated in pipelined CG to predict the stagnation point, allows for a direct implementation of the replacement criterion (51) in Algorithms 3 and 4.

Figure 5 is the analogon of Figure 4 with the incorporation of the above residual replacement strategy in the pipelined CG algorithm. The replacement criterion (51) is implemented using the estimated rounding errors given by (26). The criterion ensures that only a limited number of replacement steps (in this case two) occur, and replacement only takes place when the residual is large relative to the (estimated) rounding errors. After residual replacement the accumulated residual rounding error is set to drop back to the $\|A\|\psi$ level. Additionally, observe how after a residual replacement event proper convergence of the computed residuals is maintained. Indeed,

Matrix	N	CG		CG-CG		p-CG		p-CG-rr		
		iter	res	iter	res	iter	res	iter	res	rr
lapl50	2,500	119	2.2e-13	114	3.7e-13	102	1.6e-10	115	2.6e-13	2
lapl100	10,000	233	1.3e-12	221	2.7e-12	190	4.7e-09	224	1.4e-12	3
lapl200	40,000	460	7.0e-12	431	1.6e-11	340	1.0e-07	435	9.1e-12	4
lapl400	160,000	982	4.1e-11	841	9.3e-11	628	2.5e-06	853	5.7e-11	6
lapl800	640,000	1945	2.3e-10	1583	5.4e-10	1138	5.5e-05	1553	9.8e-10	10

Table 1: *Model problem 2D Laplacian discretization matrices for growing number of grid points. A linear system with right-hand side $b = A\hat{x}$ where $\hat{x}_j = 1/\sqrt{N}$ is solved with each of these matrices with the four different algorithms presented. The initial guess is all-zero $x_0 = 0$. The number of iterations $iter = i$ required to reach the estimated rounding error on the residual is given, along with the corresponding true residual $res = ||b - Ax_i||$. For the p-CG-rr method the table additionally indicates the number of replacement steps rr.*

up to iteration $i = 110$, the residuals shown in Figure 5 (left) are nearly indistinguishable from the residuals of standard pipelined CG in Figure 4. The pipelined CG algorithm with automated residual replacements, Algorithm 5, displays a stagnation of the true residual around iteration $i = 115$, at which point the residual norm is comparable to the final residual norm of the standard CG and Chronopoulos/Gear CG algorithms, see Figure 2. Additionally, the replacement strategy allows to reduce the irregular residual behavior after iteration $i = 110$ that was observed in Figure 4.

4 Numerical results

Numerical results on a wide range of matrices from applications are presented to compare the residual history of the different CG methods and show the improvement in maximal attainable accuracy using the residual replacement strategy. The convergence results in Sections 4.1 and 4.2 are based on a Matlab implementation of the different CG algorithms and error estimates. Parallel performance measurements in Section 4.3 result from a PETSc implementation of pipelined CG with automated residual replacements on a distributed memory machine using the message passing paradigm.

4.1 Poisson model problem

The methods presented above are initially tested on a two-dimensional Laplace PDE model (1) for growing numbers of grid points n. Table 1 shows convergence results for solving the discretized Laplacian system (2) for $n = 50, 100, 200, 400$ and 800, with corresponding moderately increasing conditions numbers, ranging from 1.5e+3 to 1.8e+5. A linear system with exact solution $\hat{x}_j = 1/\sqrt{N}$, where $N = n^2$ is the number of rows in A, such that $\|\hat{x}\| = 1$ and right-hand side $b = A\hat{x}$ is solved for each of these discretization matrices. The initial guess is always $x_0 = 0$, and the stopping criterion is $\|r_i\| < \|\Delta_i^r\|$, from which point on the true residual is expected to stagnate. No preconditioner is used for the Laplace problem. The table lists the required number of iterations *iter* and the final true residual norm *res* for the CG, Chronopoulos/Gear CG, p-CG and p-CG-rr methods.

One observes from Table 1 that although the Chronopoulos/Gear CG algorithm generally requires slightly less iterations than standard CG for the Poisson model, both methods obtain a very similar final residual. For all grid sizes, the pipelined CG algorithm stagnates at a significantly higher residual norm due to the propagation of rounding errors in the auxiliary recursions, cf. Lemma 2.2. Using residual replacements in the pipelined CG algorithm, implemented using criterion (51), the maximal attainable accuracy on the residual is improved significantly for all

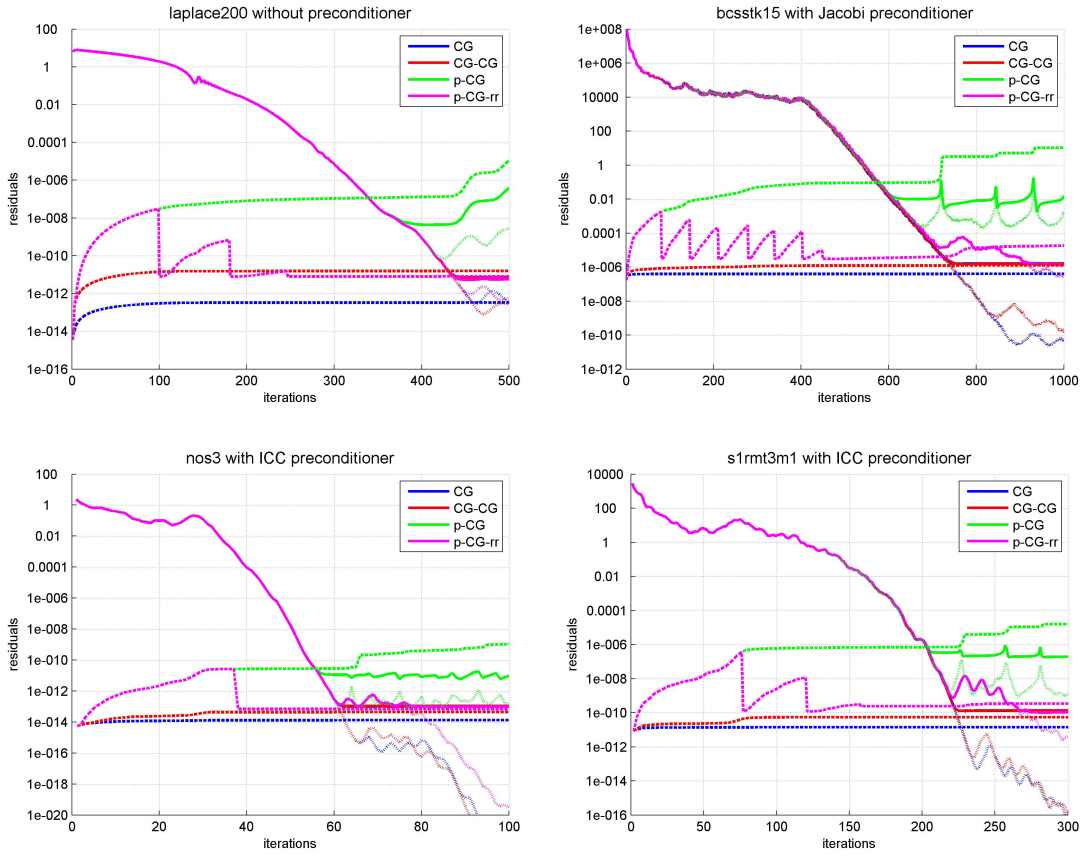


Figure 6: *Convergence history for the different CG methods applied to four different symmetric positive definite test matrices from various applications (see also Table 1 and 2). Solid lines: true residual norm* $\|b - Ax_i\|$*; dotted lines: residual error* $\|\Delta_i^r\|$*; dashed lines: recursive residual norm* $\|r_i\|$*. Convergence of CG (blue) and Chronopoulos/Gear CG (red) is nearly indistinguishable. The pipelined CG method (green) suffers from rounding error accumulation, leading to early stagnation way above the* $\|A\|\psi$ *level. The residual replacement strategy applied in the p-CG-rr method (magenta) reduces the accumulation of rounding errors, leading to a final residual accuracy that is similar to standard CG.*

problem sizes. Final residuals for the p-CG-rr method are again comparable to the residuals obtained by standard CG, as shown in the table.

The top-left panel of Figure 6 shows the convergence history of the Poisson model problem `lapl200` with $n = 200$. The residual accuracy attained by p-CG-rr, Algorithm 5, is comparable to the final residuals obtained by the CG and Chronopoulos/Gear CG Algorithms 1 and 2, whereas the p-CG method, Algorithm 3, suffers heavily from the accumulation of rounding errors (dotted green line). The application of several residual replacement steps in the initial phases of the algorithm reduces the accumulation of rounding errors (dotted magenta line), leading to significantly improved accuracy on the final residual.

4.2 Problems from Matrix Market

We present numerical results on a wide range of different linear systems to show the robustness of the proposed residual replacement strategy. Table 2 lists all square, real and symmetric positive definite matrices from Matrix Market[3], with their respective condition number κ, number of rows N and total number of nonzero elements $\#nnz$. We again solve a linear system with exact solution $\hat{x}_j = 1/\sqrt{N}$ and right-hand side $b = A\hat{x}$ for each of these matrices with the four presented methods, using an all-zero initial guess $x_0 = 0$. The stopping criterion is based on the estimated point of residual stagnation, i.e. when $\|r_i\| < \|\Delta_i^r\|$, as before. Jacobi diagonal preconditioning (JAC) and Incomplete Cholesky Factorization (ICC) are included to reduce the number of Krylov iterations if possible. Note that for some matrices the preconditioner is designated by *ICC; in this case a compensated Incomplete Cholesky factorization is performed, where a real non-negative scalar η is used as a global diagonal shift in forming the Cholesky factor. For the `nos1` and `nos2` matrices, the shift is chosen to be $\eta = 0.5$, whereas for all other *ICC preconditioners we used $\eta = 0.1$.

Table 2 lists the number of iterations $iter = i$ required such that $\|r_i\| < \|\Delta_i^r\|$, as well as the final true residual norm res for all methods. A '-' entry in the table denotes failure to meet the stopping criterion within 5,000 iterations. Note how for all matrices the residual replacement strategy incorporated in p-CG-rr improves the maximal attainable accuracy of the p-CG method.[4] For some matrices, notably `s3rmq4m1`, `s3rmt3m1` and `s3rmt3m3`, only a minor improvement of the attainable accuracy is observed. In these examples the p-CG residual stagnates significantly above the accuracy bound for standard CG, which is only partially resolved by the replacement strategy. However, despite being unable to obtain the precision of standard CG for these particular matrices, the maximal attainable accuracy of the p-CG algorithm is in general significantly improved by the incorporation of the residual replacement technique.

Apart from the convergence for the Laplace model (top-left), cf. Table 1, Figure 6 additionally illustrates the convergence history for several randomly selected matrices from Table 2. Fig. 6 (top-right) shows the true residual (solid), residual rounding error (dotted) and recursive residual (dashed) for the `bcsstk15` matrix with Jacobi preconditioning. The bottom panels of Fig. 6 show the convergence history for the `nos3` (bottom-left) and `s1rmt3m1` (bottom-right) matrices with ICC preconditioner respectively. The convergence of the standard CG and Chronopoulos/Gear CG methods is nearly indistinguishable. The accumulation of rounding errors in the pipelined CG algorithm causes the residuals to level off sooner compared to standard and Chronopoulos/Gear CG, resulting in a less accurate final solution. Based on the estimated rounding errors and (51), the residual replacement strategy resets the rounding errors in certain iterations, leading to a

[3] `http://math.nist.gov/MatrixMarket/`

[4] We note for completeness that the replacement criterion tolerance τ in (51) was manually modified in two cases in Table 2 to optimize the replacement strategy (i.e. to avoid that either an excessive or a non-sufficient number of replacements is performed). For the `bcsstk18` matrix, the tolerance was set to $\tau =$ 1e+4 $\sqrt{\psi}$, whereas for the `nos7` matrix we used $\tau =$ 1e-4 $\sqrt{\psi}$.

Matrix	Prec	$\kappa(A)$	N	$\#nnz$	$\|\|b - Ax_0\|\|$	CG		CG-CG		p-CG		p-CG-rr		
						iter	res	iter	res	iter	res	iter	res	rr
bcsstk14	JAC	1.3e+10	1806	63,454	2.1e+09	631	2.0e-06	630	2.1e-06	411	2.3e-02	570	4.6e-05	6
bcsstk15	JAC	8.0e+09	3948	117,816	4.3e+08	758	1.7e-06	743	1.9e-06	575	8.7e-02	793	1.1e-05	8
bcsstk16	JAC	65	4884	290,378	1.5e+08	288	6.3e-07	281	1.6e-06	238	6.6e-03	285	1.1e-06	4
bcsstk17	JAC	65	10,974	428,650	9.0e+07	3445	1.4e-06	3317	3.6e-06	2478	5.8e+00	3237	3.8e-05	25
bcsstk18	JAC	65	11,948	149,090	2.6e+09	2271	5.7e-06	2248	6.1e-06	1270	4.6e-01	2056	7.6e-05	11
bcsstk27	JAC	7.7e+04	1224	56,126	1.1e+05	332	4.3e-10	322	1.0e-09	268	5.7e-06	352	8.9e-10	3
bcsstm19	-	2.3e+05	817	817	3.8e+06	766	4.8e-09	800	1.1e-07	345	3.9e-01	942	6.5e-06	34
bcsstm20	-	2.6e+05	485	485	4.6e+06	465	3.8e-09	484	9.9e-08	272	2.5e-01	627	4.3e-06	24
bcsstm21	-	24	3600	3600	1.0e-04	4	1.8e-20	4	3.5e-20	4	8.9e-20	4	8.9e-20	1
bcsstm22	-	9.4e+02	138	138	3.1e-03	68	2.1e-18	67	3.8e-18	65	1.6e-14	71	7.2e-18	2
bcsstm23	-	9.5e+08	3134	3134	6.7e+05	-	9.6e-03	-	1.4e-03	1749	2.4e-02	3700	2.1e-03	38
bcsstm24	-	1.8e+13	3562	3562	1.1e+05	-	1.0e-07	-	1.1e-05	1301	3.0e-03	2445	4.6e-04	39
bcsstm25	-	6.1e+09	15,439	15,439	6.9e+07	-	1.7e+01	-	1.6e+02	2195	6.7e+01	-	1.7e+02	309
bcsstm26	-	2.6e+05	1922	1922	2.6e-01	3615	7.6e-16	3484	2.0e-15	1720	1.8e-09	3063	4.6e-13	37
gr_30_30	-	3.8e+02	900	7744	1.1e+00	54	3.7e-15	53	7.9e-15	49	6.8e-13	54	6.7e-15	2
nos1	*ICC	2.5e+07	237	1017	5.7e+07	342	8.4e-07	334	8.0e-07	270	1.9e-01	631	1.7e-06	9
nos2	*ICC	6.3e+09	957	4137	1.8e+09	3501	1.5e-04	3683	1.9e-04	1923	1.1e+04	3867	2.3e-01	56
nos3	ICC	7.3e+04	960	15,844	1.0e+01	63	1.0e-13	62	1.1e-13	56	2.3e-11	68	9.2e-14	2
nos4	ICC	2.7e+03	100	594	5.2e-02	31	9.4e-17	30	1.2e-16	29	2.6e-15	30	1.3e-16	2
nos5	ICC	2.9e+04	468	5172	2.8e+05	60	1.2e-10	60	1.2e-10	56	1.4e-08	59	5.4e-10	2
nos6	ICC	8.0e+06	675	3255	8.6e+04	40	4.2e-10	34	5.7e-10	28	4.3e-06	47	5.3e-10	2
nos7	ICC	4.1e+09	729	4617	8.6e+03	47	2.5e-10	44	3.6e-10	33	3.7e-10	56	3.4e-10	4
s1rmq4m1	ICC	1.8e+06	5489	262,411	1.5e+04	122	6.5e-11	121	7.1e-11	110	9.6e-08	142	2.9e-10	3
s1rmt3m1	ICC	2.5e+06	5489	217,651	1.5e+04	227	1.3e-10	225	1.4e-10	204	6.2e-07	258	3.2e-10	3
s2rmq4m1	*ICC	1.8e+08	5489	263,351	1.5e+03	366	1.0e-11	362	1.2e-11	309	1.2e-06	449	4.3e-10	8
s2rmt3m1	ICC	2.5e+08	5489	217,681	1.5e+03	273	1.3e-11	270	1.6e-11	240	2.1e-06	363	3.8e-11	6
s3dkq4m2	*ICC	1.9e+11	90,449	2,455,670	6.8e+01	-	1.3e-06	-	1.4e-06	2658	3.6e-06	2460	9.2e-06	37
s3dkt3m2	*ICC	3.6e+11	90,449	1,921,955	6.8e+01	-	2.0e-05	-	2.0e-05	3409	1.3e-05	3370	1.5e-05	36
s3rmq4m1	*ICC	1.8e+10	5489	262,943	1.5e+02	1692	2.2e-12	1779	2.6e-12	1396	2.4e-05	1343	6.8e-06	32
s3rmt3m1	*ICC	2.5e+10	5489	217,669	1.5e+02	2399	4.2e-12	2255	4.7e-12	1821	1.1e-04	1878	9.9e-06	28
s3rmt3m3	*ICC	2.4e+10	5357	207,123	1.3e+02	2906	4.2e-12	2706	4.8e-12	2068	1.7e-04	2477	2.1e-07	60

Table 2: *All real, square and positive definite matrices from Matrix Market, listed with their respective condition number $\kappa(A)$, number of rows/columns N and total number of nonzeros $\#nnz$. A linear system with right-hand side $b = A\hat{x}$ where $\hat{x}_i = 1/\sqrt{N}$ is solved with each of these matrices with the four different algorithms presented. The initial guess is all-zero $x_0 = 0$. Jacobi (JAC) and Incomplete Cholesky (ICC) preconditioners are included where needed. The number of iterations iter required to reach the estimated rounding error on the residual is given, along with the corresponding true residual $res = ||b - Ax_i||$. For the p-CG-rr method the table additionally indicates the number of replacement steps.*

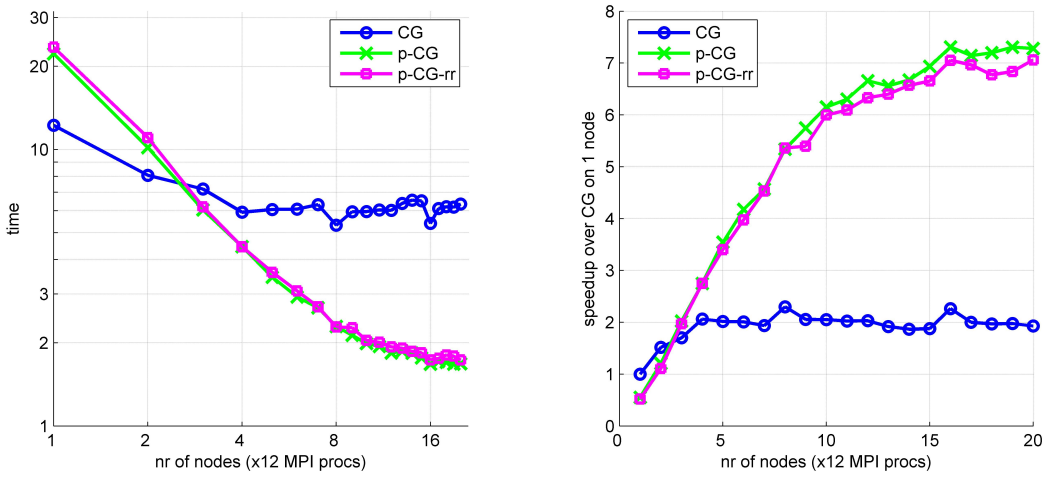


Figure 7: *Strong scaling experiment performed on up to 20 nodes (240 cores). Left: Time to solution (`log10` scale) as function of number of nodes (`log2` scale) for the 2D Poisson problem with* 1000×1000 *unknowns. Right: Speedup as function of the number of nodes over standard CG solver on a single node. All methods converged in 1474 iterations to a relative residual tolerance of* 10^{-6}*. The p-CG-rr algorithm performed 11 replacement steps.*

much more accurate final solution. Note that the behavior of the pipe-CG-rr residuals near the stagnation point differs slightly from the standard CG residuals, which is an artifact from the irregularities in the scalar coefficients α_i and β_i near the stagnation point, see Section 3.1. However, the final attainable accuracy on the pipe-CG-rr residuals is similar to standard CG.

4.3 Parallel performance

The pipelined CG method was presented in [13] in order to overcome the global communication bottleneck that is characteristic for standard CG on large parallel machines. In this section we demonstrate that the parallel performance of pipelined CG, Algorithm 4, is maintained by the addition of the automated residual replacement strategy in the extended p-CG-rr Algorithm 5.

The following parallel experiment is performed on a small cluster with 20 compute nodes, consisting of two 6-core Intel Xeon X5660 Nehalem 2.80 GHz processors each (12 cores per node), for a total of 240 cores. Nodes are connected by $4 \times$ QDR InfiniBand technology, providing 32 Gb/s of point-to-point bandwidth for message passing and I/O.

We use PETSc [1] version 3.6.3, which includes implementations of the CG and p-CG algorithms. The p-CG-rr Algorithm 5 was manually implemented by the authors as a direct extension of the p-CG method. The benchmark problem used to asses parallel performance in this section is a simple moderately-sized 2D Poisson model (1), available in the PETSc distribution as example 2 in the Krylov solvers folder. The simulation domain is discretized using a second order finite difference stencil with 1000×1000 grid points (1 million unknowns). No preconditioner is applied. The relative tolerance for Krylov solution imposed on the standard recursive residual norm $\|r_i\|_2$ is set to 10^{-6}.

Since each node consists of 12 cores, we use 12 MPI processes per node to fully exploit paral-

lelism on the machine. The MPI library used for this experiment is MPICH-3.1.3[5]. Note that the environment variables `MPICH_ASYNC_PROGRESS=1` and `MPICH_MAX_THREAD_SAFETY=multiple` are set to ensure optimal parallelism.

Figure 7 (left) shows the time to solution as a function of the number of nodes. For the given benchmark problem it is observed that pipelined CG starts to outperform standard CG when the number of cores exceeds 2. The maximum speed-up for p-CG on 20 nodes compared to CG on a single node is 7.28, see Figure 7 (right), whereas the CG method achieves a speedup of only 1.92 on 20 nodes. This implies pipelined CG attains a net speedup of 3.79 compared to standard CG when both are executed on 20 nodes. The p-CG-rr method shows very similar performance results compared to p-CG. A minor slowdown is observed, which is primarily due to the computation and communication of the two additional norms $\|s\|$ and $\|z\|$ in line 7-8 of Algorithm 5 that are required for the automated residual replacement (rounding error estimates), and to a minor extent due to the additional computational work to compute the extra SPMV's when residual replacement takes place. The maximum speedup for p-CG-rr on 20 nodes compared to CG on a single node is 7.06, yielding a total speedup of 3.66 compared to standard CG on 20 nodes.

Figure 8 shows the accuracy of the solution in function of the number of iterations (left) and computational time (right) spend by the three algorithms for the 2D Poisson benchmark problem on a 12 node setup. In only 3.17 seconds, corresponding to 2500 iterations (incl. 20 replacement steps), the p-CG-rr algorithm obtains an accurate solution with a true residual norm of 2.12e-11. Standard CG needs 9.38 s. to attain a comparable accuracy on the solution, performing 2300 iterations on the same 12 nodes. This is approximately three times slower than the time required by p-CG-rr to obtain a similar precision, see also Figure 7. The p-CG method without residual replacement is unable to reach the aforementioned accuracy regardless of computational effort. Indeed, stagnation of the true residual norm around 1.0e-7 is imminent from 1700 iterations or, equivalently, a total time of 2.15 s. onward.

5 Conclusion and Discussion

The deviation of the recursive residual from the true residual around machine precision level is a well-known aspect of the CG method. Whereas the true residual stagnates due to the accumulation of rounding errors on the solution, the recurred residual typically keeps decreasing. This observation is significantly more prominent in the pipelined Chronopoulous & Gear version of CG, leading to a stagnation of the residual norm several orders of magnitude above the accuracy attainable by standard CG.

In this paper we have analyzed the propagation of rounding errors in pipelined CG. The pipelined CG algorithm features additional auxiliary variables compared to standard CG, which are computed using extra recursions (AXPYs) and are hence all prone to rounding error accumulation. A model for rounding errors accumulation that couples all rounding error terms is derived from the recursion relations of the individual auxiliary variables. This model allows to accurately predict the influence of rounding errors on the residual in each step of the pipelined algorithm.

In preconditioned pipelined CG an additional vector, namely the preconditioned residual $u_i = M^{-1}r_i$, is also contaminated by rounding errors accumulation. However, it is argued that the accumulation of rounding errors on the preconditioned residual is much smaller than on the unpreconditioned residual, and it can thus be neglected. The incorporation of the error model in the pipelined CG method requires the calculation of two additional vector norms, requiring global communication. However, these norm computations can easily be combined with the

[5] `http://www.mpich.org/`

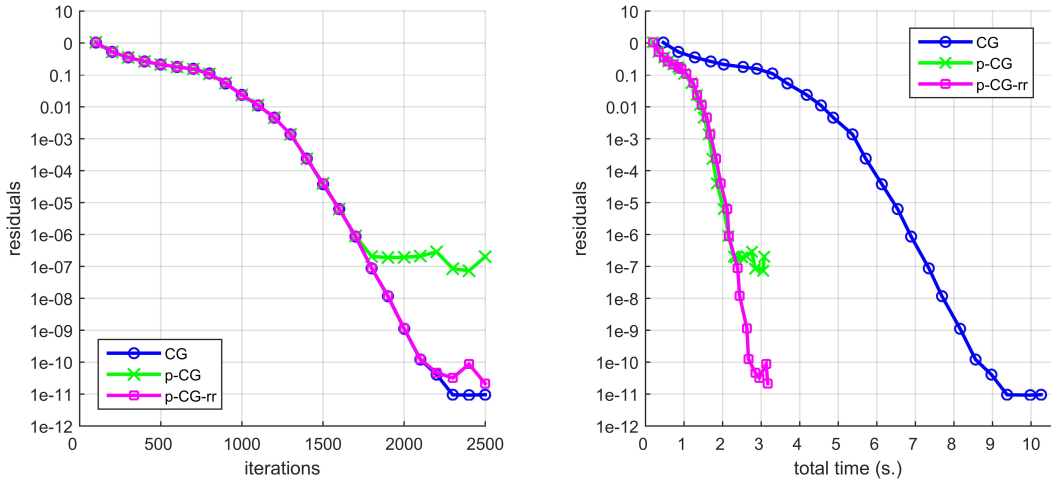


Figure 8: *Accuracy experiment performed on 12 nodes (144 cores). Left: True residual as function of iterations for the 2D Poisson problem with* 1000×1000 *unknowns. Right: True residual as function of total time spend by the algorithm. Maximal number of iterations is 2500 for all methods; the p-CG-rr algorithm performed 20 replacement steps (max.).*

existing global communication phase, such that the global performance of the algorithm remains intact.

The error propagation model is subsequently used to track the rounding error norm at run-time level. Combining the error propagation model with a residual replacement strategy, the true residual is calculated at specific times during the iteration when the accumulated rounding error becomes too large. This automated replacement strategy leads to a significantly improved solution, with a corresponding true residual norm that stagnates very close to the original CG residual norm.

The automated residual replacement strategy is validated on a variety of numerical benchmark problems, showing that the maximal attainable accuracy of pipelined CG is robustly improved by several orders of magnitude. Moreover, results using a PETSc implementation were presented to demonstrate that parallel performance is unaffected by incorporating the proposed residual replacement strategy based on the rounding error propagation model into the pipelined CG solver.

6 Acknowledgements

This work is funded by the EXA2CT European Project on Exascale Algorithms and Advanced Computational Techniques, which receives funding from the EU's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 610741. The authors thank Bram Reps and Pieter Ghysels for their insightful comments and discussions.

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. `http://www.mcs.anl.gov/petsc`, 2015.

[2] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H.A. Van der Vorst. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. *2nd ed., SIAM, Philadelphia*, 1994.

[3] 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.

[4] 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.

[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] E. De Sturler and H.A. Van der Vorst. Reducing the effect of global communication in GMRES(m) and CG on parallel distributed memory computers. *Applied Numerical Mathematics*, 18(4):441–459, 1995.

[7] J.W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[8] J.W. Demmel, M.T. Heath, and H.A. Van der Vorst. Parallel numerical linear algebra. *Acta Numerica*, 2:111–197, 1993.

[9] J. Dongarra and M.A. Heroux. Toward a new metric for ranking high performance computing systems. *Sandia National Laboratories Technical Report, SAND2013-4744*, 312, 2013.

[10] 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 Sciente Department, Technical Report UT-EECS-15-736*, 2015.

[11] P.R. Eller and W. Gropp. Non-blocking preconditioned conjugate gradient methods for extreme-scale computing. In *Conference proceedings*. 17th Copper Mountain Conference on Multigrid Methods, Colorado, US, 2015.

[12] 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.

[13] P. Ghysels and W. Vanroose. Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm. *Parallel Computing*, 40(7):224–238, 2014.

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

[15] M.H. Gutknecht and Z. Strakos. 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.

[16] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 14(6), 1952.

[17] J. Liesen and Z. Strakos. *Krylov Subspace Methods: Principles and Analysis.* Oxford University Press, 2012.

[18] C.C. Paige. Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *IMA Journal of Applied Mathematics*, 18(3):341–349, 1976.

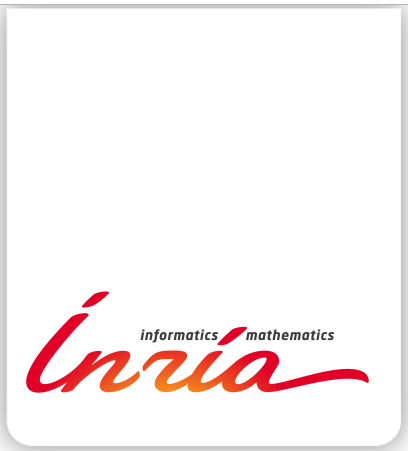
[19] Z. Strakoš and P. Tichý. On error estimation in the conjugate gradient method and why it works in finite precision computations. *Electronic Transactions on Numerical Analysis*, 13:56–80, 2002.

[20] C. Tong and Q. Ye. Analysis of the finite precision Bi-Conjugate Gradient algorithm for nonsymmetric linear systems. *Mathematics of Computation*, 69(232):1559–1575, 2000.

[21] H.A. Van der Vorst and Q. Ye. Residual replacement strategies for Krylov subspace iterative methods for the convergence of true residuals. *SIAM Journal on Scientific Computing*, 22(3):835–852, 2000.

Algorithm 5 Preconditioned pipelined CG with automated residual replacement

procedure PPIPE-CG-RR(A, M^{-1}, b, x_0)
  $r_0 := b - Ax_0$; $u_0 := M^{-1}r_0$; $w_0 := Au_0$, $\tau := \sqrt{\psi}$, **set** replace := false
  for $i = 0, \ldots$ **do**
    $\gamma_i := (r_i, u_i)$
    $\delta := (w_i, u_i)$
    if $i > 0$ **then**
      $\sigma_{i-1} := \sqrt{(s_{i-1}, s_{i-1})}$
      $\zeta_{i-1} := \sqrt{(z_{i-1}, z_{i-1})}$
    end if
    $m_i := M^{-1}w_i$
    $n_i := Am_i$
    if $i > 0$ **then**
      $\beta_i := \gamma_i/\gamma_{i-1}$; $\alpha_i := (\delta/\gamma_i - \beta_i/\alpha_{i-1})^{-1}$
    else
      $\beta_i := 0; \alpha_i = \gamma_i/\delta$
    end if
    $z_i := n_i + \beta_i z_{i-1}$
    $q_i := m_i + \beta_i q_{i-1}$
    $s_i := w_i + \beta_i s_{i-1}$
    $p_i := u_i + \beta_i p_{i-1}$
    $x_{i+1} := x_i + \alpha_i p_i$
    $r_{i+1} := r_i - \alpha_i s_i$
    $u_{i+1} := u_i - \alpha_i q_i$
    $w_{i+1} := w_i - \alpha_i z_i$
    if $i > 0$ **then**
      $e^r_{i-1} := 2\alpha_{i-1}\sigma_{i-1}\psi$
      $e^s_{i-1} := 2\beta_i\sigma_{i-1}\psi + 2\alpha_{i-1}\zeta_{i-1}\psi$
      $e^w_{i-1} := 2\alpha_{i-1}\zeta_{i-1}\psi$
      $e^z_{i-1} := 2\beta_i\zeta_{i-1}\psi$
      if $i = 1$ **or** replace = true (**set** replace := false) **then**
        $d^r_i := e^r_{i-1}$; $d^s_i := e^s_{i-1}$; $d^w_i := e^w_{i-1}$; $d^z_i := e^z_{i-1}$
      else
        $d^r_i := d^r_{i-1} + \alpha_{i-1}d^s_{i-1} + e^r_{i-1}$
        $d^s_i := \beta_i d^s_{i-1} + d^w_{i-1} + \alpha_{i-1}d^z_{i-1} + e^s_{i-1}$
        $d^w_i := d^w_{i-1} + \alpha_{i-1}d^z_{i-1} + e^w_{i-1}$
        $d^z_i := \beta_i d^z_{i-1} + e^z_{i-1}$
      end if
      if $d^r_{i-1} \leq \tau\sqrt{\gamma_{i-1}}$ **and** $d^r_i > \tau\sqrt{\gamma_i}$ (**set** replace := true) **then**
        $s_i := Ap_i$
        $q_i := M^{-1}s_i$
        $z_i := Aq_i$
        $r_{i+1} := b - Ax_{i+1}$
        $u_{i+1} := M^{-1}r_{i+1}$
        $w_{i+1} := Au_{i+1}$
      end if
    end if
  end for
end procedure



RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200, Avenue de la vieille tour
33405 Talence Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399