On rounding error resilience, maximal attainable accuracy and parallel performance of the pipelined Conjugate Gradients method for large-scale linear systems in PETSc

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

On rounding error resilience, maximal attainable accuracy and parallel performance of the pipelined Conjugate Gradients method for large-scale linear systems in PETSc*

Siegfried Cools
Applied Mathematics Group
University of Antwerp
Middelheimlaan 1, Building G
2020 Antwerp, Belgium
siegfried.cools@uantwerp.be

Wim Vanroose
Applied Mathematics Group
University of Antwerp
Middelheimlaan 1, Building G
2020 Antwerp, Belgium
wim.vanroose@uantwerp.be

Emrullah Fatih Yetkin
HiePACS
INRIA Bordeaux - Sud-Ouest
Avenue de la Vieille Tour 200
33405 Talence, France
emrullah-fatih.yetkin@inria.fr

Emmanuel Agullo
HiePACS
INRIA Bordeaux - Sud-Ouest
Avenue de la Vieille Tour 200
33405 Talence, France
emmanuel.agullo@inria.fr

Luc Giraud
HiePACS
INRIA Bordeaux - Sud-Ouest
Avenue de la Vieille Tour 200
33405 Talence, France
luc.giraud@inria.fr

ABSTRACT

Pipelined Krylov solvers typically display better strong scaling compared to standard Krylov methods for large linear systems. The synchronization bottleneck is mitigated by overlapping time-consuming global communications with computations. To achieve this hiding of communication, pipelined methods feature additional recurrence relations on auxiliary variables. This paper analyzes why rounding error effects have a significantly larger impact on the accuracy of pipelined algorithms. An algebraic model for the accumulation of rounding errors in the (pipelined) CG algorithm is derived. Furthermore, an automated residual replacement strategy is proposed to reduce the effect of rounding errors on the final solution. MPI parallel performance tests implemented in PETSc on an Intel Xeon X5660 cluster show that the pipelined CG method with automated residual replacement is more resilient to rounding errors while maintaining the efficient parallel performance obtained by pipelining.

Keywords

Pipelined Krylov methods, Conjugate Gradients, Parallelism, Latency hiding, Global communication, Rounding error resilience, Maximal attainable accuracy, PETSc.

*Extended technical report *"Analysis of rounding error accumulation in the Conjugate Gradient method to improve the maximal attainable accuracy of pipelined CG"* by Cools et al. available at http://arxiv.org/abs/1601.07068, see [6].

EASC '16, April 26-29, 2016, Stockholm, Sweden

DOI: http://dx.doi.org/10.1145/2938615.2938621

1. INTRODUCTION

Krylov subspace methods form the basis linear algebra solvers for many contemporary high-performance computing applications. The Conjugate Gradient (CG) method [14] 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 computing era in the coming years, research on the scalability of Krylov methods on massively parallel architectures is becoming 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 in the computation of dot-products.

Significant research has recently been devoted to the reduction and 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 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. [7], Demmel et al. [8] and Ghysels et al. [10].

The pipelined CG (p-CG) method proposed in [11] aims at hiding the global synchronization latency of standard pre-

Algorithm 1 Preconditioned CG

function CG(A, M^{-1}, b, x_0)
  $r_0 := b - Ax_0$; $u_0 := M^{-1}r_0$; $p_0 = u_0$
  for $i = 0, \ldots$ **do**
    $s_i := Ap_i$
    $\alpha_i := (r_i, u_i) \,/\, (s_i, p_i)$
    $x_{i+1} := x_i + \alpha_i p_i$
    $r_{i+1} := r_i - \alpha_i s_i$
    $u_{i+1} := M^{-1} r_{i+1}$
    $\beta_{i+1} := (r_{i+1}, u_{i+1}) \,/\, (r_i, u_i)$
    $p_{i+1} := u_{i+1} + \beta_{i+1} p_i$
  end for
end function

conditioned 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. [9].

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. In exact arithmetic the resulting pipelined CG algorithm is numerically equivalent to standard CG. However, when switching to finite precision floating-point representation, each of the additional recurrences accumulate rounding errors, and the extra recursions reduce the asymptotic accuracy of the solution.

In this paper we propose an analysis of the rounding errors that propagate through the CG and p-CG iterations. The rounding error model derived in this work allows to predict the estimated rounding error at run-time level. It is shown that the pipelined CG method is much more sensitive to rounding error propagation compared to the classical CG algorithm. Consequently, countermeasures are proposed in the form of a residual replacement strategy, based on the error model. The resulting pipelined CG algorithm with automated residual replacement (p-CG-rr) is able to achieve a precision that is comparable to standard CG, while displaying the good parallel scalability of pipelined CG.

The remainder of this 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. An error model that allows to predict the residual stagnation point for the different methods is introduced. Furthermore, we propose the incorporation of a residual replacement strategy in the pipelined CG method. A criterion for automated residual replacement is suggested near the end of this section. Extensive numerical tests are reported in Section 3. These experiments show the validity of the error model and the improvement in maximal attainable accuracy for the pipelined CG method. We illustrate that the proposed residual replacement strategy does not affect the parallel scalability of the p-CG method. Indeed, it is demonstrated that the pipelined CG algorithm with automated residual replacement is both accurate and computationally efficient. Finally, conclusions are formulated in Section 4.

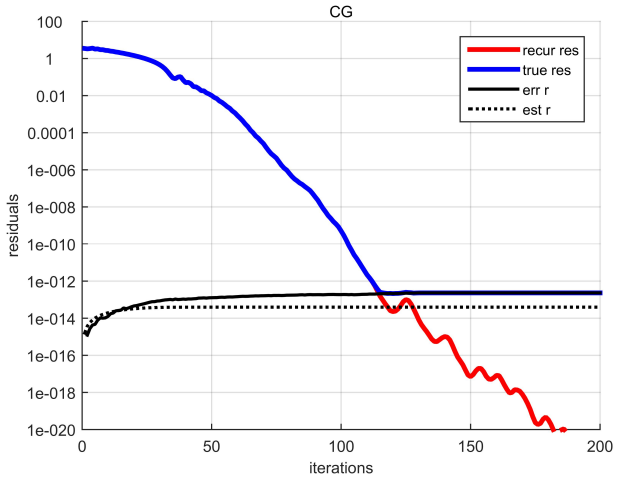

Figure 1: *Propagation of rounding errors in Conjugate Gradients. Residual rounding error* $\|r_i - (b - Ax_i)\|$ *(solid black) and estimate* $\|\Delta_i^r\|$ *computed using* (6) *(dotted black).*

2. ROUNDING ERROR PROPAGATION IN CG, CG-CG AND PIPELINED CG

The recursive residual r calculated by the Conjugate Gradient algorithm typically starts to deviate from the true residual $b - Ax$ at some point during the iteration. This is caused by the fact that the true residual is never explicitly calculated by the algorithm. Instead, the residual is computed using a recurrence relation to avoid the additional SPMV required to compute $b - Ax$ in each iteration. However, in finite precision arithmetic, rounding errors occur in each of these recurrence relations (AXPYs) and accumulate over the iteration, resulting in a significant difference between true and recursive residual near the end of the algorithm.

2.1 Resilience: CG vs. pipelined CG

We demonstrate the deviation between the true and recursive residual in both the CG and pipelined CG method on a simple 2D Poisson model problem on the domain $\Omega = [0, 1]^2$ with homogeneous Dirichlet boundary conditions. The equation is discretized using second order finite differences on a uniform grid with $n = 50$ grid points per spatial dimension. As illustrated in Fig. 1 the recursive residual computed by the CG method, Algorithm 1, keeps decreasing, while the true residual $r = b - Ax$ stagnates upon reaching the lower bound $\|A\|\psi$, where ψ denotes the machine precision. In the pipelined CG method, Algorithm 3, an additional number of auxiliary vectors defined as $s := Ap$, $w := Ar$ and $z := As = A^2 p$ 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. Fig. 3 shows the residuals corresponding to pipelined CG. 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 [11], this loss of maximal attainable accuracy is typical for pipelined CG.

Algorithm 2 Preconditioned Chronopoulos/Gear CG

function CGCG(A, M^{-1}, b, x_0)
  $r_0 := b - Ax_0$; $u_0 := M^{-1}r_0$; $w_0 := Au_0$
  $\alpha_0 := (r_0, u_0)/(w_0, u_0)$; $\beta := 0$; $\gamma_0 := (r_0, u_0)$
  for $i = 0, \ldots$ **do**
    $p_i := u_i + \beta_i p_{i-1}$
    $s_i := w_i + \beta_i s_{i-1}$
    $x_{i+1} := x_i + \alpha_i p_i$
    $r_{i+1} := r_i - \alpha_i s_i$
    $u_{i+1} := M^{-1} r_{i+1}$
    $w_{i+1} := Au_{i+1}$
    $\gamma_{i+1} := (r_{i+1}, u_{i+1})$
    $\delta := (w_{i+1}, u_{i+1})$
    $\beta_{i+1} := \gamma_{i+1}/\gamma_i$
    $\alpha_{i+1} := (\delta/\gamma_{i+1} - \beta_{i+1}/\alpha_i)^{-1}$
  end for
end function

2.2 Rounding error accumulation in CG

The approach to analyzing the propagation of rounding errors in CG and related algorithms has been discussed in Greenbaum [12], Gutknecht et al. [13], Strakos et al. [15], Vandervorst et al. [17] and Tong et al. [16]. In traditional preconditioned Conjugate Gradients, Algorithm 1, the solution x_{i+1} and residual r_{i+1} are updated respectively as

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

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 denote 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

$$\begin{aligned} \Delta_{i+1}^r &:= r_{i+1} - (b - Ax_{i+1}) \\ &= r_i - (b - Ax_i) - \alpha_i(Ap_i) + A(\alpha_i p_i) + A\delta_i^x + \delta_i^r \\ &= \Delta_i^r + \underbrace{A(\alpha_i p_i) - \alpha_i(Ap_i)}_{\text{non-commutativity error}} + \underbrace{A\delta_i^x + \delta_i^r}_{\text{local rounding error}} \end{aligned} \tag{2}$$

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 stems from the fact that $\alpha_i(Ap_i) \neq A(\alpha_i p_i)$ in finite precision floating point arithmetic.[1] The second term is due to the local rounding errors of the recurrence relations (1) in the i-th iteration.

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 clear from Fig. 1.

[1] This error is denoted as the 'non-commutativity' error by the authors, since it boils down to stating that in finite precision $\alpha_i A p_i \neq A\alpha_i p_i$, and the product $\alpha_i A$ hence appears to be non-commutative. However, more precisely, the origin of this error is the fact that scalar associativity $\alpha_i(Ap_i) = (\alpha_i A)p_i = A(\alpha_i p_i) = (Ap_i)\alpha_i$ does not hold in finite precision floating point arithmetic.

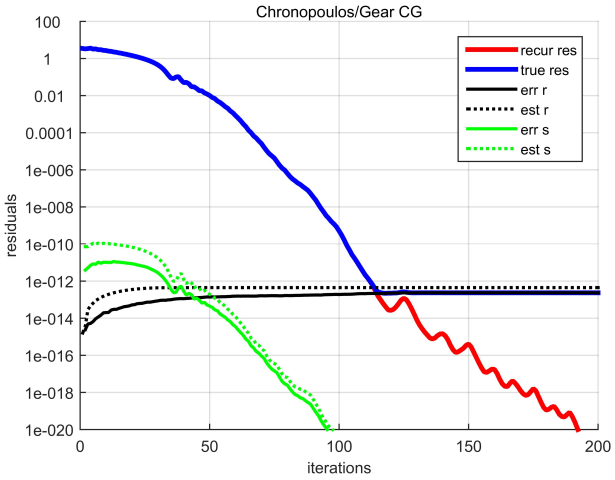

Figure 2: *Propagation of rounding errors in Chronopoulos/Gear CG. Residual rounding error* $\|r_i - (b - Ax_i)\|$ *(solid black) and auxiliary rounding error* $\|s_i - Ap_i\|$ *(solid green), and their estimates* $\|\Delta_i^r\|$ *and* $\|\Delta_i^s\|$*, see* (11) *(dotted).*

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

$$\|\Delta_{i+1}^r\| = \|\Delta_i^r\| + \|A(\alpha_i p_i) - \alpha_i(Ap_i)\|. \tag{3}$$

The norm $\|\cdot\|$ indicates the standard 2-norm throughout this work. The error model (3) 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. 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 estimate the non-commutativity error in (3) using the following lemma.

Lemma 1. *The non-commutativity error in rounding error model* (3) *can be bounded from above by*

$$\|A(\alpha_i p_i) - \alpha_i(Ap_i)\| \leq 2(m+1)\alpha_i\|s_i\|\psi, \tag{4}$$

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

Proof. We refer the reader to the full manuscript [6] for the proof of this lemma.

Lemma 1 provides an absolute upper bound on the non-commutativity error. Indeed, it bounds the largest possible value that could be computed without incorporating error cancellation caused by adding positive and negative numbers. In practice, however, rounding error cancellation does occur, such that the factor $(m+1)$ in Lemma 1 is dropped. Hence, the worst-case upper bound (4) often largely overestimates the actual non-commutativity error. We propose to use the following, more realistic estimate for the error

$$\|A(\alpha_i p_i) - \alpha_i(Ap_i)\| \approx 2\alpha_i\|s_i\|\psi. \tag{5}$$

Algorithm 3 Pipelined Chronopoulos/Gear CG

1: **function** P-CG(A, b, x_0)
2: $r_0 := b - Ax_0$; $w_0 := Ar_0$
3: **for** $i = 0, \ldots$ **do**
4: $\gamma_i := (r_i, r_i)$
5: $\delta := (w_i, r_i)$
6: $q_i := Aw_i$
7: **if** $i > 0$ **then**
8: $\beta_i := \gamma_i/\gamma_{i-1}$; $\alpha_i := (\delta/\gamma_i - \beta_i/\alpha_{i-1})^{-1}$
9: **else**
10: $\beta_i := 0$; $\alpha := \gamma_i/\delta$
11: **end if**
12: $z_i := q_i + \beta_i z_{i-1}$
13: $s_i := w_i + \beta_i s_{i-1}$
14: $p_i := r_i + \beta_i p_{i-1}$
15: $x_{i+1} := x_i + \alpha_i p_i$
16: $r_{i+1} := r_i - \alpha_i s_i$
17: $w_{i+1} := w_i - \alpha_i z_i$
18: **end for**
19: **end function**

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

$$\|\Delta^r_{i+1}\| = \|\Delta^r_i\| + 2\alpha_i\|s_i\|\psi. \tag{6}$$

Note that the incorporation of the error estimate (6) 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.

2.3 Rounding error accumulation in CG-CG

The Chronopoulos/Gear CG (CG-CG) 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 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

$$\begin{cases} p_i = r_i + \beta_i p_{i-1}, \\ s_i = w_i + \beta_i s_{i-1}, \end{cases} \qquad \begin{cases} x_{i+1} = x_i + \alpha_i p_i, \\ r_{i+1} = r_i - \alpha_i s_i. \end{cases} \tag{7}$$

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

$$\begin{aligned} \Delta^r_{i+1} &:= r_{i+1} - (b - Ax_{i+1}) \\ &= r_i - (b - Ax_i) - \alpha_i(s_i - Ap_i) + A\alpha_i p_i - \alpha_i Ap_i \\ &= \Delta^r_i - \alpha_i\Delta^s_i + A\alpha_i p_i - \alpha_i Ap_i, \end{aligned} \tag{8}$$

where again the local rounding errors are neglected to simplify the notation. Note that in Algorithm 2 s_i, defined as

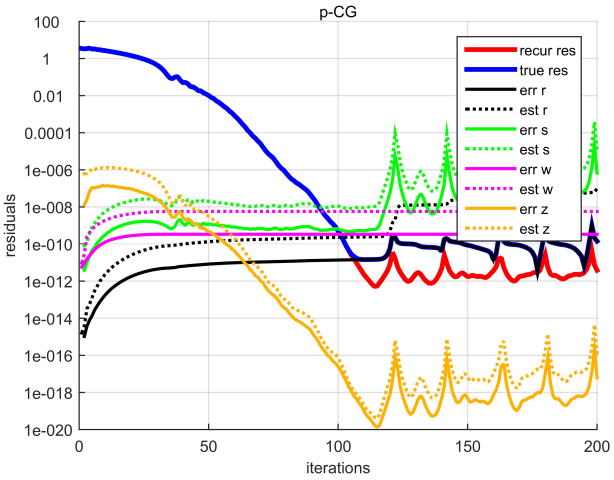

Figure 3: *Propagation of rounding errors in pipelined CG. Residual rounding error* $\|r_i - (b - Ax_i)\|$ *(solid black) and auxiliary rounding errors* $\|s_i - Ap_i\|$ *(solid green),* $\|w_i - Ar_i\|$ *(solid magenta), and* $\|z_i - As_i\|$ *(solid yellow). Error estimates* $\|\Delta^r_i\|$, $\|\Delta^s_i\|$, $\|\Delta^w_i\|$ *and* $\|\Delta^z_i\|$ *(dotted) are computed using* (14) *and* (16).

Ap_i, is also not explicitly calculated but rather derived from a recurrence relation. Hence, like the error on the residual, the rounding error $\Delta^s_i = s_i - Ap_i$ also grows in function of i. We find in a similar way to the above

$$\begin{aligned} \Delta^s_{i+1} &:= s_{i+1} - Ap_{i+1} \\ &= w_{i+1} + \beta_{i+1}s_i - A(r_{i+1} + \beta_{i+1}p_i) \\ &= \beta_{i+1}\Delta^s_i - A\beta_{i+1}p_i + \beta_{i+1}Ap_i, \end{aligned} \tag{9}$$

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

Combining (8) and (9) we obtain the following rounding error propagation model for Chronopoulos/Gear CG:

$$\begin{pmatrix} \|\Delta^r_{i+1}\| \\ \|\Delta^s_{i+1}\| \end{pmatrix} = \begin{pmatrix} 1 & \alpha_i \\ 0 & \beta_{i+1} \end{pmatrix} \begin{pmatrix} \|\Delta^r_i\| \\ \|\Delta^s_i\| \end{pmatrix} + \begin{pmatrix} \|\alpha_i Ap_i - A\alpha_i p_i\| \\ \|\beta_{i+1}Ap_i - A\beta_{i+1}p_i\| \end{pmatrix} \tag{10}$$

Note that the scalars α_i and β_i are positive per definition. The above error propagation model is the analog of the standard CG model (3). 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 (3), the model (10) in fact simulates a worst-case scenario for rounding error propagation, due to the summation of the right-hand side norms.

In analogy to (6) and by applying Lemma 1 the propagation of rounding errors in CG-CG can now be estimated by approximating the non-commutativity errors in (10) as

$$\begin{pmatrix} \|\Delta^r_{i+1}\| \\ \|\Delta^s_{i+1}\| \end{pmatrix} = \begin{pmatrix} 1 & \alpha_i \\ 0 & \beta_{i+1} \end{pmatrix} \begin{pmatrix} \|\Delta^r_i\| \\ \|\Delta^s_i\| \end{pmatrix} + \begin{pmatrix} 2\alpha_i\|s_i\|\psi \\ 2\beta_{i+1}\|s_i\|\psi \end{pmatrix}. \tag{11}$$

The effective and estimated rounding errors for the CG-CG method are illustrated in Fig. 2.

2.4 Rounding error accumulation in p-CG

A similar rounding error analysis can be performed for pipelined Krylov algorithms. We discuss the pipelined CG (p-CG) method 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:

$$\begin{cases} z_i = q_i + \beta_i z_{i-1}, \\ s_i = w_i + \beta_i s_{i-1}, \\ p_i = r_i + \beta_i p_{i-1}, \end{cases} \qquad \begin{cases} x_{i+1} = x_i + \alpha_i p_i, \\ r_{i+1} = r_i - \alpha_i s_i, \\ w_{i+1} = w_i - \alpha_i z_i. \end{cases} \tag{12}$$

Lemma 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*

$$\begin{cases} \Delta_i^r := r_i - (b - Ax_i), \\ \Delta_i^s := s_i - Ap_i, \\ \Delta_i^w := w_i - Ar_i, \\ \Delta_i^z := z_i - As_i. \end{cases} \tag{13}$$

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

$$\begin{pmatrix} \|\Delta_{i+1}^r\| \\ \|\Delta_{i+1}^s\| \\ \|\Delta_{i+1}^w\| \\ \|\Delta_{i+1}^z\| \end{pmatrix} = \begin{pmatrix} 1 & \alpha_i & 0 & 0 \\ 0 & \beta_{i+1} & 1 & \alpha_i \\ 0 & 0 & 1 & \alpha_i \\ 0 & 0 & 0 & \beta_{i+1} \end{pmatrix} \begin{pmatrix} \|\Delta_i^r\| \\ \|\Delta_i^s\| \\ \|\Delta_i^w\| \\ \|\Delta_i^z\| \end{pmatrix} + \begin{pmatrix} e_i^r \\ e_i^s \\ e_i^w \\ e_i^z \end{pmatrix}, \tag{14}$$

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

$$\begin{pmatrix} e_i^r \\ e_i^s \\ e_i^w \\ e_i^z \end{pmatrix} := \begin{pmatrix} \|\alpha_i Ap_i - A\alpha_i p_i\| \\ \|\beta_{i+1} Ap_i - A\beta_{i+1} p_i\| + \|\alpha_i As_i - A\alpha_i s_i\| \\ \|\alpha_i As_i - A\alpha_i s_i\| \\ \|\beta_{i+1} As_i - A\beta_{i+1} s_i\| \end{pmatrix}. \tag{15}$$

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

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

Note that s_i is not explicitly calculated but rather derived recursively. The error Δ_i^s is given by

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

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

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

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

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

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

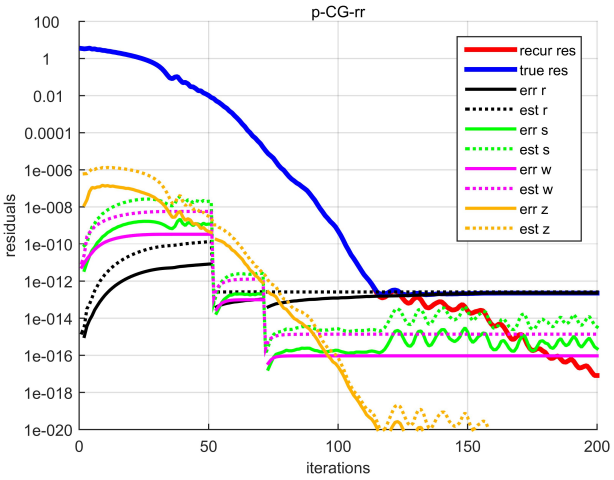

Figure 4: *Propagation of rounding errors for pipelined CG with residual replacement. Residual rounding error $\|r_i - (b - Ax_i)\|$ (solid black) and auxiliary rounding errors $\|s_i - Ap_i\|$ (solid green), $\|w_i - Ar_i\|$ (solid magenta), and $\|z_i - As_i\|$ (solid yellow). Error estimates $\|\Delta_i^r\|$, $\|\Delta_i^s\|$, $\|\Delta_i^w\|$ and $\|\Delta_i^z\|$ (dotted) are computed using* (14) *and* (16).

To incorporate the rounding error model into the pipelined CG Algorithm 3, we again propose to estimate the non-commutativity errors, (15), as follows

$$\begin{pmatrix} e_i^r \\ e_i^s \\ e_i^w \\ e_i^z \end{pmatrix} \approx \begin{pmatrix} 2\alpha_i \|s_i\| \psi \\ 2\beta_{i+1} \|s_i\| \psi + 2\alpha_i \|z_i\| \psi \\ 2\alpha_i \|z_i\| \psi \\ 2\beta_{i+1} \|z_i\| \psi \end{pmatrix}, \tag{16}$$

which only requires the vector norms $\|s_i\|$ and $\|z_i\|$ to be computed. These computations are computationally inexpensive but require additional global communication. However, they 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. Fig. 3 shows the propagated rounding errors for pipelined CG. The estimated rounding errors are computed by substituting (16) into the model (14). The error prediction model (14) allows for an accurate prediction of the stagnation point. The additional rounding errors by the recursions for the three auxiliary variables s_i, w_i and z_i cause the pipelined residual to stagnate several orders of magnitude above the standard CG residual.

Note that a very similar error analysis can be performed for the preconditioned pipelined CG method, which has been omitted here for brevity, cf. the full manuscript [6],

2.5 Pipelined CG with residual replacements

In this section we suggest a residual replacement strategy for pipelined CG methods to improve the maximal attainable accuracy of Algorithm 3 as a countermeasure to reduce the propagation of rounding errors. The main idea in using residual replacement 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.

$$\begin{cases} s_i = Ap_i, \\ z_i = As_i, \\ r_{i+1} = b - Ax_{i+1}, \\ w_{i+1} = Ar_{i+1}. \end{cases} \tag{17}$$

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.

One could inquire if such a drastic replacement strategy does not destroy the established Krylov convergence. A key result from [16] states that if r_i satisfies the typical perturbed Lanczos relation, then

$$\|r_{i+1}\| \leq C_i \min_{p \in \mathcal{P}_i, p(0)=1} \|p(A + \Delta A_i) r_1\|, \tag{18}$$

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 stable. Based on the bound (18), Van der Vorst et al. [17] 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 replacement step. Performing replacements when $\|r_i\|$ is small is generally not recommended.

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 SPMVs, an accurate criterion to determine the need for residual replacement that does not overestimate the total number of replacements is essential. As the iteration proceeds, $\|\Delta_i^r\|$ typically increases, while the residual norm $\|r_i\|$ is expected to decrease, see Fig. 3. Residual replacement should be carried out before $\|\Delta_i^r\|$ becomes too large relative to $\|r_i\|$. Hence, in analogy to [17], we use a threshold τ, typically chosen as $\tau = \sqrt{\psi}$, and introduce a residual replacement in step i of the pipelined CG Algorithm 3 if

$$\|\Delta_{i-1}^r\| \leq \tau \|r_{i-1}\| \quad \text{and} \quad \|\Delta_i^r\| > \tau \|r_i\|. \tag{19}$$

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 replacement steps are performed, keeping the total cost of the algorithm as low as possible. The rounding error model (14) allows for a direct implementation of the replacement criterion (19) in Algorithm 3.

Fig. 4 is the analog of Fig. 3 with the incorporation of the above residual replacement strategy in the pipelined CG Algorithm 4 The replacement criterion (19) is implemented using the estimated rounding errors given by (14). Proper convergence of the computed residuals is maintained after a residual replacement event. Indeed, up to the stagnation point, the residuals shown in Fig. 4 are nearly indistinguishable from the residuals of standard pipelined CG in Fig. 3. The p-CG-rr Algorithm 4 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.

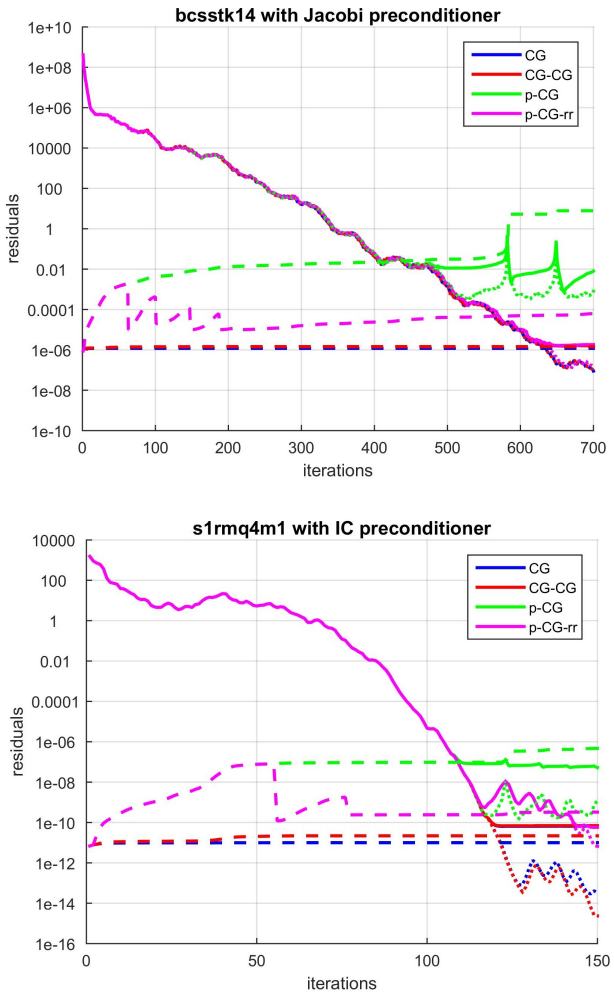

Figure 5: *Convergence history for the CG methods applied to different SPD test matrices (see also Table 1). Solid lines: true residual norm $\|b - Ax_i\|$; dotted lines: residual error $\|\Delta_i^r\|$; dashed lines: recursive residual norm $\|r_i\|$. The p-CG method (green) suffers from rounding error accumulation. The residual replacement strategy applied in the p-CG-rr method (magenta) reduces the accumulation of rounding errors.*

3. NUMERICAL RESULTS

Numerical results on a wide range of matrices are presented to compare the residual history of the different CG methods and show the improvement in attainable accuracy by using the residual replacement strategy. The parallel performance measurements in this section result from a PETSc implementation of pipelined CG with automated residual replacements on a distributed memory machine using the message passing paradigm (MPI). The MPI library used for these experiments is MPICH-3.1.3[2].

3.1 Attainable accuracy

We present numerical results on a range of different linear systems to show the robustness of the proposed residual replacement strategy. Table 1 lists a broad selection of square, real and symmetric positive definite matrices from

[2]http://www.mpich.org/

Matrix	Prec	$\kappa(A)$	N	$\#nnz$	$\|\|r_0\|\|$	CG		CG-CG		p-CG		p-CG-rr		
						iter	res	iter	res	iter	res	iter	res	rr
bcsstk14	JAC	1.3e+10	1806	63k	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	118k	4.3e+08	758	1.7e-06	743	1.9e-06	575	8.7e-02	793	1.1e-05	8
bcsstk16	JAC	65	4884	290k	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	429k	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	149k	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	56k	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	0.8k	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	0.5k	4.6e+06	465	3.8e-09	484	9.9e-08	272	2.5e-01	627	4.3e-06	24
bcsstm22	-	9.4e+02	138	0.1k	3.1e-03	68	2.1e-18	67	3.8e-18	65	1.6e-14	71	7.2e-18	2
bcsstm24	-	1.8e+13	3562	3.6k	1.1e+05	-	1.0e-07	-	1.1e-05	1301	3.0e-03	2445	4.6e-04	39
bcsstm26	-	2.6e+05	1922	1.9k	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	7.7k	1.1e+00	54	3.7e-15	53	7.9e-15	49	6.8e-13	54	6.7e-15	2
nos1	*IC	2.5e+07	237	1.0k	5.7e+07	342	8.4e-07	334	8.0e-07	270	1.9e-01	631	1.7e-06	9
nos2	*IC	6.3e+09	957	4.1k	1.8e+09	3501	1.5e-04	3683	1.9e-04	1923	1.1e+04	3867	2.3e-01	56
nos3	IC	7.3e+04	960	16k	1.0e+01	63	1.0e-13	62	1.1e-13	56	2.3e-11	68	9.2e-14	2
nos4	IC	2.7e+03	100	0.6k	5.2e-02	31	9.4e-17	30	1.2e-16	29	2.6e-15	30	1.3e-16	2
nos5	IC	2.9e+04	468	5.1k	2.8e+05	60	1.2e-10	60	1.2e-10	56	1.4e-08	59	5.4e-10	2
nos6	IC	8.0e+06	675	3.3k	8.6e+04	40	4.2e-10	34	5.7e-10	28	4.3e-06	47	5.3e-10	2
s1rmq4m1	IC	1.8e+06	5489	262k	1.5e+04	122	6.5e-11	121	7.1e-11	110	9.6e-08	142	2.9e-10	3
s1rmt3m1	IC	2.5e+06	5489	218k	1.5e+04	227	1.3e-10	225	1.4e-10	204	6.2e-07	258	3.2e-10	3
s2rmq4m1	*IC	1.8e+08	5489	263k	1.5e+03	366	1.0e-11	362	1.2e-11	309	1.2e-06	449	4.3e-10	8
s2rmt3m1	IC	2.5e+08	5489	218k	1.5e+03	273	1.3e-11	270	1.6e-11	240	2.1e-06	363	3.8e-11	6
s3dkq4m2	*IC	1.9e+11	90,449	2456k	6.8e+01	-	1.3e-06	-	1.4e-06	2658	3.6e-06	2460	9.2e-06	37
s3dkt3m2	*IC	3.6e+11	90,449	1922k	6.8e+01	-	2.0e-05	-	2.0e-05	3409	1.3e-05	3370	1.5e-05	36

Table 1: *A collection of 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 using Algorithms 1-4. The initial guess is all-zero $x_0 = 0$. Jacobi (JAC) and Incomplete Cholesky (IC) preconditioners are included where needed. Number of iterations iter required to reach the estimated rounding error on the residual and corresponding true residual $res = ||b - Ax_i||$ are given. For the p-CG-rr method the table indicates the number of replacement steps.*

Matrix Market[3], with their respective condition number κ, number of rows N and total number of nonzero elements $\#nnz$. A linear system with exact solution $\hat{x}_j = 1/\sqrt{N}$ and right-hand side $b = A\hat{x}$ is solved for each of these matrices with the four presented methods, using an all-zero initial guess $x_0 = 0$. Jacobi diagonal preconditioning (JAC) and Incomplete Cholesky factorization (IC) are included to reduce the number of Krylov iterations if possible. *IC indicates that a compensated Incomplete Cholesky factorization is performed.

The accuracy experiments in this section were performed on a mid-range laptop computer with Intel Core i7-2720QM 2.20GHz CPU. Since we only focus on attainable accuracy and robustness of the methods in this section, no MPI was used in the following experiment (single-node execution). We refer to section 3.2 for multi-core strong scaling results.

Table 1 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.

Fig. 5 illustrates the convergence history for two selected matrices from Table 1 by showing the true residual (solid), recursive residual (dashed) and residual rounding error (dotted). Convergence of the CG and Chronopoulos/Gear CG methods is nearly indistinguishable. The accumulation of rounding errors in the p-CG algorithm causes the residuals to level off sooner compared to CG and CG-CG, resulting in a less accurate final solution. Based on the estimated rounding errors and criterion (19), the replacement strategy resets the rounding errors in certain iterations, leading to a much more accurate final solution. The final accuracy on the p-CG-rr solution is similar to that of standard CG.

At first glance, Table 1 appears to suggest that the number of p-CG iterations is generally lower than for the standard CG method. However, it should be noted that the p-CG execution is often halted at an early stage, when additional iterations do no longer reduce the residual norm. As indicated above, the p-CG method is generally unable to attain the solution accuracy of classical CG, regardless of the iteration count. The p-CG-rr method, on the other hand, typically takes at least as many iterations as standard CG (and sometimes slightly more) to obtain a comparable accuracy, see Table 1. This stems from the fact that slight deviations of the p-CG-rr residual norms from the standard CG residual history can occasionally be observed close to the stagnation point, cf. Fig. 5, depending on the problem solved. The observed deviation is an artifact from the irregularities in the coefficients α_i and β_i near stagnation, see [6] for a more elaborate discussion. The proposed p-CG-rr method is hence significantly more robust than p-CG, reaching the standard CG accuracy in a comparable or slightly larger number of iterations.

3.2 Parallel performance

The pipelined CG method was presented in [11] 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 3, is maintained by the addition of the automated replacement strategy in Algorithm 4.

The following parallel strong scaling experiments are performed on a small cluster with 12 compute nodes, consisting

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

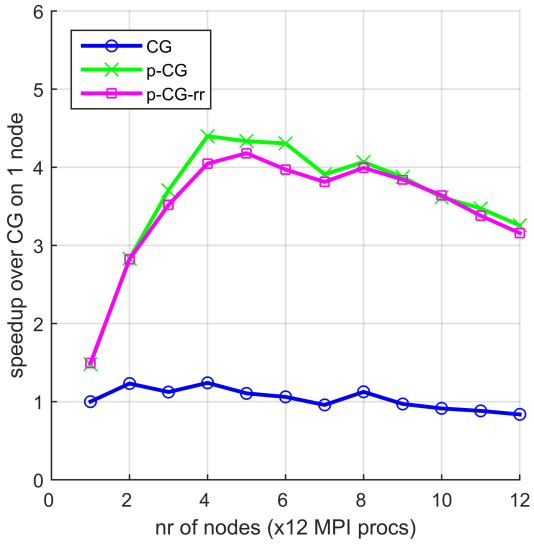

Figure 6: *Strong scaling test on up to 12 nodes (144 cores). Speedup as function of the number of nodes over standard CG solver on a single node. All methods converged in 900 iterations to a relative residual tolerance of* 10^{-4}*. The p-CG-rr algorithm performed 7 replacement steps.*

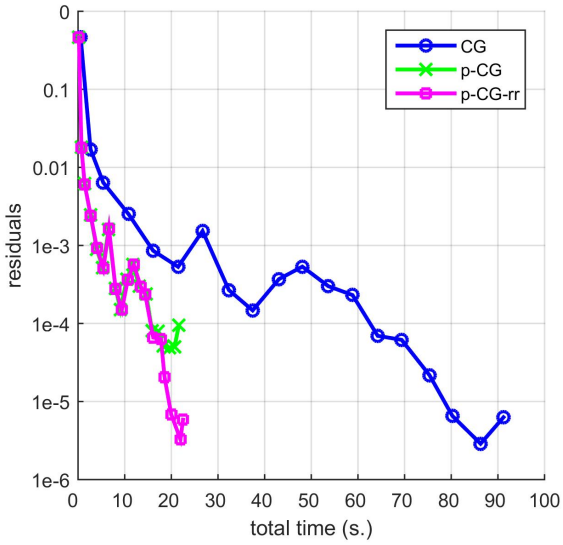

Figure 7: *Accuracy test performed on 12 nodes (144 cores). True residual as function of total time spend by the algorithm. Maximal number of iterations is 17,000 for all methods; the p-CG-rr algorithm performed 147 replacement steps (max.).*

of two 6-core Intel Xeon X5660 Nehalem 2.80 GHz processors each (12 cores per node), for a total of 144 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] v.3.6.3, which includes implementations of the CG and p-CG algorithms. The p-CG-rr Algorithm 4 was implemented as a direct extension of the p-CG method, and will be incorporated in the next PETSc release. The first benchmark problem used to asses parallel performance in this section is the small-sized, ill-conditioned `s3dkq4m2` Matrix Market system (90,449 unknowns). A simple Jacobi preconditioner is applied in this experiment. The relative tolerance for Krylov solution imposed on the standard recursive residual norm $\|r_i\|_2$ is set to 10^{-4}. Since each node consists of 12 cores, we use 12 MPI processes per node to fully exploit parallelism on the machine. The MPICH environment variables

- `MPICH_ASYNC_PROGRESS=1`
- `MPICH_MAX_THREAD_SAFETY=multiple`

are set to ensure optimal parallelism.

Fig. 6 shows the time to solution as a function of the number of nodes for the `s3dkq4m2` problem. For the given benchmark problem the maximum speed-up for p-CG compared to CG on a single node is 4.40, whereas the CG method achieves virtually no speedup on multiple nodes. Pipelined CG attains a net speedup of 3.89 compared to standard CG when both are executed on 12 nodes. The p-CG-rr method shows very similar performance results compared to p-CG. A minor discrepancy between the p-CG and p-CG-rr speedups is observed due to the additional computational work for the extra SPMVs when replacement takes place. The maximum speedup for p-CG-rr compared to CG on a single node is 4.18, yielding a total net speedup factor of 3.78 over standard CG on 12 nodes.

Fig. 7 shows the accuracy of the solution in function of the computational time spend by the three algorithms for the `s3dkq4m2` benchmark problem on a 12 node setup. The initial residual is 6.8e+1. In 20.0 seconds, corresponding to 15,000 iterations (incl. 147 replacement steps), the p-CG-rr algorithm obtains an accurate solution with a true residual norm of 6.8e-6. Standard CG needs 80.2 s. to attain a comparable accuracy on the solution (true residual norm 6.5e-6), performing 15,000 iterations on the same 12 nodes. This is roughly four times slower than the time required by p-CG-rr to obtain a similar precision, see also Fig. 6. 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 5.2e-5 is imminent from 14,000 iterations or, equivalently, a total time of 18.2 s. onward.

As a second test-case for parallel performance we consider a medium-sized 2D Poisson model, 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 residual tolerance for Krylov solution is 10^{-6}. The following strong scaling experiments were performed on up to 20 nodes.

Fig. 8 shows the time to solution as a function of the number of nodes for the 2D Poisson problem. The maximum speed-up for p-CG on 20 nodes compared to CG on a single node is 7.28. Pipelined CG attains a net speedup of 3.79 compared to standard CG on 20 nodes. The maximum speedup for p-CG-rr on 20 nodes compared to CG on a single node is 7.06, yielding a total net speedup factor of 3.66 compared to standard CG on 20 nodes. In comparison to the `s3dkq4m2` benchmark problem, the pipelined methods scale very well on up to 20 nodes (240 MPI threads) for the 2D Poisson benchmark due to the larger size of the problem.

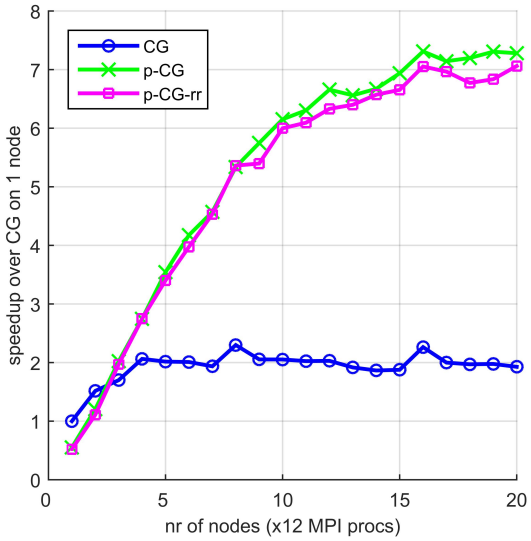

Figure 8: *Strong scaling test on up to 20 nodes (240 cores). Speedup as function of the number of nodes over standard CG solver on a single node. All methods converged in 1474 iterations to a relative residual tolerance of* 10^{-6}. *The p-CG-rr algorithm performed 11 replacement steps.*

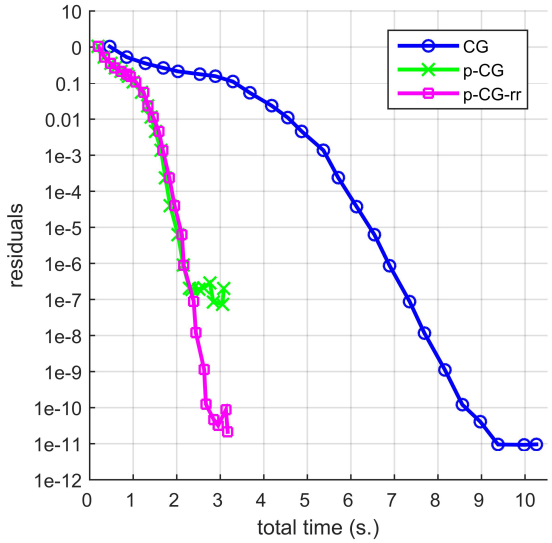

Figure 9: *Accuracy test performed on 12 nodes (144 cores). True residual as function of total time spend by the algorithm. Maximal number of iterations is 2500 for all methods; the p-CG-rr algorithm performed 20 replacement steps (max.).*

Fig. 9 shows the solution accuracy for the 2D Poisson benchmark problem on a 12 node setup. In 3.17 seconds (2500 iterations) the p-CG-rr algorithm attains a true residual norm of 2.12e-11. Standard CG needs 9.38 s. (2300 iterations) to attain a comparable accuracy on the same 12 nodes. The original pipelined CG true residual norm stagnates around 1.0e-7 from 1700 iterations (2.15 s.) onward.

4. CONCLUSIONS

The deviation of the recursive residual from the true residual around machine precision level is a well-known aspect of the Conjugate Gradients method. Whereas the true residual stagnates due to the accumulation of rounding errors on the solution, the recurred residual typically keeps decreasing. This phenomenon is significantly more prominent in the pipelined Chronopoulos & 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 analyzed the propagation of rounding errors in pipelined CG. The p-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 predict the influence of rounding errors on the residual in each step of the pipelined algorithm.

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 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 runtime 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.

Scaling results using an MPI-based PETSc implementation were presented to demonstrate the resilience of the novel p-CG-rr algorithm to rounding error accumulation, while it is illustrated that parallel performance is unaffected by incorporating the replacement strategy into the p-CG solver.

5. ACKNOWLEDGMENTS

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 no. 610741.

6. 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] S. Cools, E.F. Yetkin, E. Agullo, L. Giraud, and W. Vanroose. Analysis of rounding error accumulation in the Conjugate Gradient method to improve the maximal attainable accuracy of pipelined CG. *Technical report - preprint available at* `http://arxiv.org/abs/1601.07068`, 2016.

[7] 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.

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

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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 4 Preconditioned pipelined CG with automated residual replacement

function P-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 **then**
        $d^r_i := e^r_{i-1}$
        $d^s_i := e^s_{i-1}$
        $d^w_i := e^w_{i-1}$
        $d^z_i := e^z_{i-1}$
        set replace := false
      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** $d^r_i > \tau\sqrt{\gamma_i}$ **then**
        $s_i := Ap_i$
        $q_i := M^{-1}s_i$
        $z_i := Aq_i$
        $r_{i+1} := b - Ax_{i+1}$
        $u_{i+1} := M^{-1}r_{i+1}$
        $w_{i+1} := Au_{i+1}$
        set replace := true
      end if
    end if
  end for
end function