AN ADAPTIVE MIXED-PRECISION AND DYNAMICALLY SCALED PRECONDITIONED CONJUGATE GRADIENT ALGORITHM *

YICHEN GUO[†], ERIC DE STURLER[†], AND TIM WARBURTON[†]

Abstract. We propose an adaptive mixed precision and dynamically scaled preconditioned conjugate gradient algorithm (AMP-PCG). It dynamically adjusts the precision for storing vectors and computing, exploiting low precision when appropriate, while maintaining a convergence rate and accuracy comparable to that of double precision PCG. Our mixed precision strategy consists of three main components: (1) The residual and matrix-vector product are initially computed in double precision, and the algorithm switches these to single precision based on the chosen convergence tolerance and an estimate of the residual gap. (2) Depending on the eigenvalue distribution, the preconditioned residual and search direction are either in half precision throughout the iterations or initially in double precision and then stepwise reduced to single and half precision. (3) A dynamically scaled residual is used at every iteration to mitigate underflow in half precision. We provide theoretical support for our estimates and we demonstrate the effectiveness of AMP-PCG through numerical experiments, highlighting both its robustness and the significant performance gains (1.63× speedup) achieved compared to double precision PCG on a GPU.

Key words. preconditioned conjugate gradient, mixed precision, adaptive precision, inexact computation, attainable accuracy

MSC codes. 65G50, 65F10, 65F50, 65Y20, 65Y05

- 1. Introduction. In this paper, we propose an adaptive mixed precision and dynamically-scaled Preconditioned Conjugate Gradient (PCG) algorithm, referred to as AMP-PCG, for solving symmetric positive definite linear systems. During the PCG iterations, AMP-PCG dynamically adapts the precision in which iteration vectors are stored and operations are carried out, using low-precision where appropriate, while ensuring convergence rate and accuracy comparable to double precision implementation. This strategy can significantly reduce the runtime for the linear solver. To mitigate the potential downsides of using mixed precision, our proposed algorithm satisfies three properties:
 - **P1.** It ensures that the true residual reaches the chosen convergence tolerance by keeping the so-called residual gap sufficiently small [24, 25].
 - **P2.** It maintain roughly the rate of convergence that double precision PCG would achieve.
 - **P3.** It uses dynamic scaling of the iteration vectors, when using FP16, to ensure that vectors (and vector components) can be represented within a relatively small range.

In numerical simulations, solving large-scale linear systems is computationally intensive and often the most time-consuming part. Recent hardware accelerators provide significantly higher arithmetic throughput at lower precisions. Moreover, using lower precision storage reduces both memory usage and bandwidth demands, substantially accelerating data transfers, which in turn allows the algorithm to take better advantage of the low precision higher arithmetic throughput. These trends motivate the development of algorithms that aggressively exploit reduced precision to improve the efficiency of sparse linear solvers.

1.1. Literature review. The use of multiple precisions in solving linear systems has been explored. One well-studied approach is iterative refinement, where the solution is improved iteratively by solving a sequence of correction equations that account for the residual error. At each step, the residual is computed in high precision, while the correction equations are solved in lower precision using either a direct solver, often based on a low precision LU-factorization [39, 9], or Krylov subspace methods [2]. Mixed precision iterative refinement has been extensively studied in both theory and practice [20, 49, 33, 10]. A potential drawback of this approach is that the Krylov subspace is discarded upon each outer refinement iteration, potentially leading to a significantly higher total iteration count compared with a double precision solve.

^{*}Submitted to the editors 05/06/2025.

Funding: This work was supported under NSF DMS 2208470. The first author acknowledges the generous support from the John K. Costain Graduate Fellowship. The third author was supported in part by the John K. Costain Faculty Chair in Science at Virginia Tech.

[†]Department of Mathematics, Virginia Tech, Blacksburg, VA 24061 (ycguo@vt.edu, sturler@vt.edu, tcew@vt.edu)

Beyond iterative refinement, various works have incorporated low precision arithmetic into specific components of Krylov subspace and other iterative methods [3]. For example, low precision preconditioners have demonstrated the ability to maintain convergence while accelerating computation [19, 36, 47]. In addition, matrix-vector products (matvec) can be performed in mixed precision, either statically or adaptively based on the magnitude of entries, to reduce data movement and computational cost [41, 23, 50]. Mixed precision restarted GMRES methods have also been proposed [40], where only the residual and solution updates are in double precision, while other vectors are in single precision. Specifically for PCG, Clark et al. [15] investigate the use of half precision multigrid preconditioning and single precision arithmetic in lattice QCD solvers, where the updated residual produced by PCG is periodically replaced with the true residual computed in high precision. In [46], a strategy that determines when to perform this residual replacement without affecting the convergence is studied. Maddox et al. [41] propose performing matvec operations in FP16 while using full orthogonalization to maintain the orthogonality of the residuals and applying log-sum-exp transformations to mitigate rounding and overflow errors.

Moreover, our work is closely related to the theory of finite precision CG. In such settings, rounding errors due to the finite precision arithmetic leads to two primary effects: a reduction in the attainable accuracy and a delay in convergence, as discussed in [45, 48, 25, 29]. These challenges are magnified in communication-avoiding variants, such as pipelined CG and s-step CG, which have larger local rounding errors relative to CG; see [13, 16, 12, 8] for detailed analysis of the resulting limitations in accuracy and convergence.

Mixed precision Krylov methods can also be analyzed as *inexact Krylov subspace methods*. In [22], the perturbations introduced by preconditioning using inner solves is analyzed. In [6, 5, 18], the matvec is computed approximately using an inner iteration, and adaptive stopping criteria are proposed for inner iteration to ensure convergence of the outer iteration. Their analysis shows that the allowable inaccuracy in the matvec can increase as the outer iteration progresses toward convergence.

1.2. AMP-PCG. We distinguish two choices of precision in AMP-PCG: one for the matvec and residual vector, which mainly controls the final attainable residual accuracy [25], and the second precision for the preconditioned residual and search direction vectors, which primarily affects the convergence rate [22, 44, 27]. It has been shown in [25] that roundoff errors in residual updates can limit the maximum attainable residual accuracy. Based on this analysis, we propose an attainable accuracy indicator and a precision switching criterion to reduce the precision of residual vector and matvec from FP64 to FP32. The second choice of precision concerns the preconditioned residual and the search direction vector. If the system matrix is well-conditioned or equipped with an effective preconditioner, the convergence is typically linear, and the preconditioned residual and search direction can be stored in FP16 from the start, while still achieving convergence rates comparable to PCG in FP64. This is the case with, for example, p-multigrid when applied to high-order finite element methods (FEM) for Poisson equations [35, 43]. We demonstrate the convergence of our algorithm with a multigrid preconditioner and search direction vectors in half precision in subsection 5.3. For linear systems without many large outlying eigenvalues and if the convergence is not linear, we begin using FP64 and lower the precision of the preconditioned residual and search direction to FP32 and FP16 according to the precision selector in AMP-PCG. In these cases, convergence remains close to that of PCG in FP64. For matrices with large, outlying eigenvalues, using vectors in low precision may lead to frequent recurrence of the eigenvectors corresponding to the large(st) eigenvalues in the construction of Krylov space [28], which can degrade convergence. In such cases, the algorithm adapts the precision more conservatively.

To ensure that all vectors remain within the representable range of the current precision, AMP-PCG normalizes the residual before preconditioning (other scaling factors can be used). It can be shown that the dynamically scaled PCG is equivalent to PCG in exact arithmetic. This strategy is similar to scaling in iterative refinement [33], but is applied in each iteration rather than applied in the outer loop. Scaling in low precision has also been used effectively in deep learning [1] and fluid dynamics simulations [37] to prevent underflow and overflow. In summary, the AMP-PCG algorithm follows almost the same steps as PCG but employs mixed precision for different vectors adaptively,

and it typically achieves convergence and accuracy comparable to a double precision PCG while reducing computational cost and data movement.

In the following sections, we first motivate the careful design of mixed precision PCG with an illustrative example in section 2. Then, in section 3, we review theoretical results of finite precision CG for attainable accuracy and inexact Krylov methods for convergence rates. In subsection 3.3, we introduce a dynamically scaled PCG method designed to prevent overflow and underflow in low precision arithmetic. We present our main algorithm, adaptive mixed precision and dynamically scaled PCG, and precision switch criteria in section 4. Finally, in section 5, we evaluate the method's performance by comparing its convergence behavior to that of double precision PCG on various linear systems and assessing runtime speedups in a large-scale GPU-accelerated high-order finite element solver.

2. Motivating example. We consider the solution of the $n \times n$ linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

where **A** is a real, symmetric, positive definite (SPD) matrix. The preconditioned conjugate gradient (PCG) method [30] is widely used for solving large-scale, sparse, SPD systems. An outline of the PCG algorithm is given in Algorithm 3.1.

In exact arithmetic, the updated residual \mathbf{r}_{k+1} exactly equals the true residual $\mathbf{b} - \mathbf{A}\mathbf{x}_{k+1}$, the search directions \mathbf{p}_k are \mathbf{A} -orthogonal to all previous search directions, and the algorithm converges to the exact solution in at most n iterations. However, in finite precision arithmetic, rounding errors disrupt these properties [28, 42]. As a result, the updated residual \mathbf{r}_{k+1} , obtained from line 13 of Algorithm 3.1, may deviate from the true residual $\mathbf{b} - \mathbf{A}\mathbf{x}_{k+1}$, due to the rounding error in line 12 and 13. Additionally, the search direction \mathbf{p}_k may lose its global \mathbf{A} -orthogonality with previous directions, maintaining only local orthogonality, which can result in a degraded convergence rate. Consequently, the algorithm may require more than n iterations to converge. We present a motivating example to demonstrate how finite precision affects the convergence of PCG.

Example 1. Let $\mathbf{A} \in \mathbb{R}^{100 \times 100}$ have eigenvalues $\lambda_1, \dots, \lambda_{100}$, defined as follows:

(2.2)
$$\lambda_{100} = 1, \quad \lambda_i = 10^{-3} + \frac{i-1}{99} \left(\frac{17}{20}\right)^{100-i}, \quad \text{for } i = 95, \dots, 99, \\ [\lambda_1, \dots, \lambda_{95}] = \texttt{linspace}(10^{-3}, \lambda_{95}, 95).$$

Here, linspace(a, b, N) denotes N evenly spaced values between a and b. The matrix \mathbf{A} contains 5 isolated large eigenvalues and 95 evenly spaced eigenvalues. The eigenvector matrix of \mathbf{A} is a random orthonormal matrix. The right-hand side \mathbf{b} is a vector of all ones. The preconditioner is the identity matrix, and the initial guess \mathbf{x}_0 is the zero vector. The stopping tolerance is $tol = 10^{-10}$.

We solve the linear system using the PCG algorithm in half, in single, and in double precision. Figure 1 shows the convergence of the relative updated residual norm and the relative true residual norm for these three implementations. Here, the updated residual refers to the residual \mathbf{r}_k computed in line 13 of Algorithm 3.1, while the true residual is given by $\mathbf{b} - \mathbf{A}\mathbf{x}_k$. Initially, the convergence of PCG in half or single precision closely matches that of the double precision implementation. However, after approximately 30 iterations, the half and double precision PCG diverge. As the residual norm continues to decrease, in half precision, \mathbf{z}_k eventually underflows to zero due to the limited dynamic range, causing ρ_k to be zero and consequently β_k evaluation induces NaNs. This underflow is marked with a red star in Figure 1. Moreover, due to the reduced numerical accuracy in low precision formats, the updated residual norm deviates from the true residual norm in both half and single precision implementations. As a result, the final accuracy in low precision is substantially lower than that achieved in double precision. Additionally, the convergence of PCG in both half and single precision is slightly slower than in double precision after the 25-th iteration. This example illustrates three key challenges using PCG in low precision: unacceptable accuracy, degraded convergence rate, and data underflow.

Our goal is to develop an algorithm that satisfies P1., P2., P3., while using low precision in selected steps and variables of PCG, thereby reducing computational cost and data movement. In the

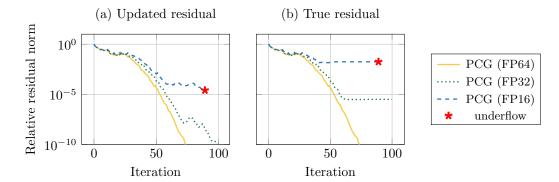


Fig. 1: Convergence of (a) the updated residual, and (b) the true residual, computed as $\mathbf{b} - \mathbf{A}\mathbf{x}_k$, for PCG implemented in half, single, and double precision for Example 1. The matrix \mathbf{A} is defined in (2.2). Compared with FP64, both FP16 and FP32 yield lower attainable accuracy and slower convergence. In the FP16 case, ρ_k eventually underflows to zero due to limited dynamic range, leading to breakdown, indicated by the red star. This example highlights that naive use of low precision in PCG can lead to reduced accuracy, delayed convergence, and data underflow.

following sections, we review key results from finite precision sensitivity analysis of PCG and inexact Krylov methods to understand what affects the attainable accuracy and the rate of convergence of PCG. Additionally, we introduce a dynamically scaling to mitigate data underflow. We propose an algorithm that selects appropriate precision levels for \mathbf{z}_k , \mathbf{p}_k , \mathbf{r}_k , and \mathbf{q}_k in Algorithm 4.1. The motivating example is revisited in subsection 5.1.

3. Analysis. We recall the finite precision analysis on the attainable accuracy and the theory of inexact Krylov subspace methods about the convergence rate of PCG. To address potential data underflow, we introduce the dynamically scaled PCG algorithm (DS-PCG). The analysis and the DS-PCG algorithm provide the foundation for the adaptive mixed precision and dynamically scaled PCG method described in section 4.

3.1. (P1) Attainable accuracy. For the attainable accuracy of PCG, we largely follow [25], with modifications tailored to our specific context. This analysis holds for any coupled recurrence of the type

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_{k-1} \mathbf{p}_{k-1},$$

$$\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1} \mathbf{A} \mathbf{p}_{k-1}.$$

Hence, the analysis remains valid even if the mixed precision PCG deviates substantially from PCG in exact arithmetic and also under changes in other parts of the algorithm.

Let $\varepsilon_{x,k}$, $\varepsilon_{r,k}$, and $\varepsilon_{q,k}$ denote the roundoff errors in computing \mathbf{x}_k , \mathbf{r}_k , and \mathbf{q}_k , respectively, as specified in lines 12, 13, and 9 of Algorithm 3.1. A finite precision implementation leads to the following (extended) recurrence (variables denote the finite precision computed variables),

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_{k-1} \mathbf{p}_{k-1} + \boldsymbol{\xi}_k,$$

(3.4)
$$\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1} \mathbf{A} \mathbf{p}_{k-1} + \boldsymbol{\theta}_k,$$

where

(3.5)
$$\|\boldsymbol{\xi}_{k}\| \leq \varepsilon_{x,k} \|\mathbf{x}_{k-1}\| + (2\varepsilon_{x,k} + \varepsilon_{x,k}^{2}) \|\alpha_{k-1}\mathbf{p}_{k-1}\|,$$

$$(3.6) \|\boldsymbol{\theta}_k\| \le \varepsilon_{r,k} \|\mathbf{r}_{k-1}\| + (2\varepsilon_{r,k} + \varepsilon_{r,k}^2) \|\boldsymbol{\alpha}_{k-1} \mathbf{A} \mathbf{p}_{k-1}\| + (1 + 2\varepsilon_{r,k} + \varepsilon_{r,k}\varepsilon_{q,k}) \|\boldsymbol{\alpha}_{k-1} \mathbf{d}_{k-1}\|.$$

These bounds follow directly from [25] using the specific roundoff errors above. The vector \mathbf{d}_{k-1} represents the numerical error arising from the floating point evaluation of \mathbf{Ap}_{k-1} , given by, \mathbf{d}_{k-1} =

 $f(\mathbf{A}\mathbf{p}_{k-1}, \varepsilon_{q,k}) - \mathbf{A}\mathbf{p}_{k-1}$, where $f(\cdot, \varepsilon_{q,k})$ denotes the result of floating point arithmetic with $\varepsilon_{q,k}$. We define ζ_k as the difference between the true residual $\mathbf{b} - \mathbf{A}\mathbf{x}_k$ and the updated residual \mathbf{r}_k ,

(3.7)
$$\boldsymbol{\zeta}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k - \mathbf{r}_k = (\mathbf{b} - \mathbf{A}\mathbf{x}_0 - \mathbf{r}_0) - \sum_{t=1}^k (\mathbf{A}\boldsymbol{\xi}_t + \boldsymbol{\theta}_t).$$

Therefore, the true residual norm $\|\mathbf{b} - \mathbf{A}\mathbf{x}_k\|$ can be bounded as follows:

(3.8)
$$|||\zeta_k|| - ||\mathbf{r}_k||| \le ||\mathbf{b} - \mathbf{A}\mathbf{x}_k|| \le ||\zeta_k|| + ||\mathbf{r}_k||.$$

In general, \mathbf{r}_k , the updated residual, satisfies [24, 25],

(3.9)
$$\mathbf{r}_k \to 0$$
, for $k \to \infty$.

When $\|\boldsymbol{\zeta}_k\| \gg \|\mathbf{r}_k\|$, the best achievable accuracy for $\|\mathbf{b} - \mathbf{A}\mathbf{x}_k\|$ is approximately $\|\boldsymbol{\zeta}_k\|$. Therefore, to ensure that the true residual norm $\|\mathbf{b} - \mathbf{A}\mathbf{x}_k\|$ converges to the chosen tolerance tol^1 , the relative convergence tolerance should be set to

(3.10)
$$\|\zeta_k\| + \|\mathbf{r}_k\| \le tol \|\mathbf{b}\|.$$

Then, when this bound is satisfied at, say, iteration m, we have

(3.11)
$$\|\mathbf{b} - \mathbf{A}\mathbf{x}_m\| \le \|\zeta_m\| + \|\mathbf{r}_m\| \le tol\|\mathbf{b}\|.$$

This approach assumes that the residual gap (3.7) (or its upper bound), stays below the convergence tolerance. If the residual gap becomes too large, we can use several residual replacement strategies to enforce a sufficiently small gap while maintaining good convergence for the PCG algorithm [48, 46, 15, 13]. However, here we focus on controlling the gap by switching precision for various vectors at appropriate points in the algorithm. This strategy ensures that the attainable precision can be reached; however, it does not address the rate of convergence, which we consider next.

3.2. (P2) Rate of convergence. Next, we consider how to maintain (close to) the rate of convergence achieved by PCG in double precision while exploiting mixed precision implementations. We follow the approach of [22], which considers the tolerance for an iteratively applied preconditioner. We can apply their analysis, for the purpose of this paper, to guide the precision of the preconditioner \mathbf{M} and the preconditioned residual \mathbf{z}_k . Moreover, as the quality of the new search direction, $\mathbf{p}_k = \mathbf{z}_k + \beta_{k-1}\mathbf{p}_{k-1}$, depends largely on \mathbf{z}_k (the direction expanding the current Krylov subspace), there is no need to compute \mathbf{p}_k in higher precision than \mathbf{z}_k . So, we use the same precision for both.

The precision of \mathbf{z}_k can be considered to reflect the accuracy with which the preconditioner has been computed. Following [22], we replace $\mathbf{M}\mathbf{z}_k = \mathbf{r}_k$ by

$$\mathbf{Mz}_k = \mathbf{r}_k + \mathbf{e}_k,$$

where \mathbf{e}_k is the perturbation in computing \mathbf{z}_k . Theorem 3.6 in [22] provides a bound for the rate of linear convergence for the Inexact Preconditioned CG (IPCG):

Theorem 3.1 (Theorem 3.6 in [22]). Let $\delta \geq \|\mathbf{e}_k\|_{\mathbf{M}^{-1}} / \|\mathbf{r}_k\|_{\mathbf{M}^{-1}}$. If $\delta' = 2\sin\theta\sqrt{\kappa} < 1$ with $\theta = \arcsin\delta$, $\kappa = \mathrm{cond}(\mathbf{M}^{-1/2}\mathbf{A}\mathbf{M}^{-1/2})$, then IPCG converges, and for even k,

(3.12)
$$\frac{\|\mathbf{r}_{k+1}\|_{\mathbf{A}^{-1}}}{\|\mathbf{r}_1\|_{\mathbf{A}^{-1}}} \le (\sigma K)^k,$$

where

$$\sigma = \frac{\sqrt{\kappa'} - 1}{\sqrt{\kappa'} + 1}, \quad K = \sqrt{\frac{2 + 16\delta'\kappa'/(\kappa' - 1)^2}{1 + \sigma^4}}, \quad and \quad \kappa' = \kappa \frac{1 + \delta}{1 - \delta}.$$

¹ for simplicity, we choose the stopping criterion based on the relative residual norm.

In [22], the perturbation δ is of the order 10^{-2} . However, in the mixed precision context, using single precision or half precision for \mathbf{z}_k and \mathbf{p}_k usually leads to $\delta \approx 10^{-7}$ and $\delta \approx 10^{-3}$, respectively. When $\delta \ll 1$, the modified condition number κ' remains close to κ . For example, if $\mathbf{M} = \mathbf{I}$ and \mathbf{z}_k is computed in half precision, $\delta \approx 10^{-3}$. Assuming $\kappa = 10^3$, we obtain $K \approx 1.06$, and the relative increase in κ' compared to κ is only about 0.2%. Thus, we anticipate if PCG is in a linear convergence regime, then using half precision for \mathbf{z}_k and \mathbf{p}_k does not significantly degrade the convergence rate.

3.3. (P3) Dynamic Scaling. As discussed above, we can use low precision vectors in PCG without significantly affecting convergence rate or attainable accuracy under certain conditions. Beyond convergence rate and accuracy, another critical issue in low precision is the risk of data underflow or overflow. The motivating example in section 2 shows that underflow may occur when entries of the residual become small, causing the preconditioned residual to drop below the dynamic range of half precision. To address this, all vectors stored in half precision must be representable within a limited dynamic range. This can generally be achieved by consistently scaling all vectors during each PCG iteration. We introduce the dynamically scaled PCG algorithm (DS-PCG), presented in Algorithm 3.2, where the modifications relative to PCG are highlighted in blue.

Algorithm 3.1 Preconditioned Conjugate Gradient (PCG) Algorithm

Require: SPD matrix A, preconditioner M, right-hand side b, initial guess x_0 , stopping tolerance tol, maximum number of iterations m

```
Ensure: Approximate solution x
   1: Initialize: \mathbf{r}_0 \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}_0, \, \beta_{-1} \leftarrow 0
  2: for k = 0 to m - 1 do
              \mathbf{z}_k \leftarrow \mathbf{M}^{-1}\mathbf{r}_k
              \rho_k \leftarrow \mathbf{r}_k^T \mathbf{z}_k
  4:
              if k > 0 then
                   \beta_{k-1} \leftarrow \rho_k/\rho_{k-1}
  6:
  7:
              end if
              \mathbf{p}_k \leftarrow \mathbf{z}_k + \beta_{k-1} \mathbf{p}_{k-1}
  8:
              \mathbf{q}_k \leftarrow \mathbf{A}\mathbf{p}_k
  9:
              \gamma_k \leftarrow \mathbf{q}_k^T \mathbf{p}_k
 10:
11:
              \alpha_k \leftarrow \rho_k/\gamma_k
              \mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k
12:
13:
              \mathbf{r}_{k+1} \leftarrow \mathbf{r}_k - \alpha_k \mathbf{q}_k
              if \|\mathbf{r}_{k+1}\| \leq tol \|\mathbf{b}\| then
14:
                    break
15:
```

end if

17: end for

18: $\mathbf{x} \leftarrow \mathbf{x}_{k+1}$

16:

Algorithm 3.2 Dynamically Scaled PCG (DS-PCG)

Require: SPD matrix **A**, preconditioner **M**, right-hand side **b**, initial guess \mathbf{x}_0 , stopping tolerance *tol*, maximum number of iterations m, and scaling factors $\{\omega_k\}_{k=0}^m$

```
Ensure: Approximate solution x
   1: Initialize: \mathbf{r}_0 \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}_0, \, \beta_{-1} \leftarrow 0
   2: for k = 0 to m - 1 do
              \mathbf{y}_k \leftarrow \omega_k \, \mathbf{r}_k, \, \mathbf{z}_k \leftarrow \mathbf{M}^{-1} \mathbf{y}_k
              \rho_k \leftarrow \mathbf{r}_k^T \mathbf{z}_k
   4:
              if k > 0 then
   5:
                    \beta_{k-1} \leftarrow \rho_k/\rho_{k-1}
   6:
              end if
   7:
              \mathbf{p}_k \leftarrow \mathbf{z}_k + \beta_{k-1} \mathbf{p}_{k-1}
   8:
              \mathbf{q}_k \leftarrow \mathbf{A}\mathbf{p}_k
  9:
              \gamma_k \leftarrow \mathbf{q}_k^T \mathbf{p}_k
10:
11:
              \alpha_k \leftarrow \rho_k/\gamma_k
12:
              \mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k
13:
              \mathbf{r}_{k+1} \leftarrow \mathbf{r}_k - \alpha_k \mathbf{q}_k
              if \|\mathbf{r}_{k+1}\| \leq tol \|\mathbf{b}\| then
14:
                    break
15:
16:
              end if
17: end for
18: \mathbf{x} \leftarrow \mathbf{x}_{k+1}
```

The key modification is to scale the residual vector prior to preconditioning (line 3 of Algorithm 3.2) to ensure that $\|\mathbf{y}_k\|$ remains within a moderate range, thereby avoiding excessively small \mathbf{z}_k values. The scaling factor ω_k in practice may be chosen as $\|\mathbf{r}_k\|^{-1}$ or $\|\mathbf{z}_{k-1}\|^{-1}$. The scaling can be fused with preconditioning step to minimize overhead. Similar scaling strategies have been successfully used in other contexts involving mixed precision computation [34, 37, 1]. For any sequence $\{\omega_k\}_{k=0}^m$ with $\omega_k \in \mathbb{R} \setminus \{0\}$ in DS-PCG, the approximate solutions and residuals produced by PCG are exactly the same as those produced by DS-PCG in exact arithmetic. This equivalence is formalized in the following theorem.

THEOREM 3.2. Let \mathbf{x}_k and \mathbf{r}_k denote the approximate solutions and residuals generated by PCG (Algorithm 3.1), and let $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{r}}_k$ be those generated by DS-PCG(Algorithm 3.2). Given the same

initial guesses \mathbf{x}_0 and a sequence of nonzero real scaling factors $\{\omega_k\}_{k=0}^m$, DS-PCG and PCG iterates are identical in exact arithmetic for all k, that is,

$$\widehat{\mathbf{x}}_k = \mathbf{x}_k$$
 and $\widehat{\mathbf{r}}_k = \mathbf{r}_k$.

The proof of Theorem 3.2 is provided in Appendix A.

4. Adaptive mixed precision and dynamically scaled PCG algorithm. Based on the theoretical discussion in section 3, we propose an adaptive mixed precision PCG algorithm with dynamic scaling that satisfies properties P1., P2., and P3., while lowering the precision as much as possible to reduce computational cost and data movement.

Since the residual gap (3.7) is influenced primarily by rounding errors in the computation of \mathbf{x}_k , \mathbf{q}_k , and \mathbf{r}_k , while the rate of convergence depends primarily on rounding errors in computing \mathbf{p}_k and \mathbf{z}_k , we define two separate precision controls:

- 1. $u_{r,k}$: the precision used to compute \mathbf{y}_k , and to compute and store \mathbf{q}_k and \mathbf{r}_k ,
- 2. $u_{z,k}$: the precision used to store \mathbf{y}_k , and to compute and store \mathbf{z}_k and \mathbf{p}_k .

The approximate solution \mathbf{x}_k , all inner products, norms, and scalar-only operations are computed in double precision to ensure the accuracy of the computed solution.

To guarantee that we reach the target accuracy, we control $\|\boldsymbol{\zeta}_k\|$ (3.7) by adaptively reducing $u_{r,k}$ from double to single precision, based on a precision switch indicator η_k detailed in subsection 4.1. The analysis of IPCG in subsection 3.2 suggests that, within a linear convergence regime and under small perturbations in \mathbf{z}_k and \mathbf{p}_k , the convergence rate remains comparable to that of double precision PCG. Often, when highly effective preconditioners (like multigrid) are used, the iteration exhibits linear convergence, allowing $u_{z,k}$ to be FP16 throughout. When linear convergence is not evident, we introduce two heuristic thresholds, $\tau_{z,s}$ and $\tau_{z,h}$, to estimate the onset of the linear regime. These thresholds determine when $u_{z,k}$ should be reduced from double to single precision and from single to half precision, respectively. For our examples, we determine the thresholds experimentally.

Let u_0 denote the initial (maximum) precision for $u_{z,k}$, and let tol be the stopping tolerance. We define the relative residual norm at iteration k as $\nu_k = \frac{\|\mathbf{r}_k\|}{\|\mathbf{b}\|}$. The precision selector function below determines $u_{z,k}$ and $u_{r,k}$ at iteration k:

$$(u_{z,k}, u_{r,k}) = \operatorname{precisionSelector}\left(\nu_{k}, u_{0}, \tau_{z,s}, \tau_{z,h}, \eta_{k}\right), \text{ with}$$

$$u_{z,k} = \begin{cases} u_{0}, & \text{if } \nu_{k} \geq \tau_{z,s}, \\ \min\left\{u_{0}, \text{FP32}\right\}, & \text{if } \tau_{z,h} \leq \nu_{k} < \tau_{z,s}, \\ \min\left\{u_{0}, \text{FP16}\right\}, & \text{if } \nu_{k} < \tau_{z,h}, \end{cases}$$

$$u_{r,k} = \begin{cases} \text{FP64}, & \text{if } \eta_{k} \geq tol, \\ \text{FP32}, & \text{if } \eta_{k} < tol, \end{cases}$$

Here, $\min(p_1, p_2)$ returns the lower precision format, and η_k is defined by (4.9) or, in the case of linear convergence, by (4.11).

The Adaptive Mixed Precision and Dynamically Scaled PCG algorithm (AMP-PCG), presented in Algorithm 4.1, implements the proposed algorithmic changes. Vectors highlighted in blue and orange reflect the precisions $u_{z,k}$ and $u_{r,k}$, respectively.

4.1. Criterion for switching precision for r_k and \mathbf{q}_k . To achieve the target tolerance tol, the inequality (3.11) must be satisfied. Therefore, we aim to derive a bound on $\|\boldsymbol{\zeta}_k\|$ (3.7) to ensures the prescribed accuracy is met. We begin by establishing the following results under three assumptions:

Assumption 4.1. There exists a constant C > 0, independent of the matrix size n, such that for all k,

(4.2)
$$\|f(\mathbf{A}\mathbf{p}_k, \varepsilon_{a,k}) - \mathbf{A}\mathbf{p}_k\| \le C\varepsilon_{a,k} \|\mathbf{A}\mathbf{p}_k\|.$$

Algorithm 4.1 Adaptive Mixed Precision and Dynamically Scaled PCG with Initial Precision u_0 (AMP-PCG (u_0))

Require: SPD matrix **A**, preconditioner **M**, right-hand side **b**, initial guess \mathbf{x}_0 , stopping tolerance tol, precision switch thresholds $\tau_{z,h}$ and $\tau_{z,s}$, precision switch indicator η_k , maximum iteration count m, the default precision u_0 , and a precision selector function

```
Ensure: Approximate solution x
  1: \mathbf{r}_0 \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}_0, \ \beta_{-1} \leftarrow 0, \ \delta_0 \leftarrow \|\mathbf{r}_0\|
  2: for k = 0 to m - 1 do
             Calculate \eta_k //Precision switch indicator for u_{r,k} (u_{z,k}, u_{r,k}) \leftarrow precisionSelector \left(\frac{\delta_k}{\|\mathbf{b}\|}, u_0, \tau_{z,s}, \tau_{z,h}, \eta_k\right) //Select precisions following (4.1)
  4:
              \mathbf{y}_k \leftarrow \mathbf{r}_k / \delta_k
  5:
              \mathbf{z}_k \leftarrow \mathbf{M}^{-1}\mathbf{y}_k
  6:
              \rho_k \leftarrow \mathbf{r}_k^T \mathbf{z}_k
  7:
              if k > 0 then
  8:
                  \beta_{k-1} \leftarrow \rho_k/\rho_{k-1}
  9:
10:
              end if
              \mathbf{p}_k \leftarrow \mathbf{z}_k + \beta_{k-1} \mathbf{p}_{k-1}
11:
              \mathbf{q}_k \leftarrow \mathbf{A}\mathbf{p}_k
12:
13:
              \gamma_k \leftarrow \mathbf{q}_k^T \mathbf{p}_k
14:
             \alpha_k \leftarrow \rho_k/\gamma_k
                                                                                                                                                                        //Update solution in FP64
15:
             \mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k
16:
             \mathbf{r}_{k+1} \leftarrow \mathbf{r}_k - \alpha_k \mathbf{q}_k
17:
             \delta_{k+1} \leftarrow ||\mathbf{r}_{k+1}||
             if \delta_{k+1} \leq tol \|\mathbf{b}\| then
18:
19:
                   break
              end if
20:
21: end for
22: \mathbf{x} \leftarrow \mathbf{x}_{k+1}
```

Assumption 4.2. There exists a constant $C_x > 0$ such that $\|\mathbf{x}_k\| \leq C_x \|\mathbf{x}\|$ for all k.

Assumption 4.3. Let \bar{k} denote the total number of iterations required by the algorithm. Then $\bar{k}\varepsilon_{x,k}\|\mathbf{A}\|\|\mathbf{x}\|$ is sufficiently small.

Remark 4.4. Assumption 4.1 adopts a less conservative error bound than is commonly used in the literature:

$$\|\mathbf{fl}(\mathbf{A}\mathbf{p}_k, \varepsilon_{q,k}) - \mathbf{A}\mathbf{p}_k\| \le \hat{C} \, \varepsilon_{q,k} \|\mathbf{A}\| \|\mathbf{p}_k\|,$$

where $\hat{C} = m_r n^{1/2}$ in [21] and m_r denotes the maximum number of nonzeros per row. While sharper bounds can be obtained through probabilistic rounding error analysis [31, 32], we adopt (4.2) for practicality, as the more rigorous alternatives often yield pessimistic estimates. In practice, C = 1 seems to yield satisfactory results, although it may be somewhat optimistic.

Assumption 4.2 holds when the initial guess \mathbf{x}_0 is comparable in the size to the exact solution \mathbf{x} , which is often the case in practice [30, 25].

Assumption 4.3 reflects the fact that \mathbf{x}_k is computed in double precision, and hence $\varepsilon_{x,k}$ corresponds to the unit roundoff for FP64. Typically, both $\|\mathbf{A}\|$ and the total iteration count \bar{k} are moderate, ensuring that the term $\bar{k} \varepsilon_{x,k} \|\mathbf{A}\| \|\mathbf{x}\|$ is negligible in comparison with the numerical errors arising from the residual updates.

We now derive a bound for $\|\boldsymbol{\theta}_k\|$, which is defined in (3.4). Since both \mathbf{r}_k and \mathbf{q}_k are computed

in precision $u_{r,k}$, it follows that $\varepsilon_{q,k} = \varepsilon_{r,k}$ in (3.6). From Assumption 4.1, we obtain (4.3)

$$\|\boldsymbol{\theta}_{k}\| \leq \varepsilon_{r,k} \|\mathbf{r}_{k-1}\| + (2\varepsilon_{r,k} + \varepsilon_{r,k}^{2}) \|\alpha_{k-1}\mathbf{A}\mathbf{p}_{k-1}\| + (1+\varepsilon_{r,k})^{2} \|\alpha_{k-1} (\mathrm{fl}(\mathbf{A}\mathbf{p}_{k-1}, \varepsilon_{r,k}) - \mathbf{A}\mathbf{p}_{k-1}) \|$$

$$\leq \varepsilon_{r,k} \|\mathbf{r}_{k-1}\| + (2\varepsilon_{r,k} + \varepsilon_{r,k}^{2}) \|\alpha_{k-1}\mathbf{A}\mathbf{p}_{k-1}\| + C(1+\varepsilon_{r,k})^{2} \varepsilon_{r,k} \|\alpha_{k-1}\mathbf{A}\mathbf{p}_{k-1}\|$$

$$\leq \varepsilon_{r,k} \|\mathbf{r}_{k-1}\| + (2\varepsilon_{r,k} + \varepsilon_{r,k}^{2} + C\varepsilon_{r,k}(1+\varepsilon_{r,k})^{2}) \|\alpha_{k-1}\mathbf{A}\mathbf{p}_{k-1}\|$$

$$\leq \varepsilon_{r,k} \|\mathbf{r}_{k-1}\| + ((2+C)\varepsilon_{r,k} + (1+3C)\varepsilon_{r,k}^{2}) (\|\mathbf{r}_{k-1}\| + \|\mathbf{r}_{k}\| + \|\boldsymbol{\theta}_{k}\|).$$

As long as $1 - \left((2+C) \varepsilon_{r,k} + (1+3C) \varepsilon_{r,k}^2 \right) > 0$, this can be written in the form

(4.4)
$$\|\boldsymbol{\theta}_{k}\| \leq (3+C)\,\varepsilon_{r,k}\|\mathbf{r}_{k-1}\| + (2+C)\,\varepsilon_{r,k}\|\mathbf{r}_{k}\| + \mathcal{O}(\varepsilon_{r,k}^{2})\,(\|\mathbf{r}_{k-1}\| + \|\mathbf{r}_{k}\|)\,.$$

Similarly, with Assumption 4.2, $\|\boldsymbol{\xi}_k\|$ can be bounded as follows:

$$(4.5) \|\boldsymbol{\xi}_{k}\| \leq \varepsilon_{x,k} \left(3 \|\mathbf{x}_{k-1}\| + 2 \|\mathbf{x}_{k}\|\right) + O\left(\varepsilon_{x,k}^{2}\right) \left(\|\mathbf{x}_{k-1}\| + \|\mathbf{x}_{k}\|\right) \leq 5 k \varepsilon_{x,k} \|\mathbf{x}\| + O\left(\varepsilon_{x,k}^{2}\right) \|\mathbf{x}\|.$$

From (3.7), (4.4), and (4.5), we obtain the following bound for $\|\boldsymbol{\zeta}_k\|$:

$$\begin{aligned} \|\boldsymbol{\zeta}_{k}\| &\leq \|\mathbf{b} - \mathbf{A}\mathbf{x}_{0} - \mathbf{r}_{0}\| + \|\mathbf{A}\| \sum_{t=1}^{k} \|\boldsymbol{\xi}_{t}\| + \sum_{t=1}^{k} \|\boldsymbol{\theta}_{t}\| \\ &\leq 5 k \,\varepsilon_{x,k} \|\mathbf{A}\| \|\mathbf{x}\| + \sum_{t=0}^{k} \varepsilon_{r,k} \left((3+C) \|\mathbf{r}_{t-1}\| + (2+C) \|\mathbf{r}_{t}\| \right) + O\left(\varepsilon_{x,t}^{2}\right) + O\left(\varepsilon_{r,t}^{2}\right) \\ &\approx \sum_{t=0}^{k} \varepsilon_{r,k} \left((3+C) \|\mathbf{r}_{t-1}\| + (2+C) \|\mathbf{r}_{t}\| \right). \end{aligned}$$

The last approximation follows by neglecting the first term (Assumption 4.3) and the second order terms. The norm $\|\boldsymbol{\zeta}_k\|$ estimates the best attainable accuracy for sufficiently large k (so that $\|\mathbf{r}_k\|$ is small), as shown in (3.11). Switching precision $u_{r,k}$ from FP64 to FP32 at iteration k, we define an indicator $\widehat{\eta}_m(k)$ to estimate the best attainable accuracy at iteration m:

$$(4.7) \quad \widehat{\eta}_m(k) := \sum_{t=0}^k \varepsilon_{64} \left((3+C) \| \mathbf{r}_{t-1} \| + (2+C) \| \mathbf{r}_t \| \right) + \sum_{t=k}^m \varepsilon_{32} \left((3+C) \| \mathbf{r}_{t-1} \| + (2+C) \| \mathbf{r}_t \| \right),$$

where ε_{32} and ε_{64} are the unit roundoff errors of single and double precision, respectively, and m denotes the final iteration index. In practice, however, $\widehat{\eta}_m(k)$ cannot be evaluated until the run completes, since it require all residual norms $\|\mathbf{r}_t\|$ for $k < t \le m$. Therefore, we introduce a computable heuristic indicator that predicts the attainable accuracy without relying on residual norms $\|\mathbf{r}_t\|$ beyond iteration k. Let integer d be a fixed delay parameter. If reducing precision at iteration k - d satisfies the condition $\widehat{\eta}_m(k - d) \le tol \|\mathbf{b}\|$, then switching at k will also satisfy the tolerance (since $\widehat{\eta}_m(k - d) > \widehat{\eta}_m(k)$). Moreover, we assume the residual decays rapidly after these d iterations, such that

(4.8)
$$\sum_{t=k+1}^{m} \|\mathbf{r}_t\| \ll \sum_{t=k-d}^{k} \|\mathbf{r}_t\|.$$

Now, we define the computable attainable accuracy indicator as the dominant contribution in $\widehat{\eta}_m(k-d)$, expressed as:

(4.9)
$$\eta_k := \sum_{t=k-d}^k \varepsilon_{32} \left((3+C) \| \mathbf{r}_{t-1} \| + (2+C) \| \mathbf{r}_t \| \right).$$

Additionally, we define the criterion for switching the precision of \mathbf{r}_k and \mathbf{q}_k from FP64 to FP32:

$$(4.10) \eta_k \le tol \|\mathbf{b}\|.$$

In our numerical experiments (see subsection 5.2), we demonstrate that choosing d = 10 and C = 1 in (4.9) provide a robust and effective compromise. This delay parameter is inspired by a similar approach used for estimating the **A**-norm of the error in [30].

When the convergence rate is nearly linear, the residual norms can be estimated without a delay parameter. We can estimate the average rate of convergence by $\rho = (\|\mathbf{r}_k\|/\|\mathbf{r}_{k-\ell}\|)^{1/\ell}$, for a chosen parameter ℓ , and estimate $\widehat{\eta}_m(k)$ using $\|\mathbf{r}_t\| = \|\mathbf{r}_k\|\rho^{t-k}$. Neglecting all terms proportional to ε_{64} , we therefore define our attainable accuracy indicator η_k and the corresponding precision switching criterion as follows:

(4.11)
$$\eta_{k} = \sum_{t=k}^{m} \varepsilon_{32} \left((3+C) \| \mathbf{r}_{t-1} \| + (2+C) \| \mathbf{r}_{t} \| \right) \\ \leq \varepsilon_{32} \left(5 + 2C \right) \| \mathbf{r}_{k-1} \| \frac{1}{1-\rho} \leq tol \| \mathbf{b} \|.$$

Therefore, when the convergence is almost linear, we use (4.11) to choose the precision switching step instead of (4.9). This criterion is tested in subsection 5.3 and subsection 5.5.

Substituting η_k as an estimate for $\max_{t \leq m} \|\zeta_t\|$ in (3.11) and combining this with the stopping criterion (line 18 in Algorithm 4.1), we obtain a bound for the true residual norm:

$$\|\mathbf{b} - \mathbf{A}\mathbf{x}_m\| \lesssim \eta_k + \|\mathbf{r}_m\| \leq 2 \operatorname{tol} \|\mathbf{b}\|.$$

Thus, given a target accuracy tol, we can choose when to switch the precision based on either (4.9) or (4.11), such that the true residual norm satisfies the accuracy.

5. Numerical Results. In this section, we test the performance of AMP-PCG (Algorithm 4.1) on the solution of a set of example linear systems. We begin by revisiting the motivating example in subsection 5.1. Then in subsection 5.2, we assess the attainable accuracy of AMP-PCG, and then in subsection 5.3 and subsection 5.4 we examine its convergence behavior. We evaluate the performance of AMP-PCG within a large-scale GPU-accelerated high-order finite element solver in subsection 5.5. Finally, we illustrate some limitations of the precision selector (4.1) in subsection 5.6 through several experiments. A summary of all examples discussed in this section is provided in Table 1.

Example	Description	n	Est. $\kappa(\mathbf{M}^{-1}\mathbf{A})$	Convergence
1	Motivating example	100	10^{3}	Superlinear
2	bcsstk08.mat	1074	3.77×10^3	Superlinear
3	622_bus.mat	662	4.46×10^4	Superlinear
4	Evenly distributed eigen.	101	10	Linear
5	FEM with multigrid prec.	9.1×10^6	187.2	Linear
6	Evenly distributed eigen.	101	10^{3}	Superlinear
7	FEM with Jacobi prec.	2.2×10^7	1.99×10^3	Linear
8	nos6.mat	675	3.49×10^6	Superlinear
9	Illustrative example	10	10^{3}	Oscillatory
10	Many large outlying eigen.	50	10^{3}	Oscillatory

Table 1: Summary of test examples. n is the matrix size, κ is the estimated condition number of the preconditioned matrix, and the last column indicates observed convergence behavior.

In this section, unless otherwise specified, in the precision selector function (4.1), η_k is defined by (4.10) with d = 10 and C = 1 under superlinear convergence and by (4.11) with $\ell = 5$ under

linear convergence. The following heuristic thresholds are used:

(5.1)
$$\tau_{z,s} = 10^{-4}, \quad \tau_{z,h} = 10^{-6}.$$

In the following subsections, we compare several PCG variants:

- 1. PCG(FP64): All vectors are computed and stored in double precision.
- 2. **AMP-PCG(FP64)**: Initially uses FP64 for both $u_{z,k}$ and $u_{r,k}$, which are then adaptively reduced during the iterations.
- 3. AMP-PCG(FP16): Use \mathbf{z}_k and \mathbf{p}_k in FP16, while $u_{r,k}$ is adaptively reduced.
- 4. **AMP-PCG** $(u_{r,k})$: Keeps $u_{z,k}$ fixed in FP64, and adaptively reduces $u_{r,k}$; used to assess the effectiveness of the precision-switching criterion for $u_{r,k}$.

Moreover, unless stated otherwise, the right hand side vector **b** is randomly generated, the initial guess \mathbf{x}_0 is the zero vector, the preconditioner is the identity matrix, and the stopping tolerance $tol = 10^{-10}$. In the numerical results, the updated residual corresponds to the residual computed within the AMP-PCG iteration (line 16 in Algorithm 4.1), whereas the true residual is defined as $\mathbf{b} - \mathbf{A}\mathbf{x}_k$.

5.1. Motivating example. As demonstrated in the motivating example (see Example 1 and Figure 1), applying the PCG algorithm in single or half precision leads to reduced attainable accuracy and slower convergence relative to PCG in double precision; half precision in particular also suffers from numerical underflow.

We now apply our AMP-PCG algorithm to the same example. In AMP-PCG(FP64), all vectors \mathbf{z}_k , \mathbf{p}_k , \mathbf{r}_k , and \mathbf{q}_k are initially computed and stored in double precision. Their precisions are adaptively reduced based on the precision selector function (4.1). In Figure 2, the red and black cross markers denote when the precision $u_{z,k}$, used for \mathbf{z}_k and \mathbf{p}_k , is reduced first to single precision and then to half precision, respectively; the blue circle markers denote when precision $u_{r,k}$, associated with \mathbf{r}_k and \mathbf{q}_k , is reduced from double to single precision.

The convergence of both the updated and true residuals under AMP-PCG(FP64) closely mirrors that of the PCG(FP64). This demonstrates the effectiveness of the proposed AMP-PCG in maintaining convergence and accuracy while using low precision. In the next subsections, we further evaluate the performance of AMP-PCG from multiple perspectives.

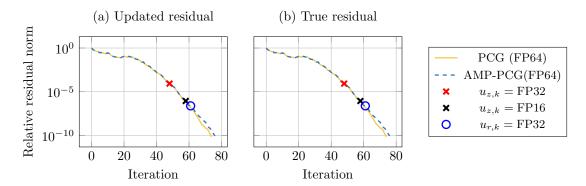


Fig. 2: Revisiting the motivating example (see Example 1 and Figure 1): convergence of (a) the relative updated residual norm and (b) the relative true residual norm for PCG(FP64) and AMP-PCG(FP64), as discussed in subsection 5.1. Cross markers indicate the iterations where the precision $u_{z,k}$, used for \mathbf{z}_k and \mathbf{p}_k , is reduced to single and then to half precision. The blue circle marks where $u_{r,k}$, used for \mathbf{r}_k and \mathbf{q}_k , is reduced from double to single precision.

5.2. Impact of the precision of \mathbf{r}_k and \mathbf{q}_k on attainable accuracy. We assess whether the mixed precision strategy for \mathbf{r}_k and \mathbf{q}_k achieves a final residual norm within the prescribed tolerance tol. All other vectors remain in double precision. We test Example 1 and two SPD matrices from the SuiteSparse Matrix Collection [17], bcsstk08.mat and 622_bus.mat, which are commonly used

for testing PCG. These two matrices are scaled on both sides by the square root of the inverse of its diagonal entries.

Example 2. Consider the 1074×1074 matrix bcsstk08.mat, for which the condition number after scaling is approximately 3.77×10^3 .

Example 3. Consider the 662×662 matrix 622_bus.mat, for which the condition number after scaling is about 4.46×10^4 .

We consider two options for the stopping tolerance tol used in line 18 of Algorithm 4.1:

$$tol = 10^{-6}$$
 and $tol = 10^{-8}$.

According to the precision selector function (4.1), varying tol changes when the precision $u_{r,k}$ is reduced, thereby impacting the attainable accuracy. In Figure 3, we present convergence curves for both the updated residual (top row) and the true residual (bottom row) for PCG and AMP-PCG($u_{r,k}$). In these experiments, the iterations are not terminated based on the stopping criterion in Algorithm 4.1, allowing us to examine the final attainable residual accuracy. Markers indicate the iteration at which the precision switch occurs for each value of tol. The results indicate that using low precision for \mathbf{r}_k and \mathbf{q}_k does not affect the convergence rate in these examples; however, it does limit the attainable accuracy. As illustrated in subfigures (b), (d), and (f), the precision selector function successfully drives the true residual below the prescribed tolerance $\|\mathbf{r}_k\| \leq tol \|\mathbf{b}\|$. Additionally, we compute the indicator $\hat{\eta}_k$, defined in (4.7). This indicator is an estimate of the final attainable accuracy, and its values are shown as dotted lines. In these examples, $\hat{\eta}_k$ closely approximates the attainable accuracy, confirming the effectiveness of the indicator.

The results show that using the precision selector function (4.1) to determine when to reduce the precision of \mathbf{r}_k and \mathbf{q}_k is effective in achieving the target accuracy. Since the updated and true residual norms closely agree prior to satisfying the stopping criterion, we report only the updated residual norm in subsequent subsections.

5.3. Cases with linear convergence. We consider two matrices that either have a small condition number or are equipped with an effective preconditioner, such that the condition number of the preconditioned system is small. In these cases, we anticipate linear convergence. As discussed in subsection 3.2, within the linear convergence regime, using \mathbf{p}_k and \mathbf{z}_k in half precision has negligible impact on the convergence rate. Therefore, we fix $u_{z,k} = \text{FP16}$, using \mathbf{z}_k and \mathbf{p}_k in half precision throughout the iterations.

Example 4. Define $\mathbf{A} \in \mathbb{R}^{(n+1)\times(n+1)}$ with uniformly distributed eigenvalues

$$\lambda_i = \frac{i}{10n} + \frac{n-i}{n}, \quad i = 0, \dots, n,$$

where n = 100. The eigenvector matrix is a random orthonormal matrix.

Example 5. We consider a matrix arising from a high-order finite element discretization of the Poisson equation:

(5.2)
$$-\Delta u(x, y, z) = f(x, y, z) \text{ in } \Omega = [-0.5, 0.5]^3,$$

subject to homogeneous Dirichlet boundary conditions. The forcing function is given by

(5.3)
$$f(x,y,z) = 1 + 3\pi^2 \sin(\pi x) \sin(\pi y) \sin(\pi z).$$

We discretize the equation in the weak form using high-order finite element method with a polynomial degree of N=7 on a $30\times30\times30$ hexahedral Kershaw mesh with $\varepsilon_{\rm Kershaw}=0.3$, which is used as the mesh of benchmark problems by the Center for Efficient Exascale Discretization [38]. The total degrees of freedom (DoFs) is about 9×10^6 . We solve the system using both PCG and AMP-PCG(FP16), preconditioned with a p-multigrid with a second-order Chebyshev-accelerated Jacobi smoother [35]. On the coarsest level (N=1), an algebraic multigrid solver is applied. The condition number of the preconditioned matrix is about 187, estimated from Ritz value of iteration 153. All experiments are performed using libParanumal [14] on a NVIDIA GTX 4090 GPU.

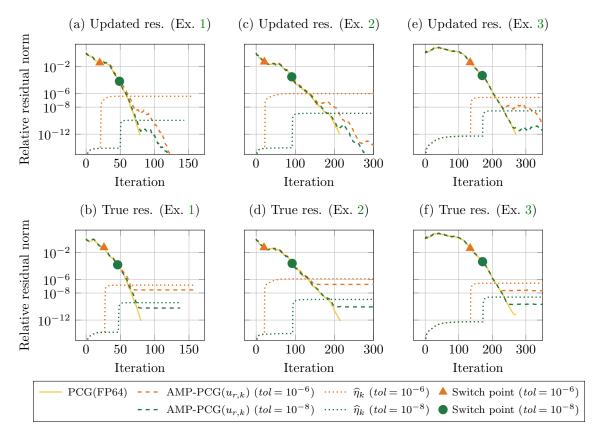


Fig. 3: Convergence curves of the relative updated residual norm ((a), (c), and (e)) and the relative true residual norm ((b), (d), and (f)) for PCG(FP64) and AMP-PCG($u_{r,k}$) applied to Examples 1 to 3, respectively. Two stopping tolerances, 10^{-6} and 10^{-8} , are tested. At the switch point, the precision of \mathbf{r}_k and \mathbf{q}_k is reduced from FP64 to FP32; all other vectors remain in FP64. The dotted curves show the indicator $\hat{\eta}_k$ (see (4.7)), which provides an estimate of the final attainable accuracy.

In Figure 4, we present the convergence curves for PCG(FP64) and AMP-PCG(FP16), where \mathbf{p}_k and \mathbf{z}_k are stored in FP16 throughout all iterations, while the precision of \mathbf{r}_k and \mathbf{q}_k is adaptively reduced according to (4.1) with η_k defined in (4.11). Subfigure (a) and (b), corresponding to the precision selector Example 4 and Example 5, respectively, shows that the convergence of AMP-PCG(FP16) is nearly identical to that of PCG(FP64). The blue circle indicates the iteration at which the precision of \mathbf{r}_k and \mathbf{q}_k is reduced to single precision.

These results demonstrate that when the convergence is linear, using half precision for \mathbf{z}_k and \mathbf{p}_k rarely affect the convergence rate, which is consistent with the analysis presented in subsection 3.2.

5.4. Matrices without large, outlying eigenvalues. We consider matrices whose eigenvalue distributions do not contain large, outlying eigenvalues. These matrices typically exhibit linear convergence after a few initial iterations, allowing the precision of \mathbf{z}_k and \mathbf{p}_k to be reduced once the linear regime is reached. In contrast, matrices with many large, outlying eigenvalues often lead to oscillatory convergence in the residual norm, as demonstrated in subsection 5.6.2. A similar distinction between matrices with and without large, outlying eigenvalues is discussed in [27, 11]. In this subsection, we examine Example 2, Example 3, and a synthetic matrix with uniformly distributed eigenvalues, defined as follows.

Example 6. We consider a matrix $\mathbf{A} \in \mathbb{R}^{(n+1)\times(n+1)}$ with uniformly distributed eigenvalues

$$\lambda_i = \frac{i}{1000n} + \frac{n-i}{n}, \quad i = 0, \dots, n,$$

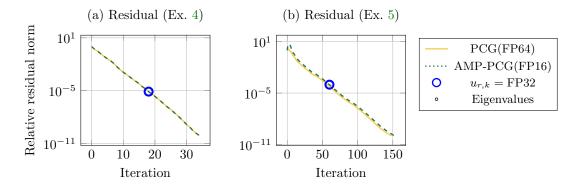


Fig. 4: Convergence of the relative residual norm for PCG(FP64) and AMP-PCG(FP16) on (a) Example 4 and (b) Example 5. In AMP-PCG(FP16), \mathbf{z}_k and \mathbf{p}_k are in half precision throughout the iterations. The blue circle marks where $u_{r,k}$, used for \mathbf{r}_k and \mathbf{q}_k , is reduced from double to single precision.

with n = 100 from $\lambda_n = 10^{-3}$ to $\lambda_0 = 1$. The eigenvector matrix is a random orthonormal matrix.

In Figure 5, we present the convergence curves for PCG(FP64), AMP-PCG(FP64), and AMP-PCG(FP16). AMP-PCG(FP64) initializes all variables in double precision and adaptively reduces their precision according to the precision selector function (4.1). In contrast, AMP-PCG(FP16) uses \mathbf{z}_k and \mathbf{p}_k in half precision throughout all iterations, while \mathbf{r}_k and \mathbf{q}_k are initially in double and subsequently reduced to single precision based on the same function.

Dotted lines in subfigures (a)–(c) show that AMP-PCG(FP16) converges more slowly than PCG(FP64) for all three matrices. In contrast, AMP-PCG(FP64), which stepwise reduces the precision of \mathbf{z}_k and \mathbf{p}_k , achieves convergence nearly identical to that of PCG(FP64). Red and black crosses mark the iterations where $u_{z,k}$ is reduced to single and half precision, respectively. The precision switch thresholds in (5.1) trigger these reductions primarily during the linear convergence regime. As shown in subsection 3.2, using \mathbf{z}_k and \mathbf{p}_k in half precision in this regime has minimal effect on the convergence rate. Blue circles indicate where $u_{r,k}$ is switched to single precision. Subfigures (d)–(f) show the eigenvalue distributions of the matrices from Examples 2, 3, and 6, respectively; none exhibit large, outlying eigenvalues.

These results demonstrate that for matrices without many large outlying eigenvalues, if the convergence is not linear in the initial iterations, it is necessary to start \mathbf{p}_k and \mathbf{z}_k in FP64 and then stepwise reduce their precision to FP32 and FP16 during the linear convergence regime. Moreover, the precision switching thresholds in (5.1) work effective.

5.5. Performance comparison. We compare and analyze the performance of PCG(FP64) and AMP-PCG(FP16) for a large-scale GPU-accelerated high-order finite element simulation.

Example 7. Consider the screened Poisson equation

(5.4)
$$-\Delta u(x, y, z) + \lambda u(x, y, z) = f(x, y, z) \text{ in } \Omega = [-0.5, 0.5]^3,$$

with homogeneous Dirichlet boundary condition and $\lambda=10^3$. The forcing function f(x,y,z) is defined in (5.3). The equation is discretized in a weak form using a high-order finite element method with a polynomial degree of N=7 on a $40\times40\times40$ tensor-product hexahedral mesh, yielding approximately 21.7×10^6 DoFs. To simplify the presentation of the performance analysis, we only consider the Jacobi preconditioner.

Since each physical element is mapped from a reference element via a trilinear mapping, the geometry factors used in matrix-free operations are computed on the fly from the coordinates of the element vertices [7]. This only requires 24 values per element, which is negligible compared to the total number of DoFs. The experiments were performed using *libParanumal* [14] on an NVIDIA

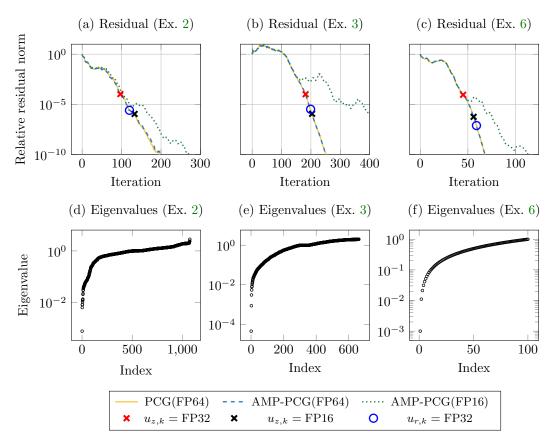


Fig. 5: Convergence curves of the relative residual norm for PCG(FP64), AMP-PCG(FP64), and AMP-PCG(FP16) applied to Examples 2, 3, and 6 in subsection 5.4. The red and black crosses indicate the iterations at which $u_{z,k}$, used for \mathbf{z}_k and \mathbf{p}_k , is set to FP32 and FP16, respectively. The blue circle marks the iterations where $u_{r,k}$ is switched to FP32.

A100 SXM 40GB GPU. For half precision arrays, we use the half2 data type and pad arrays with an additional zero entry when their length is odd.

In Figure 6, we present the convergence curves for PCG(FP64) and AMP-PCG(FP16), where the precision $u_{z,k}$ is fixed at FP16 and the precision $u_{r,k}$ is adaptively selected based on (4.11). The convergence of AMP-PCG(FP16) closely follows that of the PCG(FP64) in this example. AMP-PCG(FP16) process consists of two phases: (1) \mathbf{z}_k and \mathbf{p}_k are stored in FP16, while all other vectors remain in FP64; (2) \mathbf{z}_k and \mathbf{p}_k are in FP16, \mathbf{r}_k and \mathbf{q}_k are in FP32, and the remaining vectors are in FP64. In Figure 7, we show the runtime comparison of PCG(FP64) and the two phases of AMP-PCG(FP16) across five main kernels: \mathbf{zamx} , $\mathbf{innerProd}$, \mathbf{axpy} , \mathbf{ax} , and $\mathbf{updatePCG}$, measured with Nsight Compute. Among these, the Ax kernel is compute-bound and is expected to achieve a $2\times$ speedup since FP32 offers twice the FLOP throughput compared to FP64 on the A100. The other kernels are memory-bound, so their expected speedup can be estimated based on the amount of data movement involved, as summarized in Table 2. Each bar in Figure 7 is annotated with two values: the actual speedup (top line) and the theoretical speedup calculated using Table 2 (bottom line), both relative to the double precision implementation. The actual and theoretical speedups are closely aligned, with significant gains observed in Phase 2 of AMP-PCG. With a stopping criterion of $\|\mathbf{r}_k\| \leq 10^{-10} \|\mathbf{b}\|$, AMP-PCG(FP16) achieves an overall speedup of $1.63\times$ compared to PCG(FP64).

5.6. Limitations of the precision switch criterion. In the previous subsections, we verified the effectiveness of AMP-PCG from the perspectives of attainable accuracy, convergence rate, and GPU performance across various examples, where the assumptions for the convergence are satisfied.

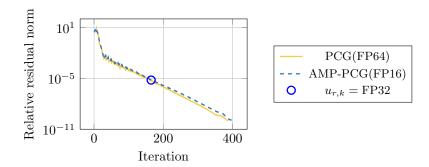


Fig. 6: Convergence curves of the relative residual norm for PCG(FP64) and AMP-PCG(FP16), applied to the screened Poisson equation with high-order finite element discretization and Jacobi preconditioner, as discussed in Example 7. \mathbf{z}_k and \mathbf{p}_k are in half precision throughout the iterations. The blue circles mark when $u_{r,k}$ is switched from double precision to single precision.

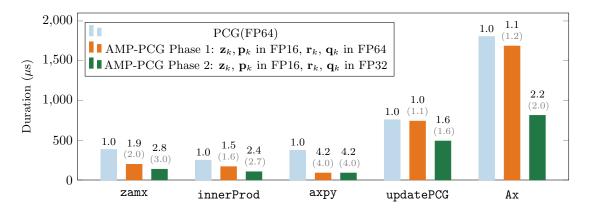


Fig. 7: Runtime of the main kernels in PCG(FP64) and AMP-PCG(FP16) for Example 7, measured on an NVIDIA A100 SXM 40GB GPU using Nsight Compute. Each bar of phase 1 and 2 is annotated with two values: the actual speedup (top line, black) relative to PCG(FP64), and the theoretical speedup (bottom line, in parentheses), estimated based on data movement detailed in Table 2. The convergence behavior is shown in Figure 6. With a stopping criterion of $\|\mathbf{r}_k\| \leq 10^{-10} \|\mathbf{b}\|$, AMP-PCG(FP16) achieves a $1.63 \times$ speedup over the PCG(FP64) baseline.

In subsection 4.1, we assume the residual decays rapidly after d iterations so that (4.8) holds. This condition ensures that η_k provides a good approximation to $\widehat{\eta}_m$ for $m \gg k$. Theorem 3.1 guarantees that the linear convergence rate of the perturbed \mathbf{z}_k is close to that of the exact arithmetic PCG. However, there is no theoretical guarantee on the degradation of the convergence rate when the residual \mathbf{r}_k exhibits oscillatory behavior. In this subsection, we present several examples where these assumptions are not satisfied to illustrate limitations of the precision selector (4.1).

5.6.1. Choice of d in the attainable accuracy indicator. The effectiveness of the attainable accuracy indicator η_k depends on the delay parameter d, which should be large enough such that the inequality (4.8) holds. In previous examples, we set d=10. However, if the convergence is slow or oscillatory, a larger value of d may be required to obtain a reliable estimate of the final attainable accuracy. To illustrate this, we consider the matrix nos6.mat from the SuiteSparse Matrix Collection [17].

Example 8. The matrix nos6.mat, of size 675×675 , is scaled on both sides by the square root of the inverse of its diagonal entries. The condition number after scaling is approximately 3.49×10^6 .

Table 2: Data movement (read and write) per degree of freedom in PCG(FP64) and AMP-						
$PCG(FP16)$, measured in bytes. Here, M denotes the Jacobi preconditioner. N_{ℓ} is the number						
of local DoFs and N_s is the number of global DoFs.						

Kernel	Phase	PCG(FP64)	Phase 1	Phase 2
	Precision	All in FP64	$u_{z,k} = \text{FP16}$	$u_{z,k} = \text{FP16}$
			$u_{r,k} = \text{FP64}$	$u_{r,k} = \text{FP32}$
zamx	$\mathbf{z} = \mathbf{Mr}$	24	12	8
innerProd	$\gamma = \mathbf{z}^T \mathbf{r}$	16	10	6
axpy	$\mathbf{p} = \mathbf{z} + \beta \mathbf{p}$	24	6	6
updatePCG	$\mathbf{r} = \mathbf{r} - \alpha \mathbf{q}, \ \mathbf{x} = \mathbf{x} + \alpha \mathbf{p}, \ \ \mathbf{r}\ $	48	42	30
Ax	$\mathbf{q} = \mathbf{A}\mathbf{p}, \mu = \mathbf{p}^T \mathbf{q}$	$pprox 28 N_{\ell}/N_s$	$pprox 24 N_{\ell}/N_s$	$pprox 14 N_{\ell}/N_s$

We set the stopping tolerance $tol = 10^{-6}$. In Figure 8, we show the convergence curves for both the updated residual (left) and the true residual (right) of PCG and AMP-PCG($u_{r,k}$). In these experiments, we do not terminate the iterations based on the stopping condition in line 18 in Algorithm 4.1, allowing us to observe the final attainable accuracy achieved by the algorithm. Markers indicate the iteration at which the precision of \mathbf{r}_k and \mathbf{q}_k is switched from FP64 to FP32 base on the precision selector (4.1). All other vectors and operations are kept in FP64. In this case, the residual norm initially decreases and then increases, so the condition (4.8) does not hold for d = 10. As a result, η_k fails to provide an upper bound for the attainable accuracy, and the final relative true residual norm exceeds $tol = 10^{-6}$. To improve the accuracy of η_k as an estimator, a larger delay parameter, at least d = 50, would be needed. The difficulty of selecting a proper d also arises in error estimation for CG using the delay strategy [30, 4], and it is also an open research area.

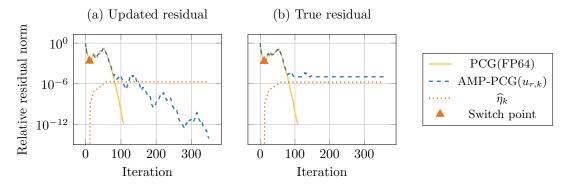


Fig. 8: Convergence curves of (a) relative updated residual norm and (b) relative true residual norm for PCG(FP64), AMP-PCG($u_{r,k}$) with a stopping tolerance 10^{-6} applied to Example 8. In AMP-PCG($u_{r,k}$), the indicator (4.9) with d=10 is used in the precision selector (4.1) to switch the precision of \mathbf{r}_k and \mathbf{q}_k from FP64 to FP32, while all other vectors are in FP64. $\hat{\eta}_k$ is the final attainable accuracy indicator defined in (4.7). Since the residual norm decreases and then increases, the assumption (4.8) does not hold for d=10. As a result, the final accuracy does not meet the target tolerance of 10^{-6} .

5.6.2. Choice of precision switch thresholds $\tau_{z,s}$ and $\tau_{z,h}$. In this subsection, we consider the impact of thresholds $\tau_{z,s}$ and $\tau_{z,h}$ on the convergence rate. Several examples in previous subsections show that applying the AMP-PCG algorithm, i.e., stepwise reduction of the precision of \mathbf{p}_k and \mathbf{z}_k in the linear convergence regime, rarely impacts the convergence rate. However, when

the matrix has many large outlying eigenvalues, it may require a significant number of iterations to reach the linear convergence regime. As a result, selecting appropriate thresholds $\tau_{z,s}$ and $\tau_{z,h}$ can be challenging. We first consider a 10×10 matrix to demonstrate that using the thresholds (5.1) to switch the precision of \mathbf{z}_k and \mathbf{p}_k in AMP-PCG leads to slow convergence relative to PCG(FP64) We then present another example illustrating how the choice of $\tau_{z,s}$ impacts the convergence of AMP-PCG. In [28, 11], it is shown that the convergence rate of PCG in finite precision can degrade when the system matrix has multiple large outlying eigenvalues. We consider a similar matrix in the following example.

Example 9. Consider an SPD matrix with eigenvalues λ_i (i = 1, ..., n). Its largest eigenvalue is $\lambda_n = 1$, while many eigenvalues are clustered near $\lambda_1 = 10^{-3}$, defined as

(5.5)
$$\lambda_i = 10^{-3} + \frac{i-1}{n-1} \rho^{n-i}, i = 1, \dots, n-1,$$

where n = 10 and $\rho = 1/4$. **b** is set to a vector of all ones.

We compare PCG(FP64) with AMP-PCG(FP64), where \mathbf{z}_k and \mathbf{p}_k are initially in double precision and stepwise reduced to single and half precision based on the thresholds in (5.1). To isolate the impact of inexactness in \mathbf{z}_k and \mathbf{p}_k , all other vectors are maintained in double precision. Figure 9 shows the convergence of both variants. In AMP-PCG(FP64), the precision of \mathbf{z}_k and \mathbf{p}_k is reduced to single precision at iteration 9 (red cross), and further reduced to half precision at iteration 11 (black cross). This progressive reduction in precision results in a few additional iterations compared to the PCG(FP64).

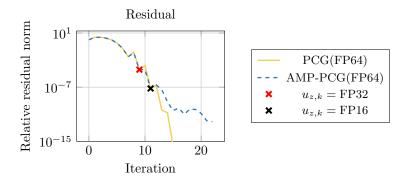


Fig. 9: Convergence curves of the relative residual norm for PCG(FP64) and AMP-PCG(FP64), applied to Example 9. The red and black crosses indicate the iterations at which $u_{z,k}$, used for \mathbf{z}_k and \mathbf{p}_k , is set to FP32 and FP16 in AMP-PCG(FP64), respectively.

In exact arithmetic, CG should converge in at most 10 iterations since $\mathbf{A} \in \mathbb{R}^{10 \times 10}$. However, both PCG(FP64) and AMP-PCG(FP64) require more than 10 iterations to converge. This behavior is explained in [28, 11], where results indicate that in finite precision arithmetic, multiple approximations to the larger eigenvalues may appear before any accurate approximation to the smaller, clustered eigenvalues is obtained. Figure 10 illustrates the Ritz values, i.e. the eigenvalues of the Lanczos tridiagonal matrices \mathbf{T}_k , as the iteration progresses. The x-axis represents the iteration, while the y-axis shows the eigenvalues shifted by $-10^{-3} + 10^{-7}$, which maps the smallest eigenvalue to 10^{-7} , allowing for clearer visualization. The gray horizontal lines represent eigenvalues of \mathbf{A} . Circles denote the Ritz values at each iteration and circles at the same position represent repeated approximation to the same eigenvalue. In the PCG(FP64), the largest eigenvalue is approximated three times. In contrast, AMP-PCG(FP64) detects large eigenvalues even more frequently; the largest eigenvalue is approximated six times, and the next two largest eigenvalues have multiplicity of four in the final iteration. This repeated discovery of large, outlying eigenvalues contributes to slower convergence, as it leads to the loss of linear independence of the Krylov basis vectors.

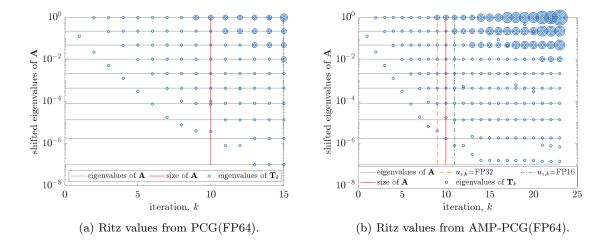


Fig. 10: Comparison of Ritz values generated by PCG(FP64) and AMP-PCG(FP64) for Example 9. The corresponding convergence curves are shown in Figure 9. For better visualization, the y-axis shows eigenvalues shifted by $-10^{-3} + 10^{-7}$, which maps the smallest eigenvalue to 10^{-7} . In AMP-PCG(FP64), the precision of \mathbf{z}_k and \mathbf{p}_k is reduced from FP64 to FP32 at the 9th iteration (dashed orange line), and further to FP16 at the 11th iteration (dash-dotted green line), while all other vectors remain in FP64. Subfigure (a) shows that PCG(FP64) approximates the largest eigenvalue three times in the final iteration, whereas subfigure (b) shows that AMP-PCG(FP64) approximates the largest eigenvalues six times. This repeated approximation of large, outlying eigenvalues contributes to the slower convergence observed in AMP-PCG(FP64).

Next, we consider a test case solved by AMP-PCG using two different thresholds, $\tau_{z,s} = 10^{-4}$ (as in (5.1)) and $\tau_{z,s} = 10^{-8}$. We set $\tau_{z,h} = 0$ (meaning conversion to half precision is never triggered), and all other vectors are represented in double precision.

Example 10. Consider a matrix A with the eigenvalues defined by (5.5) with n = 50 and $\rho = 0.9$.

In Figure 11, we present the convergence curves of the relative residual norm for PCG in FP64 and AMP-PCG(FP64), where \mathbf{z}_k and \mathbf{p}_k start in FP64 and are switched to FP32 based on $\tau_{z,s}$, with red circles indicating the switch points. In subfigure (a), with $\tau_{z,s} = 10^{-4}$, the threshold used in (5.1) and previous examples, the precision switch occurs during an oscillatory phase of the convergence. After switching to FP32, AMP-PCG(FP64) exhibits nearly linear convergence, whereas PCG(FP64) continues to show superlinear convergence. In contrast, in subfigure (b), with $\tau_{z,s} = 10^{-8}$, the switch occurs closer to the linear convergence regime, and the convergence rates of PCG(FP64) and AMP-PCG(FP64) are almost identical.

This example underscores the sensitivity of the convergence rate of AMP-PCG to the precision switch threshold. Specifically, when a matrix has multiple large, outlying eigenvalues, those eigenvalues tend to be approximated frequently, requiring a substantial number of iterations before the linear convergence regime is reached. Consequently, selecting the appropriate iteration to switch precision is nontrivial. The thresholds in (5.1) are empirical and do not perform well in this case. Furthermore, if n = 100 is used in Example 10, the convergence does not exhibit a clearly identifiable linear regime. In this case, reducing the precision of \mathbf{z}_k and \mathbf{p}_k at any iteration leads to slower convergence compared to PCG(FP64). Hence, for AMP-PCG, preconditioning should aim to avoid or reduce large, outlying eigenvalues of the preconditioned system. The impact of roundoff errors in PCG has been widely studied in the context of finite precision [26, 42, 11], and these insights are directly relevant for understanding the convergence behavior of AMP-PCG.

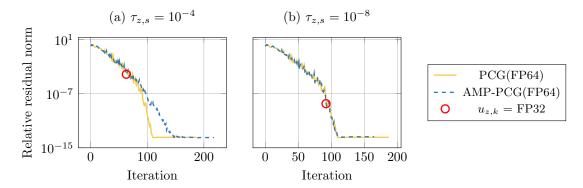


Fig. 11: Convergence curves of the relative true residual norm of PCG(FP64) and AMP-PCG(FP64) for Example 10. In AMP-PCG(FP64), the precision of \mathbf{z}_k and \mathbf{p}_k starts in FP64 and then are switched to FP32 at red circles. $\tau_{z,s}$ is used in (4.1) to determine when to switch precision of \mathbf{z}_k and \mathbf{p}_k . All other vectors are in FP64.

6. Summary. We have presented an adaptive mixed precision and dynamically scaled preconditioned conjugate gradient algorithm for solving sparse linear systems. As demonstrated in the motivating example in section 2, naively implementing PCG in low precision faces three major challenges: reduced attainable accuracy, delayed convergence, and numerical underflow due to the limited dynamic range of half precision. The proposed AMP-PCG algorithm addresses these issues by adaptively selecting the precision of vectors and dynamically scaling the right-hand side vector in the preconditioning step to ensure all vectors remain representable in low precision. An attainable accuracy indicator is introduced to guide the precision switching of \mathbf{r}_k and \mathbf{q}_k from double to single precision. Additionally, we propose an empirical criterion for adjusting the precision of \mathbf{z}_k and \mathbf{p}_k : if convergence is nearly linear, both vectors remain in half precision throughout the iterations; if convergence is superlinear, they begin in double precision and switch to lower precision once the linear convergence is reached.

Our experiments on a broad set of sparse linear systems demonstrate the strong potential of the proposed algorithm. We show that across a wide variety of problems, the convergence rate remains unaffected by adaptive mixed precision strategies, with final solution accuracy comparable to uniform double precision implementations. We also analyze the performance of a large-scale GPU-accelerated high-order finite element discretization with a Jacobi preconditioner, in which \mathbf{z}_k and \mathbf{p}_k are maintained in half precision throughout the iterations. The vectors \mathbf{r}_k and \mathbf{q}_k initially use double precision and are later reduced to single precision. This use of low precision significantly reduces data movement and floating-point operations in key kernels, achieving an overall speedup of about 1.63 on an A100 GPU while maintaining the accuracy of the computed solution.

The potential shortcomings of the precision selector function (4.1) are discussed in subsection 5.6. For matrices with many large, outlying eigenvalues, selecting an appropriate threshold for switching the precision of \mathbf{z}_k and \mathbf{p}_k can be challenging. An improper switch may degrade the convergence rate. This highlights the need for either a more robust precision-switching criterion or the use of an effective preconditioner for such matrices. Another important consideration is the trade-off between iteration count and per-iteration cost. Although AMP-PCG may require more iterations to converge, each iteration can be significantly cheaper. Balancing convergence rate and computational cost per iteration remains an interesting direction for further investigation. While our performance study focuses on linear systems arising from high-order finite element discretizations with a simple Jacobi preconditioner, greater speedups are expected when using more effective preconditioners, such as p-multigrid. In future work, we plan to optimize low precision implementations of multigrid preconditioners and apply AMP-PCG to exascale, real-world, time-dependent problems.

Appendix A. Proof of Theorem 3.2.

Proof. All variables produced in the DS-PCG process are denoted by hats. We proceed it by

induction. Starting with the first iteration (k = 1)

$$\widehat{\mathbf{z}}_0 = \omega_0 \mathbf{M}^{-1} \widehat{\mathbf{r}}_0 = \omega_0 \mathbf{M}^{-1} \mathbf{r}_0 = \omega_0 \mathbf{z}_0, \quad \widehat{\rho}_0 = \omega_0 \rho_0, \quad \widehat{\mathbf{p}}_0 = \widehat{\mathbf{z}}_0 = \omega_0 \mathbf{p}_0, \quad \widehat{\mathbf{q}}_0 = \omega_0 \mathbf{q}_0.$$

Therefore, we obtain $\hat{\gamma}_0 = \omega_0^2 \gamma_0$, $\hat{\alpha}_0 = \alpha_0/\omega_0$, and thus

$$\widehat{\alpha}_0\widehat{\mathbf{p}}_0 = \alpha_0\mathbf{p}_0, \quad \widehat{\alpha}_0\widehat{\mathbf{q}}_0 = \alpha_0\mathbf{q}_0.$$

Consequently,

$$\hat{\mathbf{x}}_1 = \mathbf{x}_1 \text{ and } \hat{\mathbf{r}}_1 = \mathbf{r}_1.$$

Thus, the claim holds for k = 1.

Next, we assume that $\widehat{\rho}_{k-1} = \omega_{k-1}\rho_{k-1}$, $\widehat{\mathbf{p}}_{k-1} = \omega_{k-1}\mathbf{p}_{k-1}$, $\widehat{\mathbf{x}}_k = \mathbf{x}_k$, and $\widehat{\mathbf{r}}_k = \mathbf{r}_k$ for some positive integer k. We aim to show that the identities also hold at the next iteration: $\widehat{\rho}_k = \omega_k \rho_k$, $\widehat{\mathbf{p}}_k = \omega_k \mathbf{p}_k$, $\widehat{\mathbf{x}}_{k+1} = \mathbf{x}_{k+1}$, and $\widehat{\mathbf{r}}_{k+1} = \mathbf{r}_{k+1}$. Similar to the initial step,

$$\widehat{\mathbf{z}}_k = \omega_k \, \mathbf{M}^{-1} \widehat{\mathbf{r}}_k = \omega_k \, \mathbf{M}^{-1} \mathbf{r}_k = \omega_k \, \mathbf{z}_k, \quad \widehat{\rho}_k = \omega_k \, \rho_k.$$

Moreover, for search direction.

$$\widehat{\mathbf{p}}_k = \widehat{\mathbf{z}}_k + \widehat{\beta}_{k-1} \widehat{\mathbf{p}}_{k-1} = \omega_k \mathbf{z}_k + \frac{\omega_k \rho_k}{\omega_{k-1} \rho_{k-1}} \omega_{k-1} \mathbf{p}_{k-1} = \omega_k \left(\mathbf{z}_k + \frac{\rho_k}{\rho_{k-1}} \mathbf{p}_{k-1} \right) = \omega_k \mathbf{p}_k.$$

By multiplying **A** to both sides, we obtain $\hat{\mathbf{q}}_k = \omega_k \, \mathbf{q}_k$. Then,

$$\widehat{\gamma}_k = \widehat{\mathbf{q}}_k^T \widehat{\mathbf{p}}_k = (\omega_k \, \mathbf{q}_k)^T (\omega_k \, \mathbf{p}_k) = \omega_k^2 \, \gamma_k.$$

Thus, the step size is

$$\widehat{\alpha}_k = \frac{\widehat{\rho}_k}{\widehat{\gamma}_k} = \frac{\rho_k}{\omega_k \, \gamma_k} = \frac{\alpha_k}{\omega_k},$$

with $\alpha_k = \rho_k/\gamma_k$. Hence, the solution update is

$$\widehat{\mathbf{x}}_{k+1} = \widehat{\mathbf{x}}_k + \widehat{\alpha}_k \, \widehat{\mathbf{p}}_k = \mathbf{x}_k + \frac{\alpha_k}{\omega_k} (\omega_k \, \mathbf{p}_k) = \mathbf{x}_k + \alpha_k \, \mathbf{p}_k = \mathbf{x}_{k+1},$$

and the residual update becomes

$$\widehat{\mathbf{r}}_{k+1} = \widehat{\mathbf{r}}_k - \widehat{\alpha}_k \, \widehat{\mathbf{q}}_k = \mathbf{r}_k - \frac{\alpha_k}{\omega_k} (\omega_k \, \mathbf{q}_k) = \mathbf{r}_k - \alpha_k \, \mathbf{q}_k = \mathbf{r}_{k+1}.$$

By induction, the result holds for all iterations k. That is, for every k, $\hat{\mathbf{x}}_k = \mathbf{x}_k$ and $\hat{\mathbf{r}}_k = \mathbf{r}_k$.

References.

- [1] Automatic mixed precision (AMP) documentation. https://pytorch.org/docs/stable/amp. html#gradient-scaling.
- [2] P. AMESTOY, A. BUTTARI, N. J. HIGHAM, J.-Y. L'EXCELLENT, T. MARY, AND B. VIEUBLÉ, Five-precision GMRES-based iterative refinement, SIAM Journal on Matrix Analysis and Applications, 45 (2024), pp. 529–552.
- [3] H. Anzt, J. Dongarra, and E. S. Quintana-Ortí, Adaptive precision solvers for sparse linear systems, in Proceedings of the 3rd International Workshop on Energy Efficient Supercomputing, 2015, pp. 1–10.
- [4] M. ARIOLI, A stopping criterion for the conjugate gradient algorithm in a finite element method framework, Numer. Math., 97 (2004), pp. 1–24, https://doi.org/10.1007/s00211-003-0500-y.
- [5] A. BOURAS AND V. FRAYSSÉ, Inexact matrix-vector products in Krylov methods for solving linear systems: a relaxation strategy, SIAM Journal on Matrix Analysis and Applications, 26 (2005), pp. 660–678.
- [6] A. BOURAS, V. FRAYSSÉ, AND L. GIRAUD, A relaxation strategy for inner-outer linear solvers in domain decomposition methods, CERFACS TR0PA000017, European Centre for Research and Advanced Training in Scientific Computation, (2000).

- [7] Z. CAO, Q. Sun, T. Zhang, and H. Li, Towards a higher roofline for matrix-vector multiplication in matrix-free HOSFEM, arXiv preprint arXiv:2504.07042, (2025).
- [8] E. CARSON, T. GERGELITS, AND I. YAMAZAKI, Mixed precision s-step Lanczos and conjugate gradient algorithms, Numerical Linear Algebra with Applications, 29 (2022), p. e2425.
- [9] E. CARSON AND N. J. HIGHAM, Accelerating the solution of linear systems by iterative refinement in three precisions, SIAM Journal on Scientific Computing, 40 (2018), pp. A817–A847.
- [10] E. Carson and N. Khan, Mixed precision iterative refinement with sparse approximate inverse preconditioning, 45, pp. C131–C153, https://doi.org/10.1137/22M1487709, https://epubs.siam.org/doi/10.1137/22M1487709 (accessed 2024-09-11).
- [11] E. Carson, J. Liesen, and Z. Strakoš, Towards understanding CG and GMRES through examples, Linear Algebra and its Applications, 692 (2024), pp. 241–291.
- [12] E. C. Carson, *The adaptive s-step conjugate gradient method*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 1318–1338.
- [13] E. C. CARSON, M. ROZLOZNIK, Z. STRAKOS, P. TICHY, AND M. TŮMA, The numerical stability analysis of pipelined conjugate gradient methods: Historical context and methodology, SIAM Journal on Scientific Computing, 40 (2018), pp. A3549–A3580.
- [14] N. CHALMERS, A. KARAKUS, A. P. AUSTIN, K. ŚWIRYDOWICZ, AND T. WARBURTON, lib-Paranumal: a performance portable high-order finite element library, 2022, https://doi.org/10.5281/zenodo.4004744, https://github.com/paranumal/libparanumal. Release 0.5.0.
- [15] M. A. CLARK, R. BABICH, K. BARROS, R. C. BROWER, AND C. REBBI, Solving lattice QCD systems of equations using mixed precision solvers on GPUs, Computer Physics Communications, 181 (2010), pp. 1517–1528.
- [16] S. COOLS, E. F. YETKIN, E. AGULLO, L. GIRAUD, AND W. VANROOSE, Analyzing the effect of local rounding error propagation on the maximal attainable accuracy of the pipelined conjugate gradient method, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 426–450.
- [17] T. A. DAVIS AND Y. Hu, The university of Florida sparse matrix collection, ACM Transactions on Mathematical Software (TOMS), 38 (2011), pp. 1–25.
- [18] X. Du, E. Haber, M. Karampataki, D. B. Szyld, et al., Varying iteration accuracy using inexact conjugate gradients in control problems governed by PDE's, in Proceedings of the 2nd Annual International Conference on Computational Mathematics, Computational Geometry and Statistics (CMCGS 2013), 2008, pp. 29–38.
- [19] F. GÖBEL, T. GRÜTZMACHER, T. RIBIZEL, AND H. ANZT, Mixed precision incomplete and factorized sparse approximate inverse preconditioning on GPUs, in Euro-Par 2021: Parallel Processing, L. Sousa, N. Roma, and P. Tomás, eds., Cham, 2021, Springer International Publishing, pp. 550–564.
- [20] D. GÖDDEKE, R. STRZODKA, AND S. TUREK, Performance and accuracy of hardware-oriented native-, emulated- and mixed-precision solvers in FEM simulations, International Journal of Parallel, Emergent and Distributed Systems, 22 (2007), pp. 221–256.
- [21] G. H. GOLUB AND C. F. VAN LOAN, Matrix computations, JHU press, 2013.
- [22] G. H. Golub and Q. Ye, Inexact preconditioned conjugate gradient method with inner-outer iteration, SIAM Journal on Scientific Computing, 21 (1999), pp. 1305–1320.
- [23] S. GRAILLAT, F. JÉZÉQUEL, T. MARY, AND R. MOLINA, Adaptive precision sparse matrix-vector product and its application to Krylov solvers, SIAM Journal on Scientific Computing, 46 (2024), pp. C30–C56.
- [24] A. Greenbaum, Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences, Linear Algebra and its Applications, 113 (1989), pp. 7–63.
- [25] A. Greenbaum, Estimating the attainable accuracy of recursively computed residual methods, SIAM Journal on Matrix Analysis and Applications, 18 (1997), pp. 535 551.
- [26] A. Greenbaum, Iterative methods for solving linear systems, SIAM, 1997.
- [27] A. Greenbaum, H. Liu, and T. Chen, On the convergence rate of variants of the conjugate gradient algorithm in finite precision arithmetic, SIAM Journal on Scientific Computing, 43 (2021), pp. S496–S515.
- [28] A. Greenbaum and Z. Strakos, Predicting the behavior of finite precision Lanczos and conjugate gradient computations, SIAM Journal on Matrix Analysis and Applications, 13 (1992),

- pp. 121-137.
- [29] 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 (2000), pp. 213–229.
- [30] M. R. HESTENES AND E. STIEFEL, Methods of conjugate gradients for solving linear systems, J. Res. Natl. Bur. Stand., 49 (1952), pp. 409–436.
- [31] N. J. Higham, Accuracy and stability of numerical algorithms, SIAM, 2002.
- [32] N. J. HIGHAM AND T. MARY, A new approach to probabilistic rounding error analysis, SIAM Journal on Scientific Computing, 41 (2019), pp. A2815–A2835.
- [33] N. J. HIGHAM AND S. PRANESH, Exploiting lower precision arithmetic in solving symmetric positive definite linear systems and least squares problems, SIAM Journal on Scientific Computing, 43 (2021), pp. A258–A277.
- [34] N. J. HIGHAM, S. PRANESH, AND M. ZOUNON, Squeezing a matrix into half precision, with an application to solving linear systems, SIAM Journal on Scientific Computing, 41 (2019), pp. A2536–A2551.
- [35] A. KARAKUS, N. CHALMERS, K. ŚWIRYDOWICZ, AND T. WARBURTON, A GPU accelerated discontinuous galerkin incompressible flow solver, Journal of Computational Physics, 390 (2019), pp. 380–404.
- [36] M. KAWAI AND K. NAKAJIMA, Low/adaptive precision computation in preconditioned iterative solvers for ill-conditioned problems, in International Conference on High Performance Computing in Asia-Pacific Region, 2022, pp. 30–40.
- [37] M. KLÖWER, S. HATFIELD, M. CROCI, P. D. DÜBEN, AND T. N. PALMER, Fluid simulations accelerated with 16 bits: Approaching 4x speedup on A64FX by squeezing shallowwaters. jl into float16, Journal of Advances in Modeling Earth Systems, 14 (2022), p. e2021MS002684.
- [38] T. Kolev, P. Fischer, A. P. Austin, A. T. Barker, N. Beams, J. Brown, J.-S. Camier, N. Chalmers, V. Dobrev, Y. Dudouit, L. Ghaffari, S. Kerkemeier, Y.-H. Lan, E. Merzari, M. Min, W. Pazner, T. Ratnayaka, M. S. Shephard, M. H. Siboni, C. W. Smith, J. L. Thompson, S. Tomov, and T. Warburton, CEED ECP milestone report: High-order algorithmic developments and optimizations for large-scale GPU-accelerated simulations, Mar. 2021, https://doi.org/10.5281/zenodo.4672664, https://doi.org/10.5281/zenodo.4672664.
- [39] J. LANGOU, J. LANGOU, P. LUSZCZEK, J. KURZAK, A. BUTTARI, AND J. DONGARRA, Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy (revisiting iterative refinement for linear systems), in Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, 2006, pp. 113–es.
- [40] N. LINDQUIST, P. LUSZCZEK, AND J. DONGARRA, Accelerating restarted GMRES with mixed precision arithmetic, IEEE Transactions on Parallel and Distributed Systems, 33 (2021), pp. 1027–1037.
- [41] W. J. MADDOX, A. POTAPCYNSKI, AND A. G. WILSON, Low-precision arithmetic for fast Gaussian processes, in Uncertainty in Artificial Intelligence, PMLR, 2022, pp. 1306–1316.
- [42] G. MEURANT, The Lanczos and conjugate gradient algorithms: from theory to finite precision computations, SIAM, 2006, https://doi.org/10.5555/1177249.
- [43] M. MIN, Y.-H. LAN, P. FISCHER, E. MERZARI, S. KERKEMEIER, M. PHILLIPS, T. RATH-NAYAKE, A. NOVAK, D. GASTON, N. CHALMERS, ET Al., *Optimization of full-core reactor simulations on Summit*, in SC22: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2022, pp. 1–11.
- [44] Y. Notay, Flexible conjugate gradients, SIAM Journal on Scientific Computing, 22 (2000), pp. 1444–1460.
- [45] C. C. Paige, Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem, Linear Algebra and its Applications, 34 (1980), pp. 235–258.
- [46] G. L. Sleijpen and H. A. van der Vorst, Reliable updated residuals in hybrid Bi-CG methods, Computing, 56 (1996), pp. 141–163.
- [47] N. Tian, S. Huang, and X. Xu, Mixed precision block-Jacobi preconditioner: Algorithms, performance evaluation and feature analysis, arXiv preprint arXiv:2407.15973, (2024).

- [48] 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 (2000), pp. 835–852.
- [49] J. H. WILKINSON, Rounding errors in algebraic processes, SIAM, 2023.
- [50] D. Yang, Y. Zhao, Y. Niu, W. Jia, E. Shao, W. Liu, G. Tan, and Z. Jin, Mille-feuille: A tile-grained mixed precision single-kernel conjugate gradient solver on GPUs, in SC24: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2024, pp. 1–16.