Mixed Precision MINRES

Romina Mahinpei *

Project Advisor: Dr. Chen Greif

Abstract

Within the scientific computing community, there is a growing interest in using lower precision formats, such as single and half precision, given the potential for increased speed, reduced energy, and reduced communication costs. Mixed precision algorithms, which combine different precisions, offer a promising avenue to balance speed and accuracy. However, the applicability of mixed precision algorithms varies across different problem domains, thus requiring domain-specific investigations. This paper addresses a gap in the domain of sparse preconditioned saddle-point problems, proposing two mixed precision variants of the Minimal Residual (MINRES) method that vary the arithmetic and storage precision of the preconditioner solves and the matrix-vector products. Using CUDA C, these variants are implemented and then used to make performance measurements on NVIDIA's GeForce RTX 3070 Ti graphics card for Maxwell and Stokes saddle-point problems. Our numerical results suggest that low precision preconditioning is an effective optimization strategy for reducing the runtime of the MINRES iterative solver when applied to the chosen saddle-point problems while also maintaining the desired accuracy of the final solution.

1 Introduction

Most numerical algorithms are implemented in double precision, corresponding to a 64-bit floating-point number format. Although double precision has benefits, there has been a growing interest in incorporating lower precision formats, such as single, half, and even quarter precision, into numerical algorithms [9]. Much of this interest originates from the increased speed, minimal energy usage, and reduced communication costs that lower precision offers relative to its higher precision counterpart [9]. However, lower precision can decrease accuracy because such formats are only accurate to a limited number of digits, thus making them more vulnerable to numerical instabilities and problems like underflow and overflow [13]. As such, to strike a balance between the speed of low precision formats and the accuracy of high precision formats, recent studies have explored *mixed precision algorithms* that combine two or more precisions within their implementation [9]. One example includes the GMRES-based iterative refinement for least-squares problems, which uses three different precisions instead of one [6].

Although mixed precision algorithms sound appealing in theory, their usefulness highly depends on the problem to which they are applied [9]. This implies that different problem domains will require their own investigations on the applicability of mixed precision algorithms. Specifically, the domain of sparse preconditioned saddle-point problems, arising in a variety of applications across computational sciences and engineering [3], remains less explored in this context. Given that saddle-point problems can often be expressed using a symmetric block matrix, the Minimal Residual (MINRES) method is often used as the iterative solver for such problems [3].

To address this gap, this paper proposes two mixed precision variants of MINRES that vary the arithmetic and storage precision of the preconditioner solves and the matrix-vector products respectively. Each of these variants was implemented in CUDA C and then applied to the Maxwell and Stokes saddle-point problems to obtain performance measurements on the GeForce RTX 3070 Ti graphics card. As such, the major contributions of our work include:

- A CUDA C implementation of MINRES for solving sparse preconditioned saddle-point systems with user-specified precisions for the arithmetic and storage of the preconditioner solves and matrix-vector products;

*Department of Computer Science, University of British Columbia, (mahinpei@student.ubc.ca).

- An experimental evaluation of the GPU-based performance gains for two different mixed precision variants of MINRES relative to a baseline variant when applied to the Maxwell and Stokes saddle-point problems.

The remainder of this paper is organized as follows. We first provide relevant background relating to floating-point arithmetic, saddle-point problems, and the MINRES algorithm in section 2 and summarize the relevant literature in section 3. We then describe our CUDA C implementation of MINRES and our mixed precision variants in section 4 before discussing our numerical experiments and results in sections 5 and 6 respectively. We finally conclude with a summary and suggestions for future work in section 7.

2 Background

2.1 Floating-Point Arithmetic

Most current computers use the IEEE Standard for Binary Floating-Point Arithmetic, which provides binary floating-point formats and precise rules for carrying out arithmetic on them [13]. The most recent iteration of the standard consists of the following four floating-point number formats:

- **16-bit half precision**: 1 sign bit, 10 significant bits, 5 exponent bits;
- **32-bit single precision**: 1 sign bit, 23 significant bits, 8 exponent bits;
- **64-bit double precision**: 1 sign bit, 52 significant bits, 11 exponent bits;
- **128-bit quadruple precision**: 1 sign bit, 112 significant bits, 15 exponent bits.

An important quantity when dealing with floating-point arithmetic is the distance from 1 to the next larger floating-point number [13]. This quantity is known as the machine epsilon ϵ and is computed as follows:

$$\epsilon = \beta^{1-t}$$

where t is the precision (number of significant bits including the sign bit) and β is the base of the system, which is 2 on almost all current computers [13].

As a result of having more significant bits and thus a smaller machine epsilon, higher precision formats can represent a *larger proportion of small numbers* [9]. Moreover, having more bits available enables high precision formats to represent a *wider rage* of numbers than their low precision counterparts [9]. Nonetheless, the same features that allow higher precision formats to obtain higher accuracy cause them to have reduced speed and higher energy consumption relative to their lower precision counterparts [9]. Specifically, given their reduced number of bits, numbers in lower precision formats lead to reduced data transfer and a faster movement of data through the data hierarchy [5]. These contrasting advantages of high precision for accuracy and low precision for computational efficiency have thus motivated the development of mixed precision algorithms, which harness the strengths of both formats to optimize performance and precision across various computing tasks [9].

2.2 Saddle-Point Problems

A saddle-point problem is one that can be expressed as a block 2×2 linear system of the form

$$\begin{bmatrix} A & B_1^T \\ B_2 & -C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

where $A \in \mathbb{R}^{n \times n}$ and $B_1, B_2 \in \mathbb{R}^{m \times n}$ and $C \in \mathbb{R}^{m \times m}$ [3]. To be considered a saddle-point problem, the above system needs to satisfy at least one of the conditions outlined in [3]:

1. A is symmetric $A = A^T$

2. The symmetric part of A is positive semidefinite
3. $B_1 = B_2 = B$
4. C is symmetric and positive semidefinite
5. $C = O$ (the zero matrix).

The focus of this paper is on the most standard case of a saddle-point problem, which occurs when all of the conditions above are satisfied. This results in a symmetric linear system of the form

$$\begin{bmatrix} A & B^T \\ B & O \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \tag{1}$$

Both the Maxwell problems, arising from electromagnetics, and the Stokes problems, arising from fluid dynamics, can be expressed as a system satisfying the form in equation 1 and are thus used as the selected problems for our numerical experiments.

2.3 Minimal Residual (MINRES) Method

MINRES is a Krylov subspace method based on Lanczos tridiagonalization that iteratively finds the solution to a symmetric linear system by minimizing the 2-norm of the residual [14]. The algorithm is a specialized version of the Generalized Minimal Residual (GMRES) method, which also supports non-symmetric matrices. As a result of working with a symmetric system, MINRES has constant memory usage and involves a short recurrence. Relative to the Conjugate Gradient (CG) method, an iterative solver specialized for symmetric positive-definite systems, MINRES supports both symmetric definite and indefinite systems but with a slightly more expensive iteration cost [14].

We outline the primary steps of the MINRES method in Algorithm 1 and direct interested readers to [14] for a more detailed derivation. Here, Q_n is the orthogonal matrix whose column space gives an orthonormal basis for the Krylov subspace $\mathcal{K}_n(A, b) = \text{span}\{b, Ab, \ldots, A^{n-1}b\}$, and $\tilde{T}_n$ is a tridiagonal matrix such that $Q_{n+1}^T A Q_n = \tilde{T}_n$. Note that this transformation can be viewed as a (partial) similarity transformation that reduces the matrix A to a tridiagonal form.

Algorithm 1 MINRES Method

Require: A is a symmetric matrix.
1: $q_1 = b/||b||$
2: **for** $k = 1, 2, 3, \ldots$ **do**
3: Perform step k of the Lanczos Algorithm to compute new entries for Q_k and $\tilde{T}_k$.
4: Find y minimizing $||\tilde{T}_k y - ||b||e_1||$.
5: Update $x_k = Q_k y$ and check for convergence.
6: **end for**

In practice, computing new entries on step 3 of Algorithm 1 is done using the Givens rotations and checking for convergence on step 5 is done by tracking an *approximation* to the residual norm that only uses already computed values. As derived by the authors of the MINRES algorithm in [14], the residual norm at the kth iteration can be approximated by:

$$||\tilde{r}_k|| = \beta_1 s_1 s_2 \ldots s_k$$

where $\beta_1 = ||b||$ and s_j is the sine entry of the jth Givens rotation for $j = 1, \ldots, k$. We refer to this approximation as the *computed residual norm* throughout the rest of this paper.

3 Relevant Literature

We briefly summarize existing literature on introducing inexactness into Krylov methods through matrix perturbations and then discuss recent efforts on implementing mixed precision variants of Krylov methods.

3.1 Inexact Krylov Methods

Inexact Krylov subspace methods are a general variant of (exact) Krylov subspace methods that introduce some degree of perturbation into the method under consideration [16, 4, 7]. This perturbation is often introduced through the matrix-vector products in the form:

$$\mathcal{A}_k v = (A + E_k)v$$

where E_k is the perturbation matrix for the matrix-vector product computed at the kth iteration. The matrix-vector perturbation is often formulated in the above manner (instead of using $\mathcal{A}_k v = Av + e_k$) since this formulation can be viewed as modelling the realistic scenario of having incomplete information about A [4].

A notable observation from [16, 4, 7] is that the accuracy of the matrix-vector products can be *relaxed* by letting $||E_k||_2$ grow *using an appropriate relaxation strategy* as the iterations progress without apparent degradation of the convergence of the iterative Krylov method. The most general formulation of this relaxation strategy corresponds to having $||E_k||_2$ bounded by a value proportional to the 2-norm of the most recent residual:

$$||E_k||_2 \leq \frac{l_k}{||r_{k-1}||_2}\varepsilon \tag{2}$$

where $\varepsilon > 0$ is the desired tolerance of the final solution.

As an example, in the numerical experiments of [16], the authors introduce inexactness into the matrix-vector products of the GMRES algorithm using the formulation in (2) with:

$$l_k = \frac{\sigma_{\min}(A)}{m}$$

where $\sigma_{\min}(A)$ is the smallest singular value of A and m is the maximum number of allowed iterations. Even though $\sigma_{\min}(A)$ is expensive to compute, the authors justified this choice by their goal of confirming a *theoretical* bound. They also point out that in cases where avoiding this expensive computation is a must, using $l_k = 1$ may be sufficient for giving a rough yet useful approximate to this bound. Similarly, the authors in [4] introduce inexactness into their matrix-vector products using a similar bound but extend their numerical experiments to cover CG and BiCGStab in addition to GMRES.

Based on the collective results of [16, 4, 7], the major finding was that Krylov methods are generally robust to perturbations in matrix-vector products *provided that the first few products are computed to full accuracy.* Nonetheless, the authors in [16] also point out that even though convergence may be achieved with the proposed relaxation strategies, the *rate of convergence* may deteriorate due to the loss of orthogonality in the Krylov vectors that result from performing the matrix-vector products inexactly.

3.2 Mixed Precision Krylov Methods

The increase in hardware support for lower precision formats has motivated a transition to mixed precision algorithms in an effort to make the best use of computational resources [9]. The reason for this is twofold. The first is that most current GPUs can complete more low precision operations per clock cycle than high precision operations [5]. As an example, the NVIDIA GeForce RTX 3070 Ti graphics card has a theoretical floating-point operations per second (FLOPS) count of 20.31 TFLOPS for single precision and 317.4 GFLOPS

for double precision. The second is that more low precision data can be held in caches relative to its high precision counterpart, and the low precision data can also be moved at a faster rate through the memory hierarchy as a result of the reduced amount of data to be transferred [5].

As such, recent effort has been put towards exploring mixed precision variants of existing algorithms as outlined in [9]. One notable example includes the work done in [6] that proposed an iterative refinement algorithm based on GMRES preconditioned by LU factors (GMRES-IR) that uses three precisions for solving nonsingular linear systems of the form $Ax = b$. In this work, the authors used single precision as their working precision, computed their LU factors in half precision, calculated the residual at each step in double precision, and showed that it is still possible for the system to be solved to full single-precision accuracy. Similar results were obtained for the work done in [10] that used low precision Cholesky factors as preconditioners in GMRES-IR and CG methods.

Most of the mixed precision works discussed above were simulated on CPUs, which offer limited to no special hardware support for low precision operations. As a result, the findings of these works focused on the convergence properties of mixed precision algorithms as opposed to the actual performance gains. More recently, some explorations have been done on the GPU-based performance gains of mixed precision Krylov methods, with most works focusing on the GMRES and preconditioned conjugate gradient (PCG) methods [2, 1]. To the best of our knowledge, little to no exploration has been done on introducing mixed precision arithmetic into the preconditioned MINRES algorithm, with a focus on its applications to sparse saddle-point problems. Our work aims to fill this gap by investigating mixed precision preconditioned MINRES in the context of sparse saddle-point problems and leads us to introduce inexactness through low precision operations into the three block matrices outlined in equation 1. We specifically carry out our exploration on a GPU and obtain performance measurements that would allow us to experimentally verify the performance gains associated with the use of low precision operations within numerical algorithms.

4 Implementation

We use the latest preconditioned MINRES implementation provided by Stanford's Systems Optimization Laboratory in [12] as a reference framework and implement our mixed precision variants of MINRES in CUDA C as further outlined in the following sections. We provide the pseudocode for the main iteration of our implementation (excluding the required set-up) in Algorithm 2. Although the majority of our implementation is consistent with the latest implementation in [12], one difference is that we have a single stopping condition rather than multiple different ones. This choice was made in an effort to better understand the effects of mixed precision operations on the convergence of MINRES across our selected saddle-point problems.

Further note that Algorithm 2 primarily tracks the *computed residual norm* of the preconditioned system in order to minimize the number of matrix-vector products that computing the true residual would require. However, once the computed residual norm of the preconditioned system satisfies the stopping condition (line 32), the true residual is computed and the stopping condition is checked once again (line 34) to ensure that the final solution truly satisfies the user-specified accuracy. If the true residual does not satisfy the stopping condition, the tolerance is scaled down by a factor of 10 (line 37), and the algorithm continues to perform more MINRES iterations until the stopping condition is satisfied once again. This convergence scheme is the same as what is implemented in [12] and is kept in our implementation to ensure consistency.

4.1 Mixed Precision Variants

Observing the steps in Algorithm 2, we note that the two most expensive operations are the matrix-vector products (line 4) and the preconditioner solves (line 12). As such, we decided to lower the precision of these two operations with the hopes of making performance gains while maintaining the desired user-specified accuracy in our final solution. This resulted in us investigating the following baseline variant along with two mixed precision variants of MINRES:

Algorithm 2 Preconditioned MINRES iteration supporting mixed precision operations

Require: User-specified matrix-vector product precision u_m and preconditioning precision u_p.

while $i \leq$ itnlim **do**
  $s \leftarrow \frac{1}{\beta}$
  $v \leftarrow s * y$
  $y \leftarrow \begin{bmatrix} A & B' \\ B & 0 \end{bmatrix} v$ ▷ **Perform matrix-vector product in precision** u_m
  if $i \geq 2$ **then**
    $y \leftarrow -\frac{\beta}{\beta_{\text{prev}}} r_1 + y$
  end if
  $\alpha \leftarrow v \cdot y$
  $y \leftarrow -\frac{\alpha}{\beta} r_2 + y$
  $r_1 \leftarrow r_2$
  $r_2 \leftarrow y$
  $y \leftarrow \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix} \backslash r_2$ ▷ **Perform preconditioning in precision** u_p
  $\beta_{\text{prev}} \leftarrow \beta$
  $\beta \leftarrow \sqrt{r_2 \cdot y}$
  $\epsilon_{\text{prev}} \leftarrow \epsilon$ ▷ Apply previous rotation
  $\delta \leftarrow cs * \bar{d} + sn * \alpha$
  $\bar{g} \leftarrow sn * \bar{d} - cs * \alpha$
  $\epsilon \leftarrow sn * \beta$
  $\bar{d} \leftarrow -cs * \beta$
  $\gamma \leftarrow \sqrt{\bar{g}^2 + \beta^2}$ ▷ Compute next plane rotation
  $\gamma \leftarrow \max(\gamma, \epsilon)$
  $cs \leftarrow \frac{\bar{g}}{\gamma}$
  $sn \leftarrow \frac{\beta}{\gamma}$
  $\phi \leftarrow cs * \bar{\phi}$
  $\bar{\phi} \leftarrow sn * \bar{\phi}$
  $w_1 \leftarrow w_2$ ▷ Update x and the computed rnorm
  $w_2 \leftarrow w$
  $w \leftarrow (v - \epsilon_{\text{prev}} * w_1 - \delta * w_2) * \frac{1}{\gamma}$
  $x \leftarrow x + \phi * w$
  rnorm $\leftarrow \bar{\phi}$

  if rnorm $\leq ||b|| *$rtol **then** ▷ Check stopping condition against the computed rnorm
    rnormk $\leftarrow ||b - \begin{bmatrix} A & B' \\ B & 0 \end{bmatrix} x||$
    if rnormk $\leq ||b|| *$rtol0 **then** ▷ Check stopping condition against the actual rnorm
      break
    else
      rtol $\leftarrow$ rtol/10
    end if
  end if
end while

Table 1: MATLAB CPU time measurements for the baseline and Variant 1 execution of MINRES across three Maxwell saddle-point problems of varying sizes, with the reported measurements reflecting the time (averaged across five runs) taken by each scenario to converge to a final solution with a tolerance of 10^{-6}. The baseline variant stores all matrices in sparse double precision format while Variant 1 stores the matrices A and B for the matrix-vector products in sparse double precision format and the matrices for the preconditioner solves M_1 and M_2 in full single precision format.

Problem Name	**Size**	**Baseline Avg Time (s)**	**Variant 1 Avg Time (s)**
Maxwell 1	1,985	5.09e-03	2.23e-02
Maxwell 2	8,065	2.24e-02	2.71e-01
Maxwell 3	32,513	2.58e-01	4.18e+01

- **Baseline**: Double precision preconditioner solves and double precision matrix-vector products.
- **Variant 1**: Single precision preconditioner solves and double precision matrix-vector products.
- **Variant 2**: Double precision preconditioner solves and single precision matrix-vector products.

For the single precision preconditioner solves that are a part of Variant 1, we use single precision copies of matrices M_1 and M_2 (line 12) so that the overall computation is performed in that precision. We then *cast* the computed value to double precision to ensure that the remainder of the algorithm is performed in double precision. Similarly, for the single precision matrix-vector products that are a part of Variant 2, we use single precision copies of matrices A and B (line 4) so that the overall computation is performed in that precision and then cast the computed value to double precision.

Note that Variant 2 must store both single precision *and* double precision copies of matrices A and B. This arises from the need to compute the actual residual to double precision accuracy on Line 33. In contrast, this requirement does not apply to the single precision preconditioner solves, which enables us to only store single precision copies of matrices M_1 and M_2.

4.2 CUDA C Implementation

Although MATLAB is a commonly used software within the numerical linear algebra community, it has its limitations when exploring mixed precision algorithms, especially when dealing with *sparse* matrices. Currently, MATLAB only supports double precision sparse matrices and does not support its single precision counterparts. As a result, one is forced to convert a sparse matrix to full format, which can then be stored in single precision for mixed precision schemes. As highlighted by the measurements in table 1, this conversion to a full formatted matrix has significant impacts on performance, especially for large problems. As such, we took the latest implementation of MINRES in [12] as a reference and implemented a CUDA C version of the algorithm, which allowed us to store our low precision matrices in sparse format and make all of our performance measurements on a GeForce RTX 3070 Ti graphics cards. Moreover, we maintain our CUDA C implementation as an open-source project for others to use for further investigations.

5 Numerical Experiments

Our numerical experiments were executed with our CUDA C implementation of MINRES on an NVIDIA GeForce RTX 3070 Ti graphics card. We focused on two classes of problems, Maxwell and Stokes, and made performance measurements for three problems of varying sizes for each class. We provide the metadata on the dimensionality of our chosen Maxwell and Stokes problems in table 2 and the sparsity patterns of our selected problems in appendix A.1.

The chosen Maxwell problems arise from finite element discretization of the mixed formulation of the time-harmonic Maxwell equations in lossless media with perfectly conducting boundaries, with the problem deriva-

Table 2: Metadata on the Maxwell and Stokes saddle-point problems used in our numerical experiments.

Problem Name	**Size**	**Dimensions of A**	**Size of B**
Maxwell 1	1,985	1,504 × 1,504	481 × 1504
Maxwell 2	8,065	6,080 × 6,080	1,985 × 6,080
Maxwell 3	32,513	24,448 × 24,448	8,065 × 24,448
Stokes 1	2,946	2,178 × 2,178	768 × 2,178
Stokes 2	11,522	8,450 × 8,450	3,072 × 8,450
Stokes 3	45,579	33,282 × 33,282	12,288 × 33,282

tion being further described in [8]. We use the proposed block diagonal preconditioner in [8] to precondition our Maxwell saddle-point systems:

$$\mathcal{P} = \begin{bmatrix} A+M & 0 \\ 0 & L \end{bmatrix}$$

where A is the discrete curl-curl operator, M is the mass matrix, and L is the scalar Laplacian. We also use the MATLAB code provided by one of the co-authors in [8], to generate the Maxwell matrices and preconditioners.

The chosen Stokes problems arise from a classic fluid dynamics test problem, known as cavity-driven flow. As described in [11], this problem is a model of the flow in a square cavity with the lid moving from left to right, with different choices of the nonzero horizontal velocity resulting in different computational models. Our selected problems specifically focus on the regularized variant with non-uniform streamlines and Q2-P1 discretization. We used the Incompressible Flow & Iterative Solver Software (IFISS) to generate the Stokes matrices and preconditioners [15], with the block diagonal preconditioner being of the form:

$$\mathcal{P} = \begin{bmatrix} \tilde{A} & 0 \\ 0 & Q \end{bmatrix}$$

where $\tilde{A}$ is an approximation to A represented by its incomplete Cholesky factors and Q is the mass matrix.

Note that our MINRES implementation outlined in Algorithm 2 requires a direct preconditioner solve on line 12, which was also the case in the latest implementation in [12]. As such, we precomputed the Cholesky factors for all of the preconditioners in our test problems and stored our preconditions via these factors. We then used these factors to perform lower and upper triangular solves in order to directly perform our preconditioner solves. Faster procedures may be used in practice for these preconditioner solves, such as using incomplete Cholesky factors instead of full Cholesky factors, but we leave this as future work.

6 Results and Discussion

6.1 Low Precision Preconditioning

We first compare the results between the baseline variant and Variant 1, which performed the preconditioner solves in single precision. Observing the measurements in table 3, we note that the baseline variant and Variant 1 took a similar number of iterations to converge, with Variant 1 sometimes taking one more iteration than the baseline. On the other hand, Variant 1 always took less time than the baseline, resulting in speedup factors between 1.3-1.6, as shown in table 4.

We also provide the convergence plots comparing the trends in the relative residual of the preconditioned system between the baseline variant and Variant 1 across all test problems in figure 1. For the Maxwell problems, the baseline variant and Variant 1 start with overlapping relative residuals that diverge near the

Table 3: CUDA C performance measurements for the executions of the baseline, Variant 1, and Variant 2 of MINRES across three Maxwell and three Stokes saddle-point problems of varying sizes, with the reported measurements reflecting the number of iterations and the time (averaged across five runs) taken by each scenario to converge to a final solution with a tolerance of 10^{-6}.

Problem Name	**Size**	**Baseline # Iterations**	**Variant 1 # Iterations**	**Variant 2 # Iterations**	**Baseline Avg Time (s)**	**Variant 1 Avg Time (s)**	**Variant 2 Avg Time (s)**
Maxwell 1	1,985	4	5	—	8.54e-03	5.46e-03	—
Maxwell 2	8,065	4	5	—	4.20e-02	2.85e-02	—
Maxwell 3	32,513	4	5	—	3.32e-01	2.22e-01	—
Stokes 1	2,946	59	59	59	3.44e-02	2.37e-02	3.36e-02
Stokes 2	11,522	92	93	93	1.12e-01	7.87e-02	1.11e-01
Stokes 3	45,570	196	197	198	5.51e-01	4.03e-01	5.45e-01

Table 4: Speedup factors for Variant 1 and Variant 2 relative to the baseline across all Maxwell and Stokes test problems. Note that Variant 2 does not have speedup factors for the Maxwell problems since none of the executions converged.

Problem Name	**Variant 1 Speedup**	**Variant 2 Speedup**
Maxwell 1	1.56	—
Maxwell 2	1.48	—
Maxwell 3	1.49	—
Stokes 1	1.45	1.02
Stokes 2	1.42	1.01
Stokes 3	1.37	1.01

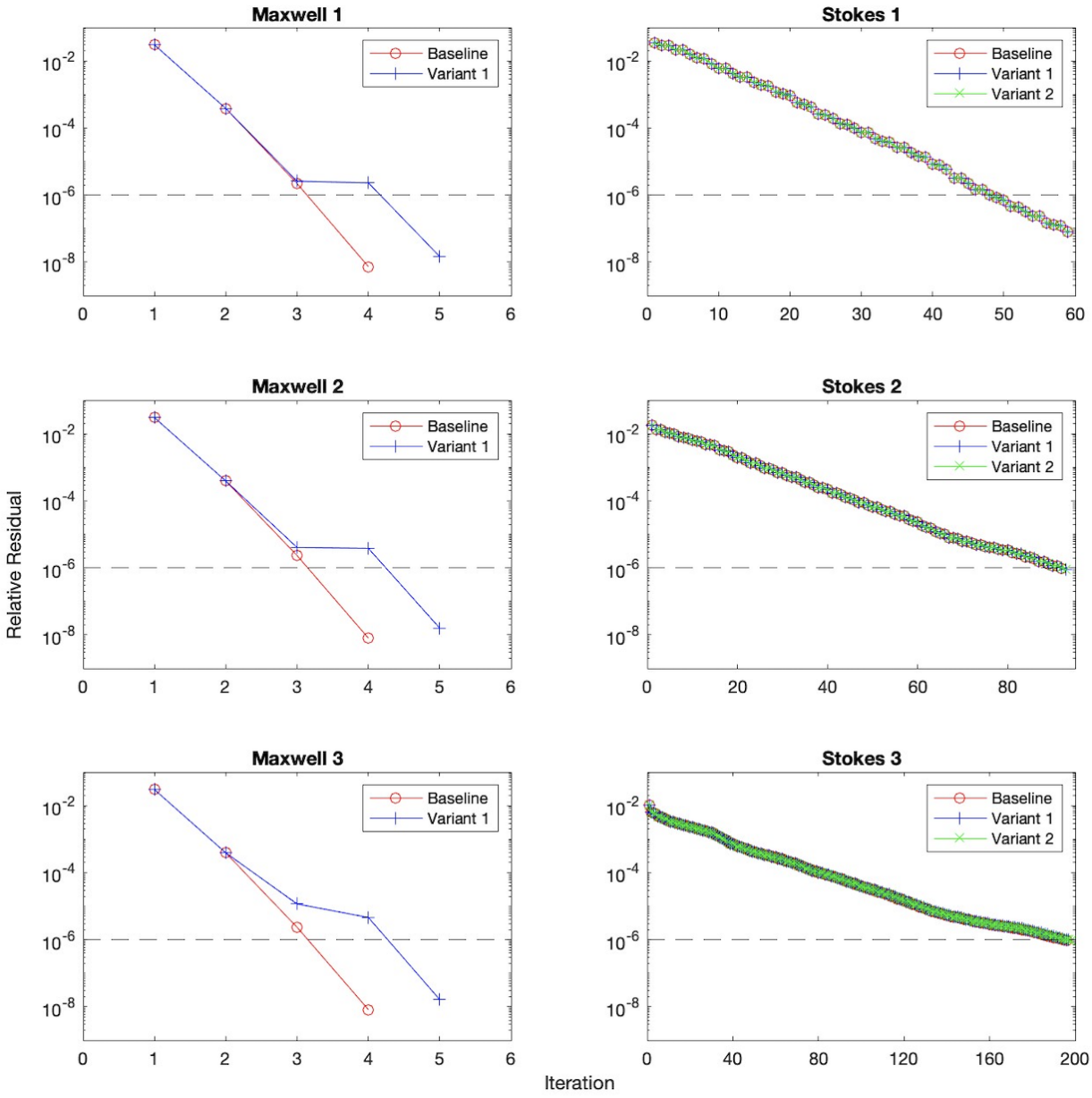
final few iterations. For the Stokes problems, the baseline variant and Variant 1 have overlapping relative residuals for all the iterations at the graphical scale. Furthermore, note that Stokes 3 continued performing more MINRES iterations even after reaching a tolerance of 1e-06. This is a byproduct of the fact that our MINRES implementation tracks the *computed* relative residual but also checks the *true* relative residual once the computed value satisfies the user-specified tolerance. As was the case with Stokes 3, if the true residual does not satisfy the stopping condition, the tolerance is scaled down by a factor of 10 and the algorithm continues performing more MINRES iterations until the stopping condition is satisfied once again.

The numerical results above suggest that low precision preconditioning is indeed an effective optimization strategy for reducing the overall runtime of the MINRES iterative solver without sacrificing the desired accuracy of the final solution. This observation can be reasoned by noting that the preconditioner itself acts as an approximation to the inverse matrix and that the error introduced by this approximation is likely to be more significant than the error introduced by using low precision arithmetic. We further note that these results obtained with Variant 1 also extend to cases where a smaller tolerance is set by the user. As an example, when running Maxwell 3 with a tolerance of 1e-10, a speedup of 1.34 was obtained.

6.2 Low Precision Matrix-Vector Products

We now compare the results between the baseline variant and Variant 2, which performed the matrix-vector products in single precision. For the Maxwell problems, none of the executions of Variant 2 converged, with the actual relative residual plateauing at approximately 3.0e-05, 1.0e-04, and 5.0e-04 for problems 1 through 3 respectively. In contrast, for the Stokes problems, executions of Variant 2 did indeed converge and often took the same number of iterations as the baseline. However, as shown in table 4, the speedup gained from Variant 2 is relatively negligible, suggesting that the switch to single precision matrix-vector products does not offer much performance gains. For the sake of comparison, we also provide the convergence plots

Figure 1: Convergence plots tracking the trajectory of the *computed* relative residual of the preconditioned system between the baseline and Variants 1 and 2 across all test problems. The dashed horizontal line in each plot indicates the user-specified tolerance for the final solution (1e-06). Note that the Maxwell problems do not show the trajectory for Variant 2 since none of the executions converged.



comparing the trends in the relative residual of the preconditioned system between the baseline variant and Variant 2 across the Stokes problems in figure 1.

The absence of convergence with some executions of Variant 2 may seem surprising at first, but existing

numerical experiments and the relevant literature discussed in section 3 suggest that these results are to be expected due to the *loss of orthogonality in the Kyrlov basis vectors*. Firstly, we note that similar conjectures were made of the GPU-based experiments performed by [2, 1] regarding the loss of orthogonality. In [2], the authors stored the diagonals of the preconditioner to the PCG method in different precisions and noted that applying the preconditioner at lower precisions may result in the Krylov vectors no longer being orthogonal. Similarly, the authors in [1] stored the Krylov vectors computed by the GMRES algorithm in different precisions while all arithmetic operations were performed in double precision. By tracking the orthogonality between the Krylov vectors for their different variants, the authors observed that the orthogonality of the Krylov basis vectors degraded when stored in lower precision formats.

In a similar manner, the existing literature in [16] and [4] discuss that inexact Krylov subspace methods may suffer from a lack of convergence due to the loss of orthogonality in the computed Krylov vectors. This is what likely caused the relative residual for the Maxwell problems to plateau at a value larger than the user-specified tolerance. Nonetheless, both [16] and [4] also suggest that it is indeed possible to introduce inexactness into the matrix-vector products that constitute Krylov subspace methods while still achieving convergence.

A major observation from [16] and [4] is that the amount of inexactness introduced into a matrix-vector product is *inversely proportional* to the norm of the computed residual as highlighted by the formulation in equation 2 from section 3.1. This suggests that the accuracy of the matrix-vector products can be relaxed for later iterations, with [4] further emphasizing that the *first few Krylov vectors* must be computed to full accuracy. We further explore this idea in the subsection that follows.

6.3 Adaptive Precision Matrix-Vector Products

Although both [16] and [4] increased the inexactness in their matrix-vector products through a perturbation matrix, we believe that these results can also be extended to the inexactness that is introduced by performing the matrix-vector products in lower precision. To numerically confirm this, we implemented an *adaptive* version of Variant 2 that performed the matrix-vector products of the first two MINRES iterations in double precision and the products of the remaining iterations in single precision. We specifically applied this adaptive scheme to the Maxwell problems with the corresponding convergence plots being provided in figure 2. In contrast to the original version of Variant 2, the Maxwell problems did indeed converge with the adaptive version of Variant 2, confirming their need to perform the first few matrix-vector products to full accuracy before transitioning to a lower precision.

To verify whether the loss of orthogonality had an impact on these convergence behaviours, we also measured the absolute value of the dot product between the two most recently computed Krylov vectors (v_k and v_{k-1}) as a proxy measure for the loss of orthogonality. Given that the Krylov vectors are supposed to be orthogonal in the absence of floating-point issues, a positive value of this measure indicates a loss in orthogonality. Note that we only computed the dot product between the two most recently computed Krylov vectors (as opposed to including all the previous Krylov vectors) since each iteration of our MINRES implementation already computes v_{k-1} (line 3 in Algorithm 2) and v_k (product of y from Line 12 and $1/\beta$ from line 14 in Algorithm 2). As such, no additional storage nor significant computation costs were added to our MINRES iterations as a part of tracking this proxy measure.

We provide the trajectory plots of our proxy measure for the baseline, Variant 2, and the adaptive version of Variant 2 across all Maxwell test problems in figure 3. As expected, we observed that the baseline and the adaptive version of Variant 2 had corresponding orthogonality trajectories, with both these variants seeing a reduction in the loss of orthogonality as the iterations progressed. In contrast, Variant 2 experienced a spike in the loss of orthogonality at iteration 3 and diverged from the matching trajectories of the two other variants for the remaining iterations, thus confirming the need to perform the first few matrix-vector products to full accuracy before transitioning to a lower precision.

Figure 2: Convergence plots tracking the trajectory of the *computed* relative residual of the preconditioned system between the baseline and the adaptive version of Variant 2 across all Maxwell test problems. The dashed horizontal line in each plot indicates the user-specified tolerance for the final solution (1e-06).

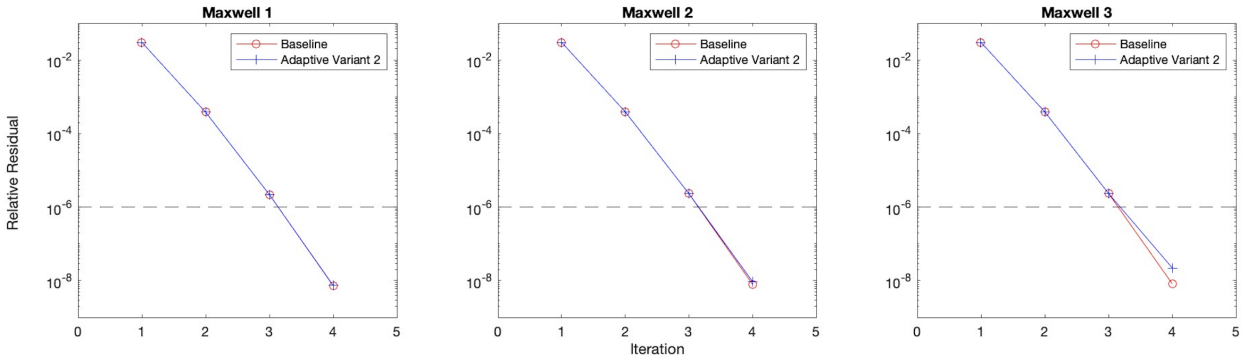
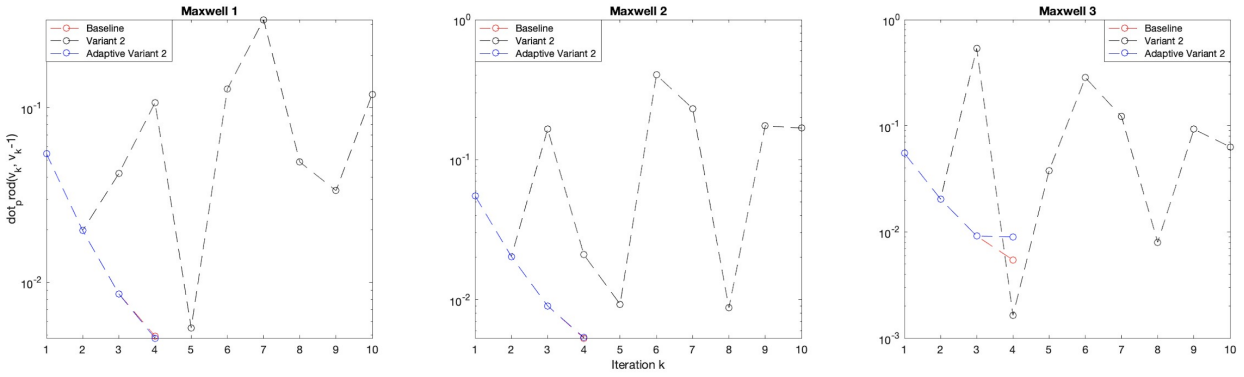


Figure 3: Orthogonality trajectory plots tracking the (absolute value of) the dot product between the two most recently computed Lanczos vectors (v_k and v_{k-1}) for the baseline, Variant 2, and the adaptive version of Variant 2 across all Maxwell test problems. Note that Variant 2 never converged and reached the maximum number of allowed iterations (500), but the trajectory plots only exhibit the behaviour up to iteration 10.



7 Conclusion

In this paper, we proposed two mixed precision variants of MINRES for solving preconditioned sparse saddle-point systems and implemented them in CUDA C. Our first variant (Variant 1) performed the preconditioner solves in single precision while our second variant (Variant 2) performed the matrix-vector products in single precision. We then made performance measurements for both these variants on NVIDIA's GeForce RTX 3070 Ti graphics card for three Maxwell and three Stokes saddle point problems of varying sizes, with our mixed precision results being compared against a baseline implementation that performed all operations in double precision.

We found that Variant 1 with single precision preconditioning obtained speedup factors ranging between 1.3-1.6 while still obtaining a final solution with a tolerance of 10^{-6}. In contrast, we did not obtain significant speedup factors using Variant 2 with single precision matrix-vector products. Moreover, Variant 2 did not even converge for any of our Maxwell problems. As suggested by the existing literature and our

numerical experiments, this was most likely caused by the loss of orthogonality in the Krylov vectors and could potentially be mitigated by lowering the precision of the matrix-vector products at later iterations. Following this recommendation and lowering the precision of the matrix-vector products from double to single precision after two iterations, we indeed observed convergence using the adaptive version of Variant 2 across all three Maxwell problems. Regardless, further investigation is required on when this transition from high to low precision matrix-vector products should exactly occur.

As for future work, we plan to investigate performance measurements on larger saddle-point problems in order to explore how the speedup factors for our mixed precision variants vary when the dimensionality of the system under investigation increases. In addition, we plan to investigate performance measurements for the same problems covered in this paper on different GPU architectures in an effort to highlight how the specs of the underlying architectures can impact the performance gains of mixed precision algorithms.

References

[1] J. I. Aliaga, H. Anzt, T. Grützmacher, E. S. Quintana-Ortí, and A. E. Tomás. "Compressed basis GMRES on high-performance graphics processing units". In: *The International Journal of High Performance Computing Applications* 37.2 (2023), pp. 82–100. DOI: 10.1177/10943420221115140.

[2] H. Anzt, J. Dongarra, G. Flegar, N. J. Higham, and E. S. Quintana-Ortí. "Adaptive precision in block-Jacobi preconditioning for iterative sparse linear system solvers". In: *Concurrency and Computation: Practice and Experience* 31.6 (2019). DOI: 10.1002/cpe.4460.

[3] M. Benzi, G. H. Golub, and J. Liesen. "Numerical solution of saddle point problems". In: *Acta Numerica* 14 (2005), pp. 1–137. DOI: 10.1017/S0962492904000212.

[4] A. Bouras and V. Fraysse. "Inexact Matrix-Vector Products in Krylov Methods for Solving Linear Systems: A Relaxation Strategy". In: *SIAM Journal on Matrix Analysis and Applications* 26.3 (2005), pp. 660–678. DOI: 10.1137/S0895479801384743.

[5] A. Buttari, J. Dongarra, J. Kurzak, P. Luszczek, and S. Tomov. "Using Mixed Precision for Sparse Matrix Computations to Enhance the Performance while Achieving 64-bit Accuracy". In: 34.4 (2008). ISSN: 0098-3500. DOI: 10.1145/1377596.1377597.

[6] E. Carson and N. J. Higham. "Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions". In: *SIAM Journal on Scientific Computing* 40.2 (2018), A817–A847. DOI: 10.1137/17M1140819.

[7] L. Giraud, S. Gratton, and J. Langou. "Convergence in Backward Error of Relaxed GMRES". In: *SIAM Journal on Scientific Computing* 29.2 (2007), pp. 710–728. DOI: 10.1137/040608416.

[8] C. Greif and D. Schotzau. "Preconditioners for the discretized time-harmonic Maxwell equations in mixed form". In: *Numer. Linear Algebra Appl.* 14.4 (2007), pp. 281–297. DOI: 10.1002/nla.515.

[9] N. J. Higham and T. Mary. "Mixed precision algorithms in numerical linear algebra". In: *Acta Numerica* 31 (2022), pp. 347–414. DOI: 10.1017/S0962492922000022.

[10] N. J. Higham and S. Pranesh. "Exploiting Lower Precision Arithmetic in Solving Symmetric Positive Definite Linear Systems and Least Squares Problems". In: *SIAM Journal on Scientific Computing* 43.1 (2021), A258–A277. DOI: 10.1137/19M1298263.

[11] D. S. Howard Elman and A. Wathen. "Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics". In: Oxford University Press, June 2014. Chap. 3, pp. 119–188. ISBN: 9780199678792. DOI: 10.1093/acprof:oso/9780199678792.001.0001.

[12] S. S. O. Laboratory. *MINRES: Sparse Symmetric Equations*. 2020. URL: https://web.stanford.edu/group/SOL/software/minres/ (visited on 02/18/2024).

[13] M. L. Overton. *Numerical Computing with IEEE Floating Point Arithmetic*. Society for Industrial and Applied Mathematics, 2001. DOI: 10.1137/1.9780898718072.

[14] C. C. Paige and M. A. Saunders. "Solution of Sparse Indefinite Systems of Linear Equations". In: *SIAM Journal on Numerical Analysis* 12.4 (1975), pp. 617–629. DOI: 10.1137/0712047.

[15] D. Silvester, H. Elman, and A. Ramage. *Incompressible Flow Iterative Solver Software*. 2014. URL: https://personalpages.manchester.ac.uk/staff/david.silvester/ifiss/default.htm (visited on 02/22/2024).

[16] V. Simoncini and D. B. Szyld. "Theory of Inexact Krylov Subspace Methods and Applications to Scientific Computing". In: *SIAM Journal on Scientific Computing* 25.2 (2003), pp. 454–477. DOI: 10.1137/S1064827502406415.

A Appendix

A.1 Sparsity Patterns

Figure 4: Sparsity patterns of the Maxwell and Stokes saddle-point problems used in our numerical experiments.