DOI: 10.14529/jsfi210301

Evaluating Performance of Mixed Precision Linear Solvers with Iterative Refinement

Boris I. Krasnopolsky[1], *Alexey V. Medvedev*[1]

The solution of systems of linear algebraic equations is among the time-consuming problems when performing the numerical simulations. One of the possible ways of improving the corresponding solver performance is the use of reduced precision calculations, which, however, may affect the accuracy of the obtained solution. The current paper analyzes the potential of using the mixed precision iterative refinement procedure to solve the systems of equations occurring as a result of the discretization of elliptic differential equations. The paper compares several inner solver stopping criteria and proposes the one allowing to eliminate the residual deviation and minimize the number of extra iterations. The presented numerical calculation results demonstrate the efficiency of the adopted algorithm and show about the decrease in the solution time by a factor of 1.5 for the turbulent flow simulations when using the iterative refinement procedure to solve the corresponding pressure Poisson equation.

Keywords: systems of linear algebraic equations, elliptic equations, algebraic multigrid methods, iterative refinement, mixed precision calculations.

Introduction

The solution of systems of linear algebraic equations (SLAEs) is a typical task when performing most of the high performance computing-related applications. This issue motivates the researchers to both develop novel mathematical algorithms for solving SLAEs and improve the implementation aspects to provide better correspondence between the numerical methods and modern compute platforms hardware.

The use of reduced precision floating-point numbers [4, 8, 14] or lower size integer numbers [19] is a well-known way to improve the performance of the calculations, which is discussed in the literature for some time. The reduced precision floating-point calculations can be beneficial for both memory-bound and compute-bound applications [18]. Decreasing the size of the data types allows decreasing the amount of memory traffic. Besides, multiple more floating-point arithmetic operations can be performed per each CPU processor cycle for the vectorized code sections when performing the calculations.

A new wave of interest in the topic of mixed precision calculations is associated with the emergence of the compute devices bringing the hardware support for the half-precision floating-point numbers (e.g., NVIDIA Pascal and Volta GPUs). While the driving force of this innovation is related to machine learning applications, the potential of fold calculations speed up compared to the double precision calculations raised the interest in many other areas including numerical modeling. For example, the use of mixed precision calculations is among the key directions of improving the performance for the exascale systems and computational codes [2]. The publication [2] reviews the popular linear algebra algorithms and libraries of numerical methods capable of using mixed precision calculations and highlights the current achievements and future research trends.

The iterative methods for solving large sparse SLAEs, which this paper focuses on, belong to the memory-bound applications with very low compute intensity of the order $O(10^{-1})$ [7, 18]. Thus, the use of mixed precision can be an attractive way to increase the compute intensity

[1]Lomonosov Moscow State University, Moscow, Russian Federation

and significantly improve the performance of the calculations. Reducing the precision of the floating-point numbers for the whole linear solver, however, may lead to a significant iterative process convergence rate degradation or even to divergence. The potential of using the single (or even lower) precision calculations depends on the properties of the specific SLAE, but for many practical applications, like structural mechanics problems or incompressible turbulent flow calculations, the use of single-precision for the whole solver is unacceptable.

The compromise and widely accepted variant is the use of reduced precision floating-point calculations when performing the preconditioning while preserving operations with the SLAE matrix and solution vector in the basic precision. This approach does not affect the resulting solution tolerance and is more robust compared to performing the whole solver with reduced precision. However, a significant portion of calculations is still performed with the basic precision, which reduces significantly the potential calculations speedup.

An alternative way of utilizing the mixed precision calculations is the use of the iterative refinement procedure [17]. Initially developed in the 1940s, this method receives great attention in the light of recent activity with introducing the mixed precision calculations in iterative methods for solving SLAEs [3, 5]. These publications show promising results with attractive speedup numbers. The conjugate gradient (CG) method with adaptive geometric multigrid preconditioner was considered by the developers of the QUDA library, and the combination of the top-level 64-bit solver with inner 32-bit CG solver and 16-bit preconditioner was analyzed. The proposed methods combination allowed to speed up the corresponding SLAE solver for NVIDIA Volta GPUs compared to the basic solver performed in double precision by a factor of 4–6. In [3] the authors investigated the combination of CG solver with a lightweight plain Jacobi (diagonal) preconditioner and the top-level iterative refinement procedure. The outer solver calculations were performed in double precision, and the preconditioned CG solver was performed in the single precision. The solver was evaluated with 123 SLAEs, and in about 60 % of the cases, the total energy consumption improvement by a factor of 1.1–1.8 was demonstrated compared to the preconditioned CG solver performed in double precision.

While the results mentioned above show good potential in using mixed precision calculations, the numerical methods and test matrices considered have their specific, which hardens the conclusion on the applicability of the methods proposed to the wider range of applications. Additionally, the energy consumption results presented in [3] can only give a rough estimate to the real calculation time improvement by using the mixed precision iterative refinement procedure. The current paper focuses on the applicability of mixed precision calculations to solve the elliptic equations, and, specifically, the pressure Poisson equation occurring in incompressible flow simulations. For example, high-fidelity turbulent flow simulations require multiple solutions of SLAEs, and the corresponding time to solve these SLAEs typically prevails in the overall calculation time. This aspect motivates the further tuning of numerical methods applicable for solving the corresponding SLAEs, and the use of iterative refinement with reduced precision solvers can be an attractive option.

The rest of the paper is organized as follows. The first section outlines the potential speedup due to the use of reduced precision calculations. The second section presents the iterative refinement procedure and highlights the main algorithm pitfalls. Various residual replacement strategies used to resolve some of these problems are discussed. The description of the XAMG library used in the numerical experiments to demonstrate the efficiency of the iterative refinement procedure is presented in the third section. The fourth section discusses the results of the

numerical experiments and the evaluation of the mixed precision iterative refinement procedure for modeling incompressible turbulent flows. The conclusion section summarizes the paper.

1. Potential of Mixed Precision Calculations

Before discussing the aspects of the iterative refinement procedure, it is important to estimate the potential performance gain due to reducing the precision of floating-point calculations. The object of interest is the algorithm combining the Krylov subspace method (specifically, BiCGStab [15]) with the classical algebraic multigrid preconditioner and Gauss-Seidel smoother. This combination of the methods represents a robust and scalable solver, which can be effectively used to solve various SLAEs, including the ones derived from the elliptic partial differential equations.

The methods mentioned above consist of the combination of linear operations with vectors and matrix-vector multiplications. Accounting that both of these operations are memory-bound, the expected calculation time reduction would be in proportion with the corresponding reduction in the amount of memory accesses. The overall execution time, T_{exec}, is a sum of vector operations time, T_{vec}, and matrix-vector multiplications time, T_{mat}:

$$T_{exec} = T_{vec} + T_{mat} = \frac{\Sigma_{vec} + \Sigma_{mat}}{B}, \tag{1}$$

where Σ is the amount of memory traffic when performing the corresponding operations, and B is the memory bandwidth of the compute system.

Reduction of the floating-point data size, e.g., from 8 to 4 bytes, leads to a twofold decrease of the amount of memory traffic for the vector operations and the corresponding calculation time. The matrix-vector operations also demonstrate significant calculation speedup, but it is typically less than by a factor of 2 due to the use of specialized sparse matrix storage formats and the need to store additional integer indices. For example, for the CSR data storage format [12], the overall amount of memory traffic when performing matrix-vector multiplication can be represented as:

$$\Sigma_{mat} = n\,I(C+1) + n\,F(2C+1). \tag{2}$$

Here n is the matrix size, I and F are the sizes of the integer and floating-point data types respectively, and C is the average number of nonzero elements per each matrix row. The ratio of the amount of data for 8 and 4 byte floating-point numbers shows the potential of the reduced precision matrix-vector operations speedup (expecting here that I is equal to 4 bytes):

$$P = \frac{n\,4(C+1) + n\,8(2C+1)}{n\,4(C+1) + n\,4(2C+1)} = \frac{5C+3}{3C+2}. \tag{3}$$

In the typical case $C \gg 1$ the expression above reduces to $P \approx 1.67$.

Using the expression (1), one can expect the overall speedup due to using the reduced precision floating-point calculations. Generally, the speedup lies in the range of 1.7–1.9 in case the overall convergence rate is not affected by the round-off errors. The use of mixed precision calculations for the preconditioner only further decreases the potential speedup, as only a portion of vector and matrix-vector operations in (1) is performed with reduced precision. In practice, this speedup depends on the size of the multigrid matrix hierarchy and can be roughly estimated in the range of 5–30 %.

Algorithm 1 Mixed precision iterative refinement algorithm.

1: x_0 – initial guess (basic precision)
2: $r_0 = b - Ax_0$ (basic precision)
3: $k = 0$
4: **while** not converged **do**
5: $r_k \to \hat{r}_k$ (convert to reduced precision)
6: solve $\hat{A}\hat{y}_k = \hat{r}_k$ (reduced precision)
7: $\hat{y}_k \to y_k$ (convert to basic precision)
8: $x_{k+1} = x_k + y_k$ (basic precision)
9: $r_{k+1} = b - Ax_{k+1}$ (basic precision)
10: $k = k + 1$
11: **end while**

The iterative refinement procedure, discussed in detail in the sections below, requires some additional calculations compared to the pure 32-bit solver. Even in case the SLAE solver is not suffering the convergence rate degradation due to lower precision calculations, the expected speedup will be lower than for the 32-bit solver. In the negative scenario, the overall calculation time can be even higher than that for the basic precision solver.

Summarizing the discussion above, the following conclusions can be stated:

- performance of iterative refinement procedure combining 64-bit and 32-bit floating-point calculations is expected to be lower than for the pure 32-bit solver and to not exceed the range of 1.7–1.9;
- mixed precision iterative refinement procedure can be found useful in case the obtained calculations speedup will exceed the one for the method configuration with the preconditioner only performed with reduced precision.

2. Iterative Refinement Procedure

The main idea of the mixed precision iterative refinement procedure is to combine two iterative algorithms, the inner iterative method operating with reduced precision and the outer method operating with the basic precision. The mixed precision iterative refinement algorithm is presented in Algorithm 1. The calculations start with computing the initial residual vector r in basic precision (line 2). The residual is converted to lower precision (line 5) and the reduced precision solver is used to get the solution update $\hat{y}$ (line 6). The obtained vector is converted back to basic precision (line 7) and accumulated with solution vector x (line 8). Finally, the residual vector is recalculated in basic precision (line 9) and is used to control the overall convergence.

This rather simple algorithm, however, has several pitfalls when used in practice:

1. The outer loop of the iterative refinement procedure performs some additional calculations, like the calculation of residual vector, requiring matrix-vector multiplication, vector updates, and floating-point data conversion. These operations provide some overhead compared to the original solver performed in basic precision, and the number of outer iterations should be minimized to minimize the corresponding overhead.
2. To compete with the reduced precision preconditioner calculations, the number of inner solver iterations should not exceed 20–30 %. For the iterative methods considered in this paper the typical number of iterations of the basic solver is in the range 10–20. This means,

an increase in the cumulative number of inner solver iterations in the range 3–5 is permissible, otherwise the use of iterative refinement procedure will lead to the calculations slowdown.

3. The Krylov subspace iterative methods (e.g., CG, BiCGStab) use recurrent expressions to calculate the residual vector at the next iteration. The long recurrent chains are sensitive to the accumulation of the round-off errors, and as a result, the deviation of this residual from the true one, $r = b - Ax$, may be observed. This issue can lead to the situation when the convergence of the recurrent residuals will be accompanied by the true residuals stagnation. The use of reduced precision calculations for the inner solver can even complicate the situation, as the calculations with reduced precision can accelerate the accumulation of the round-off errors.

The problem of residuals deviation for the Krylov subspace iterative methods has been analyzed in the literature, e.g., in [3, 13, 16]. The observed convergence issue in some cases can be resolved by the correction of the recurrent residuals, which can be done in several ways. In [13, 16] the authors propose the reliable updates to correct the recurrent residual with the true one and several criteria when these updates must be performed. Alternatively, in [3] the inner solver restarts are considered for the mixed precision iterative refinement algorithm. Following [3], the proper choice of the conditions to perform the restarts allows this approach to outperform the reliable updates strategy.

It should be noted that the publications mentioned above use for the numerical evaluation the lightweight iterative methods with the simple preconditioner or even without it. The typical number of iterations till convergence in the presented results is counted by at least several hundred, and several additional residual checks or extra iterations do not play a significant role. The current paper focuses on the algorithms with robust preconditioners performing typically 10–20 iterations. In that case, the penalty for the several extra iterations can be critical, and the applicability of the corresponding residual replacement algorithms must be analyzed in detail. To the best of our knowledge, there are no publications discussing the applicability of the mixed precision iterative refinement algorithms with robust and fast converging iterative methods.

3. XAMG Library

The performance evaluation results presented in this paper are performed with the XAMG library [9, 10]. The XAMG library is a novel open-source project which implements sparse linear algebra subroutines and solvers in modern C++. The library provides the implementation of a set of widely used numerical methods for solving large sparse SLAEs, including the Krylov subspace iterative methods, classical algebraic multigrid method, and others. The XAMG library focuses on the optimized implementation for the solve part of the methods and reuses the *hypre* library [1] for the multigrid hierarchy matrices construction.

The library design feature is a template-based generalization of basic objects and subroutines. This feature makes it possible to implement a few important design principles. First, the specialization of the number of right-hand side vectors as a template parameter allows for a generalization of the subroutines for an arbitrary number of right-hand sides in the solution. The specialization of the generalized subroutines takes place automatically in compile time, and the automated vectorization of loops is being done without additional effort. Second, all the data types for base library objects are also specialized with template parameters, so the features like the advanced sparse matrix formats and reduced precision storage options can be implemented with ease.

The library consists of four major groups of elements: (i) matrices and vectors data structures, representing basic data objects; (ii) basic sparse linear algebra subroutines gathered in "blas" and "blas2" groups; (iii) solver classes inherited from an abstract interface, which exposes the main library function "solve()"; (iv) solver parameter classes. The solver classes are arranged in a structure allowing easy addition of the new numerical methods. The solvers can be combined with each other as well. The established solver parameters subsystem supports the solver code extension and code combinations.

An important design feature of the XAMG library is the hierarchical hybrid parallel programming model. This model, called MPI+ShM, is integrated into basic data objects and basic subroutines of the library. The MPI+ShM model implies that MPI ranks, which appear to be placed on the same compute node, allocate and use the common POSIX shared memory regions to store the vector data structures. The communication and data access within a single compute node is performed using these shared memory regions. This hybrid approach to parallel programming improves the productivity and scalability of basic library subroutines compared to a pure MPI-only parallel programming model. It makes it possible to get leading scalability figures on modern multicore and manycore HPC systems.

To summarize, the XAMG library is a flexible tool to implement new iterative methods and their combinations and do experiments with their traits and parameters. On the other hand, the MPI+ShM hybrid model ensures the good productivity and scalability of the resulting code. It makes the XAMG library an attractive platform for the research made in this work.

4. Numerical Experiments

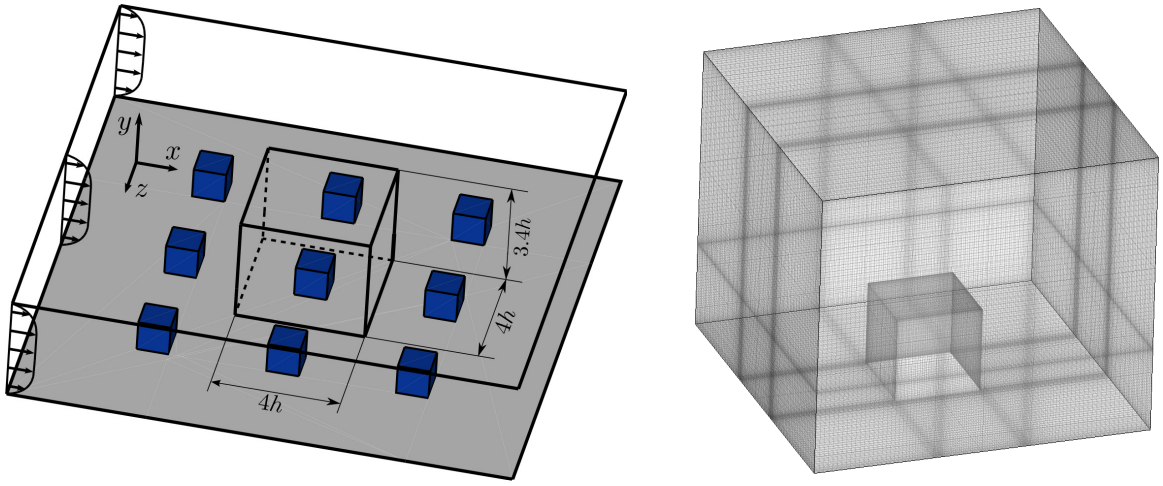
The current section presents results of the numerical experiments investigating the applicability of the mixed precision iterative refinement procedure to accelerate the solution of the SLAEs, and, specifically, the ones obtained as a result of the spatial discretization of the pressure Poisson equation when modeling incompressible turbulent flows. The investigation is performed for the SLAE solver comprising the BiCGStab method with algebraic multigrid preconditioner and symmetric Gauss-Seidel smoother. The four solver configurations with the same numerical method configurations are considered. The *basic* solver configuration performs all the calculations with double precision. The *mixed* solver configuration assumes the use of reduced precision for the preconditioning when operating with the constructed multigrid matrices hierarchy. The *IR* solver utilizes the mixed precision iterative refinement procedure with the inner solver, corresponding to the *basic* solver, but performed in the single precision. Finally, the *reduced* solver configuration performs all the calculations with single precision. All the numerical experiments are performed on the Lomonosov-2 supercomputer (compute nodes with single 14-core Intel E5-2697v3 processor); the calculation runs utilize all available 14 physical CPU cores per node.

4.1. Testing Methodology

The evaluation of different approaches of introducing the reduced precision calculations in the iterative SLAE solvers is performed in several steps. The first test series confirms the potential speedup limits by performing the same calculations in double precision (*basic*), in single precision (*reduced*), and for the double-precision solver with reduced precision preconditioning calculations (*mixed*). The various inner solver stopping criteria, affecting the frequency of residual replacement in the mixed precision iterative refinement algorithm (*IR*) and the corre-

sponding solver convergence rate, are investigated in the second test series. The third group of tests compares the convergence rates and calculation times for *basic*, *mixed*, and *IR* solver configurations. Finally, the turbulent flow simulations are performed with various solver configurations to demonstrate the potential decrease in the solution time in the real calculations.

The two groups of test SLAEs are used in the tests. The first one corresponds to the discretization of the 3D Poisson equation in the cubic domain ("cube") with the uniform grid and constant right-hand side vector. The size of the grid varies in the range of 100^3–500^3. The second group includes the two test systems corresponding to the pressure Poisson equation in the computational domain used for the simulation of the turbulent flow in a channel with a matrix of wall-mounted cubes [6] ("channel with cube"; the geometry of the computational domain and the corresponding computational grid are presented in Fig. 1); the size of the grids is equal to 2.3 and 9.7 million unknowns. The right-hand side vectors are chosen corresponding to the physical problem statement, and they are defined as the divergence of the random velocity fields. Both groups of the test matrices are available with the internal data generator of the integration test application provided with the XAMG library [10].



(a) Computational domain (b) Computational grid

Figure 1. Sketch of the computational domain and the computational grid

4.2. Performance Tests

4.2.1. Performance limitations

The performance evaluation starts with the numerical experiments, demonstrating the real calculation time reduction with mixed precision calculations and the calculations with the solver operating in single precision. This test series performs the constant number of iterations and focuses on the calculation time reduction as an upper bound estimate for the calculation results that can be observed with the iterative refinement procedure (in case the convergence rate is not affected by the reduced precision calculations).

The calculations presented in this section are performed with the single node of the Lomonosov-2 supercomputer for the "cube" test system with 150^3 unknowns and the "channel with cube" system with 2.3 million unknowns, and with 4 compute nodes for the "cube" system with 500^3 unknowns. The obtained calculation results are summarized in Tab. 1. These include

the single iteration times and the corresponding speedup compared with the *basic* solver configuration performed in double precision. The speedup by a factor of about 1.1 can be expected for the *mixed* solver configuration and by a factor of 1.65 – for the *reduced* solver configuration operating with 32-bit floating-point numbers. The results presented are similar for all test matrices considered and are in agreement with the proposed theoretical estimates (see, section 1).

The obtained calculation times are also compared with the ones for the well-known open-source *hypre* library, which allows using the same numerical method configurations as in XAMG. The *hypre* library does not provide the functionality to utilize the mixed precision calculations, thus only double precision calculations are performed, which correspond to the *basic* configuration of the XAMG library. The results presented in Tab. 1 demonstrate an advantage of the XAMG library compared to *hypre*: the decrease in the execution time is about 10–15 %. Some more details on comparing the performance of the XAMG and *hypre* libraries can be found in [9].

Table 1. The single iteration calculation times for the XAMG and *hypre* libraries and the corresponding decrease in the solution time compared to the double precision solver (speedup) for different solver configurations and test matrices

Test case	*basic*	*mixed*		*reduced*		*hypre*
	time, s	time, s	speedup	time, s	speedup	time, s
Cube, 150^3	0.151	0.143	1.06	0.091	1.66	0.170
Cube*, 500^3	1.65	1.51	1.09	1.01	1.64	1.91
Channel with cube, 2M	0.123	0.109	1.12	0.076	1.62	0.139

*The corresponding calculations are performed with 4 compute nodes

4.2.2. Residual replacement

The choice of the inner solver stopping criteria is among the key practical aspects affecting the usability of the mixed precision iterative refinement procedure. The current paragraph analyzes and compares some of them, including the periodic restarts and the explicit residual replacement [3]. The corresponding experiments are performed for two test matrices, the "cube" case with 150^3 unknowns and the "channel with cube" case with 2.3 million unknowns and investigate 8 different combinations of the parameters responsible for stopping the inner solver and residual replacement. The frequency of the periodic restarts is chosen according to the typical number of iterations till convergence, which varies in the range 10–20: the experiments with restarts every 1, 2, 3, and 5 iterations are performed. The explicit residual deviation checks once in $t = 100$ iterations with the allowed deviation ratio $\gamma = 10$ were used in [3]. In our case, these values are inapplicable because the periodicity of the residual checks outnumbers significantly the expected number of iterations till convergence. Alternatively, the residual deviation checks every $t = 2$ and 3 iterations with the allowed deviation ratio $\gamma = 1.01$ and 2 are considered. The decrease of the initial residual norm by a factor of $\varepsilon = 10^{-8}$ is used as the convergence criterion.

The obtained calculation results are presented in Tab. 2. The restarts for the inner solver performed every iteration lead to the convergence degradation due to the loss of the information about the already constructed Krylov subspace basis. An increase in the number of inner solver iterations leads to the overall solver convergence acceleration. However, reduction of the

frequency of restarts strengthens the residual deviation effect, and for the case with restarts once in 5 iterations, the overall number of iterations starts to increase. Summarizing the results presented in the table, the frequency of restarts every 2–3 iterations can be outlined as the one providing the best convergence rate for the SLAEs and numerical method configurations considered.

The explicit residual deviation check assumes explicit control of the deviation of the true and recurrent residuals every t iterations and the inner solver restart only in case the residual deviation exceeds the limiting value γ. The presented calculation results show that the use of a high deviation ratio is inefficient: the higher value γ only tightens the inner solver restart, while performing the inner solver with even slightly deviated residuals reduces the solver efficiency in terms of the true residual decay. The decrease of the deviation value γ allows reducing in some cases the overall number of iterations, and, consequently, the calculation time. However, the obtained calculation times with explicit residual deviation checks are systematically higher than the ones for the periodic restarts approach, which provides a simple and efficient way to avoid the residual deviation issue and convergence rate degradation.

Table 2. Comparison of various inner solver stopping criteria

Stopping criteria		Cube, 150^3		Channel with cube, 2M	
Method	Parameters	Time, s	Iter.	Time, s	Iter.
Periodic restart	$t = 1$	1.54	15 (15)	1.01	12 (12)
	$t = 2$	1.14	6 (12)	0.79	5 (10)
	$t = 3$	1.13	4 (12)	0.79	4 (10)
	$t = 5$	1.12	3 (12)	0.86	3 (11)
Explicit residuals check	$t = 2$, $\gamma = 2$	1.32	2 (14)	1.04	3 (13)
	$t = 3$, $\gamma = 2$	1.22	2 (13)	1.09	3 (14)
	$t = 2$, $\gamma = 1.01$	1.24	3 (13)	0.88	3 (11)
	$t = 3$, $\gamma = 1.01$	1.22	2 (13)	0.87	3 (11)

4.2.3. Single SLAE tests

The next step of performance evaluation focuses on the investigation of various solver configurations with a set of test matrices. The three solver configurations (*basic*, *mixed*, and *IR* solvers) with four "cube" SLAEs and two "channel with cube" SLAEs are analyzed. The *IR* solver is performed with the inner solver periodic restarts every 3 iterations.

The overall solution times and the number of iterations (for the *IR* solver – the number of outer iterations and the cumulative number of inner iterations) are collected in Tab. 3. For all the test cases considered the observed calculation results, in general, demonstrate similar behavior. The *basic* and *mixed* solver configurations converge in the same number of iterations: the use of reduced precision for the multigrid hierarchy preconditioning calculations does not affect the overall convergence rate. The speedup for the *mixed* solver varies in the range 1.06–1.13, and these values correspond to the basic theoretical estimates. The *IR* solver demonstrates even faster convergence than the *basic* one, decreasing the number of iterations by 1–3. The reduction in the number of iterations for the *IR* solver on par with the lower calculation time

for the inner solver performed in reduced precision allows obtaining the significant speedup for all the test systems considered, which varies in the range 1.67–1.88.

Table 3. The calculation times, the number of iterations, and the achieved relative speedup for the several test matrices and solver configurations

Test case	*basic*		*mixed*			*IR*		
	time, s	iter.	time, s	iter.	speedup	time, s	iter.	speedup
Cube, 100^3	0.53	13	0.50	13	1.06	0.30	4 (11)	1.74
Cube, 150^3	2.11	14	1.99	14	1.06	1.12	4 (12)	1.88
Cube, 200^3	5.69	15	5.34	15	1.07	3.10	5 (13)	1.83
Cube*, 500^3	39.5	24	36.3	24	1.09	22.0	7 (21)	1.80
Channel with cube, 2M	1.35	11	1.20	11	1.13	0.79	4 (10)	1.71
Channel with cube*, 9M	2.12	14	1.88	14	1.13	1.27	5 (13)	1.67

*The corresponding calculations are performed with 4 compute nodes

The calculation times presented in Tab. 3 show a stable tendency in decreasing the number of iterations for the *IR* solver configuration. Despite the use of lower precision for the inner solver calculations, the corresponding number of iterations till convergence is decreased in all the cases considered by about 10–15 %. To investigate the reasons for this effect, the same calculations were repeated for the double precision solver with residual replacement. This test series reproduced the same effect of reducing the number of iterations for the basic solver with residual replacement, observed for the *IR* solver configuration (Tab. 4). Despite an additional amount of calculations to perform the residual replacement and solver restarts, the use of this technique can be advantageous even for double precision calculations.

Table 4. The calculation times and the number of iterations for the *IR* and double precision and solver configurations with residual replacement

Test case	*basic + RR*		*IR*	
	time, s	iterations	time, s	iterations
Cube, 100^3	0.46	4 (11)	0.30	4 (11)
Cube, 150^3	1.87	4 (12)	1.12	4 (12)
Cube, 200^3	5.11	5 (13)	3.10	5 (13)
Cube*, 500^3	35.5	7 (21)	22.0	7 (21)
Channel with cube, 2M	1.26	4 (10)	0.79	4 (10)
Channel with cube*, 9M	1.85	4 (12)	1.27	5 (13)

*The corresponding calculations are performed with 4 compute nodes

4.2.4. Turbulent flow calculations

The results presented in the previous paragraph demonstrate attractive speedup for the mixed precision iterative refinement procedure which can be easily incorporated into practical applications. The final stage of the performance evaluation considers the test runs for the direct numerical simulation of incompressible turbulent flow with three solver configurations and shows the practical calculations speedup.

The turbulent flow simulations are performed with in-house computational code, based on the second order in space and third order in time computational algorithm, operating with curvilinear orthogonal coordinates [11]. This algorithm requires three solutions of pressure Poisson equation per each time step; the SLAE matrix remains unchanged during the calculations allowing to perform the construction of the multigrid matrix hierarchy only once at the initialization stage. The calculations are performed for the test case of modeling the turbulent flow in a channel with a matrix of wall-mounted cubes, described in subsection 4.1, on the grid with 2.3 million unknowns. The test runs are performed with 8 compute nodes and model short time interval of $T = 3$ time units, which corresponds to about 1100 time steps.

The results of the three test runs are outlined in Tab. 5. These include the linear solver calculation times and the overall simulation time. The simulation with the *basic* solver configuration takes 486 seconds, where 455 seconds are spent with the SLAE solver. The cumulative number of iterations performed by the linear solver is about 30000. The run with the *mixed* precision solver takes 452 seconds and shows the speedup by a factor of 1.08. The convergence rate of the linear solver remains unaffected by the reduction of the preconditioner calculations precision – the number of iterations to solve the corresponding SLAEs is about the same as for the *basic* solver configuration. The lowest calculation times are observed with the *IR* solver configuration. The overall simulation time takes 323 seconds, which is 1.5 times faster than the one with the *basic* solver configuration. The linear solver fraction of the calculation time is reduced by a factor of 1.56 and takes 292 seconds. The *IR* solver requires about 9800 iterations for the outer solver and about 26000 iterations for the inner solver. The overall number of iterations for the *IR* solver is about 10 % less than for the *basic* solver, which indicates that the regular residual replacement allows to reduce the round-off errors and increase the convergence rate.

Table 5. The calculation times, the total number of iterations, and the achieved speedup for the direct numerical simulation of turbulent flow performed with several linear solver configurations

Stage	*basic*		*mixed*			*IR*		
	time, s	iter.	time, s	iter.	speedup	time, s	iter.	speedup
SLAE solver	455	29978	421	29982	1.08	292	9816 (25687)	1.56
Total	486		452		1.08	323		1.50

Conclusions

The paper investigates the efficiency of the several approaches of introducing the reduced precision calculations for solving large sparse systems of linear algebraic equations, and, specifically, the ones occurring when solving the elliptic differential equations. These include the use of reduced precision calculations for the preconditioning and the mixed precision iterative refinement procedure, combining the outer solver performed with basic precision and the inner solver operating with reduced precision. The paper reviews the examples of using the mixed precision iterative refinement procedure known in the literature and significantly extends them with the obtained calculation results for the numerical methods with robust preconditioners. The various residual replacement strategies are discussed and the optimal one for the test cases considered is proposed.

The results of the numerical experiments performed for several test matrices are presented. These results show that the use of reduced precision calculations for the preconditioner allows decreasing the solution time by only about 10 %, while the use of mixed precision iterative refinement procedure with the proper choice of the residual replacement strategy can provide the SLAE solver speedup by a factor of 1.8. The proposed residual replacement algorithm is examined by performing the calculations of incompressible turbulent flow. The several runs performed for the direct numerical simulation of the turbulent flow in a channel with a matrix of wall-mounted cubes demonstrate about 1.5 times decrease in the solution time due to acceleration of the pressure Poisson equation SLAE solver when using the mixed precision iterative refinement procedure.

Acknowledgements

The presented work is supported by the RSF grant No. 18-71-10075. The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University.

References

1. HYPRE: High performance preconditioners (2020). http://www.llnl.gov/CASC/hypre/, accessed: 2020-12-27

2. Abdelfattah, A., Anzt, H., Boman, E.G., et al.: A survey of numerical linear algebra methods utilizing mixed-precision arithmetic. The International Journal of High Performance Computing Applications 35(4), 344–369 (2021). https://doi.org/10.1177/10943420211003313

3. Anzt, H., Flegar, G., Novaković, V., et al.: Residual replacement in mixed-precision iterative refinement for sparse linear systems. In: Yokota, R., Weiland, M., Shalf, J., Alam, S. (eds.) High Performance Computing. pp. 554–561. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02465-9_39

4. Baboulin, M., Buttari, A., Dongarra, J., et al.: Accelerating scientific computations with mixed precision algorithms. Computer Physics Communications 180(12), 2526–2533 (2009). https://doi.org/10.1016/j.cpc.2008.11.005

5. Clark, M., Babich, R., Barros, K., et al.: Solving lattice QCD systems of equations using mixed precision solvers on GPUs. Computer Physics Communications 181(9), 1517–1528 (2010). https://doi.org/10.1016/j.cpc.2010.05.002

6. Krasnopolsky, B.: An approach for accelerating incompressible turbulent flow simulations based on simultaneous modelling of multiple ensembles. Computer Physics Communications 229, 8–19 (2018). https://doi.org/10.1016/j.cpc.2018.03.023

7. Krasnopolsky, B.: Revisiting performance of BiCGStab methods for solving systems with multiple right-hand sides. Computers & Mathematics with Applications 79(9), 2574–2597 (2020). https://doi.org/10.1016/j.camwa.2019.11.025

8. Krasnopolsky, B., Medvedev, A.: Acceleration of large scale OpenFOAM simulations on distributed systems with multicore CPUs and GPUs. In: Parallel Computing: On the Road to Exascale. Advances in Parallel Computing, vol. 27, pp. 93–102 (2016). https://doi.org/10.3233/978-1-61499-621-7-93

9. Krasnopolsky, B., Medvedev, A.: XAMG: A library for solving linear systems with multiple right-hand side vectors. SoftwareX 14 (2021). https://doi.org/10.1016/j.softx.2021.100695

10. Krasnopolsky, B., Medvedev, A.: XAMG: Source code repository (2021). https://gitlab.com/xamg/xamg, accessed: 2021-03-31

11. Nikitin, N.: Finite-difference method for incompressible Navier-Stokes equations in arbitrary orthogonal curvilinear coordinates. Journal of Computational Physics 217, 759–781 (2006). https://doi.org/10.1016/j.jcp.2006.01.036

12. Saad, Y.: Iterative methods for sparse linear systems, 2nd edition. SIAM, Philadelpha, PA (2003)

13. Sleijpen, G.L.G., van der Vorst, H.A.: Reliable updated residuals in hybrid Bi-CG methods. Computing 56, 141–163 (1996). https://doi.org/10.1007/BF02309342

14. Sumiyoshi, Y., Fujii, A., Nukada, A., Tanaka, T.: Mixed-precision AMG method for many core accelerators. In: 21st European MPI Users' Group Meeting, EuroMPI/ASIA '14. pp. 127–132. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2642769.2642794

15. van der Vorst, H.A.: BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. 13(2), 631–644 (1992). https://doi.org/10.1137/0913035

16. van der Vorst, H.A., Ye, Q.: Residual replacement strategies for Krylov subspace iterative methods for the convergence of true residuals. SIAM Journal on Scientific Computing 22(3), 835–852 (2000). https://doi.org/10.1137/S1064827599353865

17. Wilkinson, J.H.: Rounding Errors in Algebraic Processes. Englewood Cliffs, Prentice-Hall, New Jersey (1963)

18. Williams, S., Waterman, A., Patterson, D.: Roofline: An insightful visual performance model for multicore architectures. Communications of the ACM 52(4), 65–76 (2009). https://doi.org/10.1145/1498765.1498785

19. Zarechnev, S., Krasnopolsky, B.: Improving performance of linear solvers by using indices compression for storing sparse matrices. In: Voevodin, V. (ed.) Russian Supercomputing Days: Proceedings of the International Conference. pp. 119–120. MAKS Press, Moscow (2020). https://doi.org/10.29003/m1406.RussianSCDays-2020