Proceedings of the Twelfth International Conference on
Engineering Computational Technology
Edited by: P. Iványi, J. Kruis and B.H.V. Topping
Civil-Comp Conferences, Volume 8, Paper 5.4
Civil-Comp Press, Edinburgh, United Kingdom, 2024
ISSN: 2753-3239, doi: 10.4203/ccc.8.5.4
©Civil-Comp Ltd, Edinburgh, UK, 2024

GPU-Accelerated Iterative Refinement Based on Induced Dimension Reduction

Y. Jiang[1], F. Magoulès[2], X. Wang[1] and Q. Zou[1]

[1] School of Science, Beijing University of Posts and Telecommunications, China
[2] CentraleSupélec, Paris-Saclay University, Gif-sur-Yvette, France

Abstract

The resolution of large scale linear systems is still a significant challenge in scientific computing. In this paper we consider the induced dimension reduction method and iterative refinement scheme for solving nonsymmetric linear systems. We formulate an IR outer scheme based on the IDR inner solver and perform a numerical study of its finite precision behavior. The algorithm is implemented on a graphics processing unit using the CUDA programming tool.

Keywords: induced dimension reduction, iterative refinement, mixed precision, Krylov subspace methods, linear systems, parallel computing.

1 Introduction

Krylov subspace methods are the standard choice for solving large scale linear systems

$$Ax = b, \tag{1}$$

where $A \in \mathbb{R}^{N\times N}$ is a nonsymmetric matrix and $b \in \mathbb{R}^N$ is a nonzero vector. For instance, the generalized minimal residual (GMRES) algorithm [1,2] and biconjugate gradient stabilized (BICGSTAB) algorithm [3] are widely used and are successful in practice for a wide variety of problems. For more discussion about Krylov subspace methods, we refer the reader to [4, 5]. The induced dimension reduction (IDR) algorithm is originally proposed in 1980 by Wesseling and Sonneveld [6], and improved in 2009 by Sonneveld and van Gijzen [7] to the IDR(s) variant.

The role of iterative refinement (IR) in solving linear systems (1) has received increased attention across a number of disciplines in recent years [8–10]. It is used to improve a computed solution $\hat{x}$ of a linear system $Ax = b$ by computing the residual $r = b - A\hat{x}$ in higher precision $\bar{u}$ (such as $\bar{u} = u^2$), solving $Ad = r$ in working precision u, and updating $\hat{x} \leftarrow \hat{x} + d$ with precision u. The IR scheme can date back to the pioneering work of Wilkinson [11] in 1948. At that time dot products were often accumulated at twice the usual precision with hardware support. With nearly two decades of neglect, IR has received renewed attention since 2000, especially in the last ten years due to the theoretical and hardware breakthroughs; see [10] for more details. We also refer the reader to [12–14] for more recent development on this subject.

Traditionally, the inner solver is performed with LU factorization. In the paper of Carson and Higham [8,9] they prove that using iterative scheme may show advantages in terms of stability; see also [15] for more discussion on stability analysis. Here we perform the IR scheme based on IDR(s) iterations, denoted for simplicity by IDR-IR(s), with working precision u and residual precision $\bar{u}$. We compare different u and maximum IDR iterations in GPU environment and illustrate the convergence behavior.

2 IDR-based iterative refinement

The IDR(s) algorithm defines a sequence

$$\mathcal{G}_j = (I - \omega_j A)(\mathcal{G}_{j-1} \cap \mathcal{S}), \quad j = 1, 2, \ldots, \tag{2}$$

where $\mathcal{G}_0$ is the full Krylov subspace

$$\mathcal{K}_N(A, v_0) = \text{span}\,\{v_0,\ Av_0,\ \ldots,\ A^{N-1}v_0\},$$

for some nonzero vector v_0, and $\mathcal{S}$ is some subspace of $\mathbb{C}^N$ such that $\mathbb{C}$ and $\mathcal{G}_0$ do not share a nontrivial invariant subspace of A. Here ω_j denotes some nonzero scalars, usually chosen as the value that minimizes the norm of residual.

The IDR(s) process is shown in Algorithm 1. Let $r_n = b - Ax_n$ be the residual. Since r_n can be expressed as $\Phi_n(A)r_0$ for some nth degree polynomial Φ_n, we have

$$r_{n+1} = r_n - \alpha A v_n - \sum_{l=1}^{s} c_l \Delta r_{n-l},$$

where s denotes the depth of the recursion, v_n is any computable vector in $\mathcal{K}_n(A, r_0)$ but not in $\mathcal{K}_{n-1}(A, r_0)$, and $\Delta r_n = r_{n+1} - r_n$. Then

$$A\Delta x_n = -\Delta r_n = [\Phi_n(A) - \Phi_{n+1}(A)]r_0$$

Algorithm 1: The IDR(s) algorithm.

```
Data: $A \in \mathbb{C}^{N\times N}; x_0, b \in \mathbb{C}^N; P \in \mathbb{C}^{N\times s}; \varepsilon \in (0,1); \mathtt{MAXIT} > 0$.
Result: $x_n$ such that $\|b - Ax_n\| \le \varepsilon$.
Calculate $r_0 = b - Ax_0$;
for $n = 0$ to $s-1$ do
    $v = Ar_n; \omega = (v^H r_n)/(v^H v)$;
    $\Delta x_n = \omega r_n; \Delta r_n = -\omega v$;
    $r_{n+1} = r_n + \Delta r_n; x_{n+1} = x_n + \Delta x_n$;
end
$\Delta R_{n+1} = (\Delta r_n \dots \Delta r_0); \Delta X_{n+1} = (\Delta x_n \dots \Delta x_0)$;
$n = s$;
while $\|r_n\| > \varepsilon$ and $n < \mathtt{MAXIT}$ do
    for $k = 0$ to $s$ do
        Solve $c$ from $P^H \Delta R_n c = P^H r_n$;
        $v = r_n - \Delta R_n c$;
        if $k = 0$ then
            $t = Av$;
            $\omega = (t^H v)/(t^H t)$;
            $\Delta r_n = -\Delta R_n c - \omega t$;
            $\Delta x_n = -\Delta X_n c - \omega v$;
        else
            $\Delta x_n = -\Delta X_n c - \omega v; \Delta r_n = -A\Delta x_n$;
        end
        $r_{n+1} = r_n + \Delta r_n$;
        $x_{n+1} = x_n + \Delta x_n$;
        $n = n + 1$;
        $\Delta R_n = (\Delta r_{n-1} \dots \Delta r_{n-s})$;
        $\Delta X_n = (\Delta x_{n-1} \dots \Delta x_{n-s})$;
    end
end
```

The IDR process ensures that the residual r_n lies in the subspaces $\mathcal{G}_j$, where j is non-decreasing with increasing n. The system (1) will be solved in exact arithmetic after at most N dimension reduction steps. Moreover, the relation (2) can be implemented by enforcing

$$\mathcal{G}_j = (I - \omega_j A)(\mathcal{G}_{j-1} \cap \mathcal{N}(P^H)), \quad j = 1, 2, \dots,$$

where

$$P = [p_1, \dots, p_s]$$

is a rank-s matrix. We do not pursue the background of IDR(s) and refer the reader to [7] for more details.

Algorithm 2: Iterative refinement.

```
Data: Nonsingular matrix $A \in \mathbb{C}^{N\times N}$; $b \in \mathbb{C}^N$.
Result: Approximate solution $\hat{x}$ to $Ax = b$.
1 Solve $Ax_0 = b$;
2 for $i = 0 : i_{max} - 1$ do
3     Compute $r_i = b - Ax_i$ in precision $\bar{u}$; store in precision $u$;
4     Solve $Ad_i = r_i$;
5     Compute $x_{i+1} = x_i + d_i$;
6     if converged then
7         return $x_{i+1}$, quit.
8     end
9 end
```

On the other hand, the IR scheme is shown in Algorithm 2. Here we use Algorithm 1 to solve $Ad_i = r_i$ and try different precisions to investigate the numerical behavior. We perform experiments with IDR-IR(s) on graphics processing units (GPUs). We test two precisions $u = 2^{-24}$ and $u = 2^{-53}$, namely, single and double precisions, respectively, and fix the residual precision $\bar{u} = u^2$ to evaluate the convergence behavior of our implementation of the IDR-IR(s) algorithm. To gauge the benefits of using GPUs to accelerate the resolution, we use NVDIA GeForce RTX3060 Laptop GPU with 12GB memory, with NVIDIA's CUDA version 11.6. We use a set of large-scale sparse matrices extracted from the SuiteSparse matrix collection [16] to obtain the numerical results. Recall that only the calculation of residuals is done with higher precision $\bar{u}$.

In all the tests we set 30 as the maximum number of iterative refinement steps since we observe that the number of iterative refinements is generally not too high. For the inner IDR(s) process, we vary the maximum number of iteration steps to observe its effect on the accuracy and the number of iteration refinements. The convergence threshold is fixed as 10^{-8}. From Figures 1–4 we can see that when the maximum number of IDR(s) iterations is limited to 5 or 10, the solution fails to converge even when the number of iterative refinements is large. On the other hand, when the number of IDR(s) iterations is sufficient, the accuracy of the approximate solution is reduced by an order of magnitude of 10^{-5} compared to the case when u is in double precision.

3 Concluding remarks

This paper investigates the iterative refinement based on the induced dimension reduction process. Actually this is a preliminary study of a large project, which exploits the effect of approximate computing on parallel algorithms. For instance, pre-

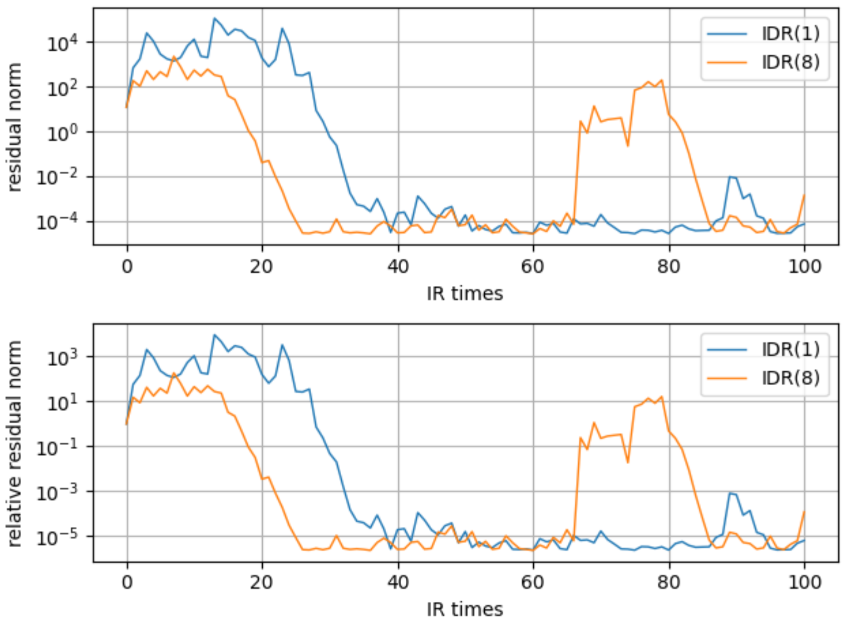

Figure 1: Performance of IDR-IR(s) with $\texttt{MAXIT} = 5$ and $u = 2^{-24}$.

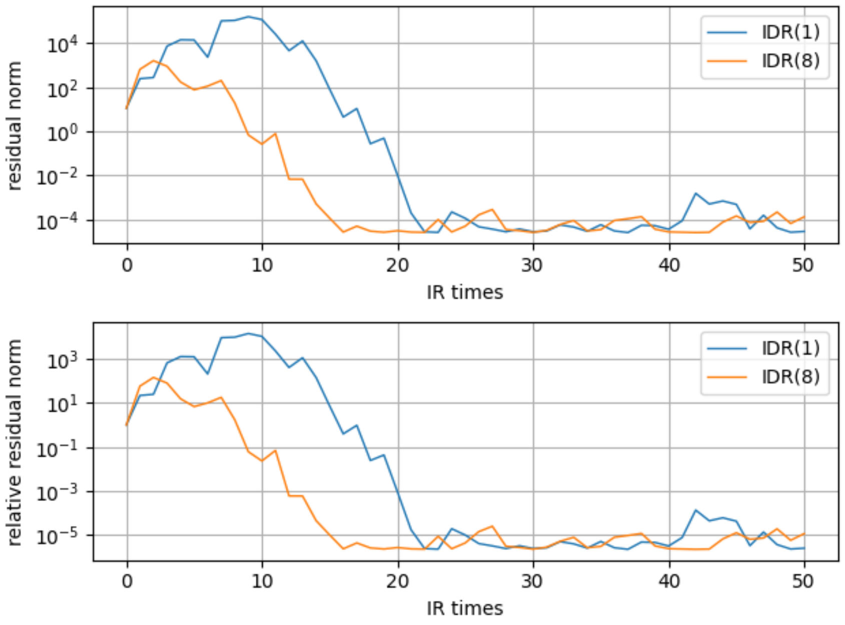

Figure 2: Performance of IDR-IR(s) with $\texttt{MAXIT} = 10$ and $u = 2^{-24}$.

conditioning, mixed precision, and randomization are widely used techniques for ill-conditioned problems. At the same time, one often implements s-step, pipelined, or hierarchical variants on GPUs to accelerate the resolution of linear systems. Furthermore, both deterministic and probabilistic rounding error analysis approaches can be used for ensuring the stability of a numerical algorithm. A combination of approxi-

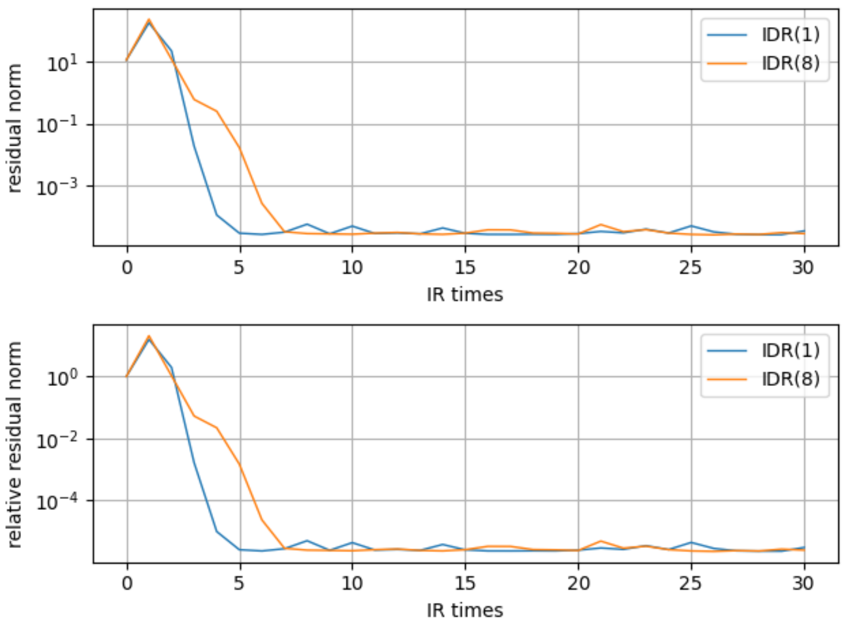

Figure 3: Performance of IDR-IR(s) with $\mathtt{MAXIT} = 30$ and $u = 2^{-24}$.

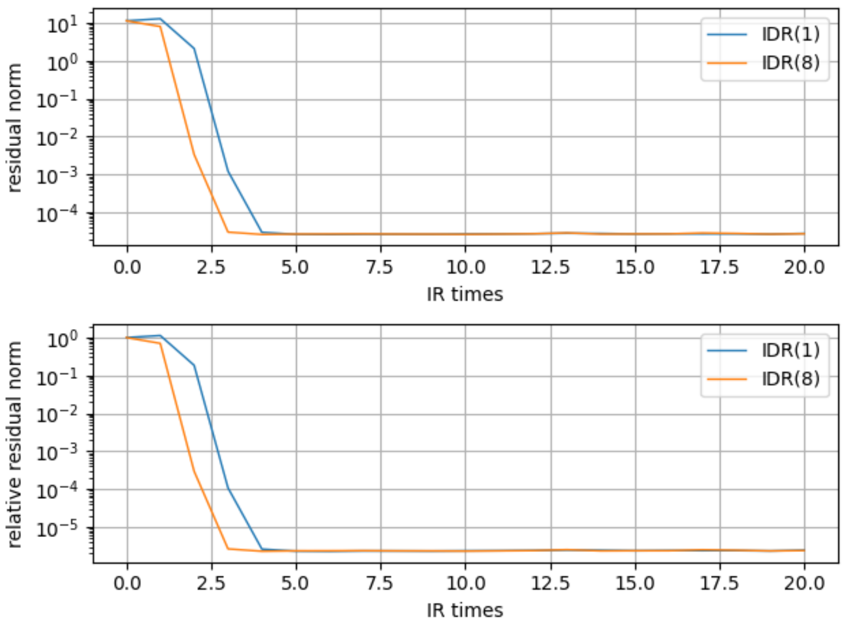

Figure 4: Performance of IDR-IR(s) with $\mathtt{MAXIT} = 50$ and $u = 2^{-24}$.

mation, parallelization, and rounding error analysis appears to be a successful stack of techniques in the exascale era. In the future we will investigate the communication-reducing IDR variants and pursue the optimal implementation on GPUs. We will also exploit practical preconditioners adapted to the parallel algorithms. Finally for the entire elements we will study the mixed precision technique and try to formulate a

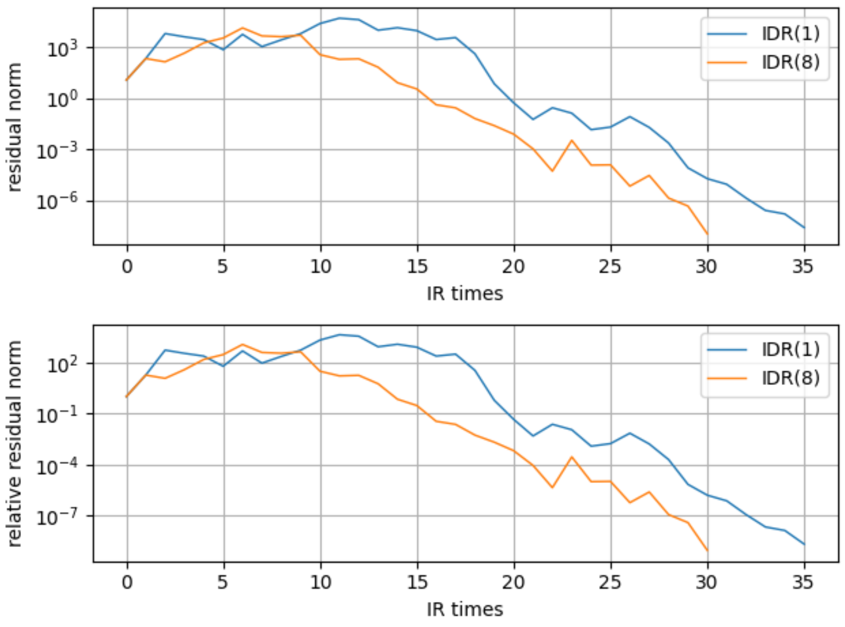

Figure 5: Performance of IDR-IR(s) with $\texttt{MAXIT} = 5$ and $u = 2^{-53}$.

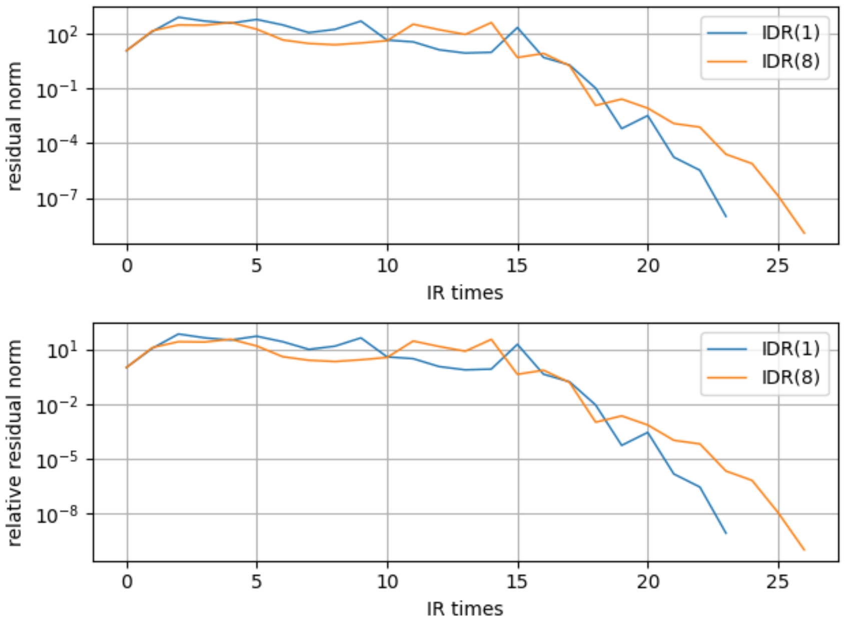

Figure 6: Performance of IDR-IR(s) with $\texttt{MAXIT} = 10$ and $u = 2^{-53}$.

nontrivial scheme to further accelerate the algorithms.

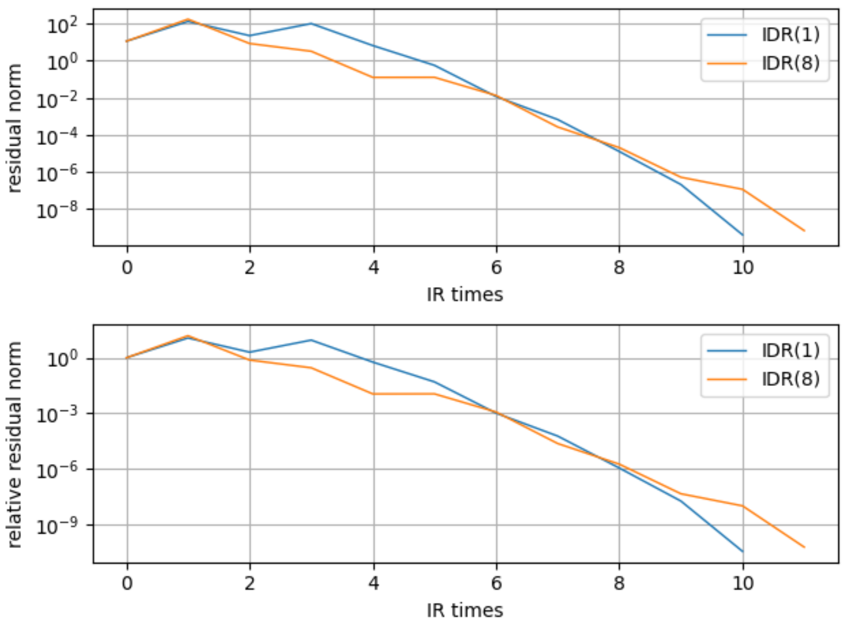

Figure 7: Performance of IDR-IR(s) with $\texttt{MAXIT} = 30$ and $u = 2^{-53}$.

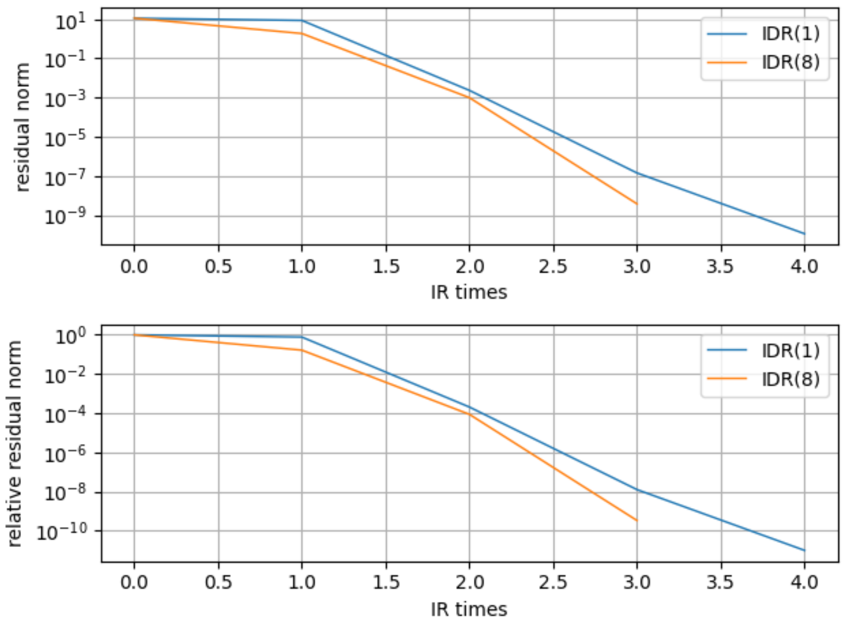

Figure 8: Performance of IDR-IR(s) with $\texttt{MAXIT} = 50$ and $u = 2^{-53}$.

Acknowledgements

This work was partly funded by National Natural Science Foundation of China under grant numbers 12101071, 12171051 and 12171052, by Fundamental Research Funds for the Central Universities under grant number 2023ZCJH02, and by French National Research Agency as part of project ADOM, under grant number ANR-18-CE46-0008.

References

[1] Y. Saad, M.H. Schultz, "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems", SIAM Journal on Scientific and Statistical Computing, 7, 856-869, 1986.

[2] Q. Zou, "GMRES Algorithms over 35 Years", Applied Mathematics and Computation, 445, 127869, 2023.

[3] H.A. van der Vorst, "Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems", SIAM Journal on Scientific and Statistical Computing, 13, 631-644, 1992.

[4] Y. Saad, Iterative Methods for Sparse Linear Systems, SIAM, Philadelphia, 2003, 2nd edition.

[5] G. Meurant, J. Duintjer Tebbens, Krylov Methods for Nonsymmetric Linear Systems: From Theory to Computations, Springer, Cham, 2020.

[6] P. Wesseling, P. Sonneveld, "Numerical Experiments with a Multiple Grid and a Preconditioned Lanczos Type Method", Approximation Methods for Navier-Stokes Problems, Springer, 543-562, 1980.

[7] P. Sonneveld, M.B. van Gijzen, "IDR(s): A Family of Simple and Fast Algorithms for Solving Large Nonsymmetric Systems of Linear Equations", SIAM Journal on Scientific Computing, 31, 1035-1062, 2009.

[8] E.C. Carson, N.J. Higham, "A New Analysis of Iterative Refinement and Its Application to Accurate Solution of Ill-Conditioned Sparse Linear Systems", SIAM Journal on Scientific Computing, 39, A2834-A2856, 2017.

[9] E.C. Carson, N.J. Higham, "Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions", SIAM Journal on Scientific Computing, 40, A817-A847, 2018.

[10] N.J. Higham, T. Mary, "Mixed Precision Algorithms in Numerical Linear Algebra", Acta Numerica, 31, 347-414, 2022.

[11] J.H. Wilkinson, "Progress Report on the Automatic Computing Engine", National Physical Laboratory, Teddington, 1948.

[12] P. Amestoy, A. Buttari, N.J. Higham, et al., "Combining Sparse Approximate Factorizations with Mixed-precision Iterative Refinement", ACM Transactions on Mathematical Software, 49, 4:1-4:29, 2023.

[13] E.C. Carson, N. Khan, "Mixed Precision Iterative Refinement with Sparse Approximate Inverse Preconditioning", SIAM Journal on Scientific Computing, 45, C131-C153, 2023.

[14] P. Amestoy, A. Buttari, N.J. Higham, et al., "Five-Precision GMRES-Based Iterative Refinement", SIAM Journal on Matrix Analysis and Applications, 45, 529-552, 2024.

[15] N.J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, Philadelphia, 2002, 2nd edition.
[16] T.A. Davis, Y. Hu, “The University of Florida Sparse Matrix Collection”, ACM Transactions on Mathematical Software, 38, 1:1-1:25, 2011.