The 9th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2024
3–7 June 2024, Lisboa, Portugal

ACCELERATING THE FLOWSIMULATOR: MESH DEFORMATION PERFORMANCE ENHANCEMENT THROUGH MIXED PRECISIONS

Marco Cristofaro*, **Johannes Wendler**, **Immo Huismann** and **Arne Rempke**

Institute of Software Methods for Product Virtualization
German Aerospace Center (DLR), Zwickauer Straße 46, 01069 Dresden, Germany
web page: https://www.dlr.de/sp, *e-mail: Marco.Cristofaro@dlr.de

Key words: Mesh Deformation, Mixed Precision, High–Performance Computing

Summary.
This study investigates the performance enhancement of mesh deformation processes on High-Performance Computing systems through the integration of mixed precision techniques. The linear system, derived from the elastic analogy theory, is solved with iterative methods provided by the library `Spliss`. Reducing the memory size occupied by floating-point values, i.e. switching from double- to single-precision, at key points of the linear solver algorithm yields a noteworthy reduction in runtime without compromising the accuracy of the solution. Through industrial relevant numerical experiments, the feasibility and benefits of this approach in the context of mesh deformation problems are presented. In conjunction with algebraic multigrid methods, mixed precision techniques increase the industrial readiness level of mesh deformation methods based on the elastic analogy theory. These findings offer valuable performance enhancements for High-Performance Computing simulations within the aerospace industry.

1 INTRODUCTION

Global regulations establish guidelines to reduce greenhouse gas emissions, with the ultimate aim of achieving climate neutrality. Among others, aviation contributes a significant portion of the total CO_2 emissions, making advancements in cleaner aircraft design crucial. These innovations can substantially reduce pollution, especially given the growing number of flights. During the design phase, engineers can leverage advanced simulation tools to significantly reduce the cost and time required for testing procedures. Current regulations also permit certification by simulation, provided the simulation tools are demonstrated to offer the same reliability as real-world testing. Therefore, the availability of accurate and fast simulation methods can significantly impact the aviation market. Industrial-grade simulation tools, supported by a robust computing infrastructure, are then expected to deliver high-fidelity solutions within engineering-acceptable timeframes.

The availability of modern computing resources should yield increasingly accurate results without extending the expected time-to-solution. These advancements include faster computing processors, larger memory bandwidth, more computing nodes, and faster interconnecting networks, among other improvements. Although most metrics have improved over the past years in the High-Performance Computing (HPC) world, disparities exist. Notably, memory bandwidth has not generally increased as rapidly as the execution speed of floating-point operations. Consequently, this shifts the optimal execution point of algorithms on modern hardware toward higher operational intensity [1]. In other words, to simultaneously take advantage from the maximum memory bandwidth and floating-point operations execution speed, modern algorithms should either perform more operations on already loaded data or reduce the size of data required for the operations. In this regards, mixed precision methods have been developed and implemented. Their application to linear solvers involves using different numerical precisions, such as single- and double-precision, within the same computational algorithm to balance performance and accuracy. Single-precision calculations use half the memory compared to double-precision, potentially achieving a theoretical speedup of up to 2 in memory-bound algorithms. This approach can be applied to parts of the algorithm that are less sensitive to truncation errors, such as preconditioning steps, without compromising the overall accuracy of the solution.

The application of mixed precision techniques to Computational Fluid Dynamics (CFD) simulations has been extensively explored in various studies. Execution time and energy consumption for both CPU and GPU architectures were analyzed in a 2D CFD problem, revealing significant impacts in [2]. In another study, performance was evaluated on GPUs using both steady flow and large-scale unsteady turbulent flow simulations [3]. Additionally, mixed precision effects were examined in multiphase simulations involving up to 8,000 CPUs in [4]. The technique was also assessed across a wide range of benchmarks within the open-source CFD solver OpenFOAM, demonstrating its broad applicability [5]. In contrast, research on the application of mixed precision techniques in Computational Structural Mechanics (CSM) solvers is less extensive. Among other works, in [6] double-precision and mixed-precision solvers were compared for linear systems of equations typical of finite element discretizations. More recently, mixed precision has been implemented for simulating nonlinear dynamic problems on GPUs [7].

This paper investigates the runtime benefits of using mixed precision to solve the linear problem arising from mesh deformation using the elastic analogy method. The simulations are conducted within the *FlowSimulator* environment maintained by the German Aerospace Center (DLR, *Deutsches Zentrum für Luft- und Raumfahrt e.V.*). Similar to other coupling library, e.g. see [8], this enables multidisciplinary parallel simulations by providing exchange layers between different simulation blocks, such as CFD and CSM. In this work, the mesh deformation plugin implementing the elastic analogy method in the *FlowSimulator* is extended to incorporate recent developments in mixed precision within the Sparse linear system solver (`Spliss`) library [9]. Six different cases are then considered, featuring various mesh topologies and vertex counts ranging from $2,200$ to 31 million. These cases include basic 2D airfoils as well as a complex full-aircraft configuration. The deformation problem is defined either by a rigid body rotation or by structural deformation resulting from the steady aeroelastic equilibrium solution, as the one found in a previous work [10]. The simulations are conducted with varying numbers of MPI-ranks, pre-computed to ensure comparable parallelization levels across the different cases, with mesh vertices per MPI-rank ranging from 1 million down to 100. The linear solver is set to use either the Generalized Minimal Residual (GMRes) or the Bi-Conjugate Gradient Stabilized (BiCGStab) method as iterative solver preconditioned by the Algebraic Multigrid (AMG) with Jacobi or Gauss-Seidel as smoother on each level. This variety of cases and settings aims to cover a wide range of applications, ensuring that the conclusions are as broadly applicable as possible.

2 THE FLOWSIMULATOR

Airbus and DLR have been working together for the last couple of decades to develop a unified environment for parallel flow simulations. This initiative, called *FlowSimulator*, aims to improve interoperability between various simulation tools by offering a common framework with a shared data-exchange layer [11]. At the core of the *FlowSimulator* is the *FlowSimulator* Data Manager (`FSDM`). This is designed to store mesh-based data to facilitate information exchange between different simulation blocks, e.g. fluid and structural domains. These comprises structural deformation, aerodynamic loads, but also heat fluxes. `FSDM` also provides a vast variety of functionalities that can be used in any part of a simulation toolchain to enhance and simplify the capabilities of each simulation block. Among the most commonly used are mesh import, export, and partitioning. The functionalities at the mesh-elements level are implemented in C++ to enable fast processing of large data sets, while a Python wrapper using SWIG is employed for higher-level operations. This programming language combination allows easy implementation of control scripts without compromising the performance of critical processes. DLR's CSM solvers `b2000++pro` [12], as well as DLR's high-fidelity CFD solvers, `CODA` [13] and `TAU` [14], are integrated in the *FlowSimulator*. Furthermore additional interfaces exist to couple externally developed solvers, e.g. `TRACE` [15], `NASTRAN` [16], `CalculiX` [17], taking full advantage from the environment interoperability.

3 MESH DEFORMATION

When conducting multi-disciplinary optimization and fluid-structure interaction simulations, obtaining CFD solutions around multiple geometries is essential. Despite advancements in immersed boundary techniques have improved their accuracy and applicability, surface-conforming meshes remain the standard for accurate flow simulations. This comes with the drawback that high-quality meshes are complex and time-consuming to generate, and the process is far from being fully automated. For these reasons, a reliable mesh deformation method is essential for adapting existing meshes to modified geometries.

Among others, the elastic analogy method has proven to be a robust approach for generating high-quality CFD meshes following surface deformations. This method treats the volumetric CFD domain as an elastic body, subject to prescribed deformation as boundary conditions [18]. In this work, the plugin implemented in the *FlowSimulator* framework is considered. A detailed description of the method implementation specifics are described in [19]. The plugin can use various linear solver libraries, but, for the purpose of this study, DLR's internal linear algebra library `Spliss` [20] is used. This library is specifically implementated for block sparse matrices (characterizing most of CFD and finite element methods) and it can take advantage from modern HPC systems architectures thanks to the native support to hybrid parallelization and GPU exploitation.

4 MIXED PRECISION

All numerical simulations rely on the computer representation of numbers. Although double-precision is considered as standard in most scientific calculations, the provided accuracy may not actually be necessary in all parts of the algorithm. For example, the preconditioning step in a linear system solver algorithm aims to provide just a rough solution improvement. For this step, the accuracy provided with double-precision values may be unnecessary. Reducing the memory occupancy per value from 64 to 32 bit, i.e. double- to single-precision, may have a minor impact on the overall convergence rate. Having top-level solvers in double-precision and bottom-level solvers in single-precision is then called mixed precision.

Switching from double- to single-precision floating-point values can significantly impact the performance of algorithms, particularly in memory-bound operations. The reduced size of the data accelerates the transfer rate through the memory interface, which is crucial for improving speed in memory-intensive tasks like matrix-vector operations within linear solvers. These operations are often bottlenecked by memory bandwidth rather than computational speed. By decreasing the memory footprint of matrices and vectors, mixed precision techniques reduce the volume of data that needs to be transferred between the memory and the CPU. This reduction minimizes CPU idle time spent waiting for data, leading to faster execution of the algorithm. Further details about the mixed precision implementation in the library `Spliss` are available in [9].

5 TEST CASES

In order to best evaluate the performance improvements brought by mixed precision, a range of mesh deformation test cases have been adopted: the selected mesh have sizes range between $2,200$ and 31 million vertices with a variety of mesh types. These are shown in Fig. 1.

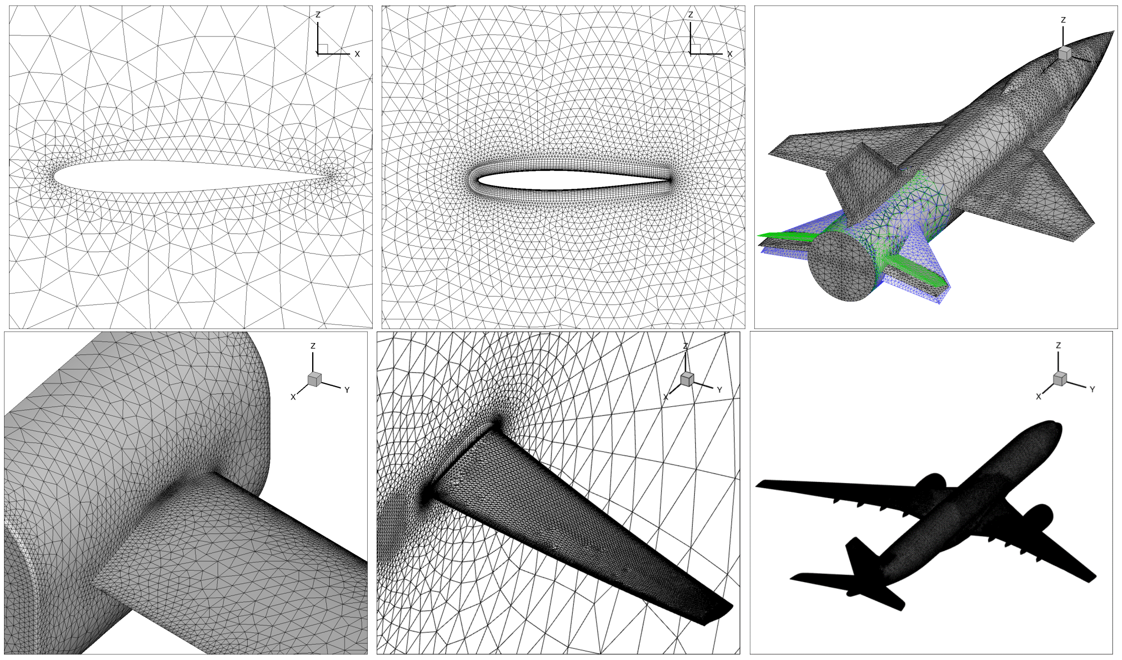

Figure 1: Test cases meshes. From left to right: NACA0012, NACA64A010, and SDM on the first row, and Wing-body, LANN wing, and XRF1 on the second row.

In order to overcome the difference in the number of mesh vertices between the different cases, the number of MPI-ranks is chosen to match a predefined MPI-domain size (i.e. averaged number of vertices per MPI-rank). Given the available meshes and computational resources, this ranges from 10^6 down to 100 vertices per MPI-domain, with a factor 10 between different cases.

Table 1: Test cases matrix: number of MPI-ranks used for each case to obtain predefined MPI-domain sizes.

Case name	Total number of vertices	Number of MPI-ranks				
NACA0012	$2.2 \cdot 10^3$	22	2			
NACA64A010	$21 \cdot 10^3$	210	21	2		
SDM	$59 \cdot 10^3$	590	59	6		
Wing-body	$230 \cdot 10^3$	-	230	23	2	
LANN	$1.2 \cdot 10^6$	-	-	120	12	1
XRF1	$31 \cdot 10^6$	-	-	-	310	31
Vertices per MPI-domain		10^2	10^3	10^4	10^5	10^6

The resulting number of ranks adopted for each simulation is reported in the test cases matrix presented in Table 1. The top-right corner of the table cannot be filled as the corresponding meshes are too small to reach the required MPI-domain size, while the bottom-left corner has not been computed in order to limit the overall analyses cost.

Furthermore, in order to cover the maximum number of configurations, various solver stacks are used to set up the linear solver library `Spliss`. Figure 2 presents a graphical representation of the linear solver stack definition. The thick vertical arrows indicate that for each iteration of the current solver block (large light blue blocks), the solver block below is invoked. Specifically, for each iterative solver iteration, an AMG V-cycle is applied. This cycle includes a smoother for each coarsening and refining step. Finally, an inversion method is applied for each smoother iteration. For each of the main blocks, the corresponding methods are linked with smaller blocks (light green): the selected iterative solvers are GMRes or BiCGStab, Jacobi or Gauss-Seidel are used in the smoother step together with blocks or lines inversion techniques [19]. Although convergence behavior may be improved by adopting case specific settings, the default values are used here for sake of generality. These consist of a Krylov space of dimension 100 for GMRes,

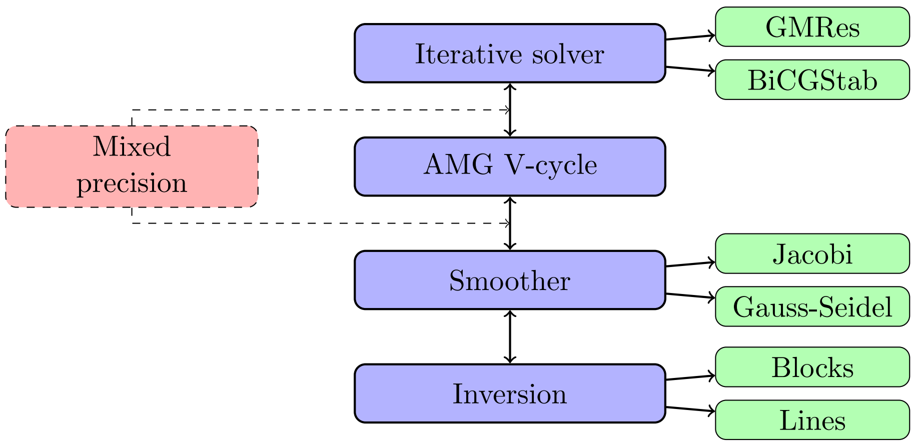

Figure 2: Linear solver stack representation. The blue large blocks represent the different solver components and the green small blocks the possible methods for each component. The dashed red block shows where the conversion between double- and single-precision may be applied.

2 Gauss-Seidel iterations, and 4 Jacobi iterations with a 0.4 damping factor for the smoothers. The number of multigrid levels is computed dynamically based on the average size of the local MPI-domain, using the formula $\log_8(\mathrm{n}_v/\mathrm{n}_{MPI})$, where n_v is the total number of vertices and n_{MPI} is the number of MPI-ranks. This approach stems from the fact that in a structured mesh, a geometric multigrid method would ideally group 8 neighboring hexahedral cells for each coarsening level. Consequently, the degrees of freedom are reduced by a factor of 8 between each layer, leading to a maximum number of local coarsening levels determined by the logarithm in base 8 of the local number of vertices. Although this method is an approximation, it effectively estimates the required number of grid levels given the local MPI-domain size, minimizing the risk of selecting too many or too few levels.

The eight solver stack combinations from the methods defined in Figure 2 are then applied to each of the case presented in Table 1. In turns, 128 simulation setups are generated by combining meshes, MPI-parallelizations and linear solver setups. A common residual tolerance of 10^{-12} is used for all cases, as this has been found to be sufficient to obtain a high-quality deformed mesh that allows to carry further CFD simulations [10]. A short description of each mesh is provided in the next sections.

5.1 NACA0012

The NACA0012 airfoil mesh is unstructured and composed by $2.1 \cdot 10^3$ prismatic cells and $2.2 \cdot 10^3$ vertices. Since the case is two-dimensional, the volumetric mesh is obtained by extruding the planar mesh in the perpendicular dimension by one chord length and applying symmetry conditions (i.e. no-normal movement) on both sides. Further boundary conditions for the mesh deformation problem are the fixed position of the vertices belonging to the farfield and airfoil surface. The deformation is computed with a rigid rotation of $10°$ of the airfoil at 25% of the chord.

5.2 NACA64A010

The NACA64A010 airfoil is a hybrid mesh (unstructured with structured boundary layer refinement) composed by $11 \cdot 10^3$ prismatic cells, $4.8 \cdot 10^3$ hexahedral cells and $21 \cdot 10^3$ vertices. Similarly to the NACA0012, a single layer volumetric mesh is used to model a two-dimensional case. The mesh deformation boundary conditions are of no-normal movement on the two symmetric sides and fixed position for the vertices laying on the farfield and on the airfoil. Also in this case, a $10°$ rotation around 25% of the chord is imposed to the airfoil surface.

5.3 SDM

The SDM represents a generic fighter aircraft geometry. The mesh is unstructured with $310 \cdot 10^3$ tetrahedral cells and $59 \cdot 10^3$ vertices. The mesh deformation problem is set by rotating of $10°$ the horizontal tail surface and setting no-normal movement conditions on the horizontal tail and on the tail of the fuselage, while keeping fixed all remaining boundary vertices. The visualization of the imposed deformation is shown in Fig. 1 (top-right image).

5.4 Wing-body

The Wing-body case represents a critical part of an aircraft CFD mesh, consisting in the connection region between fuselage and wing. The mesh is unstructured and made from $398 \cdot 10^3$ tetrahedra, $9 \cdot 10^3$ pyramids, $303 \cdot 10^3$ prisms and a total of $230 \cdot 10^3$ vertices. The deformation is computed from a rotation of 10° of the wing. All boundary vertices are set as fixed, with exception for the ones belonging to fuselage, for which a no-normal movement boundary condition is imposed.

5.5 LANN

The LANN wing geometry was originally presented in [21] and consists of a half-wing, as used for wind tunnel testing. The hybrid mesh is composed of $780 \cdot 10^3$ tetrahedra, $22 \cdot 10^3$ pyramids, $6 \cdot 10^3$ prisms, 10^6 hexahedra and a total of $1.2 \cdot 10^6$ vertices. Detailed geometry and mesh characteristics are available in [10]. A 10° rotation is imposed as deformation around 25% of the chord at the wing root. No-normal movement boundary condition is set on the symmetry plane, while the position of all remaining boundary vertices is fixed.

5.6 XRF1

The XRF1 (eXternal Research Forum) aircraft case is a full aircraft configuration, including powered engines [22]. The volume mesh is a hybrid mesh: unstructured with strucutured boundary layer refinement. The mesh consists of $65 \cdot 10^6$ tetrahedra, $117 \cdot 10^3$ pyramids, $38 \cdot 10^6$ prisms, $665 \cdot 10^3$ hexahedra and a total of $31.5 \cdot 10^6$ vertices. The resulting deformation field obtained from the steady aerolastic equilibrium simulations presented in [10] are adopted to deform the mesh in a single step. All boundary vertices position are then fixed.

6 RESULTS

6.1 Computational setup

The calculations are carried on DLR's high-performance computer, CARO. The computational nodes comprise two AMD EPYC 7702 CPUs. Each CPU has 64 physical cores, with 16 MB of L3 cache shared among groups of 4 cores. A total of 256 GB DDR4 RAM is installed in each node. As changes in the CPU frequency can strongly influence runtimes, a fixed clock frequency of 1.8 GHz is imposed during calculation. In order to take advantage from the hardware architecture, a hybrid parallelization technique is adopted: a single MPI-rank with 4 OpenMP threads is assigned to each group of 4 cores sharing the L3 cache. With reference to Table 1, the calculations used then up to $2,360$ cores, occupying 19 compute nodes.

As mixed precision affects only the linear solver, i.e. not the problem setup and the matrix assembly, only the runtime needed to effectively compute the linear problem solution is presented in this work. The roofline model performance analyses of the matrix assembly can be found in a previous work [23]. The resulting linear solver runtimes of the different cases and setups presented in Sec. 5 range between $0.2\,\mathrm{s}$ and $6\,\mathrm{min}$.

6.2 Strong scaling example

As showcase, the strong scaling measurements obtained from the LANN wing are presented. To justify the use of multigrid in the preconditioning phase, Fig. 3 compares runtimes between applying a multigrid approach and a single-grid approach with the full solver stack in double-precision. The linear solver is configured with GMRes as the outermost iterative solver, using

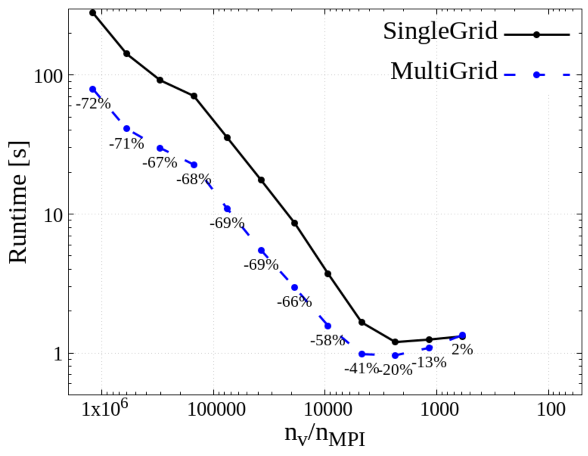

Figure 3: Multigrid effects on runtime for the LANN wing.

Jacobi with line inversion. The results highlight the significant impact of the multigrid approach, demonstrating its ability to reduce the runtime of the mesh deformation problem by substantial margins, with improvements of up to 72%.

The effect of mixed precision on the same strong scaling measurements of the LANN wing is shown in Figure 4. Single-precision values are employed in all preconditioning steps, meaning that the precision adapter lays between the iterative solver GMRes and the AMG. Results show a runtime improvements up to 33% when applying mixed precision. However, the runtime benefit

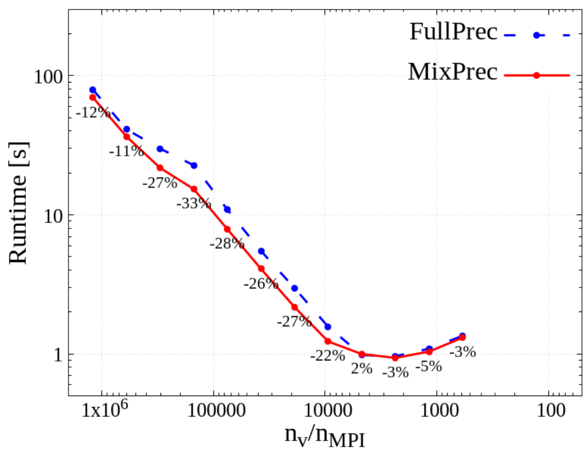

Figure 4: Mixed precision effects on runtime for the LANN wing.

strongly reduces for parallelization levels of 10^4 vertices per MPI-rank and below. This is further discussed in section 6.4.

To facilitate comparison across all test cases in the following analyses, the runtime benefits relative to the full double-precision linear solver stack are averaged for each predefined parallelization level (see Table 1). These averages are calculated across all test cases and linear solver configurations, allowing for a clear view in the single graphs presented hereafter.

6.3 Mixed precision strategies

As presented in Sec. 5, the linear solver stack can be composed by multiple components, ranging from the outermost iterative solver to the innermost sub-matrices inversion. In this regards, the conversion between single- and double-precision can be set between any of these components. In this study, two configurations are tested and compared to the baseline results of using double-precision throughout. With reference to Fig. 2, the configurations are: using double-precision for the iterative solver while keeping the rest in single-precision (i.e. setting the precision adapter between the iterative solver and AMG) and maintaining double-precision for the iterative solver and AMG while keeping the smoother and inversion in single-precision. In this work, all values from the matrix, the right-hand-side vector and the solution vector are casted between double- and single-precision, as no major difference has been detected in applying the casting only to the matrix [9].

In Figure 5, the runtime benefits with respect to the full double-precision linear solver stack are presented for the different strategies. The error bars represent the standard deviation of the averaged data, which is significant given the diversity of test cases and solver settings.

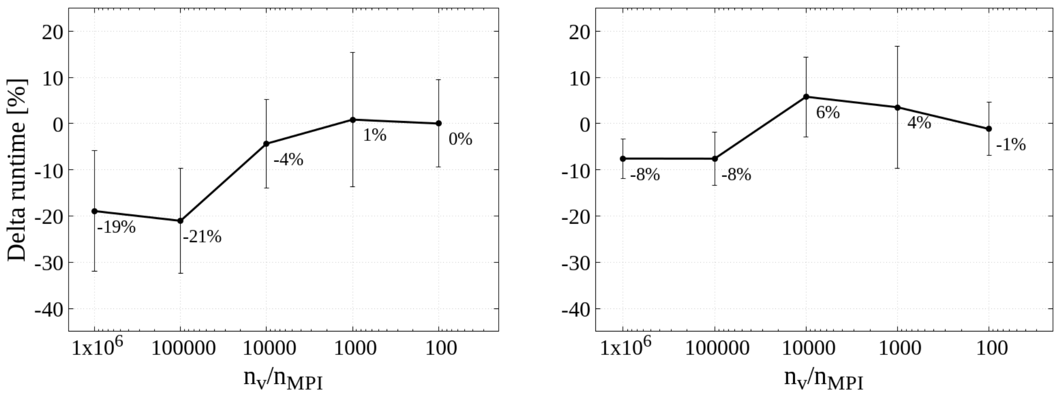

Figure 5: Effects of mixed precision strategies in terms of relative runtime difference with respect to the full double-precision solver stack. Precision adapter applied between the iterative solver and AMG (left) and between AMG and the smoother (right).

The conclusions that can be drawn from the results are:

- larger part of the linear solver in single-precision leads to larger benefits in terms of runtime
- having the iterative solver in double-precision and the AMG, smoother and inversion in single-precision leads to the most runtime benefit among the analyzed strategies.

6.4 Mixed precision effects analysis

In this section, the effects of mixed precision on the linear solver are investigated. Based on the findings in Sec. 6.3, the most effective strategy has been identified as using double-precision for the iterative solver while employing single-precision for the AMG, smoother, and inversion processes. This approach is exclusively analyzed in the following discussion. The total runtime of the linear solver can be broken down into two key components: the number of iterations required by the iterative solver and the average time taken for each iteration. Figure 6 illustrates the impact of mixed precision on these components by showing the relative differences compared to a fully double-precision solver stack. The results demonstrate that while the number of iterations increases on average by up to 7% with mixed precision, the time per iteration is significantly reduced, by as much as 26%. This reduction leads to an overall runtime improvement of up to 21% (refer to Fig. 5, left).

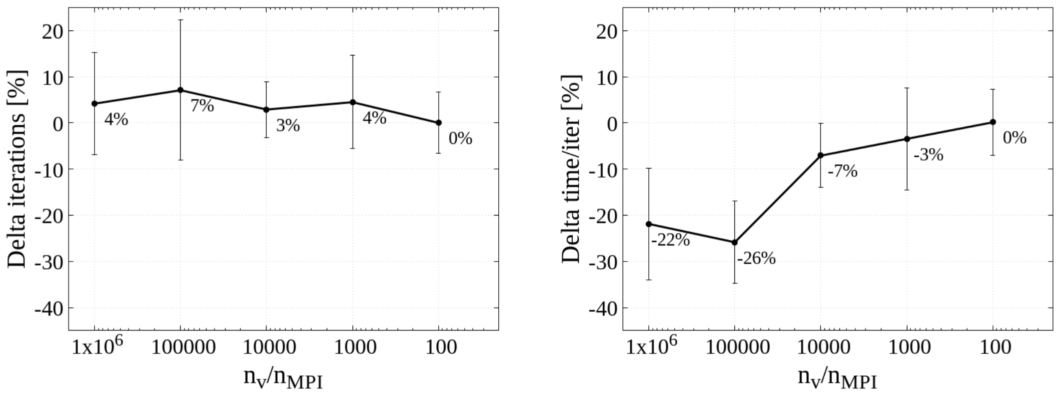

Figure 6: Decomposed runtime effects of mixed precision applied between iterative solver and AMG. Relative difference of number of iterative solver iterations (left) and time spent for each iterative solver iteration (right), with respect to the full double-precision solver stack.

As discussed in previous sections, the performance benefits of mixed precision diminish significantly when the parallelization level corresponds to 10^4 vertices per MPI-rank or fewer. Figure 6 shows that in these cases the improvements in time per iteration become negligible. This reduced effectiveness of mixed precision is primarily due to the fact that with such small local MPI-domains, the memory required by the matrix and vectors fits within the L3 cache (16 MB per MPI-rank on the used hardware). As a result, the amount of data passing through the memory interface is significantly reduced, limiting the benefits of mixed precision in terms of improving memory bandwidth utilization. In these cases, the primary runtime bottleneck may shift from memory bandwidth to MPI-communication latency and speed, where mixed precision offers only minimal advantage.

7 CONCLUSIONS

This work investigates the potential of applying mixed precision techniques to the mesh deformation problem within the *FlowSimulator* environment using the `Spliss` linear solver library. In multidisciplinary simulations, a mesh deformation method is essential for modeling geometric changes, such as those caused by structural deformations or shape optimization, in CFD meshes. The linear elastic analogy method, which involves solving a large linear system, offers a robust and efficient alternative to traditional interpolation methods.

Since linear solver algorithms typically operate in memory-bound regions, mixed precision techniques can be employed to reduce the data volume transferred between RAM and CPUs, thereby decreasing calculation time. Results from two mixed precision strategies applied to the linear solver are presented, with comparisons made relative to a fully double-precision solver stack. To ensure general applicability to the linear elastic analogy mesh deformation method, a wide range of cases has been examined. The analysis includes results from 128 cases, combining six meshes of sizes ranging from $2,200$ to 31 million vertices, eight linear solver configurations, and parallelization levels spanning from 1 million down to 100 vertices per MPI-rank. All calculations are run on DLR supercomputer CARO, using up to $2,360$ cores.

The findings indicate that using double-precision for the iterative solver and single-precision for the AMG, smoother, and inversion processes provides the greatest runtime benefit, achieving up to a 21% reduction. Further analysis reveals that mixed precision significantly reduces the time per iteration, while the number of iterations only slightly increases. However, for parallelization levels corresponding to 10^4 vertices per MPI-rank or fewer, the benefits of mixed precision decrease and factors other than memory bandwidth become dominant.

This work provides valuable insights into applying mixed precision techniques to the elastic analogy-based mesh deformation problem in High-Performance Computing environments, demonstrating the expected runtime benefits across a wide range of parallelization levels. Future steps include the analyses of fluid-structure interaction simulations, in which mixed precision is applied to all the most time-consuming components, CFD, CSM, and mesh deformation. This would allow for an assessment of runtime benefits both within each individual component (with specific linear solver settings) and in the overall fluid-structure interaction simulation.

Acknowledgements The authors thank Airbus for supplying the industrial–relevant XRF1 testcase. This research was funded in parts via the DLR–internal projects COANDA. The authors gratefully acknowledge the scientific support and HPC resources provided by the German Aerospace Center (DLR). The HPC system CARO is partially funded by "Ministry of Science and Culture of Lower Saxony" and "Federal Ministry for Economic Affairs and Climate Action".

REFERENCES

[1] M. Kronbichler, *High-Performance Implementation of Discontinuous Galerkin Methods with Application in Fluid Flow*, pp. 57–115. Cham: Springer International Publishing, 2021.

[2] H. Anzt, V. Heuveline, B. Rocker, M. Castillo, J. C. Fern´ndez, R. Mayo, and E. S. Quintana-Orti, "Power consumption of mixed precision in the iterative solution of sparse linear systems," in *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, pp. 829–836, 2011.

[3] A. Walden, E. Nielsen, B. Diskin, and M. Zubair, "A mixed precision multicolor point-implicit solver for unstructured grids on GPUs," in *ACM 9th Workshop on Irregular Applications: Architectures and Algorithms*, pp. 23–30, 2019.

[4] T. Ina, Y. Idomura, T. Imamura, S. Yamashita, and N. Onodera, "Iterative methods with mixed-precision preconditioning for ill-conditioned linear systems in multiphase CFD simulations," in *12th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA)*, pp. 1–8, 2021.

[5] F. Brogi, S. Bnà, G. Boga, G. Amati, T. Esposti Ongaro, and M. Cerminara, "On floating point precision in computational fluid dynamics using OpenFOAM," *Future Generation Computer Systems*, vol. 152, pp. 1–16, 2024.

[6] R. S. Dominik Göddeke and S. Turek, "Performance and accuracy of hardware-oriented native-, emulated- and mixed-precision solvers in fem simulations," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 22, no. 4, pp. 221–256, 2007.

[7] S. Wang, C. Wang, Y. Cai, and G. Li, "A novel parallel finite element procedure for nonlinear dynamic problems using GPU and mixed-precision algorithm," *Engineering Computations*, 02 2020.

[8] H.-J. Bungartz, F. Lindner, B. Gatzhammer, M. Mehl, K. Scheufele, A. Shukaev, and B. Uekermann, "preCICE – a fully parallel library for multi-physics surface coupling," *Computers & Fluids*, vol. 141, pp. 250–258, 2016. Advances in Fluid-Structure Interaction.

[9] J. Wendler, I. Huismann, O. Krzikalla, and A. Rempke, "Accelerating the FlowSimulator: Mixed precision linear solvers in industrial grade CFD," in *9th European Congress on Computational Methods in Applied Sciences and Engineering*, June 2024.

[10] M. Cristofaro, J. A. Fenske, I. Huismann, A. Rempke, and L. Reimer, "Accelerating the FlowSimulator: Improvements in FSI simulations for the HPC exploitation at industrial level," in *X. International Conference on Computational Methods for Coupled Problems in Science and Engineering*, June 2023.

[11] L. Reimer, R. Heinrich, S. Geisbauer, T. Leicht, S. Görtz, M. R. Ritter, and A. Krumbein, "Virtual aircraft technology integration platform: Ingredients for multidisciplinary simulation and virtual flight testing," in *AIAA SciTech Forum*, Januar 2021.

[12] N. Ebrahimi Pour and H. Klimach, "Node-level performance analysis of the structural mechanics solver b2000++pro," in *9th European Congress on Computational Methods in Applied Sciences and Engineering*, June 2024.

[13] T. Leicht, J. Jägersküpper, D. Vollmer, A. Schwöppe, R. Hartmann, J. Fiedler, and T. Schlauch, "DLR-project Digital-X – next generation CFD solver 'Flucs'," in *Deutscher Luft- und Raumfahrtkongress*, 2016.

[14] L. Reimer, "The FlowSimulator – a software framework for CFD-related multidisciplinary simulations," in *NAFEMS European Conference: Computational Fluid Dynamics (CFD) – Beyond the Solve*, 2015.

[15] K. Becker, K. Heitkamp, and E. Kuegeler, "Recent progress in a hybrid-grid CFD solver for turbomachinery flows," 06 2010.

[16] R. H. MacNeal, "Nastran theoretical manual," 1969.

[17] M. Freimuth, C. Berthold, and F. Herbst, "Towards computational efficient fully coupled aeroelastic simulations of turbomachinery blades with TRACE and CalculiX," in *X. International Conference on Computational Methods for Coupled Problems in Science and Engineering*, June 2023.

[18] R. P. Dwight, "Robust mesh deformation using the linear elasticity equations," in *Computational Fluid Dynamics*, pp. 401–406, 2006.

[19] A. Rempke, "Deformation of CFD meshes with anisotropic cells in a viscous boundary layer using line-implicit methods," in *New Results in Numerical and Experimental Fluid Mechanics XIV*, pp. 273–283, June 2023.

[20] O. Krzikalla, A. Rempke, A. Bleh, M. Wagner, and T. Gerhold, "Spliss: A sparse linear system solver for transparent integration of emerging HPC technologies into CFD solvers and applications," in *STAB-Symposium 2020*, pp. 635–645, Springer International Publishing, July 2020.

[21] G. C. Firth, "LANN wing design," *NASA. Langley Research Center Cryogenic Wind Tunnel Models*, 1983.

[22] N. Kroll, M. Abu-Zurayk, D. Dimitrov, T. Franz, T. Führer, T. Gerhold, S. Görtz, R. Heinrich, C. Ilic, J. Jepsen, *et al.*, "DLR project Digital-X: towards virtual aircraft design and flight testing based on high-fidelity methods," *CEAS Aeronautical Journal*, vol. 7, no. 1, pp. 3–27, 2016.

[23] N. Ebrahimi Pour, M. Cristofaro, I. Huismann, J. Gericke, and J. Wendler, "Accelerating the flowsimulator: Performance analysis of finite element methods on high–performance computers," in *International Parallel Tools Workshop*, October 2023.