Deformation of CFD Meshes with Anisotropic Cells in a Viscous Boundary Layer Using Line-Implicit Methods

Arne Rempke[1]

German Aerospace Center (DLR), Institute of Software Methods for Product Virtualization, Zwickauer Straße 46, 01069 Dresden, Germany, arne.rempke@dlr.de

Abstract. Various numerical applications in the context of multidisciplinary high-fidelity aircraft design, e.g. fluid-structure interaction or shape optimization, require the consideration of changes in the geometry of the aircraft shape. Such geometry changes can be realized using mesh deformations, and we demonstrate such a volume mesh deformation method based on elasticity analogy in conjunction with line-implicit solvers. Line-implicit solvers have already been successfully used in CFD simulations with strongly anisotropic cells in a viscous boundary layer, and are now applied to mesh deformation problems of such meshes. In this approach, the solution of a block-tridiagonal subsystem of the Jacobian matrix is computed exactly using the Thomas algorithm. Here, the selection of the tridiagonal part – representing the lines – is of crucial importance for the speed of convergence. A new algorithm for line identification is presented and compared with existing ones. Also, a comparison and combination of these methods with multigrid methods is performed and the implications for industrially relevant test cases are demonstrated.

Keywords: mesh deformation, CFD mesh, anisotropic cells, viscous boundary layer, line-implicit method, Thomas algorithm, line identification, numerical method, linear algebra, multigrid

1 Introduction

Mesh deformation is an important ingredient for multidisciplinary simulation or geometry optimization [6,10]. Since it is crucial for succeeding simulation parts that the resulting mesh stays intact (i.e. no collapsing elements), choosing a robust implicit method makes sense especially for large deformations or difficult meshes. The robust mesh deformation method we examine in this work is based on linear elasticity analogy [11] and will be described in section 2.

We describe the numerical algorithms that are employed in order to speed up solving the linear equation systems occuring in this method in section 3. These comprise line-implicit methods in general [5], but also a new algorithm for line detection (section 3.1), that is particularly well suited for finite-element-based discretizations, and the algebraic multigrid setup (section 3.2), that is important for an efficient handling of large meshes.

In section 4 we will present the results that were obtained from using, combining and comparing these methods on two industry-relevant test cases. Finally, section 5 contains a summary and some closing remarks.

2 Volume Mesh Deformation using Linear Elasticity Analogy

Mesh deformation using elasticity analogy is a robust form of volume mesh deformation. The method we use is based on the idea that the spacial CFD domain behaves like an artificial body under elastic deformation. We will shortly wrap up the idea in this section and refer to [11,9,1] for a more detailed description.

Following a linear approach, the strain tensor $\boldsymbol{\varepsilon}$ is written in dependency of the deformations $\mathbf{u}$ by

$$\boldsymbol{\varepsilon}(\mathbf{u}) = \frac{\mathbf{1}}{\mathbf{2}} \left((\nabla \mathbf{u}) + (\nabla \mathbf{u})^{\top} \right). \tag{1}$$

The assumption of a homogenous, isotropic material law gives us the following stress $\boldsymbol{\sigma}$:

$$\boldsymbol{\sigma}(\boldsymbol{\varepsilon}) = \frac{E}{1+\nu}\boldsymbol{\varepsilon} + \frac{\nu E}{(1+\nu)(1-2\nu)}\mathrm{tr}(\boldsymbol{\varepsilon})\mathbf{I}, \tag{2}$$

where E and ν are material parameters of the virtual elastic body, tr is the trace operator and $\mathbf{I}$ is the identity. Then we provide a deformation of the boundary $\partial\Omega$ of our CFD domain Ω using Dirichlet boundary conditions and solve the deformations according to the system of partial differential equations

$$\begin{aligned} \nabla \cdot \boldsymbol{\sigma}(\boldsymbol{\varepsilon}(\mathbf{u})) &= \mathbf{0} \quad && \text{on } \Omega, \\ \mathbf{u} &= \mathbf{u_D} && \text{on } \partial\Omega. \end{aligned} \tag{3}$$

2.1 Artificial Jacobian-based Stiffening

In order to ensure the geometrical properties of the elements like orthogonality, aspect ratio, and, most importantly, non-overlapping positive volume are conserved, especially the cells in the viscous boundary layer should behave stiff under the deformation.

This is achieved by choosing a higher Young's modulus E for those elements to be conserved, while other elements that are not that crticial can be modeled using a smaller E. The criticalness is indicated from the volume size of each element, as suggested by [11]: When choosing $E = \mathrm{vol}(e)^{-\chi}$ for each element e of the mesh, the smaller elements obtain a larger stiffness, while larger elements behave more flexible. Such a model is very convenient, since the size of the element is directly obtained from the transformation of the mesh element to a reference element in order to evaluate the quadrature formulas of the discretization.

For the computations presented in this paper, we used $\chi = 1$ and $\nu = 0.3$ for the Poisson ratio ν.

Using this approach, mesh deformations like presented in Figure 1 can be achieved, where the wing of an aircraft is bend quite a large amount and the volume mesh is successfully transformed to match the new surface geometries while preserving positive cells and the properties of the viscous boundary layer. The implementation operates on unstructured meshes with various element types.

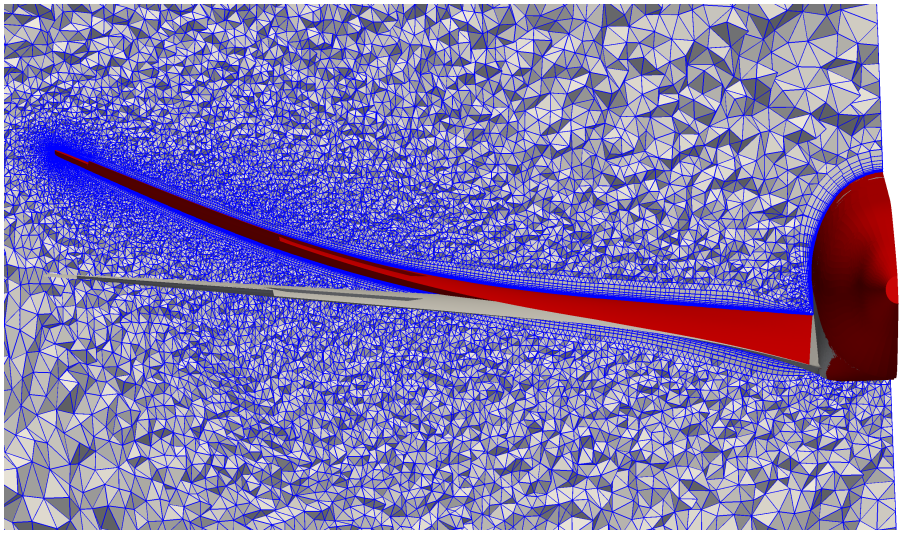

Fig. 1: Example of a CFD volume mesh deformation using linear elasticity analogy for the deflection of the wing of an aircraft: Undeformed (gray) and deformed (red) aircraft surface and unstructured deformed volume mesh (blue) with boundary layer.

3 Line-Implicit Methods

When solving the linear equation system resulting from a discretization of (3), iterative methods usually work quite well for the elliptical problem. However, when the discretization mesh contains very anisotropic cells with aspect ratios of more than 1000 : 1, these anisotropic problems are difficult to solve with standard methods. This is well known for the finite volume method in computational fluid dynamics, see [8,5,12,2,4].

The idea of line-implicit methods is to combine multiple cells to a line of cells, and to invert the tridiagonal part of the equation system instead of a purely diagonal part. Solving a tridiagonal equation system is an exact method when following the Thomas algorithm with no additional infill or approximation.

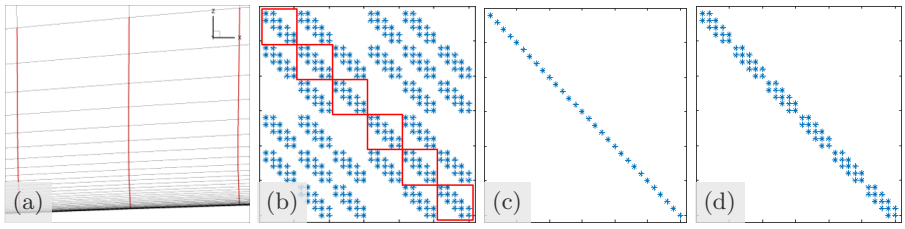

Fig. 2: Possible lines within the viscous boundary layer in the mesh (a) and the sparsity structure (after permutation) of a typical resulting matrix (b). Entries corresponding to the lines are marked by red boxes. Sparsity structure of the approximating matrices that are used as preconditioner in point-implicit (c) or line-implicit (d) methods.

Figure 2 illustrates the idea of the line-implicit method in contrast to the point-implicit method. In fact, we actually combine the 3 degrees of freedom (displacement in 3D space) for each vertex to a dense block and solve the block-tridiagonal system

$$\mathbf{A}_i\mathbf{x}_{i-1} + \mathbf{B}_i\mathbf{x}_i + \mathbf{C}_i\mathbf{x}_{i+1} = \mathbf{d}_i \quad i = 1, \ldots, n \tag{4}$$

with $\mathbf{A}_1 = \mathbf{C}_n = \mathbf{0}$ using the block Thomas algorithm [13]:

$$\begin{aligned}
\mathbf{B}'_1 &:= \mathbf{B}_1^{-1} \\
\mathbf{d}'_1 &:= \mathbf{B}'_1 d_1 \\
\left.\begin{aligned}
\mathbf{C}'_{i-1} &:= \mathbf{B}'_{i-1}\mathbf{C}_{i-1} \\
\mathbf{B}'_i &:= (\mathbf{B}_i - \mathbf{A}_i\mathbf{C}'_{i-1})^{-1} \\
\mathbf{d}'_i &:= \mathbf{B}'_i(\mathbf{d}_i - \mathbf{A}_i\mathbf{d}'_{i-1})
\end{aligned}\right\} & \; i = 2, \ldots, n \\
\mathbf{x}_n &:= \mathbf{d}'_n \\
\mathbf{x}_{i-1} &:= \mathbf{d}'_{i-1} - \mathbf{C}'_{i-1}\mathbf{x}_i \qquad i = n, \ldots, 2
\end{aligned} \tag{5}$$

3.1 Lines detection

For identifying good lines in an anisotropic mesh, algorithms like described in [5,4] are well known. They select the strongest anisotropy in the mesh as a seeding point to form a new line and then greedily append new elements to the line in both directions until all possible continuations seem to belong to an isotropic mesh region. This process is then looped until no starting point is available, because all anisotropic elements have been incorporated to a line.

In this work we introduce a different algorithm to detect lines, and we will describe it in this paragraph. It is also based on the criterion to detect anisotropy using the inverse Euclidean distances as weights in the graph to identify stronger couplings that should form a line. However, instead of the greedy approach to grow a single line, it first identifies for each vertex those two neighbors with the highest edge weight as line continuation candidates. Then, a tridiagonality check is done whether those two candidates are also directly connected within the graph, in which case only the highest edge weight is considered for a line segment and the connection to the second highest edge weight is rejected. If now two vertices did select each other as candidates and potential line neighbors, and the corresponding edge weight is the highest of all neighbors for at least one of these vertices, a line segment between them is formed. If both vertices are only connected by the second highest edge weight, the line segment is only formed, if and only if the two vertices with the highest edge weights are unconnected. Now, all line segments have been created and can finally be collected and followed to form lines. Special care needs to be taken to avoid cycles of line segments: Although the weights from typical meshes usually do not result in such specialities, the algorithm did not prohibit them until now. Either the line solver needs to be able to solve a cyclic tridiagonal system (e.g. using a transformation based on the Sherman–Morrison formula), or all cyclic lines should be split up to classical lines with two ends. In our observations it was sufficient to just split cyclic lines at the lowest edge weight.

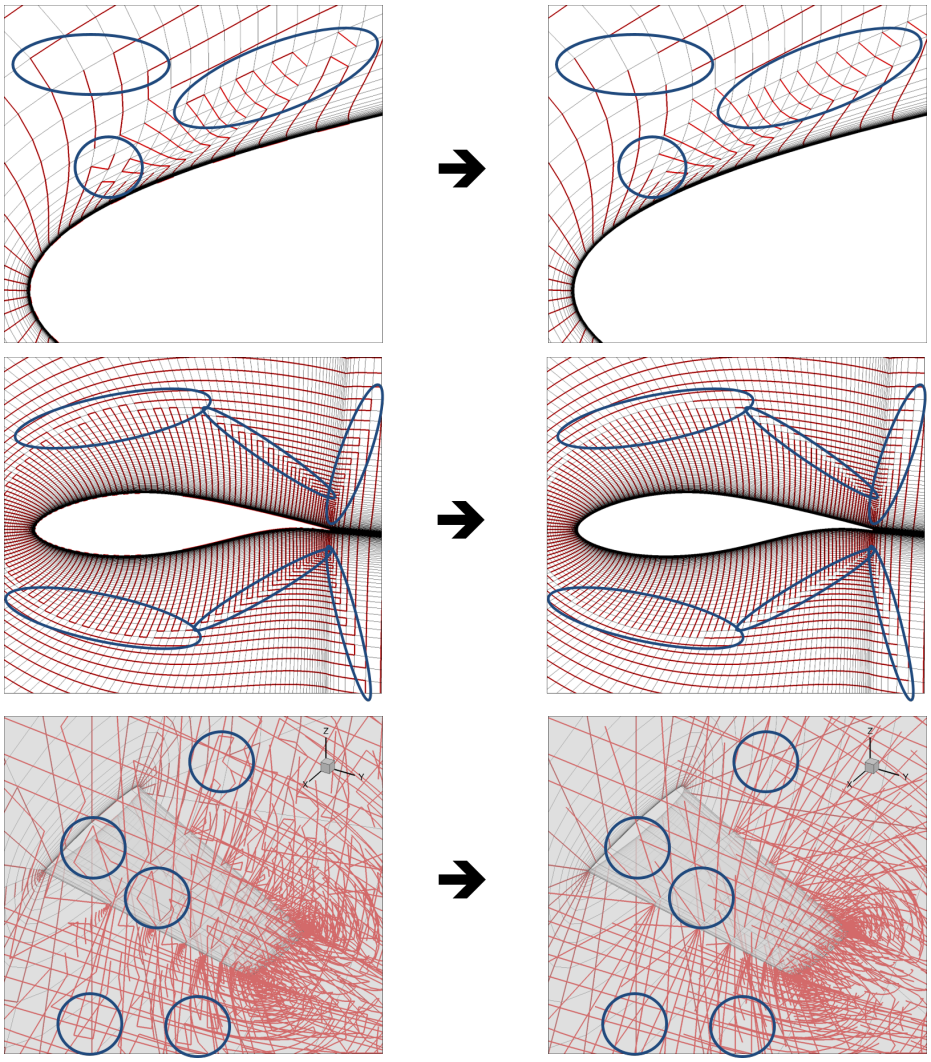

Fig. 3: Comparison of the old (left) and the new (right) lines identification algorithm on three different meshes. Regions of differently selected lines are highlighted. While the greedy selection chooses bends and turns, the new algorithm leads to mostly parallel lines.

The algorithm introduced in the last paragraph will be referred to as the new one, while we compare it to the greedy one described in [5,4]. Comparing the two algorithms leads to some observations:

1. The new algorithm does not stop at isotropic regions. Although it is possible to add a respective check, it turned out, that in general it is more beneficial for solver convergence rates to continue the lines and reach into isotropic regions of the mesh.
2. The new algorithm does involve local tridiagonality checks of the mesh. A tridiagonality check as described by [7] ensures that the submatrix that contains all relevant entries for a line is actually tridiagonal and therefore can be inverted exactly by the Thomas algorithm. In a nontridiagonal submatrix, there still exist nonzero entries apart from the tridiagonal regardless of permutations. As [7] described, ensuring tridiagonality helps to gain better solver convergence rates. However, instead of full tridiagonality checks for the whole line, we observed that local tridiagonality checks that involve at most three potential line segments are sufficient enough. This reduces the complexity and runtime of the line detection algorithm compared to a full tridiagonality check.
3. The new algorithm can be implemented in parallel, since it does not require to process one line after another, and only needs knowledge of a quite local part of the mesh when selecting the relevant line segments. Just the final step, the collection of constructed line segments to lines, needs to be consolidated between threads.
4. Performance-wise, both algorithms basically scale with the number of vertices in the mesh. However, while the greedy one requires to maintain and update a sorted list of the not-yet-incorporated vertices, the new one gets along without such central data structure, which also could become a bottleneck especially for parallel runs.

The improved line detection algorithm helps to find good lines, especially for FEM-based connectivities. See Figure 3 for a comparison of the results of the algorithm presented in [5,4] and the new line identification algorithm. Note that [5] uses a minimum aspect ratio of $\alpha = 4$ to stop the continuation of lines to isotropic regions. However, we observed a better convergence rate with long lines also reaching to isotropic regions, which is why we used $\alpha = 1$ for Figure 3. This makes the old algorithm also produce longer lines, but introduces artifacts like lines snapping off or even turning around that are handled much better with the new algorithm.

3.2 Multigrid

Since both the line-implicit and even more the point-implicit methods operate on a relatively local scope, reducing global errors can take many iterations on large meshes. In order to improve the convergence rates, a widespread method is to apply a multigrid technique, which is also what we did. We employed an isotropic

agglomeration based coarsening, where, in hexaedral structured regions of the mesh, approximately 8 vertices of the fine mesh are combined to one vertex of the next coarser level. The restriction operator just adds the different contributions of all agglomerated vertices with equal weight. Therefore, in contrast to semi-coarsening, we get quite a nice downsizing of the matrix with each level. Note, that this is a purely algebraic multigrid technique, since the agglomerates are identified via analysis of the nonzero sparsity pattern, and the coarse operators are created as restriction from the fine level matrix.

As smoother we use the line-implicit method only on the finest level. It was completely sufficient to only use a point-implicit method on the coarser levels. Depending on the mesh size we use up to 8 levels, and a V-cycle without pre-smoothing and 2 iterations of a multi-color Gauss-Seidel scheme is used as post-smoothing. This multigrid method is used as preconditioner for a GMRES Krylov subspace solver as illustrated in figure 4.

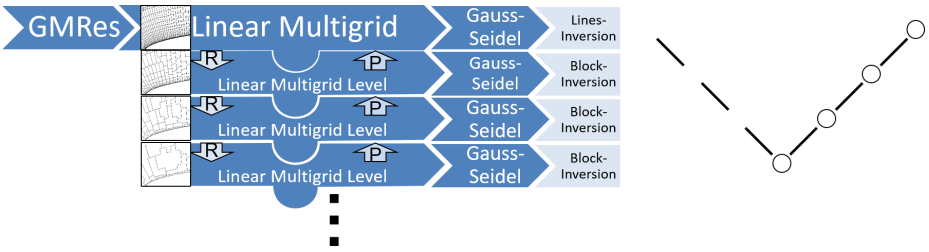

Fig. 4: Schematic solver chain (only four of the eight levels of multigrid are shown) used for the multigrid setup using line-implicit method on the finest level and point-implicit smoothing on coarser levels. V-cycle without pre-smoothing illustrated on the right.

4 Results

This section shows the results from combining the previously described methods. The investigations are realized by extending the implementation FSMeshDeformation [9] and using the Spliss linear algebra library [3]. Note that all computations converge to the same deformed meshes (except for small floating-point errors), which do not contain any collapsed cells. The main focus of this comparison is on compute time and effectiveness of the different methods to solve the large and distributed linear equation systems. All computations are performed on a cluster of two chips AMD EPYC 7601 (32 cores; 2,2 GHz) per node with 64 processes per node.

4.1 NASA Common Research Model

The first testcase is a 3D mesh of the Common Research Model (CRM) from the Drag Prediction Workshop, containing 29M nodes and aspect ratios of $4000 : 1$ in the viscous boundary layer. A comparison of the convergence of the different methods for solving the linear equation system on 512 processes in parallel

can be found in figure 5. While one can see that the single level point-implicit method gets stuck after a reduction of approximately 12 orders of magnitude, both adding multigrid or line-implicit method help to converge further. Multigrid always gives a speed-up of a factor of at least 2 compared to single grid when reducing to 14 orders of magnitude. The new lines detection algorithm also reduces the runtime compared to the old line detection algorithm by 20-50%.

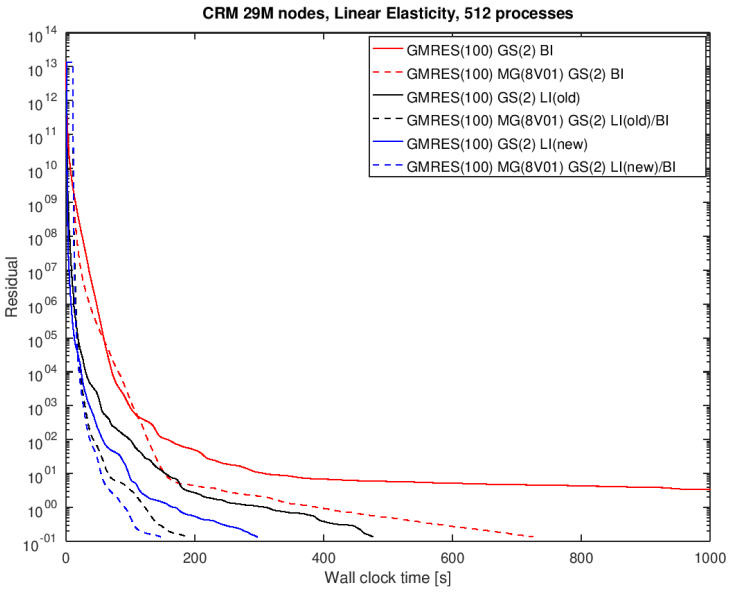

Fig. 5: Convergence time of different methods for the CRM test case.

4.2 ONERA M6 Wing

The second testcase is the ONERA M6 wing from NASA's Turbulence Modelling Resource. It is a 3D mesh with 61M nodes, 52M hexaedron and 17M prism cells, and aspect ratios of 18000 : 1 in the viscous boundary layer. A comparison of the convergence of the different methods for solving the linear equation system on 2048 processes in parallel can be found in figure 6. For the single level computations, both point-implicit and the line-implicit methods using the old line detection behave quite comparable, while the method using the new line detection is significantly faster.

For the multigrid methods, the additional time of setting up the transfer operators and coarse systems is more significant than for the CRM case, so at first we see no progress in the residual reduction at all. But after the setup phase, the multigrid methods surpass the single grid methods easily and show similar

characteristics as for the CRM case: The method using line-implicit smoother on the finest grid saves about 70% of the time of point-implicit, and the new lines detection reduces the runtime by another 30% compared to the old lines detection.

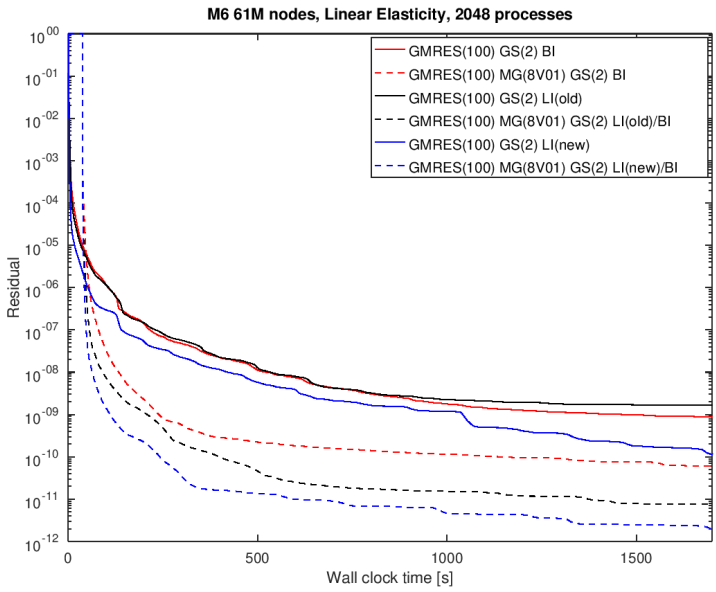

Fig. 6: Convergence time of different methods for M6 wing.

4.3 Comparison

Comparing both test cases, for M6 the gain by multigrid is much larger than that of line-implicit, compared to CRM. This is probably due to the fact that the partitioning did not take lines into account, and our line-implicit implementation is only able to handle lines that are fully available locally. Since the partitioner was not aware of the lines and the M6 test case contains fewer elements per domain than the CRM case, it is more likely that more lines were split by the partitioning, and thus the algorithmic strength is weakened. However, this explanation currently is just an assumption and needs to be confirmed in the future.

5 Conclusion

We have investigated the effectiveness of line-implicit methods for mesh deformation problems. In particular, these methods are very beneficial for CFD meshes

with highly anisotropic cells in a viscous boundary layer. We have presented a new line identification algorithm that is particularly useful for node-based discretizations, and have found lines that lead to improved convergence rates when used in line-implicit solvers, reducing solution time by 20-50%. We also compared these methods to multigrid methods and combined them, which improves convergence as well, especially for large grids. The multigrid line-implicit method using the new line detection algorithm is a very effective method and leads to a 90-95% reduction in solution time for industry-relevant meshes compared to the single-level point-implicit method.

6 Acknowledgement

This is an Author Accepted Manuscript version of the following chapter: *Arne Rempke, Deformation of CFD Meshes with Anisotropic Cells in a Viscous Boundary Layer Using Line-Implicit Methods*, published in *New Results in Numerical and Experimental Fluid Mechanics XIV*, edited by: *Andreas Dillmann et al, 2023, Springer* reproduced with permission of *Springer Nature Switzerland AG*. The final authenticated version is available online at: http://dx.doi.org/10.1007/978-3-031-40482-5_26

References

1. Dwight, R P: Robust Mesh Deformation using the Linear Elasticity Equations. In: Deconinck H, Dick E (eds) Computational Fluid Dynamics 2006, Springer: Berlin, Heidelberg (2006)
2. Eliasson, P, Weinerfelt, P, Nordström, J: Application of a line-implicit scheme on stretched unstructured grids. In: Proc. 47th AIAA Aerospace Sciences Meeting (2009)
3. Krzikalla O, Rempke A, Bleh A, Wagner M, Gerhold T: Spliss: A Sparse Linear System Solver for Transparent Integration of Emerging HPC Technologies into CFD Solvers and Applications. In: 22nd STAB/DGLR Symposium on New Results in Numerical and Experimental Fluid Mechanics XIII, Seiten 635-645. Springer International Publishing. STAB-Symposium 2020, Germany. (2021)
4. Langer, S.: Application of a line implicit method to fully coupled system of equations for turbulent flow problems. In: International Journal of Computational Fluid Dynamic 27:3, pp. 131–150 (2013)
5. Mavriplis, D J: Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. In: Journal of Computational Physics 145, pp. 141–165 (1998)
6. Ojha V, Fidkowski, K J, Cesnik C E S: Adaptive High-Order Fluid-Structure Interaction Simulations with Reduced Mesh-Motion Errors. AIAA Journal 59.6: 2084-2101 (2021)
7. Okusanya, T O: Algebraic Multigrid for Stabilized Finite Element Discretizations of the Navier-Stokes Equations. PhD dissertation, M.I.T., Department of Aeronautics and Astronautics (2002)

8. Pierce, N A, Giles, M B, Jameson, A and Martinelli, L: Accelerating Three-Dimensional Navier–Stokes Calculations. In: 13th AIAA Computational Fluid Dynamics Conference, no. 1997-1953 in Conference Proceeding Series, AIAA (1997)
9. Rempke, A: Netzdeformation mit Elastizitätsanalogie in multidisziplinärer FlowSimulator-Umgebung. In: Jahresbericht 2016 zum 20. DGLR-Fach-Symposium der STAB (2016)
10. Secco, N R, et al. Efficient mesh generation and deformation for aerodynamic shape optimization. AIAA Journal 59.4: 1151-1168 (2021)
11. Stein K, Tezduyar T E and Benney R (2003) Mesh moving techniques for fluid-structure interactions with large displacements. Journal of Applied Mechanics, 70(1):58–63 (2003)
12. Swanson, R C, Turkel, E, and Rossow, C-C: Convergence Acceleration of Runge–Kutta Schemes for Solving the Navier–Stokes Equations. Journal of Computational Physics 224: 365-388 (2006)
13. Thomas, L H: Elliptic problems in linear difference equations over a network, Watson Scientific Computing Laboratory Report, Columbia University (1949)