AMG in TAU: Adjoint Equations and Mesh Deformation

M. Förster[1] and A. Pal[2]

[1] Fraunhofer Institute for Algorithms and Scientific Computing SCAI
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
[2] Deutsches Zentrum für Luft- und Raumfahrt (DLR)
Lilienthalplatz 7, 38108 Braunschweig
malte.foerster@scai.fraunhofer.de,
anna.pal@dlr.de

1 Introduction

Dealing with aerodynamic and aeroelastic tasks typically involves large and ill conditioned linear systems of equations. Usually the solution of these equations is a time critical component of the overall simulation. While common one-level solution techniques tend to be rather inefficient, the appliance of a hierarchical method like algebraic multigrid (AMG) seems to be more promising in order to deal with the increasing demands for the linear solver. However, due to various sources of stiffness within the discretized problem, applying AMG in a straightforward way is not always possible. In the context of ComFliTe, the usage of classical AMG was evaluated for mesh deformation applications based on linear elasticity. For the solution of flow adjoint equations new and more sophisticated AMG methods were developed. All approaches have been integrated into the state-of-the-art linear solver library SAMG. The following report describes the modified algorithms utilizing AMG and summarizes the results obtained within the DLR simulation codes throughout the project.

2 Adjoint Equations

2.1 Governing Flow Equations and Their Discretization

The governing flow equations considered in this work are the steady compressible Reynolds-averaged Navier-Stokes equations supplied with Spalart-Allmaras turbulence model [1]. In conservative form, the equations are written as

$$\nabla \cdot (f^c(W) - f^v(W)) = S(W), \tag{1}$$

where W is the conservative state vector defined as $W=(\rho,\, \rho u,\, \rho v,\, \rho w,\, \rho E,\, \rho\tilde{\nu})^T$, ρ is the density, $U = (u, v, w)^T$ is the velocity vector, E is the total energy, and $\tilde{\nu}$ is the turbulence variable in the Spalart-Allmaras model. $f^c(W)$ and $f^v(W)$ are vectors of convective and viscous fluxes, respectively, and $S(W) = (0, 0, 0, 0, 0, s(W))^T$ denotes a source term.

N. Kroll et al. (eds.), *Computational Flight Testing*, Notes on Numerical Fluid Mechanics and Multidisciplinary Design 123,
DOI: 10.1007/978-3-642-38877-4_1,

A finite volume discretization is performed on unstructured grids of non-overlapping dual cells of a hybrid primary grid. For the discretization of convective fluxes in mean-flow equations we employ the Jameson-Schmidt-Turkel (JST) scheme with either scalar or matrix dissipation. The convective fluxes in the turbulence equation are discretized with an upwind scheme with piecewise-constant face reconstruction. Viscous flux discretization in the mean-flow equations is done either via thin shear layer (TSL) approximation or via Green-Gauss formula. TSL approximation is always employed for viscous fluxes in the turbulence equation. For more details concerning the employed discretization scheme we refer to the description of discretization used in DLR TAU-code, [2].

2.2 The Discrete Adjoint Equations

An important and also rather costly component of gradient-based optimization is the computation of gradients of the cost function with respect to the parameters of the problem (a set of so-called design variables). The adjoint method provides an efficient way of doing this for a large amount of design variables. The effort required for this evaluation is only weakly dependent on the number of design variables.

For convenience of representation, we write the finite volume discretization of the system (1) in the following way

$$R(W, X, D) = 0, \tag{2}$$

where R, called the residual of the discretizations, contains the discretization of both, inviscid and viscous fluxes as well as the turbulence source terms. D is a set of design variables and $X = X(D)$ is the computational mesh.

Suppose we want to minimize a cost function $I(W, X, D)$ with respect to the set of design variables D, subject to constraints

$$\begin{aligned} I(W, X, D) &\rightarrow \min_{D}, \\ R(W, X, D) &= 0, \\ T(X, D) &= 0 \ . \end{aligned} \tag{3}$$

where constraint $T(X, D)$ describes the mesh deformation. Following the adjoint-based approach, the gradient dI/dD can be computed via the formula:

$$\frac{dI}{dD} = \frac{\partial I}{\partial D} + \Lambda^T \frac{\partial R}{\partial D} + \hat{\Lambda}^T \frac{\partial T}{\partial D}, \tag{4}$$

where Λ and $\hat{\Lambda}$ are so-called flow- and mesh- adjoint variables, which are obtained from solving the flow-adjoint and mesh-adjoint equations, respectively. In this work we are solely concerned with solution of the flow-adjoint equations, which can be written as

$$\left(\frac{\partial R}{\partial W}\right)^T \Lambda = -\left(\frac{\partial I}{\partial W}\right)^T \ . \tag{5}$$

An advantage of the adjoint-based approach is that in order to evaluate the gradient of I with respect to any number of design variables, the adjoint equation must be only solved once.

Note that system (5) is a large sparse system of linear algebraic equations, which is generally very stiff for large practical applications, and thus requires a powerful linear solver. In this work we develop iterative solution methods for (5) which are based on algebraic multigrid.

3 Algebraic Multigrid

As opposed to geometric multigrid methods, algebraic multigrid does not explicitly use geometric information of a problem. All coarse levels, inter-grid transfer and coarse level operators are constructed purely algebraically, based only on the entries of a system matrix. Basic ideas of AMG methods were first introduced in the early 1980's by Brandt, McCormick, Ruge and Stüben [3], [4], [5], [6]. As geometric multigrid, algebraic multigrid relies on the efficient interplay of smoothing and coarse grid correction. However, algebraic multigrid is based on a different definition of smoothness.

In geometric multigrid the definition of smoothness depends on the chosen grid hierarchy. The error is called smooth if it can be approximated by a geometric interpolation of the correction from the next coarser grid. In contrast to this, in AMG coarser levels are constructed in such a way that all error frequencies are successfully reduced by the combination of smoothing and coarse grid correction. That is, the low frequencies or smooth error components that have to be taken care of in the correction step are indirectly defined by those components which are slow to converge by the chosen smoothing procedure.

The classical AMG algorithm consists of two phases - the setup phase and the solution phase. In the setup phase the coarse levels, intergrid transfer operators and coarse grid operators are constructed. During the solution phase, all AMG components including the smoother are employed in a multigrid cycling algorithm.

The construction of coarse levels and interpolation operators in a classical AMG method assumes M-matrices, which typically arise in the discretizations of elliptic scalar PDEs. Based on the graph of strong couplings between variables, the classical coarsening process tries to find an independent subset of nodes that maximizes the F(ine) to C(oarse) connectivity in order to allow for efficient interpolation routines from C to F with weights based on the matrix entries. Restriction operators are defined as the transpose of interpolation, and galerkin operators are used as coarse-grid operators (see e.g. [6]). For s.p.d. problems, this choice of operators ensures a variational principle which guarantees that coarse-grid correction processes are optimal w.r.t. the energy norm.

For the class of symmetric positive definite M-matrices classical AMG is known to be a very robust and efficient solver. However, extensions of classical AMG are required in order to solve efficiently non-elliptic PDEs or systems of PDEs. In both cases the matrices coming from the discretization are usually far from s.p.d. M-Matrices. Non-symmetry usually does not cause problems for

AMG as long as the given matrix is still definite. However, large (both positive and negative) off-diagonal entries, often leading to a strong violation of a diagonal dominance in the matrix, typically cause problems for algebraic multigrid. Development of special extensions of AMG are necessary to treat such problems efficiently. Extensions are also required in order to solve discretized systems of PDEs with AMG, where one has to distinguish different physical quantities and their different couplings in the linear system.

4 Defect Correction

Our goal is the solution of linear problems (5) corresponding to second-order accurate finite volume discretizations. Due to a typically low numerical dissipation in these discretizations the resulting linear systems are often very stiff with no diagonal dominance. Therefore, the construction of an efficient numerical method for these linear problems is a challenging task. In order to compensate for the lack of diagonal dominance, many iterative solution methods involve the introduction of a pseudo-timestep in order to converge to a steady state.

The defect correction approach [7], [8] is another alternative for solving second-order accurate problems. It employs an auxiliary first-order accurate discretization of the same continuous problem. The benefit of this method is due to the fact that a first-order accurate system is usually better-conditioned, sparser, and, in general, allows more efficient solution with iterative methods.

4.1 The DC Algorithm

Let us rewrite our target second-order accurate problem (5) in a compact way, employing common notations of linear algebra:

$$Ax = b, \tag{6}$$

where

$$A = \left(\frac{\partial R}{\partial W}\right)^T, \quad x = \Lambda, \quad \text{and} \quad b = -\left(\frac{\partial I}{\partial W}\right)^T .$$

If operator A_1 is a first-order accurate linear operator, corresponding to the same underlying problem, then the defect correction algorithm can be written as following

- Choose an initial approximation $x^{(0)}$ to the solution of (6)
- For $n = 1, 2, \ldots$ until convergence do:
  1. Compute a correction $e^{(n)}$ by solving

$$A_1 e^{(n)} = b - Ax^{(n-1)} \tag{7}$$

  2. Update solution vector by adding the correction

$$x^{(n)} = x^{(n-1)} + e^{(n)}$$

- End of do loop.

Obviously, the direct solution of the stiff target problem is avoided, since in (7) the second-order discretization is only used to evaluate the right-hand side vector. Instead, a first-order system with the same matrix but different right-hand sides must be solved in each iteration of the algorithm. Therefore, since a large number of first-order problems has to be solved in this method, the efficiency of the employed linear solver is crucial to the performance of the algorithm as a whole.

4.2 The Choice of First-Order Accurate Operator

In order to construct a suitable first-order accurate operator corresponding to problem (6), we modify the underlying discretization (2). Namely, we simplify the dissipation term so that the resulting operator has only the immediate-neighbors fill-in. It corresponds to getting rid of the pressure switch and second order differences as well as discarding Martinelly coefficients in case they are used in the dissipation formula. If the discretization of a viscous flux relies on the Green-Gauss formula, in the first-order operator we switch to thin shear layer (TSL) discretization instead.

5 AMG for Adjoint Equations

As practical experience shows, applying AMG directly to linear systems based on second order discretizations within the context of CFD is not favorable. In the following we will describe two different strategies on how to combine the AMG methodology with the idea of defect correction.

5.1 AMG Components

In this section we will describe the components that form the AMG method used in the defect correction strategies described later on. In principal, every AMG method consists of a coarsening, smoothing and cycling strategy. As for the cycling, we use a simple V-cycle with a direct LU-solver for the coarsest grid.

Coarsening Strategy. Given the matrix for AMG corresponding to a linear system of equations with multiple physical unknowns, the choice of a coarsening strategy is not trivial. In our method we choose an aggregation type approach in combination with piecewise constant interpolation. Aggregates are selected to ensure a convection-determined alignment away from the body of an aircraft as well as to preserve the direction of aggregates in the boundary layer near viscous walls, as seen in Figure 1.

The choice of aggregates is identical for every physical unknown. The construction of the aggregates is based on a scalar matrix with a point level connectivity pattern. This matrix can be either the diagonal block of one 'primary'

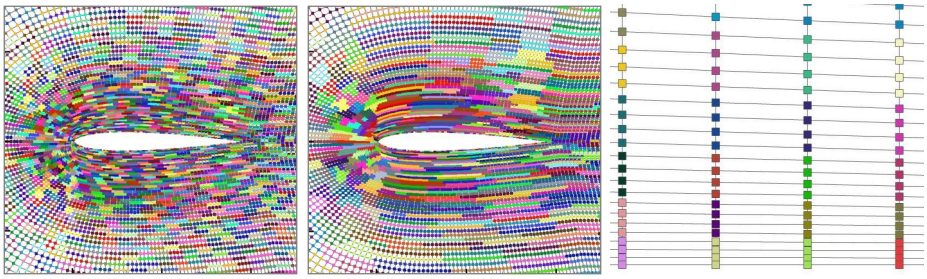

Fig. 1. 1st and 2nd-level aggregates on NACA0012 with predefined clustersize 4 for Euler computation (left and middle), 1st-level aggregates on flat plate directly above the horisontal viscous wall for RANS computation (right)

unknown (like density) or some norm using the information of the whole block. Nevertheless, piecewise constant interpolation has to be applied separately for every physical unknown afterwards.

Smoothing. For scalar elliptic problems AMG can usually use cheap variable-based smoothers as Gauss-Seidel or w-Jacobi. For our application we focus on a more robust smoothing algorithm in form of ILU(0). Prior to building the ILU decomposition, a reordering of variables according to the Reverse-Cuthill-McKee algorithm (RCM) has proven to be very efficient. Furthermore, two ILU-iterations are made in each pre- and post-smoothing step.

5.2 Outer Defect Correction (SAMG2)

The most straightforward way to integrate AMG into the defect correction process is to apply it as a solver to the defect equation within each defect correction step (7) (as described in Section 4). While the DC-residuals are computed based on the 2nd order linear problem, the system passed to AMG is only of first order accuracy. For increased stability of the overall approach, the defect correction loop itself is employed as a preconditioner within a Krylov method, namely GMRES(100). To ensure the linearity of the preconditioner we have to perform a fixed number of iterations within the AMG. For the presented results the number of AMG iterations is fixed to one. The right part of Figure 2 shows the structure of a single SAMG2 iteration.

The computational cost of one preconditioning step is dominated by the cost of the ILU within the AMG-Smoothing process. Given an AMG operator complexity of about 1.25 we can estimate the overall ILU cost to be about 5 times the cost of a standard ILU(0) iteration **on the 1st order matrix**. Transfer operators as well as residual evaluations within AMG steps cost less than 1 ILU(0) approximately. The residual evaluation of the 2nd order matrix within the defect correction does not need to be computed, since it is already given by the GMRES outer loop. While the cost for the GMRES is independent of the choice of the preconditioner we will ignore it.

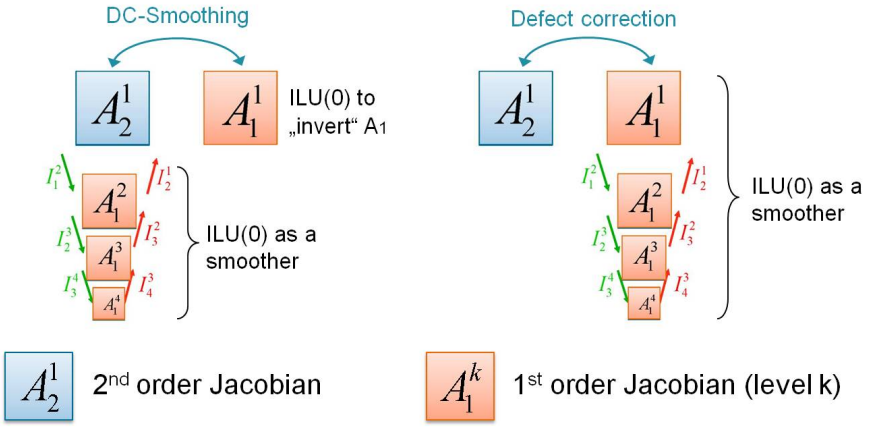

Fig. 2. Structure of one Inner-DC/SAMG1 (left) and Outer-DC/SAMG2 (right) iteration

Looking at Table 1 one can see that the memory requirements for the SAMG2 approach are even less that for a GMRES/ILU(0) combination. This is due to the fact that most data constructs are based upon the much smaller first order matrix rather than the large 2nd order operator. Overall the approximate memory consumption is about 2.2 times the amount needed for explicit storage of the main 2nd order system.

Table 1. Approximate Memory Requirements. 1 Unit corresponds to the explicit storage for the Jacobian 2nd order.

	ILU(0)	SAMG2	SAMG1
TAU PETSc Matrix (2nd order)	1	1	1
TAU GMRES(100)	0.5	0.5	0.5
SAMG matrix copy (2nd order)	-	-	1
SAMG matrix copy (1st order)	-	0.25	0.25
SAMG Hierarchy	-	0.1	0.1
ILU(0) first level	1	0.25	0.25
ILU(0) Hierarchy	-	0.1	0.1
	2.5	**2.2**	**3.2**

AMG vs. 1-level Solver. Since the development of an AMG method compared to an ILU(0) solver is quite time consuming, one could ask whether it might be sufficient to use a one-level method within the defect correction scheme. Although AMG usually has much better convergence rates, there is no indication if - and to what extend - this accuracy is transferable to the outer equation.

Therefore we will also compare SAMG1 and SAMG2 to a one-level solver within the outer defect correction algorithm. As we will see in our benchmarks later on, using the one-level solver the convergence rates and therefore the number of iterations needed to achieve the desired accuracy grow significantly. Overall the growth in iteration count is stronger than the reduction in run time per iteration, leading to a less efficient method. This statement holds for the full spectrum of test examples in Section 7. Even more drastically, some of them (e.g. MEGAFLUG, see Table 2) tend to diverge without the use of a hierarchical solver within the defect correction.

5.3 Inner Defect Correction (SAMG1)

Another possibility of combining AMG and defect correction for the solution of a 2nd order accurate linear system is to integrate DC within the smoothing process of the first AMG level. While AMG logically works directly on the second order matrix, most operations within are mapped to the first order problem. Also the creation of the coarser levels will be based on the first order matrix, giving the same coarse-grid correction operator as used for SAMG2. In a sense SAMG1 corresponds to an integration of DC within AMG rather than the other way around. As in the SAMG2 algorithm the whole iteration is accelerated via the Krylov method GMRES(100). The left part of Figure 2 shows the basic structure of a single SAMG1 iteration in comparison to SAMG2.

The core operations of one preconditioning step based on SAMG1 is almost the same as that for SAMG2. The only difference is the matrix used for residual evaluation on the finest grid. All five residual evaluations - four within the Richardson iteration of ILU(0) and one for the coarsegrid defect equation - will be based on the much larger matrix 2nd order instead of the 1st order matrix. This has a quite large effect on the run time, making one iteration of SAMG1 about twice as costly as one iteration of SAMG2.

The only difference in memory usage of the SAMG1 variant compared to SAMG2 is the explicit storage of the 2nd order Jacobian within the AMG library. Since residuals based on the 2nd order operator have to be evaluated within SAMG as well as within the GMRES, the data currently has to be stored in both environments, increasing the overall memory requirements from 2.2 up to 3.2 times the 2nd order operator.

6 Parallelization

In a parallel environment the intuitive way of processing large data sets is via *domain decomposition*. However, in the context of linear solvers this is usually not sufficient. Many components used for or within linear solution processes are inherently sequential (Gauss Seidel, ILU, ...) and can therefore only be applied for solving the domains separately from each other. Although these local solutions can be merged, e.g. via a global Jacobi smoothing or a Krylov method, the resulting approximation quality strongly suffers depending on the degree of parallelism.

The parallel AMG cycle itself is a fully parallel algorithm, because it only consists of sparse matrix vector multiplications. Given a fixed hierarchy and a parallel smoothing process (like w-Jacobi), AMG will produce exactly the same results for any given number of processors involved. Nevertheless, AMG's performance strongly depends on the setup process and the choice of the smoothing method.

Parallel Coarsening. As described in Section 5, we use an aggregation type strategy for coarsening. Since the used algorithm is inherently sequential, the coarsening is done independently for each subdomain, leading to different coarsenings for varying processor counts. However, the quality of the coarsenings is only affected near the boundaries of each domain. To reduce the resulting negative effect on the convergence rates, we reduce the aggregation size at the boundaries, effectively limiting the amount of *parallelism* of the coarsening. In other words, we might end up with a slightly reduced coarsening rate in order to keep stable convergence rates.

Parallel Smoothing. AMG smoothing within our solution approaches is based on ILU(0). Since ILU is not parallel at all, we can only apply it domain-wise as described at the start of this section. To merge the results given by each subdomain we use the additive Schwarz method (ASM), effectively overlapping the subdomains at their boundaries by a given depth. In our methods, we chose the depth 1.

Parallel vs. Numerical Scalability. AMG methods - if applied to proper problems - are known to be scalable. In terms of numerical methods this means that the computational effort to solve to a given accuracy grows linearly with the size of the problem. However, this understanding of scalability often conflicts with the definition of *parallel* scalability which is often used to measure the efficiency of a parallel program in terms of speedups gained for increased numbers of participating cores.

In theory the parallel scalability of the AMG based algorithms of section 5 should only be limited by amount of boundary points in comparison to the inner points of each domain. Huge amounts of halo data will increase communication cost within all sparse matrix vector operations as well as the local ILU compute load via the ASM overlap. The numerical effects introduced by parallelizing coarsening and smoothing can only be measured by looking at the convergence rates. In Figure 3 the corresponding results are shown for the test case VELA (see Table 2) running on the CASE cluster of the DLR. The (strong) parallel scalability in terms of run time reduction can be shown up to moderate number of processors, intuitively limited by the size of the model. In addition, numerical scalability seems to be ensured for even higher processor counts.

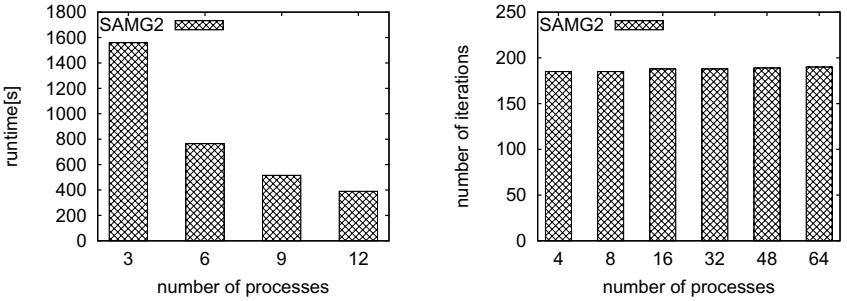

Fig. 3. Strong scaling of VELA test case (EULER), solver accuracy eps=$1d-10$. Parallel scaling of runtime in seconds (left), numerical scaling in terms of iteration count (right).

7 Results

7.1 Considered Configurations

In this section we present the results obtained from extensive testing of the solvers for various configurations. The configurations, computational grids, flow parameter settings as well as employed cost functions are summarized in Table 2. As one can see from the table, one Euler example was considered, while the rest of test cases covers the solution of RANS equations. Concerning computational grids, all but one of RANS test cases are discretized on hybrid grids, and only one (Dornier 728) is discretized on a structured grid, whereas the Euler equations are solved on an unstructured grid. A transonic flow regime is considered in all the test cases except for DLR F12 configuration, where a subsonic flow is simulated. As a cost function we used either lift or drag.

Table 2. Summary information on considered test cases. Abbreviations used in the table: U. - unstructured grid, H. - hybrid grid, S. - structured grid, M - million points.

Configuration	Equations	Grid type/size	Mach nr.	Reyn. nr.	Angle	Cost fn.
VELA	Euler	U. 1.1 M	0.85	-	1.8	drag
$LANN_1$	RANS	S. 1.1 M	0.82	$3.7 \cdot 10^6$	0.6	drag
$LANN_2$	RANS	H. 5.2 M	0.82	$5.4 \cdot 10^6$	0.59	drag
Dornier 728	RANS	S. 1.9 M	0.8	$20 \cdot 10^6$	0	lift
DLR F6	RANS	H. 5.8 M	0.75	$3 \cdot 10^6$	0.1	drag
DLR F12	RANS	H. 9.6 M	0.21	$1.3 \cdot 10^6$	0	lift
Megaflug	RANS	H. 5.8 M	0.9	$3.3 \cdot 10^6$	2.0	lift
DPW 4	RANS	H. 11.7 M	0.85	$5 \cdot 10^6$	2.3	drag

7.2 Convergence

The following results are all based on the TAU-configurations in Table 2 in combination with the algorithms SAMG1 and SAMG2 described in Section 5. Additionally a 1-level approach based on SAMG2 is shown for comparison. More precisely this approach only performs four steps of ILU(0) with RCM-reordering within each defect correction step. Results are shown for scalar dissipation (SD) and/or matrix dissipation (MD). Note that the matrix dissipation operator is modified as compared to the standard one used in TAU code. Namely, the pressure switch is computed differently, and moreover, the Martinelly coefficients are omitted. The latter has a positive effect on performance of defect correction. Note that this modified dissipation operator is not a standard dissipation operator used in TAU code and it is currently under evaluation.

As shown in Figure 4, test case LANN_1 can be solved by all three methods. SAMG1 needs only half of the iterations of SAMG2 to reach the desired accuracy. However, due to the increased run time per iteration as explained in Section 5.3 the absolute run time benefit is much smaller. For LANN_2 the convergence advantage of SAMG1 disappeared, making SAMG2 the fastest method. Note that in comparison to 1-level - due to the harsh coarsening - the computation for a single SAMG2 iteration is only slightly more expensive.

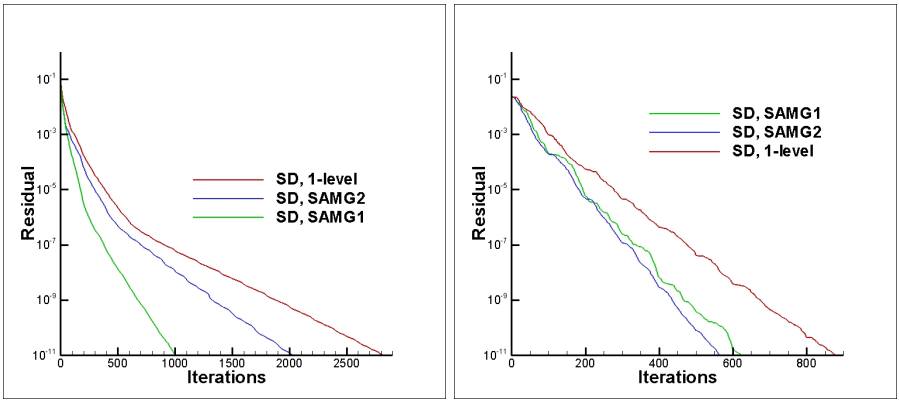

Fig. 4. Convergence of LANN_1(left) and LANN_2(right)

Figure 5 shows the convergence histories for two large models, DPW4 and MEGAFLUG. Results are shown for scalar dissipation (SD) and matrix dissipation (MD). Additionally, one test run for each model was computed with Green-Gauss (GG) viscous flux discretization instead of default TSL. With MD fewer defect correction steps are needed to reach the desired accuracy. However, the corresponding 1st order system becomes harder to solve. While SAMG2 can still handle the increased difficulty, the ILU(0) within the DC-Smoothing of SAMG1 is not sufficient anymore. Therefore the following results of MD based configurations did not converge with SAMG1 anymore. As for DPW4 the multilevel

approaches still have a large advantage in terms of convergence rates over their 1-level counterparts, looking at MEGAFLUG there is currently no alternative for a hierarchical method.

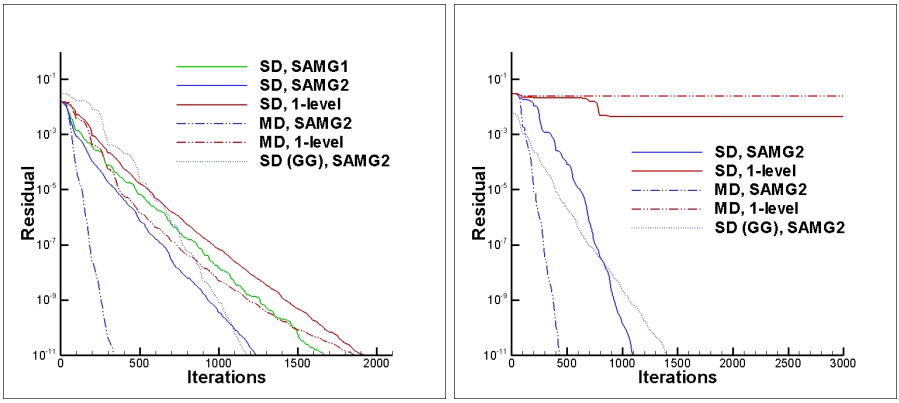

Fig. 5. Convergence of DPW4(left) and MEGAFLUG(right)

Table 3 summarizes the results for all major test cases that were considered within the project. Due to the best overall performance and stability absolute run times are only shown for the SAMG2 approach. All benchmarks were computed on the CASE-cluster at the DLR based on TAU revision 15915.

Table 3. Absolute run times of SAMG2 on CASE cluster (Accuracy $1e-10$)

				Abs. runtimes	
Configuration	#points	Jacobian	#procs	SD	MD
Vela	1.1 M	20 GB	8	10m	6m
$LANN_1$	1.1 M	11 GB	4	2h13m	8m
$LANN_2$	5.2 M	83 GB	20	50m	16m
Dornier 728	1.9 M	20 GB	8	3h2m	21m
DLR F6	5.8 M	110 GB	24	3h26m	3h48m
DLR F12	9.6 M	190 GB	12	4h53m	2h7m
Megaflug	5.8 M	104 GB	24	2h2m	52m
DPW 4	11.7 M	170 GB	44	2h37m	46m

8 Mesh Deformation

8.1 Problem Description

In this part of our work, we are concerned with solving large sparse linear systems generated by a mesh deformation tool. In this tool, displacements of mesh nodes

are governed by equations of linear elasticity. As input for this tool, a deformation of mesh surface is provided, which supplies Dirichlet boundary conditions for the elasticity problem. As output of this tool, from the original mesh and a deformed mesh surface, a deformed volumetric mesh is generated. Furthermore, in order to prevent collapsing of mesh cells during mesh deformation process, the magnitude of deformation is made locally dependent on sizes and shapes of mesh cells. This is achieved by introducing artificial element stiffness via an additional parameter, which controls stiffness of elements in the elasticity system. For the details of this particular elasticity model based on artificial element stiffness we refer to [9]. The discretization of the system is performed by classical continuous finite elements with linear shape functions on hybrid grids.

The resulting linear systems are often very stiff. The following factors increase this stiffness even further. Firstly, cells in the thin boundary layer regions have typically very high aspect ratios and introduce anisotropy into the linear system and therefore have a negative effect on its conditioning. Secondly, the element-based artificial stiffness induces further negative influence on the conditioning of the linear system.

8.2 AMG Method

In order to solve arising linear systems we employ algebraic multigrid. In particular, we suggest the unknown-based AMG approach. Unknown-based approach is a solution approach for coupled systems of PDEs, in which the coarsening process is done separately for the physical quantities.

This means that for each physical unknown its own coarse grid hierarchy via an individual coarsening process is built. The coarsening is performed via one-stage classical Ruge-Stüben algorithm, and it is done aggressively on the finest level. Interpolation is also done separately for each physical unknown, but the Galerkin matrices are computed with respect to the full set of unknowns. As a smoother we employ unknown-wise Gauss-Seidel, which causes that the resulting error is smooth separately for each physical unknown. GMRES method is applied as an accelerator in order to stabilize the solution approach.

8.3 Results

The test cases considered in this work are summarized in Table 4.

In case of three-element airfoil TC11, the deformation is induced only on the flap, which is depicted on the left side in Figure 6. For DLR F6 test case, the deformation is a 20 degrees rotation around y-axis as depicted on the right side of Figure 6 (pitching motion). For both structured and unstructured Dornier 728 test cases, prescribed deformation corresponds to a 10 degrees rotation around y-axis.

The test cases were converged until machine precision in order to demonstrate that no stagnation of convergence occurs at lower residual values. Convergence histories for all the test cases are gathered on the left and right side of Figure 7. For comparison, convergence of the corresponding one-level solution approach

Table 4. Summary information on mesh deformation test cases. Abbreviations used in the table: U. - unstructured grid, H. - hybrid grid, S. - structured grid, M - million points.

Configuration	2D / 3D	Grid type / size
TC11	2D	H. 1.1 M
DLR F6	3D	S. 1.1 M
Dornier 728	3D	S. 1.9 M
Dornier 728	3D	U. 5.2 M

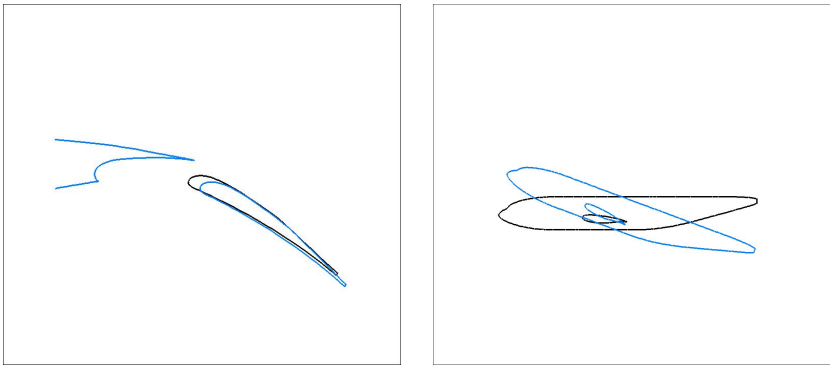

Fig. 6. Given surface deformation for TC11 (left) and DLR F6 (right) test cases

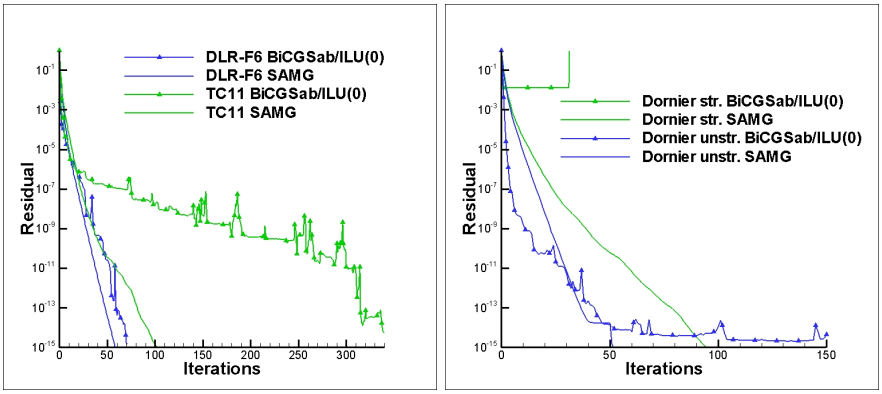

Fig. 7. Convergence of SAMG and of one-level solver for mesh deformation test cases

(unknown-wise Gauss-Seidel stabilized with BiCGStab) is presented in this figure as well. Note that each BiCGStab iteration includes two steps of Gauss-Seidel, while each step of GMRES includes only one step of AMG. As one can see, solver based on algebraic multigrid converges very fast, reaching 15 orders reduction in

less than 100 iterations for all presented test cases. Moreover, from the convergence plots one can see that convergence does not slow down much from iteration to iteration. In the same time, convergence of the considered one-level solver is not stable and very much test case dependent: while it shows fast convergence for the DLR F6 case, a divergence occurs in case of structured Dornier 728. One can also observe that convergence of the one-level solver in general deteriorates from iteration to iteration.

9 Conclusion

Within ComFliTe AMG has been successfully employed in the CFD as well as mesh deformation applications. A new method inspired by defect correction has been described for the solution of the adjoint equations. This method has been integrated and extensively tested within the DLR TAU code. The resulting method has been found to be very efficient. Without the need for complicated parameter optimization a single solver configuration managed to handle all considered test cases. Additionally, AMG has been integrated in the mesh deformation tool provided by the DLR. Based on classical strategies we observed a significant increase in performance compared to conventional one-level solvers. Both approaches were implemented and tested in parallel, therefore being capable of meeting the challenge of even larger, state-of-the-art models.

References

1. Spalart, P., Allmaras, S.: A one-equation turbulence model for aerodynamic flows. La recherche aérospatiale 1(1), 5–21 (1994)
2. Gerhold, T., Friedrich, O., Evans, J., Galle, M.: Calculation of complex three-dimensional configurations employing the dlr-tau-code. AIAA Paper 167 (1997)
3. Brandt, A., MacCormick, S., Ruge, J., Survey, N.G., of Scientific Research, A.F.O., Foundation, N.S.: Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations (1983)
4. Stüben, K.: Algebraic multigrid (amg): experiences and comparisons. Applied Mathematics and Computation 13(3-4), 419–451 (1983)
5. Brandt, A.: Algebraic multigrid theory: The symmetric case. Applied Mathematics and Computation 19(1), 23–56 (1986)
6. Ruge, J., Stüben, K.: Algebraic multigrid. Multigrid Methods 3, 73–130 (1987)
7. Böhmer, K., Hemker, P., Stetter, H.: The defect correction approach (1984)
8. Koren, B.: Multigrid and defect correction for the steady navier-stokes equations. Journal of Computational Physics 87(1), 25–46 (1990)
9. Stein, K., Tezduyar, T., Benney, R.: Mesh moving techniques for fluid-structure interactions with large displacements. Journal of Applied Mechanics 70, 58 (2003)