Anderson-Accelerated Convergence for Incompressible Navier-Stokes Equations

Auke van der Ploeg
MARIN, Wageningen/Netherlands, a.v.d.ploeg@marin.nl

1 Introduction

We consider iterative solvers for the systems of non-linear equations that need to be solved to compute incompressible flows. To ensure that the computed solution is not spoiled by grid dependence, the grids have to be sufficiently fine, which causes these systems to become large. For instationary flows such systems have to be solved every timestep, and to be sure that the instationary behavior is computed correctly, the time step cannot be chosen too large and at each time step the system of non-linear equations has to be solved sufficiently accurate. As a result, for many cases the computational effort is quite substantial. Therefore, in this paper we study the effectiveness of an acceleration strategy introduced in Anderson, 1965 to reduce the computational effort. In recent years, this strategy has been analyzed in the context of solution methods for fixed point problems, Fang and Saad, 2009. In the sequel of this report, this strategy will be referred to as Anderson Acceleration (AA).

The basic idea is that, if the problem to solve were linear, at each iteration in which AA is applied some of the history is used to optimize the next update of the approximate solution. Therefore, several vectors from previous iterations have to be stored, and frequently updated. This is very similar to the basic idea of the well-known minimal residual method GMRES, Saad and Schultz, 1986 to solve a non-symmetric system of linear equations.

In Pollock et al., 2018, Anderson-accelerated Picard iterations are analyzed for solving incompressible Navier-Stokes equations, and tested by computing the steady, laminar flow in a 2D and a 3D lid-driven cavity. In Pollock et al., 2018 it is shown that Anderson Acceleration can provide a significant, and sometimes dramatic, improvement in the convergence behavior, and it is even proven analytically that, as long as the underlying fixed-point problem satisfies some constraints, AA provides guaranteed improved convergence behavior.

2 DERIVATION AND ALGORITHMS

When solving the discretized, incompressible Navier-Stokes equations, the system of non-linear equations $\mathbf{r}(\mathbf{x}) = \mathbf{0}$ with $\mathbf{r} : \mathbb{R}^{\mathbf{n}} \rightarrow \mathbb{R}^{\mathbf{n}}$, has to be solved. Herein the vector-valued function $\mathbf{r}$ indicates the residual vector that contains the values of the residual for the transport equations. For the results presented in this paper, AA will be used to solve the coupled system of mass and momentum equations arising from a cell-centered discretization. In that case, in three space dimensions $\mathbf{r}$ contains four residual values for each cell center: three components from the momentum equation and the residual of the mass equation:

$$\mathbf{r} \;=\; \left(\mathbf{r_u}, \mathbf{r_v}, \mathbf{r_w}, \mathbf{r_p}\right)^T$$

Hence the vector length n is four times the number of cells. In general, $\mathbf{r}$ will also contain the residuals from other transport equations, like those coming from a turbulence model, or a transport equation for the volume fraction. For time-dependent flows $\mathbf{r}(\mathbf{x}) = \mathbf{0}$ has to be solved at each time step to such an accuracy that the iterative error does not affect the computed instationary behavior.

Suppose that the 'basic' solution technique without AA to solve $\mathbf{r}(\mathbf{x}) = \mathbf{0}$ is denoted by B. This can be a coupled solver as described in Klaij and Vuik, 2013, or a segregated method like 'SIMPLE' or 'SIMPLER'. The basic method constructs a sequence of estimates $\mathbf{x}_k$, $k = 1, 2, \ldots$ of the solution vector, such that $\mathbf{x}_{k+1} = \mathrm{B}(\mathbf{x}_k)$. If AA is applied at the k-th step of the basic solution technique, first an 'optimized' update $\widetilde{\mathbf{x}_k}$ is constructed. Next, $\mathbf{x}_{k+1} = \mathrm{B}(\widetilde{\mathbf{x}_k})$.

To be able to construct $\widetilde{\mathbf{x}_k}$, the residual vectors together with the solution vectors of some previous outer loops have to be stored in memory. Of course, we have to choose a maximum number of such vectors. This number will be denoted by m, and the resulting AA method to accelerate the computation will be denoted by AAm. For ease of presentation, we will first assume that $k > m$. Solving $\mathbf{r}(\mathbf{x}) = \mathbf{0}$ is

equivalent to solving the fixed point problem $\mathbf{x} = \mathbf{g}(\mathbf{x})$ in which the operator $\mathbf{g}$ is defined by

$$\mathbf{g}(\mathbf{y}) = \alpha\mathbf{r}(\mathbf{y}) + \mathbf{y} \qquad \text{for any given vector } \mathbf{y}$$

Herein α is a parameter to be chosen in advance such that $0 < \alpha \leq 1$. In the next subsection, the effect of this parameter will be discussed.

The AAm-algorithm updates the solution vector as

$$\widetilde{\mathbf{x}_k} \;=\; \sum_{i=0}^{m} \theta_i \mathbf{g}(\mathbf{x}_{k-i}) \qquad \text{satisfying the constraint} \qquad \sum_{i=0}^{m} \theta_i = 1. \tag{1}$$

Herein $\mathbf{x}_{k-1}$ indicates the solution vector from the previous outer loop, $\mathbf{x}_{k-2}$ the solution vector from the previous previous outer loop etc. The zero-sum constraint is necessary to guarantee that at a converged stage, at which $\mathbf{x} = \mathbf{x}_{k-i}$ for, $i = 0, .., m$, $\widetilde{\mathbf{x}_k}$ is indeed a solution to the fixed point problem $\mathbf{x} = \mathbf{g}(\mathbf{x})$. From this constraint it follows that $\theta_0 = 1 - \sum_{i=1}^{m} \theta_i$. To ensure that new information is incorporated into $\widetilde{\mathbf{x}_k}$, θ_0 should be positive. From Eq. (1) it follows that

$$\widetilde{\mathbf{x}_k} \;=\; (1 - \sum_{i=1}^{m} \theta_i)\mathbf{g}(\mathbf{x}_k) + \sum_{i=1}^{m} \theta_i \mathbf{g}(\mathbf{x}_{k-i}) \;=\; \mathbf{g}(\mathbf{x}_k) + \sum_{i=1}^{m} \theta_i \left[\mathbf{g}(\mathbf{x}_{k-i}) - \mathbf{g}(\mathbf{x}_k)\right] \tag{2}$$

The question is now how to determine the coefficients $\theta_1, ...\theta_m$. This is done in such a way that

$$\|\mathbf{r}(\widetilde{\mathbf{x}_k})\|_2 \qquad \text{is minimized, if } \mathbf{r} \text{ were linear.} \tag{3}$$

Therefore, we want to express $\mathbf{r}(\widetilde{\mathbf{x}_k})$ as a function of $\theta_1, ...\theta_m$. First, note that, if $\mathbf{r}$ were linear, the operators $\mathbf{r}$ and $\mathbf{g}$ commute:

$$\mathbf{r}(\mathbf{g}(y)) \;=\; \mathbf{r}(\alpha\mathbf{r}(y) + y) \;=\; \alpha\mathbf{r}(\mathbf{r}(y)) + \mathbf{r}(y) \;=\; \mathbf{g}(\mathbf{r}(y)) \qquad \text{for any given vector } \mathbf{y}.$$

From Eq. (2) it follows that $\mathbf{r}(\widetilde{\mathbf{x}_k}) = \mathbf{r}\left(\mathbf{g}(\mathbf{x}_k) + \sum_{i=1}^{m} \theta_i \left[\mathbf{g}(\mathbf{x}_{k-i}) - \mathbf{g}(\mathbf{x}_k)\right]\right)$. From this equation, and the fact that the operators $\mathbf{r}$ and $\mathbf{g}$ commute we obtain, if $\mathbf{r}$ were linear,

$$\mathbf{r}(\widetilde{\mathbf{x}_k}) \;=\; \mathbf{g}\left(\mathbf{r}(\mathbf{x}_k) + \sum_{i=1}^{m} \theta_i \left[\mathbf{r}(\mathbf{x}_{k-i}) - \mathbf{r}(\mathbf{x}_k)\right]\right) \;=\; \mathbf{g}\left(\mathbf{r}(\mathbf{x}_k) - \mathbf{R}\boldsymbol{\theta}\right) \tag{4}$$

in which the i-th column of the $n \times m$ matrix $\mathbf{R}$ consist of $\mathbf{r}(\mathbf{x}_k) - \mathbf{r}(\mathbf{x}_{k-i})$ and the vector $\boldsymbol{\theta} = (\theta_1, ..., \theta_m)^T$. If $\alpha = 0$, $\mathbf{g}$ is the identity operator and in that case Eq. (4) holds as well. From Eq. (3) and Eq. (4) it follows that $\|\mathbf{r}(\mathbf{x}_k) - \mathbf{R}\boldsymbol{\theta}\|_2$ must be minimized, independent of the value of α.

The minimization problem can be solved by first making a QR-decomposition of $\mathbf{R}$: an orthogonal $n \times n$ matrix $\mathbf{Q}$, a permutation matrix $\mathbf{P}$ and an $m_k \times m_k$ upper-triangular matrix $\mathbf{U}$ are made such that the matrix product $\mathbf{RP} = \mathbf{Q}(\mathbf{U}\ \ \mathbf{0})^T$. This is equivalent to $\mathbf{Q}^T\mathbf{RP} \;=\; (\mathbf{U}\ \ \mathbf{0})^T$. The equivalence follows from the fact that since $\mathbf{Q}$ is orthogonal it obeys $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$. If $\mathbf{R}$ has full rank m_k the matrix $\mathbf{U}$ also has full rank and, therefore, is nonsingular. Applying $\mathbf{Q}^T$ to a vector does not change the length of that vector (think of it as a rotation). In addition, the permutation matrix satisfies $\mathbf{PP}^T = \mathbf{I}$. Therefore, if we denote the residual vector $\mathbf{r}(\mathbf{x}_k)$ simply as $\mathbf{r}$,

$$\min_{\boldsymbol{\theta}} \|\mathbf{r} - \mathbf{R}\boldsymbol{\theta}\|_2 \;=\; \min_{\boldsymbol{\theta}} \left\|\mathbf{Q}^T(\mathbf{r} - \mathbf{R}\boldsymbol{\theta})\right\|_2 \;=\; \min_{\boldsymbol{\theta}} \left\|\mathbf{Q}^T\mathbf{r} - (\mathbf{Q}^T\mathbf{RP})\mathbf{P}^T\boldsymbol{\theta}\right\|_2 \;=\; \min_{\boldsymbol{\theta}} \left\|\mathbf{Q}^T\mathbf{r} - (\mathbf{U}\ \ \mathbf{0})^T\mathbf{P}^T\boldsymbol{\theta}\right\|_2$$

If we put $\mathbf{y} = \mathbf{P}^T\boldsymbol{\theta}$ and partition $\mathbf{Q}^T\mathbf{r}$ as $(\mathbf{r_1}, \mathbf{r_2})^T$ in which $\mathbf{r_1}$ has m_k components, this can be written as

$$\min_{\boldsymbol{\theta}} \|\mathbf{r} - \mathbf{R}\boldsymbol{\theta}\|_2 \;=\; \min_{\mathbf{y}} \left\| \begin{pmatrix} \mathbf{r_1} \\ \mathbf{r_2} \end{pmatrix} - \begin{pmatrix} \mathbf{U} \\ \mathbf{0} \end{pmatrix} \mathbf{y} \right\|_2 \;=\; \min_{\mathbf{y}} \sqrt{\|\mathbf{r_1} - \mathbf{U}\mathbf{y}\|_2^2 + \|\mathbf{r_2}\|_2^2}$$

The solution can be obtained from $\mathbf{Uy} = \mathbf{r_1}$ by back substitution. In our current implementation, the QR-factorization is not yet implemented in parallel: the matrix $\mathbf{R}$ is gathered on the master processor

and on that processor the factorization is performed. Therefore, the timings reported in the next section contain some overhead, which can be reduced in future implementations.

Both the residual vectors and solution vectors of m previous outer loops should be stored in memory. In our implementation, these vectors are shifted: the current residual vector and solution vector are added and, if the resulting number of residual vectors exceeds m, the 'oldest' vectors are deleted.

The AA acceleration can be applied at every step of the iteration, but to reduce the overhead, it is also possible to apply it only every second, third or fourth iteration. If the frequency of application of AA is controlled by the parameter $freq$, the basic iterative method B accelerated by Anderson Acceleration is given by Algorithm 1.

Algorithm 1: **Basic method B accelerated by Anderson Acceleration**:
Given $\mathbf{x}_0$ and $m \geqslant 1$: Set $\mathbf{x}_1 = \mathrm{B}(\mathbf{x}_0)$;
For $k = 1, 2, ...$
IF mod$(k, freq) == 0$ THEN
Set $m_k = \min(m, k)$;
Determine $\boldsymbol{\theta} = (\theta_1, ..., \theta_{m_k})^T$ in such a way that $\|\mathbf{r}(\mathbf{x}_k) - \mathbf{R}\boldsymbol{\theta}\|_2$ is minimized;
IF $\sum_{i=1}^{m_k} \theta_i < 1.0$ THEN
Set $\widetilde{\mathbf{x}_k} = \mathbf{g}(\mathbf{x}_k) + \sum_{i=1}^{m_k} \theta_i \left[\mathbf{g}(\mathbf{x}_{k-i}) - \mathbf{g}(\mathbf{x}_k)\right]$; Set $\mathbf{x}_{k+1} = \mathrm{B}(\widetilde{\mathbf{x}_k})$
ELSE Set $\mathbf{x}_{k+1} = \mathrm{B}(\mathbf{x}_k)$
ELSE Set $\mathbf{x}_{k+1} = \mathrm{B}(\mathbf{x}_k)$

2.1 Effect of the parameter α

As already mentioned in the introduction, in Pollock et al., 2018 it is proven analytically that, as long as the underlying fixed-point problem satisfies some constraints, AA provides guaranteed improved convergence behavior. The main constraint is that the fixed-point operator should be a contraction operator. To be precise, in Pollock et al., 2018 it is proven that if the spectral radius of the Jacobian of the operator $\mathbf{g}$ is below 1, the fixed point iteration $\mathbf{x_{k+1}}$=$\mathbf{g}(\mathbf{x_k})$ is locally convergent.

From $\mathbf{g}(\mathbf{y}) = \alpha\mathbf{r}(\mathbf{y}) + \mathbf{y}$, for any $\mathbf{y}$ it follows that if λ is an eigenvalue of the Jacobian of the operator $\mathbf{r}$, then $1 + \alpha\lambda$ is an eigenvalue of the Jacobian of the operator $\mathbf{g}$. Therefore, the choice of α directly influences the spectral properties of the Jacobian of the fixed-point operator. If all possible values for λ would be real and negative, choosing a positive, small enough value for α would guarantee the fixed-point operator to be a contraction operator. For real-life applications it is hard to guarantee such conditions.

If $\alpha = 0$, the operator $\mathbf{g}$ is simply the identity operator and solving the equation $\mathbf{r}(\mathbf{x})$ is no longer equivalent to solving the fixed point problem $\mathbf{x} = \mathbf{g}(\mathbf{x})$. In that case, if AA is applied at the k-th step of the iteration $\widetilde{\mathbf{x}_k}$ is constructed as

$$\widetilde{\mathbf{x}_k} \;=\; \sum_{i=0}^{m} \theta_i \mathbf{x}_{k-i} \;=\; \mathbf{x}_k + \sum_{i=1}^{m} \theta_i \left[\mathbf{x}_{k-i} - \mathbf{x}_k\right] \qquad \text{again satisfying the constraint} \qquad \sum_{i=0}^{m} \theta_i = 1.$$

So the difference vector $\widetilde{\mathbf{x}_k} - \mathbf{x}_k$ is a combination of m previous difference vectors. Again the coefficients θ_i are determined in such a way that $\|\mathbf{r}(\widetilde{\mathbf{x}_k})\|_2$ is minimized, if $\mathbf{r}$ were linear. Again, from this condition it follows that $\|\mathbf{r}(\mathbf{x}_k) - \mathbf{R}\boldsymbol{\theta}\|_2$ must be minimized. The algorithm is as listed in Algorithm 1, with $\mathbf{g}$ simply replaced by the identity operator.

3 RESULTS

The results described in this section were obtained with the RANS code REFRESCO (www.refresco.org), a community based CFD code for the maritime world. It solves multiphase (unsteady) incompressible viscous flows using the Navier-Stokes equations, complemented with turbulence models, cavitation models and volume-fraction transport equations for different phases. The equations are discretised using a finite-volume approach with cell-centered collocated variables, in strong-conservation form. In REFRESCO, segregated methods like 'SIMPLE' or 'SIMPLER', or a coupled solver as described in

Klaij and Vuik, 2013 can be chosen as 'basic' solution methods. We will first show the results obtained for some relatively easy 2D test cases from the so-called verification Suite of REFRESCO. Next, we will show results for a more difficult 3D test case with a turbulence model.

The verification Suite consists of a set of test cases in which for each case a range of grids of different density is used to establish the order of grid convergence using the method of manufactured solutions, Eça et al., 2012. The exact solution, the 'manufactured solution' and the error are known because the right-hand side is determined by simply substituting this solution in the momentum equations and the continuity equation. For example, in case of a Taylor-vortex, the pressure $p = -\frac{1}{4}(\cos(2\pi x) + \cos(2\pi y))$ and the velocity components are $u = -\cos(\pi x)\sin(\pi y)$ and $v = \sin(\pi x)\cos(\pi y)$. In the 2D-Poiseuille test cases, the pressure $p = 8\mu(2 - x) + 1$ in which $\mu = 0.1$ and the velocity components are $u = 4y(1 - y)$ and $v = 0$. Different types of grids are used, as shown in Fig. 1. For illustration of the type of grid, this figure shows only the coarsest grids. In the numerical experiments descibed below we use strongly refined grids.

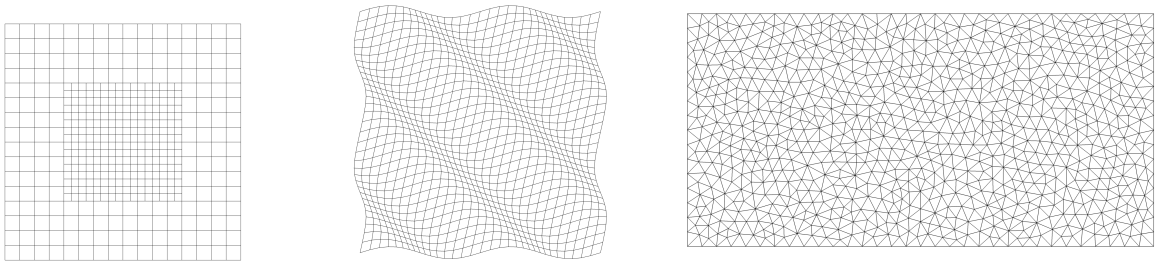

Fig. 1: Examples of grids used in the verification Suite. Left: hgrid. Middle: sgrid. Right: Delauney grid.

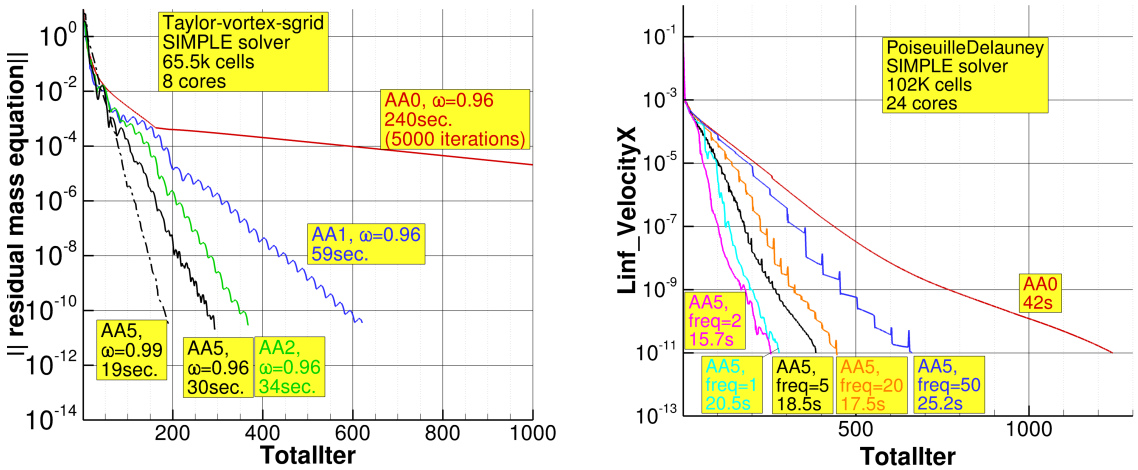

Fig. 2: Left: Effect of AA for two cases from the verification Suite. $\alpha = 1$ in both cases.

The left picture in Fig. 2 shows the results obtained on the s-grid when the manufactured solution is a Taylor-vortex. The text in the box has the same color as the corresponding line, and shows the dimension of the search space m in AAm together with the required wall clock times, using 8 cores and $freq = 1$. Relaxation parameters were determined in such a way that the solution method without AA, indicated by AA0, gives the smallest wall clock time. Anderson Acceleration gives a significant improvement of the convergence behavior and a considerable reduction in computational effort. The parameter m does not have to be chosen large, so the overhead in extra memory usage and cpu time is relatively small. The amount of implicit underrelaxation for the velocity is indicated by ω. For values of ω larger than 0.96, the computation without AA diverges. Only in combination with AA it is possible to choose $\omega = 0.99$, giving superior convergence behavior (dashed black line) and the smallest required wall clock time.

The right picture in Fig. 2 shows results obtained for a Poiseuille flow computed on a Delauney grid. Also on this type of grid Anderson Accelaration can significantly improve the convergence behavior. The wall clock times can be reduced with more the a factor of 2.5 and the choice of $freq$ is not very critical for the required wall clock time.

For the cases from the verification Suite, we obtain similar results with $\alpha = 0$ as with $\alpha = 1$. We illustrate this in Fig. 3 with the case that computes the Poiseuille flow on the hgrid. Again we see that AA strongly improves the convergence behavior and significantly reduces the required wall clock time. For this case, a five-dimensional search space seems to be a good choice. Increasing the dimension of the search space does not further improve the convergence behavior, and the required wall clock time increases due to the extra overhead.

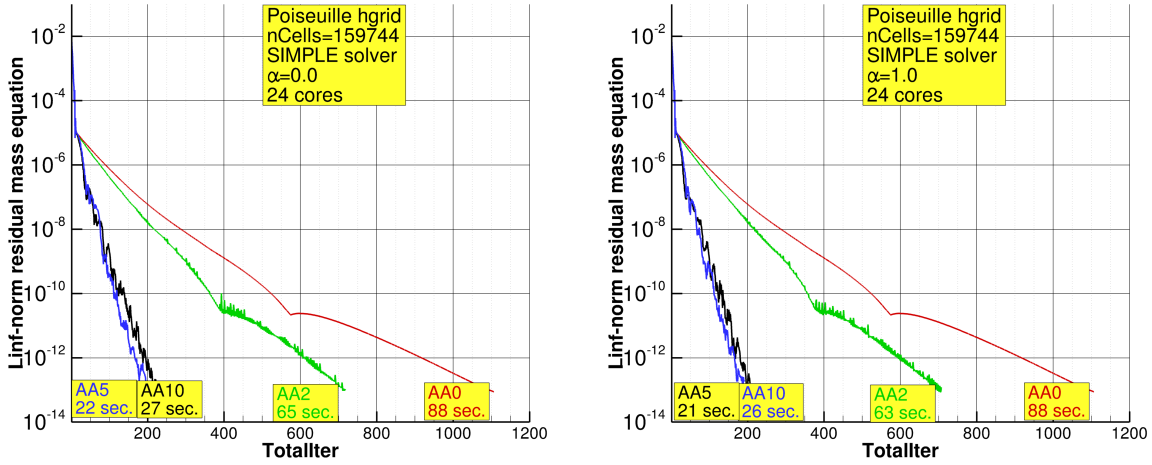

Fig. 3: Effect of α on the convergence behavior for the Poiseuille-hgrid case. Left: $\alpha = 0$. Right: $\alpha = 1$.

To illustrate the effect of AA for a 3D case with turbulence, we compute the flow around the KVLCC2 tanker with undisturbed watersurface at model-scale Reynoldsnumber 4.6E6, including the boundary layer and k-omega turbulence model. We use two grids, a structured mesh generated by Gridpro containing 0.26M cells, and an unstructured mesh with many hanging nodes generated by Hexpress containing 0.18M cells. For this application, use of Anderson Acceleration leads to divergence for all positive values of α. The effect of Anderson Acceleration on the convergence behavior with $\alpha = 0$ is shown in Fig. 4. Again the red lines show the convergence behavior without AA. With $\alpha = 0$, we obtain a reduction in wall clock time, although for this case it is not as strong as shown in the previous section for the easier, 2D test cases without turbulence. A reduction of about 40% can be achieved for the mesh generated by Gridpro, and about 30% for the one generated by Hexpress.

The green and orange lines in the right picture show that AA causes a very irregular convergence behavior at the early stage of the iteration and a severe increase of the maximum norm of the residual of the mass equation. Therefore, the blue line shows the result obtained when AA is not applied at the first 200 steps of the iteration. In that case, the use of AA gives an irregular convergence behavior between iteration number 200 and 400. About the same amount of wall clock time is required as with AA applied from the start. The reason for the irregular convergence behavior is still unknown. A possible reason is that AA is used only for the coupling between the momentum equations and the mass equation, and not for the transport equations of the turbulence model.

4 Conclusions

Anderson Acceleration (AA) can significantly speed up the solution of the systems of non-linear equations as they occur in the computation of incompressible flows. Especially for 2D problems without turbulence, the improvement in convergence rate can be spectacular and a reduction in the required wall clock time of more than one order of magnitude can be obtained. For some applications AA allows to

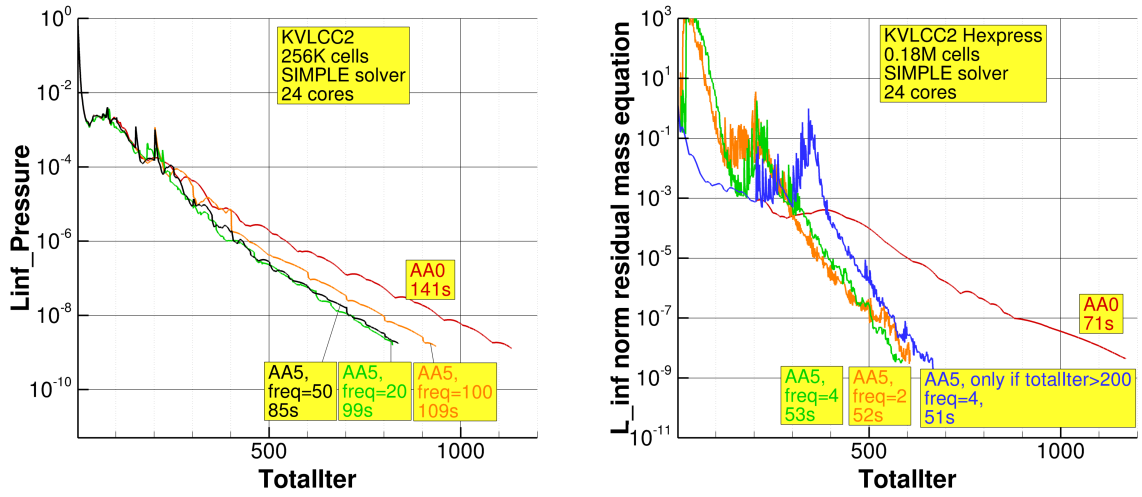

Fig. 4: Effect of AA on the computation of the flow around the KVLCC2 tanker. Left: Gridpro grid. Right: Hexpress grid. $\alpha = 0$ in both cases.

use less implicit underrelaxation, which results in a significant further reduction of the wall clock time.

For more realistic 3D test cases with turbulence, AA also can improve the convergence but the reduction of the wall clock time is not as spectacular. Typically, a reduction of 30% can be achieved when a relatively coarse mesh and up to 24 cores are used. This may be caused by the fact that in the current implementation in REFRESCO, the transport equations of the turbulence model are not yet included in the set of equations that is accelerated by the AA algorithm.

In the current implementation in the CFD-code REFRESCO, the solution of the minimization problem required by the AA algorithm has not yet been parallelized. Parallelization of this step will strongly reduce the overhead of AA. It will become mandatory for very large applications in which the required matrices will not fit in the memory of only one node.

References

Anderson, D. (1965). Iterative Procedures for Nonlinear Integral Equations. *Journal of the Association for Computing Machinery*, 12(4):547–560.

Eça, L., Hoekstra, M., and Vaz, G. (2012). On the use of Method of Manufactured Solutions for Code Verification of RANS solvers based on Eddy-viscosity Models. *ASME V&V Conference, Las Vegas, USA*.

Fang, H. and Saad, Y. (2009). Two classes of multisecant methods for nonlinear acceleration. *Numer. Linear Algebra Appl.*, 16:197–221.

Klaij, C. M. and Vuik, C. (2013). SIMPLE-type preconditioners for cell-centered, colocated finite volume discretization of incompressible Reynolds-averaged Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 71(7):830–849.

Pollock, S., Rebholz, L., and Xiao, M. (2018). Anderson-accelerated convergence of Picard iterations for incompressible Navier-Stokes equations.

Saad, Y. and Schultz, M. (1986). A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 7:856–869.

Walker, H. (2011). Anderson Acceleration: Algorithms and Implementations. Technical Report MS-6-15-50, Worcester Polytechnic Institute Mathematical Sciences Department.