11th World Congress on Computational Mechanics (WCCM XI)
5th European Conference on Computational Mechanics (ECCM V)
6th European Conference on Computational Fluid Dynamics (ECFD VI)
E. Oñate, J. Oliver and A. Huerta (Eds)

ON THE USAGE OF FINITE DIFFERENCES FOR THE DEVELOPMENT OF DISCRETE LINEARISED AND ADJOINT CFD SOLVERS

ANNA ENGELS-PUTZKA, JAN BACKHAUS, CHRISTIAN FREY

German Aerospace Center (DLR), Institute of Propulsion Technology
Linder Höhe, 51147 Cologne, Germany
e-mail: Anna.Engels-Putzka@dlr.de, Jan.Backhaus@dlr.de, Christian.Frey@dlr.de

Key words: Time-linearised Methods, Adjoint Methods, Finite Differences, Algorithmic Differentiation

Abstract. In this paper we discuss the usage of finite differences for the computation of the flux Jacobian in the framework of a discrete adjoint or time-linearised flow solver, in particular the associated choice of an appropriate step size. For comparison, we apply algorithmic differentiation to obtain an exact flux Jacobian. It turns out that the results depend strongly on the choice of the slope limiter. A careful choice of this function is crucial for computations with exact flux linearisations as well as for finite difference approximations.

1 INTRODUCTION

For many applications of computational fluid dynamics, adjoint and linear methods nowadays play an important role. The adjoint method [1, 2] is for example used to efficiently compute sensitivities for large numbers of free parameters, e.g. for optimisation problems (see e.g. [3]). Time-linearised methods can be used as an efficient approximation to unsteady simulations for applications e.g. in aeroelasticity and aeroacoustics.

There are two basic approaches for the development of adjoint methods. Either, adjoint differential equations can be derived and then discretised and solved ("continuous adjoint"), or the adjoint equations can be derived from the discretised flow equations ("discrete adjoint"). A discussion of both approaches can be found for example in [4]. In this work we focus on discrete adjoint methods. One advantage of these is that it is easier to obtain an adjoint solver which is consistent with the underlying nonlinear solver, which is an important issue for the accuracy of the adjoint sensitivities.

A central ingredient for the development of discrete adjoint flow solvers is the computation of the flux Jacobian, i.e. the derivatives of the numerical flux functions with respect to the flow variables, which can be obtained in different ways. An overview and discussion of various differentiation methods is given in [5]. In principle it is possible to differentiate the flux functions analytically “by hand”, but this is a very challenging task [6]. Alternatively, finite differences can be used to obtain the derivatives in a “black-box” manner. The main problem with this approach is the choice of an appropriate step size for the finite differences. It should be small to minimise the approximation error but it must not be too small in order to avoid cancellation errors. Another possibility is to use of algorithmic (sometimes also called automatic) differentiation (AD) [7]. This has been done e.g. by Mader et al. [8, 9] and Courty et al. [10].

Within the flow solver TRACE, adjoint [11, 12] and time-linearised [13, 14] methods have been developed since 2006 using the discrete approach. Since an analytical differentiation “by hand” would be imprатical for an industrially used, complex flow solver, which is constantly developed, finite differences are used for the computation of the flux Jacobian. Additionally we have now, based on the work described in [15], implemented a version which makes use of algorithmic differentiation.

We discuss both variants in section 2.2. Before, in section 2.1 we summarise important properties of the underlying nonlinear solver. In particular we discuss the implementation and properties of various limiter functions. In section 3 we present numerical examples where we apply these limiter functions in combination with different linearisation methods, i.e. algorithmic differentiation and finite differences with different step sizes. Finally, in section 4 we summarise our results and draw some conclusions.

2 THEORY

The flow solver TRACE has been constantly developed for more than twenty years and offers many different solver modes and features for the simulation of turbomachinery flows [16, 17]. We discuss here briefly those features which are relevant for the current study.

2.1 Nonlinear flow solver

TRACE is a hybrid-grid cell-centered finite volume solver. In the steady nonlinear mode, the Reynolds-averaged Navier–Stokes equations are solved using a time-marching algorithm. Time integration and spatial discretisation are separated. Since the adjoint solver is so far only available on structured grids, we concentrate on the spatial discretisation for structured grids.

The convective fluxes at the cell interfaces are approximated using the Roe scheme [18]. The so-called Roe-averaged variables are computed from left and right states $q^L_{i+\frac{1}{2}}, q^R_{i+\frac{1}{2}}$. Using MUSCL extrapolation [19], these are given by

$$q^L_{i+\frac{1}{2}} = q_i \quad + \tfrac{1}{4}\left[(1-\kappa)(q_i - q_{i-1}) + (1+\kappa)(q_{i+1} - q_i)\right] \quad =: q_i \quad + \tfrac{1}{2}\Delta q^L \tag{1}$$

$$q^R_{i+\frac{1}{2}} = q_{i+1} - \tfrac{1}{4}\left[(1+\kappa)(q_{i+1} - q_i) + (1-\kappa)(q_{i+2} - q_{i+1})\right] =: q_{i+1} - \tfrac{1}{2}\Delta q^R \tag{2}$$

respectively, compare e.g. [20, 21].

To avoid oscillations in the numerical solutions, the corrections $\Delta q^L, \Delta q^R$ are modified by multiplying the state differences by appropriate slope limiters:

$$\begin{aligned}
\tilde{q}^L_{i+\frac{1}{2}} &= q_i \quad + \tfrac{1}{2}\widetilde{\Delta q}^L \\
&= q_i \quad + \tfrac{1}{4}\left[(1-\kappa)\varphi\left(r^+_{i-\frac{1}{2}}\right)(q_i - q_{i-1}) + (1+\kappa)\varphi\left(r^-_{i+\frac{1}{2}}\right)(q_{i+1} - q_i)\right] \qquad (3)\\
\tilde{q}^R_{i+\frac{1}{2}} &= q_{i+1} - \tfrac{1}{2}\widetilde{\Delta q}^R \\
&= q_{i+1} - \tfrac{1}{4}\left[(1+\kappa)\varphi\left(r^+_{i+\frac{1}{2}}\right)(q_{i+1} - q_i) + (1-\kappa)\varphi\left(r^-_{i+\frac{3}{2}}\right)(q_{i+2} - q_{i+1})\right], \qquad (4)
\end{aligned}$$

where we use the following ratios of differences:

$$r^+_{i+\frac{1}{2}} = \frac{q_{i+2} - q_{i+1}}{q_{i+1} - q_i}, \quad r^-_{i+\frac{1}{2}} = \frac{q_i - q_{i-1}}{q_{i+1} - q_i}, \quad r^+_{i-\frac{1}{2}} = \frac{q_{i+1} - q_i}{q_i - q_{i-1}}, \quad r^-_{i+\frac{3}{2}} = \frac{q_{i+1} - q_i}{q_{i+2} - q_{i+1}}. \qquad (5)$$

Considering for the moment only the left state, we can rearrange

$$\widetilde{\Delta q}^L = \tfrac{1}{2}\left[(1-\kappa)\varphi\left(r^+_{i-\frac{1}{2}}\right) + (1+\kappa)\varphi\left(\frac{1}{r^+_{i-\frac{1}{2}}}\right) r^+_{i-\frac{1}{2}}\right](q_i - q_{i-1}). \qquad (6)$$

If the limiter function φ has the symmetry property

$$\varphi\left(\frac{1}{r}\right) = \frac{\varphi(r)}{r} \qquad (7)$$

equation (6) simplifies to $\widetilde{\Delta q}^L = \varphi\left(r^+_{i-\frac{1}{2}}\right)(q_i - q_{i-1})$, i.e. the correction is independent of κ and we have

$$\tilde{q}^L_{i+\frac{1}{2}} = q_i + \tfrac{1}{2}\varphi\left(r^+_{i-\frac{1}{2}}\right)(q_i - q_{i-1}). \qquad (8)$$

Analogous transformations can be applied for $q^R_{i+\frac{1}{2}}$.

We consider the following choices for the limiter function φ which fulfill the symmetry condition (7):

Minmod $\varphi(r) = \mathrm{minmod}(1, r) = \begin{cases} 0 & r < 0 \\ r & 0 \le r \le 1 \\ 1 & r > 1 \end{cases}$

van Albada $\varphi(r) = \begin{cases} 0 & r < 0 \\ \dfrac{r^2 + r}{r^2 + 1} & r \ge 0 \end{cases}$

van Leer $\varphi(r) = \dfrac{r + |r|}{1 + |r|} = \begin{cases} 0 & r < 0 \\ \frac{2r}{1+r} & r \ge 0 \end{cases}$

All these limiters also have the so-called TVD (total variation diminishing) property [21].

Another possible symmetry property for limiter functions is

$$\varphi\left(\frac{1}{r}\right) = \varphi(r) \tag{9}$$

(which applies in particular to constant functions, i.e. if no limiter is used). Then (6) can be written as

$$\begin{aligned} \widetilde{\Delta q}^{L} &= \tfrac{1}{2}\left[(1-\kappa) + (1+\kappa) r^{+}_{i-\frac{1}{2}}\right] \varphi\left(r^{+}_{i-\frac{1}{2}}\right)(q_i - q_{i-1}) & (10)\\ &= \tfrac{1}{2}\left[(1-\kappa)(q_i - q_{i-1}) + (1+\kappa)(q_{i+1} - q_i)\right] \varphi\left(r^{+}_{i-\frac{1}{2}}\right). & (11) \end{aligned}$$

One possible function with this property is

$$\varphi(r) = \frac{2r}{r^2+1}, \tag{12}$$

cf. for example [20]. This function does not behave well if both the numerator and the denominator of r approach zero. This can be changed if the limiter function is reformulated in terms of these differences $\delta q_+, \delta q_-$ and an additional constant is added to numerator and denominator of the resulting function:

$$\varphi(\delta q_+, \delta q_-) = \frac{2\delta q_+ \delta q_- + \varepsilon_{\mathrm{vA}}}{(\delta q_+)^2 + (\delta q_-)^2 + \varepsilon_{\mathrm{vA}}}. \tag{13}$$

Such a limiter formulation is used for example by Anderson et al. [22] and Benetschik and Gallus [23]. We use in this study a further modification by applying (13) to $(\delta q_+)^2, (\delta q_-)^2$. We refer to this as a "van Albada type" limiter. Note that this limiter function does not vanish if δq_+ and δq_- have different signs, so in particular it does not have the TVD property. Enforcing the TVD property would prevent the differentiability near zero [23].

All other limiters are implemented as stated above, except that the denominator of the ratio r is prevented from becoming exactly zero by adding an offset of 10^{-8} (with appropriate sign). Equations (3) and (4) are implemented componentwise for primitive variables in absolute frame of reference.

2.2 Discrete linearised and adjoint solver

We consider the linearisation of the discrete equation $R(q, x) = 0$, where q denotes as before the flow state (solution) and x represents the computational grid. The residual in cell i is given by

$$R_i = V_i^{-1} \sum_{\sigma \in \partial i} F_\sigma - S_i, \tag{14}$$

where V_i denotes the cell volume, σ runs over the faces of cell i and F_σ represents the numerical flux through the face σ, which depends of course on the employed discretisation scheme. S_i stands here for the rotational source terms which appear since the flow equations are formulated in the relative frame of reference.

The Jacobian $\frac{\partial R}{\partial q}$ therefore consists of two contributions, the flux Jacobian and the source term Jacobian. While the latter can be relatively easily derived and implemented for rotational source terms, the former would be very difficult to treat explicitly. Therefore, in the current implementation the derivatives of the fluxes are approximated by finite differences. Different levels of accuracy can be chosen for the finite differences, we discuss here only the (default) case of second order central differences, i.e. we have for each flow component q^j:

$$\frac{\partial F_\sigma}{\partial q_i^j} \approx \frac{F(q + \delta q_i^j) - F(q - \delta q_i^j)}{2\delta q_i^j}. \tag{15}$$

The perturbation amplitude δq_i^j is determined as

$$\delta q_i^j = (|q_i^j| + \mu)\varepsilon \tag{16}$$

where μ and ε are parameters which have to be chosen appropriately. The perturbation is basically proportional to the magnitude of the respective flow component with the step size ε as proportionality factor, while the threshold μ prevents the perturbation from becoming too small for very small flow components.

As a benchmark to assess the accuracy of the finite differences we apply algorithmic differentiation to obtain an exact version of the flux Jacobian. For this we use the AD tool ADOL-C [24], which employs the operator overloading technique, in forward mode and simply replace the finite differences in the computation of the flux Jacobian by algorithmically differentiated routines. For technical details on the usage of AD in TRACE we refer to [15]. The rest of the solver remains unchanged, in particular we use the same solution algorithm – a preconditioned GMRes algorithm with restarts – for the resulting linear equation system as in the finite difference case.

3 APPLICATION

3.1 Numerical test case: wave propagation

The linear solver is used to simulate the propagation of an acoustic wave through an annular duct. In this case the steady state at which the flow equations are linearised is just a constant flow state. The frequency of the wave can be varied to simulate different grid resolutions (in relation to the wave length). We use four different frequencies, namely 1250 Hz, 2500 Hz, 5000 Hz, and 10000 Hz. An example of the resulting solutions is shown in Fig. 1.

Ideally, the wave should pass the duct unchanged, but in practice the numerical solution introduces a certain amount of dissipation, i.e. the amplitude of the waves decreases towards the end of the duct. This numerical dissipation rate is given by

$$D = 1 - \left(\frac{|\Delta p_{\text{out}}|}{|\Delta p_{\max}|} \right)^{\frac{\lambda}{l}}, \tag{17}$$

where λ is the wave length and l the length of the duct.

We vary the step size for the finite differences between 10^{-1} and 10^{-4}, which is still large enough to avoid cancellation effects. To keep the number of computations manageable, we do not vary the size of the threshold μ independently, but set always $\mu = \varepsilon$. The results are compared to those obtained using algorithmic differentiation. Moreover, different limiter setups are used. For comparison, we also carried out a simulation with first order spatial accuracy. In this case the differentiation method does not influence the results.

Exemplary results are shown in Figures 2 and 3. In Fig. 2 we compare the various differentiation methods for selected limiters, while in Fig. 3 we keep the differentiation fixed and vary the limiter.

We see that if no limiter is used, the numerical dissipation shows a second order decay, and it is relatively independent of the differentiation parameters. Only for $\varepsilon = 10^{-1}$ a small deviation can be observed. In contrast, when using the Minmod limiter, the dissipation values are the same as in the first order case for all finite difference computations. Algorithmic differentiation yields only slightly better results, which are still far from beeing second order accurate. The limiters van Albada and van Leer show a very similar behaviour (compare Fig. 3).

For the case of finite differences, this can be explained as follows: Since the state q at which the Jacobian is computed is constant, i.e. $q_i = \bar{q}$ for all i, if we evaluate the extrapolated states $q^L_{i+\frac{1}{2}}, q^R_{i+\frac{1}{2}}$ starting from a state in which only one component in one cell is perturbed by Δq^j_i, the correction terms $\widetilde{\Delta q}^L, \widetilde{\Delta q}^R$ in eqns. (3), (4) are always zero (since either $r \leq 0$ and therefore $\varphi(r) = 0$, or the state difference is zero) and so the extrapolation reduces to a first order scheme.

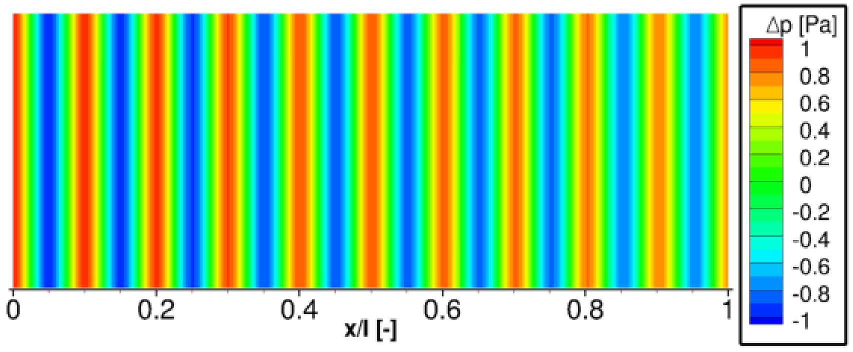

Figure 1: Solution produced by the linear solver for the propagation of a wave with a frequency of 5000 Hz using the following setting: second order spatial discretisation, van Albada type limiter with $\varepsilon_{\text{vA}} = 10^{-4}$, step size $\varepsilon = 10^{-2}$ for finite differences.

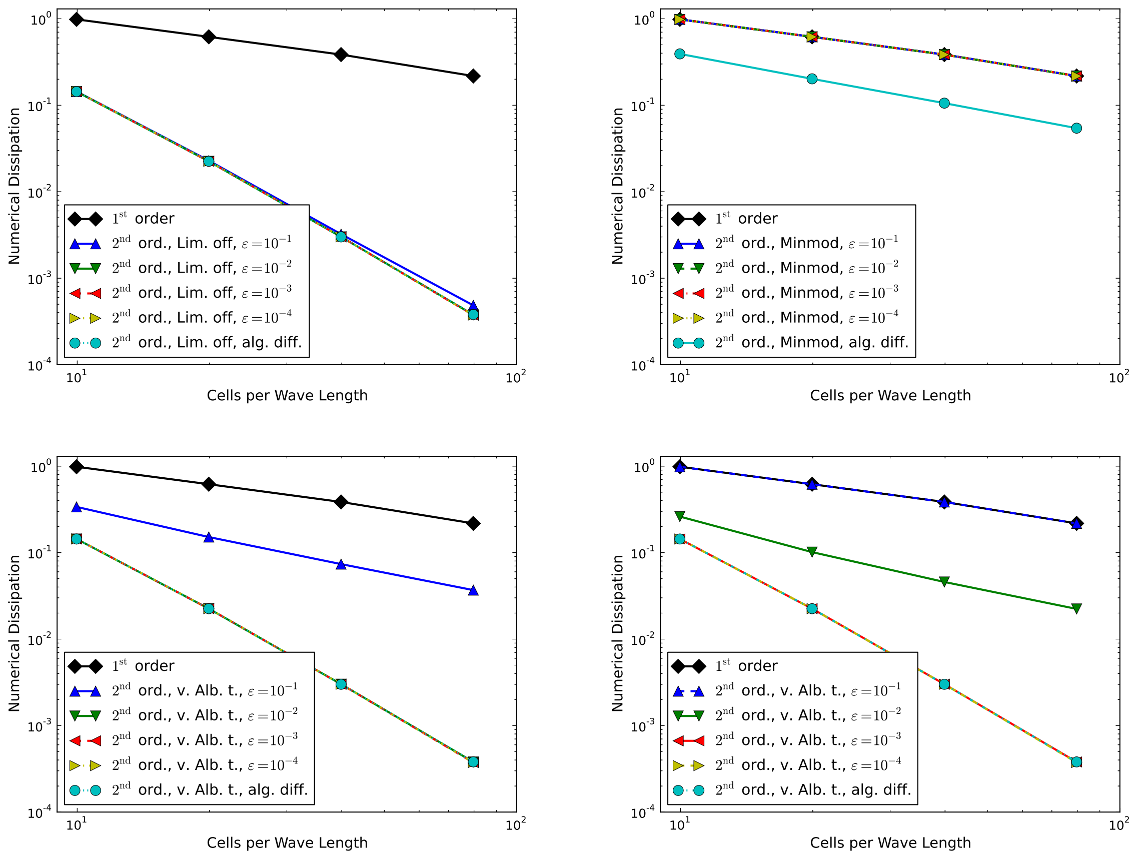

Figure 2: Comparison of numerical dissipation curves using different differentiation methods for four limiter settings: limiter off (top left), Minmod (top right), van Albada type with $\varepsilon_{\mathrm{vA}} = 10^{-4}$ (bottom left) , van Albada type with $\varepsilon_{\mathrm{vA}} = 10^{-8}$ (bottom right). For comparison, the first order curve is also shown in each plot.

The van Albada type limiter yields always the second order curve if algorithmic differentiation is used, while for finite differences the behaviour depends on the value of the constant $\varepsilon_{\mathrm{vA}}$. For $\varepsilon_{\mathrm{vA}} = 10^{-4}$ only the largest step size $\varepsilon = 10^{-1}$ leads to worse results (see bottom left plot in Fig. 2), for $\varepsilon_{\mathrm{vA}} = 10^{-8}$ a step size of (at most) $\varepsilon = 10^{-3}$ is needed to keep the second order decay. An analysis shows that the relevant quantity is the ratio $\frac{\varepsilon^4}{\varepsilon_{\mathrm{vA}}}$. If this is small (about 10^{-2} or smaller), the finite differences are very similar to the case where no limiter is used, if it is large (about 10^2 or greater), the extrapolation scheme reduces effectively to first order.

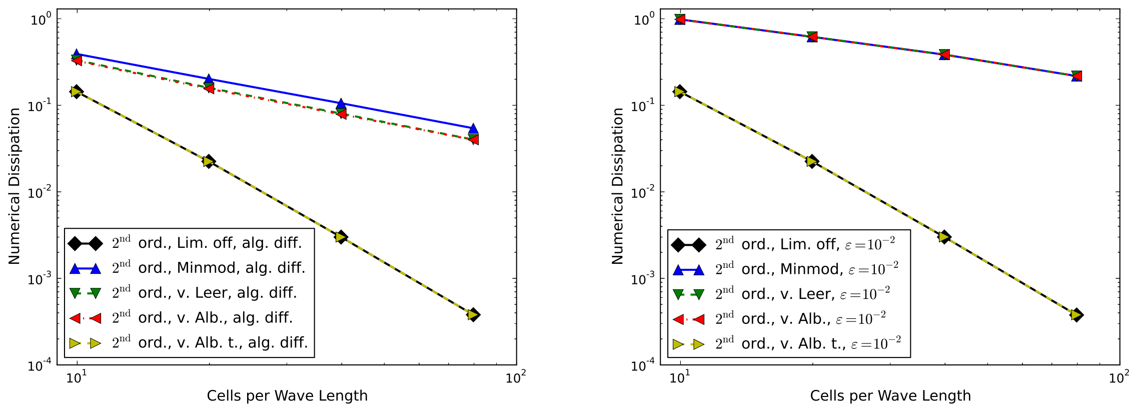

Figure 3: Comparison of numerical dissipation curves using different limiters for exact differentiation (left) and finite differences with step size 10^{-2} (right). The van Albada type limiter is here used with $\varepsilon_{\text{vA}} = 10^{-4}$.

3.2 Adjoint sensitivities for a compressor rotor

As a second test case we use a one-stage turbomachinery configuration, the Darmstadt Transonic Compressor with the baseline rotor geometry (Rotor 1) [25], see Fig. 4. The considered operating point is at a rotor velocity of 20,000 rounds per minute and a mass flow of approximately 16.3 kg/s. The resulting pre-shock Mach number is about 1.5.

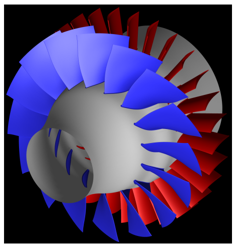

Figure 4: Computational model of the Darmstadt Transonic Compressor.

We use mass flow at the exit as objective functional and compute sensitivities with respect to small variations of the stagger angle at eight different radial heights, as described in [12]. For simplicity we keep the numerical setup for the solution of the linear equation system fixed, using ILU as preconditioner and a restart interval of 120 for the GMRes algorithm. We study the same combinations of limiters and differentiation parameters as in the previous section, except that the smallest value for ε_{vA} which we consider here is 10^{-6}. To compare the different setups, we look at the computed sensitivities as well as the residuals of the GMRes solver (see figures 5 and 6).

As before, we note that if no limiter is used, the convergence behaviour as well as the

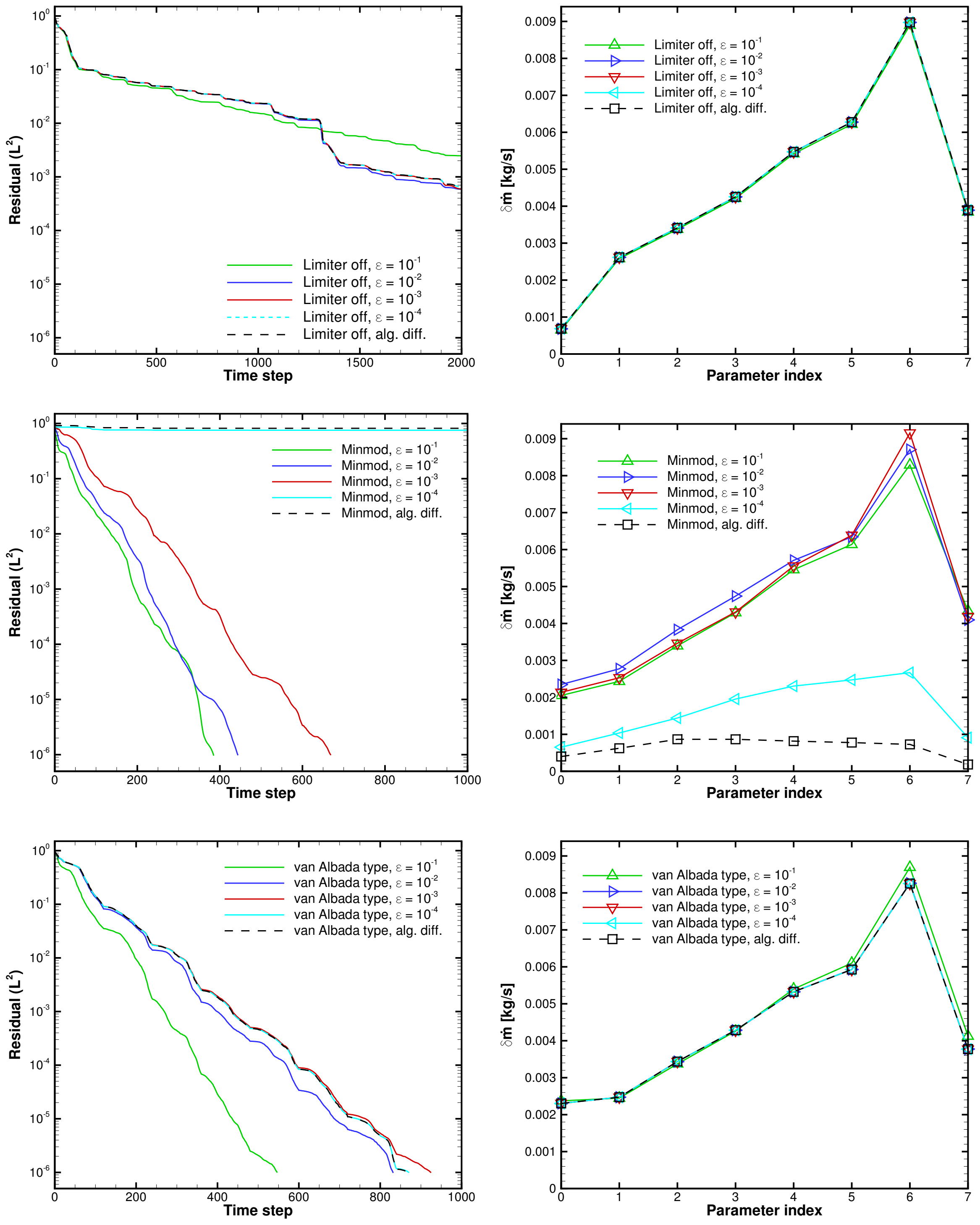

Figure 5: Comparison of convergence behaviour (left) and sensitivities (right) using different differentiation methods for three limiter settings: limiter off (top), Minmod (mid), van Albada type with $\varepsilon_{\mathrm{vA}} = 10^{-4}$ (bottom).

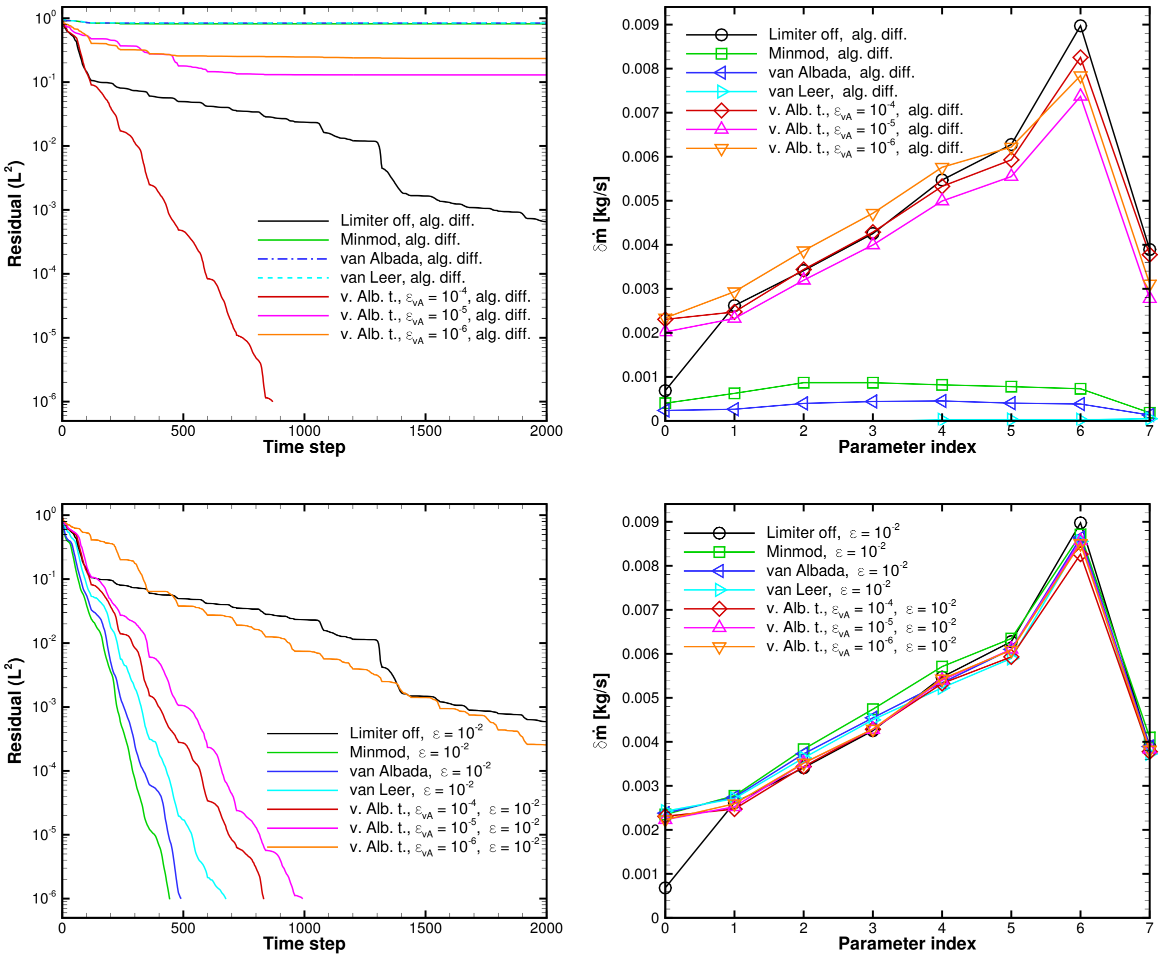

Figure 6: Comparison of convergence behaviour (left) and sensitivities (right) using different limiters for exact differentiation (top) and finite differences with step size 10^{-2} (bottom).

sensitivities depend hardly on the differentiation method. The van Albada type limiter behaves quite similarly, except that the differences between $\varepsilon = 10^{-1}$ and the other cases are a bit larger. But it can also be seen that for all differentiation methods the convergence is much faster than without limiter. The results for the Minmod limiter depend more strongly on the differentiation method. In particular, for finite differences with step size 10^{-4} and for algorithmic differentiation the GMRes algorithm does not converge any more. Therefore, the computed sensitivities are of no use in these cases.

Figure 6 shows that the other limiters (van Albada, van Leer), and also the van Albada type limiter with $\varepsilon_{\mathrm{vA}} < 10^{-4}$ have similar problems in the case of exact differentiation. In principle, the same applies to the finite difference results, although the minimal step size for which the computation converges at all depends on the choice of the limiter. For the cases with convergence problems, different combinations of preconditioner and restart interval, e.g. ILU with increased level of fill-in and restart interval 300, did not improve the convergence behaviour significantly.

We also see in Fig. 6 that for a given differentiation method the resulting sensitivities depend – in some cases very strongly – on the limiter used, even if all solutions are converged equally well. This is of course not very surprising, since also the steady and the adjoint solution change considerably when a different limiter is used. The very large deviation for the first parameter in the case where no limiter is used is due to significant differences in the adjoint solution near the hub, which do not have a counterpart in the steady solution and have to be further investigated.

4 CONCLUSION

We have shown that it is possible to construct the flux Jacobian for a discrete adjoint or linear solver using finite differences, but the choice of the limiter function is a very critical point for any discrete linearisation. On the other hand, if no limiter is used, the convergence of the nonlinear and the linear/adjoint solvers becomes considerably slower. For a scheme without limiter as well as for an appropriate choice of the limiter function (van Albada type), the computations based on finite differences become nearly identical to those with the "exact" flux Jacobian (computed using algorithmically differentiated routines) if the step size is chosen sufficiently small. In the examples considered, "sufficiently small" means in the order of 10^{-3} to 10^{-4}, and already for a step size of 10^{-2} the resulting sensitivities and numerical dissipation rates are very close to the AD reference. However, for many choices of the limiter functions, in particular for all considered TVD limiters, the behaviour of the linearization is problematic. For the "numerical dissipation" test case the accuracy reduces to first order in these cases. For the compressor test case, the linear system based on the exact linearisation or finite differences with small step size becomes very stiff so that the GMRes algorithm does not converge any more or only very slowly. The reason for this has to be studied in more detail. We have also seen that the behaviour of the van Albada type limiter is very sensitive with respect to the choice of the constant $\varepsilon_{\mathrm{vA}}$.

Acknowledgements Financial support by MTU Aero Engines (co-sponsorsip of the first author) and the German Ministry of Economy and Energy (Project R&E TURB, project number 20T1104B) is gratefully acknowledged.

REFERENCES

[1] A. Jameson, "Aerodynamic design via control theory," *Journal of Scientific Computing*, vol. 3, no. 3, pp. 233–260, 1988.

[2] M. B. Giles and N. A. Pierce, "An introduction to the adjoint approach to design," *Flow, Turbulence and Combustion*, vol. 65, pp. 393–415, 2000.

[3] J. E. V. Peter and R. P. Dwight, "Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches," *Computers & Fluids*, vol. 39, no. 3, pp. 373–391, 2010.

[4] K. C. Giannakoglou and D. I. Papadimitriou, "Adjoint methods for shape optimization," in *Optimization and Computational Fluid Dynamics* (G. Janiga and D. Thévenin, eds.), Springer, 2008.

[5] J. R. R. A. Martins and J. T. Hwang, "Review and unification of methods for computing derivatives of multidisciplinary computational models," *AIAA Journal*, vol. 51, pp. 2582–2599, November 2013.

[6] M. B. Giles, M. C. Duta, and N. A. Pierce, "Algorithm developments for discrete adjoint methods," *AIAA J.*, vol. 41, no. 2, pp. 198–205, 2003.

[7] A. Griewank, *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation.* Philadelphia: SIAM, 2000.

[8] A. C. Marta, C. A. Mader, J. R. R. A. Martins, E. Van der Weide, and J. J. Alonso, "A methodology for the development of discrete adjoint solvers using automatic differentiation tools," *Int. J. Comput. Fluid D.*, vol. 21, no. 9-10, pp. 307–327, 2007.

[9] C. A. Mader, J. R. R. A. Martins, J. J. Alonso, and E. van der Weide, "ADJoint: An approach for the rapid development of discrete adjoint solvers," *AIAA JOURNAL*, vol. 46, pp. 863–873, APR 2008. AIAA/ISSMO 11th Multidisciplinary Analysis and Optimization Conference, Portsmouth, VA, SEP 06-08, 2006.

[10] F. Courty, A. Dervieux, B. Koobus, and L. Hascoet, "Reverse automatic differentiation for optimum design: from adjoint state assembly to gradient computation," *Optimization Methods and Software*, vol. 18, no. 5, pp. 615–627, 2003.

[11] C. Frey, D. Nürnberger, and H.-P. Kersken, "The discrete adjoint of a turbomachinery RANS solver," in *Proceedings of ASME-GT2009*, 2009.

[12] C. Frey, G. Ashcroft, J. Backhaus, E. Kügeler, and J. Wellner, "Adjoint-based flow sensitivity analyis using arbitrary control surfaces," in *Proceedings of ASME-GT2011*, 2011.

[13] H.-P. Kersken, C. Frey, C. Voigt, and G. Ashcroft, "Time-Linearized and Time-Accurate 3D RANS Methods for Aeroelastic Analysis in Turbomachinery," *J. Turbomach.*, vol. 134, no. 5, 2012.

[14] C. Frey, G. Ashcroft, H.-P. Kersken, and C. Weckmüller, "Advanced numerical methods for the prediction of tonal noise in turbomachinery — Part II: Time-linearized methods," *Journal of Turbomachinery*, vol. 136, no. 2, pp. 021002–021002, 2013.

[15] M. Sagebaum, E. Özkaya, and N. R. Gauger, "Challenges in the automatic differentiation of an industrial CFD solver," in *Evolutionary and Deterministic Methods for Design, Optimization and Control with Application to Industrial and Societal Problems (EUROGEN 2013)*, 2013.

[16] D. Nürnberger, F. Eulitz, S. Schmitt, and A. Zachcial, "Recent progress in the numerical simulation of unsteady viscous multistage turbomachinery flow," in *ISABE 2001-1081*, Sept. 2001.

[17] K. Becker, K. Heitkamp, and E. Kügeler, "Recent progress in a hybrid-grid CFD solver for turbomachinery flows," in *Proceedings Fifth European Conference on Computational Fluid Dynamics ECCOMAS CFD 2010*, 2010.

[18] P. L. Roe, "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, vol. 43, no. 2, pp. 357–372, 1981.

[19] B. van Leer, "Towards the ultimate conservative difference scheme. V. a second-order sequel to Godunov's method," *Journal of Computational Physics*, vol. 32, no. 1, pp. 101–136, 1979.

[20] J. Blazek, *Computational fluid dynamics: principles and applications.* Elsevier Science, 2001.

[21] C. Hirsch, *Numerical Computation of Internal and External Flows – Computational Methods for Inviscid and Viscous Flows*, vol. 2. Wiley, 1 ed., 1990.

[22] W. Anderson, J. Thomas, and B. van Leer, "Comparison of finite volume flux vector splittings for the Euler equations," *AIAA JOURNAL*, vol. 24, pp. 1453–1460, SEP 1986.

[23] H. Benetschik and H. E. Gallus, "Inviscid and viscous transonic flows in cascades using an implicit upwind algorithm," *Journal of Propulsion and Power*, vol. 8, no. 2, pp. 403–409, 1992.

[24] A. Griewank and A. Walther, "Getting started with ADOL-C," in *Combinatorial Scientific Computing* (U. Naumann and O. Schenk, eds.), CRC Press, Taylor and Francis Group, 2012.

[25] G. Schulze, D. K. Hennecke, J. Sieber, and B. Wöhrl, "Der neue Verdichterprüfstand an der TH Darmstadt," *VDI Berichte Nr. 1109, Germany.*, 1994.