INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
*Int. J. Numer. Meth. Fluids* 2009; **59**:1–18
Published online 27 March 2008 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1798

# On CFL evolution strategies for implicit upwind methods in linearized Euler equations

H. M. Bücker[1], B. Pollul[2,*,†] and A. Rasch[1]

[1]*Institute for Scientific Computing, RWTH Aachen University, D-52056 Aachen, Germany*
[2]*Institut für Geometrie und Praktische Mathematik, RWTH Aachen University, D-52056 Aachen, Germany*

## SUMMARY

In implicit upwind methods for the solution of linearized Euler equations, one of the key issues is to balance large time steps, leading to a fast convergence behavior, and small time steps, needed to sufficiently resolve relevant flow features. A time step is determined by choosing a Courant–Friedrichs–Levy (CFL) number in every iteration. A novel CFL evolution strategy is introduced and compared with two existing strategies. Numerical experiments using the adaptive multiscale finite volume solver QUADFLOW demonstrate that all three CFL evolution strategies have their advantages and disadvantages. A fourth strategy aiming at reducing the residual as much as possible in every time step is also examined. Using automatic differentiation, a sensitivity analysis investigating the influence of the CFL number on the residual is carried out confirming that, today, CFL control is still a difficult and open problem. Copyright © 2008 John Wiley & Sons, Ltd.

Received 16 October 2007; Revised 12 February 2008; Accepted 17 February 2008

KEY WORDS: CFL number; implicit upwind methods; Euler equations; sensitivity analysis; automatic differentiation; QUADFLOW; ADIFOR

## 1. INTRODUCTION

Implicit time integration methods for the solution of stiff systems of partial differential equations arise from a variety of different areas in computational physics. Examples include plasma physics, reactive and geophysical flows, radiation diffusion, radiation hydrodynamics, and computational fluid dynamics; see the references in the survey by Knoll and Keyes [1]. Implicit schemes offer the advantage of allowing large time steps, potentially leading to fast convergence. However, when the

*Correspondence to: B. Pollul, Institut für Geometrie und Praktische Mathematik, RWTH Aachen University, D-52056 Aachen, Germany.
†E-mail: pollul@igpm.rwth-aachen.de

Contract/grant sponsor: German Research Foundation (DFG)
Contract/grant sponsor: Collaborative Research Center SFB 401

system of nonlinear equations resulting from the spatial discretization is solved in every time step, a small time step is desirable to accelerate the convergence of Newton's method. Finding a balance between these two conflicting objectives is one of the challenges to develop efficient implicit techniques for the numerical simulation of physical phenomena governed by partial differential equations.

Practically, balancing large and small time steps is carried out by choosing a Courant–Friedrichs–Levy (CFL) number for each time step. The CFL condition [2] for any explicit method requires that the time step is limited by the time for some significant action to occur, and preferably considerably less. In an implicit scheme, however, larger CFL numbers are allowed leading to larger time step sizes. Despite its high relevance in various practical applications, the choice of the sequence of CFL numbers in an implicit time integration method still remains an open problem, which has not yet been solved.

The new contributions of this article are threefold: (i) a new CFL evolution strategy called residual difference method (RDM) is proposed; (ii) experimental results are reported showing that a CFL evolution strategy that minimizes the residual as a function of the CFL number is not necessarily favorable; and (iii) a sensitivity analysis assessing the influence of the CFL number on the residual is carried out.

The utility of the new methodologies is demonstrated in a challenging fluid dynamical problem using the QUADFLOW [3–6] package. This adaptive multiscale finite volume solver for stationary and non-stationary compressible flow computations implements a Newton–Krylov approach that is often used in practice [1, 7–12].

The structure of this paper is as follows. In Section 2, the formulation of the Euler equations and the solver QUADFLOW used throughout this work for numerical experiments are briefly described. In Section 3, two known evolution strategies determining the CFL numbers for implicit time integration methods are sketched and a new CFL evolution strategy is proposed. In Section 4, numerical experiments are reported comparing the three CFL evolution strategies. A different CFL evolution strategy in which the CFL numbers are selected in a such way that the residual decreases as much as possible in every time step is explored in Section 5. In Section 6, a sensitivity analysis of CFL evolution strategies is carried out using automatic differentiation.

## 2. DISCRETE EULER EQUATIONS

In this study, we consider the conservative formulation of the Euler equations for a compressible gas. For an arbitrary control volume $V \subset \mathbb{R}^d$ $(d=2,3)$ with boundary $\partial V$ and an outward unit normal vector $\mathbf{n}$ on the surface element $\mathrm{d}S \subset \partial V$, we have equations of the form

$$\int_V \frac{\partial \mathbf{u}}{\partial t}\,\mathrm{d}V + \oint_{\partial V} \mathbf{F}(\mathbf{u})\mathbf{n}\,\mathrm{d}S = 0 \tag{1}$$

Here, $\mathbf{u} = (\rho, \rho\mathbf{v}, \rho e_{\mathrm{tot}})^{\mathrm{T}}$ denotes the vector of unknown conserved quantities, and $t$ denotes the time. The convective flux is given by

$$\mathbf{F}(\mathbf{u}) = \begin{pmatrix} \rho\mathbf{v} \\ \rho\mathbf{v}\circ\mathbf{v} + p\mathbf{I} \\ \rho e_{\mathrm{tot}}\mathbf{v} + p\mathbf{v} \end{pmatrix} \tag{2}$$

where $\rho$ denotes the density, $p$ the static pressure, $\mathbf{v}$ the velocity vector of the fluid, and $e_{\text{tot}}$ the total energy. The symbol $\circ$ denotes the dyadic product, and $\mathbf{I}$ represents the identity. The system is closed by the equation of state for a perfect gas and suitable initial and boundary conditions.

For the numerical simulation, we use the finite volume solver QUADFLOW [3–6] containing methods for the solution of two- and three-dimensional compressible Euler– and Navier–Stokes equations. It is based on block-structured grids using tensor-product B-splines. A key ingredient in QUADFLOW is the use of local grid refinement in regions of high activity.

The computation of an accurate approximation of a stationary solution is based on a nested iteration approach. Starting with an initial coarse grid, the time integration is computed until a tolerance criterion for the residual is satisfied. Thereafter, a grid adaptation, i.e. a (local) grid refinement, is performed, and the time integration procedure starts again with an interpolated initial condition. The indicator for the local grid refinement is based on a multiscale analysis using wavelets. To give an impression of the multi-block and adaptivity features of QUADFLOW, we show two computational grids that are used in a simulation of an inviscous flow around a BAC 3-11/RES/30/21 airfoil [13] in Figure 1. This grid is also used later in test case 2 of Section 4.

Although we are interested in a stationary solution, we perform a non-stationary computation resulting in a numerical method for approximating the stationary solution. This approach is also known as pseudo-transient continuation, cf. [14]. To obtain fast convergence towards the stationary solution, one wishes to use large time steps and thus an implicit time discretization method is preferred. On every level of adaptation we start with an initial CFL number, which determines the first time step. In QUADFLOW, the relation between the CFL number $\gamma$ and the local time step $\Delta t_i$ for the $i$th cell is given by [6]

$$\Delta t_i = \gamma \frac{V_i}{\lambda_i^{\text{c}}} \tag{3}$$

where $V_i$ is the volume of the cell $i$. The quantity $\lambda_i^{\text{c}}$ is related to the maximum eigenvalue of the Euler equations and is defined by the following integral over the bounding surface of the control

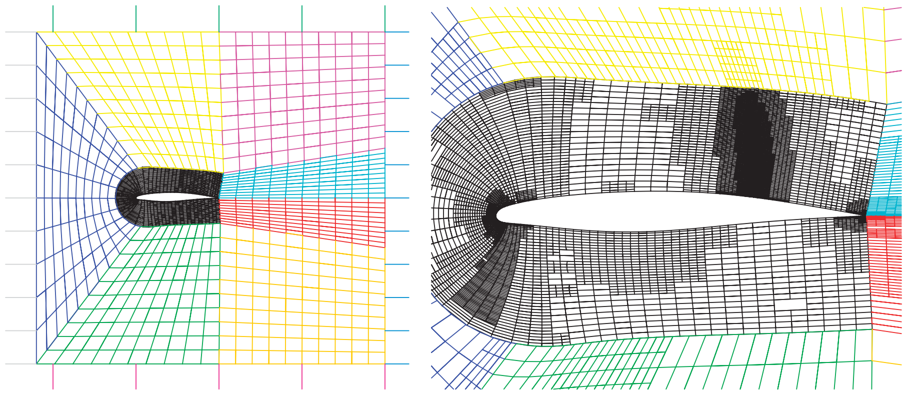

Figure 1. Computational grids for the BAC 3-11/RES/30/21 airfoil. The whole grid consists of 12 blocks. A part of the initial grid is depicted on the left. The right picture shows a part of the grid after 10 adaptations of local refinement.

volume:

$$\lambda_i^c = \oint_{\partial V} (|\mathbf{vn}| + a)\, \mathrm{d}S$$

where $a$ is the speed of sound.

In the following time steps, the CFL number (and thus the time step) is varied by one of the three CFL evolution strategies that are described in Section 3. This approach then results in a nonlinear system of equations in each time step. In each time step, one inexact Newton iteration is applied, using a lower-order approximation of the Jacobian $J$. This means that the linearization of the convective fluxes is based on a first-order accurate method in space. It solely facilitates solution information provided by compact stencils, which only couple direct face neighbors. The Jacobian matrix has the structure

$$J(\mathbf{u}) = \mathrm{diag}\left(\frac{V_i}{\Delta t_i}\right) + \frac{\partial \mathbf{R}(\mathbf{u})}{\partial \mathbf{u}} \tag{4}$$

where $\mathbf{R}(\mathbf{u})$ denotes the residual vector. Details are given in [5]. Note that in general a smaller time step will improve the condition number of the approximated Jacobian in (4). The resulting linear systems are solved using preconditioned Krylov subspace methods that are available in the PETSc library [15, 16]. The choice of the time step is crucial for the rate of convergence.

*Remark*
In the present study, the flux-vector splitting by Hänel and Schwane [17] is combined with the Venkatakrishnan limiter [18]. A linear reconstruction technique is applied to obtain second-order accuracy in regions where the solution is smooth. We use the B2-scheme by Batten *et al.* [19]. A point-block-ILU(0) preconditioned BiCGSTAB [20] method is used to solve the system of linear equations. The actual implementation is detailed in [21]. The CFL evolution strategies described in Section 3 show a similar behavior if a different upwind scheme, a different implicit time integration method, or an alternate Krylov method is employed.

## 3. CFL EVOLUTION STRATEGIES

Implicit time integration methods allow large time steps. In contrast to explicit methods, the time-stepping is not limited by the CFL condition [2]. That is, values $\gamma > 1$ can be used for the CFL number. For stationary flows, it is known that large time steps are allowed only near the steady-state solution and that one should choose small time steps as long as the flow is setting up.

For steady flows, the CFL number $\gamma_k = \gamma(t_k)$ at a time step $t_k$ is usually varied in a prescribed interval $\gamma_k \in [\gamma_{\min}, \gamma_{\max}]$. With small CFL numbers $\gamma_k$, one has to perform many time steps in order to achieve convergence. Choosing the CFL number $\gamma_k$ too large may result in a breakdown of the iteration process. There are two possible reasons for breakdowns:

- Too much time has been elapsed in the corresponding time step. Therefore, some flow features could not be resolved correctly so that the iteration process diverges towards a non-physical flow.
- The condition number of the Jacobian (4) is too large such that the Krylov solver cannot solve the corresponding system of linear equations within a prescribed number of iterations.

In the following, we review two different methods for choosing CFL numbers for stationary problems, and we also suggest a new strategy.

### 3.1. *Exponential progression (EXP)*

The exponential law (EXP)

$$\gamma_{k+1} = \gamma_0 \cdot (\gamma_{\text{EXP}})^k, \quad k \in \mathbb{N}_0 \tag{5}$$

is used in many flow solvers [6, 22–24]. In every time step, the CFL number is increased by a constant factor i.e. $\gamma_{k+1} = \gamma_{\text{EXP}} \cdot \gamma_k$. The control parameters $\gamma_0$ and $\gamma_{\text{EXP}}$ completely determine a sequence of CFL numbers. Especially the choice of $\gamma_{\text{EXP}}$ is crucial, because choosing a larger value for $\gamma_{\text{EXP}}$ may speed up the overall convergence process. However, selecting $\gamma_{\text{EXP}}$ too large may result in a breakdown of the iteration process. Appropriate values for the control parameters are problem-dependent and in general not known *a priori*. Typical values are $\gamma_0 = \gamma_{\min}$ and $\gamma_{\text{EXP}} \in [1.05, 1.5]$.

### 3.2. *Switched evolution relaxation (SER)*

In contrast to the EXP strategy, the switched evolution relaxation (SER) [25] incorporates information from the iteration process. The norm of the relative residual, denoted by

$$R_k := \|\mathbf{R}_k\|_2$$

is directly coupled with the CFL number of the next time step:

$$\gamma_{k+1} = \gamma_{\text{SER}} \cdot \left(\frac{R_0}{R_k}\right)^{\alpha_{\text{SER}}}, \quad k \in \mathbb{N}_0 \tag{6}$$

In this manner, the sequence of CFL numbers selected by the SER method is not only determined by the choice of the control parameters $\gamma_{\text{SER}}$ and $\alpha_{\text{SER}}$ but also by the particular flow problem at hand. However, inappropriate values for the control parameters $\gamma_{\text{SER}}$ and $\alpha_{\text{SER}}$ can yield slow convergence or breakdown of the computation. In this study, we use $\alpha_{\text{SER}} \in [1.0, 5.0]$ and $\gamma_{\text{SER}} = \gamma_{\min}$ as in [6, 26]. In [24, 25] $\alpha_{\text{SER}} = 1.0$ is chosen and the value of $\gamma_{\text{SER}}$ is varied.

Reconsidering (6) the CFL numbers increase in every time step as follows:

$$\gamma_{k+1} = \gamma_k \cdot \left(\frac{R_{k-1}}{R_k}\right)^{\alpha_{\text{SER}}}, \quad k \in \mathbb{N}$$

This suggests a more general approach involving the previous residual $R_k$ and $R_{k-\ell}$ for fixed $\ell \in \mathbb{N}$.

$$\gamma_{k+1} = \gamma_k \cdot \left(\frac{R_{k-\ell}}{R_k}\right)^{\alpha_{\text{SER}}} \tag{7}$$

This variant may allow one to increase the CFL numbers faster, if the residual falls over $\ell$ iterations.

While we are unaware of any convergence theory using any of the presented CFL evolution strategies in a solver such as QUADFLOW, convergence theory for some pseudo-transient continuation methods can be found in [14] and subsequent articles.

### *3.3. Residual difference method (RDM)*

This new strategy is based on the following idea: consider an iteration towards a steady-state solution. If the iterates seem to be converged, i.e. the solution does not change significantly between two consecutive iterations, one may increase the time step to check whether the solution remains stable. On the other hand, if the solution $\mathbf{u}_k$ varies much from one time step to another, smaller time steps should be chosen in order to resolve all flow features. Note that the difference of the solutions of two consecutive iterations equals the difference of the corresponding errors. As the residual is taken as an estimate for the error, we have $\|\mathbf{u}_k - \mathbf{u}_{k-1}\| \approx \|\mathbf{R}_k - \mathbf{R}_{k-1}\|$. Therefore, we suggest the following evolution strategy referred to as residual difference method (RDM):

$$\gamma_{k+1} = \begin{cases} \gamma_{\min} & \text{if } k < k_0, \\ \gamma_{\mathrm{RDM}} \cdot \left( \dfrac{1}{|R_k - R_{k-1}|} \right)^{\alpha_{\mathrm{RDM}}} & \text{if } k \geqslant k_0, \end{cases} \quad k \in \mathbb{N}_0 \tag{8}$$

where $k_0 \geqslant 1$ denotes the first index satisfying $R_{k_0} \leqslant R_{k_0-1} - \varepsilon$. That is, we start the CFL evolution after the first step in which the reduction in the residual by an additive constant $\varepsilon$ occurs. The smaller the difference between the two consecutive residuals $|R_k - R_{k-1}|$, the larger the CFL number $\gamma_{k+1}$ for the next iteration is chosen. Before starting the CFL evolution, the CFL numbers are kept fixed.

The control parameters for RDM are $\gamma_{\mathrm{RDM}}$ and $\alpha_{\mathrm{RDM}}$. Similar to those in the EXP and SER strategies, they must be selected carefully in order to obtain rapid convergence on one hand and to avoid breakdowns on the other hand. In this study, we set $\varepsilon = 10^{-2}$ and choose $\gamma_{\mathrm{RDM}} \in \{1, 2, 5\}$ and $\alpha_{\mathrm{RDM}} \in [0.6, 6.0]$.

*Remark*

All three strategies may lead to breakdowns during the iteration process and the user might have to restart the computation with modified parameters. In [24] an expert system is proposed that switches between different CFL evolution strategies. While the SER and the RDM strategies are sensitive to changes in the residuals and may generate a sequence of oscillating CFL numbers, the EXP strategy increases the CFL number continually and may result in a sequence of too rapidly increasing CFL numbers.

## 4. NUMERICAL EXPERIMENTS

In this section, we present results of numerical experiments using the different CFL evolution strategies described above, where the CFL numbers are allowed to vary in the interval $[\gamma_{\min}, \gamma_{\max}] = [1, 10^5]$. All computations are carried out with QUADFLOW. We perform a time integration on every level of discretization until the residual of the density is reduced by a factor of $10^2$. On the finest level, we require the factor to be $10^4$. After every adaptation the CFL evolution restarts with $\gamma_1 = \gamma_{\min}$. We allow eight levels of refinement, i.e. each cell can be subdivided at most eight times. In total, 11 adaptations are actually performed. The initial solution on each adaptation level is obtained by interpolation of the solution of the previous adaptation level.

In the following, we investigate the number of iterations (time steps) as well as the actual central processing unit (CPU) time of the implicit time integration method needed to achieve convergence on the finest level of adaptation. Note that with larger CFL numbers the systems of linear equations in Newton's method are typically harder to solve, taking more CPU time.

In our experiments, the linear systems are solved until the relative residual is less than $10^{-2}$. We observed that using a lower accuracy might speed up the overall method in some cases, but also often diverges towards a non-physical flow. On the other hand, solving the linear systems with higher accuracy significantly increases the number of Krylov iterations without actually decreasing the number of time steps.

### 4.1. *Test problems*

The first configuration is a standard test problem for inviscid compressible flow solvers. We consider the transonic stationary flow around the NACA0012 airfoil [27], where $M_\infty = 0.8$ and $\alpha = 1.25^\circ$. In the following, this setting is denoted by test case 1A. We also investigate another test case, called 1B, mimicking a non-adaptive scheme. That is, the calculation on the finest adaptation level is initialized with free instream conditions rather than an interpolated solution of the previous adaptation level. In this case, more computational effort is typically needed to resolve all flow features. Thus, smaller times steps must be used in the initial phase of the computation.

The second test problem is the standard cruise configuration of the BAC 3-11/RES/30/21 transonic airfoil [13] with $M_\infty = 0.77$ and $\alpha = 0.0^\circ$. The computation of the corresponding stationary flow, referred to as test case 2, is also used in the Collaborative Research Center SFB 401 at RWTH Aachen University [28, 29].

### 4.2. *Parameter study on the CFL control parameters*

In this section, we present the results of a parameter study on the CFL control parameters for the three evolution strategies EXP, SER, and RDM. Here, the parameters $\gamma_{\text{EXP}}$, $\alpha_{\text{SER}}$, $\alpha_{\text{RDM}}$, and $\gamma_{\text{RDM}}$ are varied, while $\gamma_0$ and $\gamma_{\text{SER}}$ are assigned the fixed value 1.0. We use the SER strategy with the parameter $\ell = 1$, unless stated otherwise.

The results for test case 1A are displayed in Table I, showing, for varying parameter values, the number of time steps needed to achieve convergence, denoted by '# ts', as well as the actual CPU time, given in seconds. All timing results are obtained on an Intel Xeon processor running at 3 GHz clock speed. It turns out that, for all three CFL evolution strategies, the choice of the control parameters has a great impact on the number of time steps needed for convergence. For EXP, the number of time steps varies between 35 and 124 depending on the value of $\gamma_{\text{EXP}}$. For SER, this number varies between 34 and 172. For RDM, between 36 and 90 time steps are required to achieve convergence. A similar observation can be made for the total number of CPU seconds required for the calculation. For each CFL evolution strategy, we focus on the 'best' case, i.e. the set of control parameters resulting in the fastest overall convergence in terms of CPU time. For the EXP strategy, choosing $\gamma_{\text{EXP}} = 15$ results in an overall computation taking 31.6 s. Applying SER with $\alpha_{\text{SER}} = 4.5$ requires 27.2 s, and RDM needs 28.4 s, if the control parameters $\alpha_{\text{RDM}}$ and $\gamma_{\text{RDM}}$ are set to the values 5.0 and 1.0, respectively. These 'best' parameter values for the three CFL evolution strategies are given in boldface in Table I. For these values, the actual selected CFL numbers $\gamma_k$ as well as the corresponding residual history are depicted in Figure 2. Note that the progress of the residual, $R_k$, is very similar for all three strategies, which is not surprising as they all choose relatively large CFL numbers $\gamma_k$ already in the early iterations, and for time steps 5 and greater they always choose $\gamma_k = \gamma_{\max}$.

Table I. Test case 1A: time steps needed for convergence on the finest grid and the corresponding CPU times in seconds. Different values for the control parameters of the three CFL evolution strategies are compared.

| EXP | | | SER | | | RDM | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\gamma_{\mathrm{EXP}}$ | # ts | CPU | $\alpha_{\mathrm{SER}}$ | # ts | CPU | $\alpha_{\mathrm{RDM}}$ | $\gamma_{\mathrm{RDM}}$ | # ts | CPU |
| 1.1 | 124 | 80.8 | 1.1 | 172 | 121.7 | 1.0 | 1 | 90 | 67.2 |
| 1.2 | 88 | 60.0 | 1.2 | 138 | 109.2 | 1.0 | 2 | 77 | 63.9 |
| 1.5 | 52 | 41.7 | 1.5 | 90 | 75.9 | 1.5 | 1 | 37 | 32.7 |
| 2.0 | 53 | 42.8 | 2.0 | 44 | 43.1 | 1.5 | 2 | 40 | 31.5 |
| 3.0 | 48 | 41.6 | 2.5 | 47 | 42.0 | 2.0 | 1 | 41 | 32.2 |
| 5.0 | 45 | 37.9 | 3.0 | 43 | 37.5 | 2.0 | 2 | 41 | 33.0 |
| 10 | 40 | 36.0 | 3.5 | 41 | 34.5 | 3.0 | 1 | 42 | 34.5 |
| **15** | **37** | **31.6** | 4.0 | 39 | 32.9 | 4.0 | 1 | 39 | 29.3 |
| 20 | 38 | 32.4 | **4.5** | **37** | **27.2** | **5.0** | **1** | **36** | **28.4** |
| 50 | 35 | 33.4 | 5.0 | 34 | 28.5 | 6.0 | 1 | 36 | 29.0 |

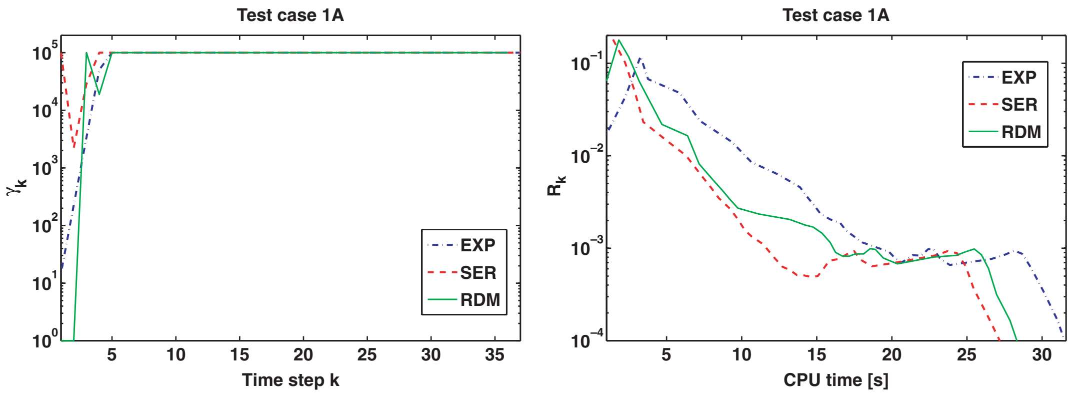

Figure 2. CFL numbers $\gamma_k$ selected by the three CFL evolution strategies (left) and the corresponding residual history (right) on the finest grid for test case 1A where the boldface values from Table I, $\gamma_{\mathrm{EXP}}=15$, $\alpha_{\mathrm{SER}}=4.5$, $\alpha_{\mathrm{RDM}}=5.0$, and $\gamma_{\mathrm{RDM}}=1.0$, are taken.

The situation is, however, different in test case 1B. The results of a corresponding parameter study are presented in Table II. It seems more difficult to find feasible values for the control parameters because values assumed to be appropriate for test case 1A may not even yield a converging iteration process in test case 1B. Such a divergence in the iteration process is indicated by '—' in the table. From all three CFL evolution strategies, EXP with $\gamma_{\mathrm{EXP}}=1.09$ is the best choice, both in terms of number of iterations and overall CPU time. However, when increasing the parameter $\gamma_{\mathrm{EXP}}$ to 1.1 or higher the strategy EXP does not yield any result at all because the iteration process diverges. Compared with EXP, the SER strategy leads to a much slower convergence, even when the 'best' parameter value, $\alpha_{\mathrm{SER}}=2.8$, is chosen. The reason for this slow convergence of SER is that it chooses only relatively small CFL numbers $\gamma_k$ in the first 1200 iterations. No convergence is achieved by SER when $\alpha_{\mathrm{SER}}$ is set to 2.7, 3.0, or greater than 3.0.

Table II. Test case 1B: time steps needed for convergence on the finest grid and the corresponding CPU times in seconds. Different values for the control parameters of the three CFL evolution strategies are compared.

| EXP | | | SER | | | RDM | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\gamma_{\text{EXP}}$ | # ts | CPU | $\alpha_{\text{SER}}$ | # ts | CPU | $\alpha_{\text{RDM}}$ | $\gamma_{\text{RDM}}$ | # ts | CPU |
| 1.05 | 189 | 116.3 | 1.3 | 1767 | 446.2 | 0.60 | 1 | 596 | 242.1 |
| 1.06 | 165 | 106.3 | 1.5 | 1642 | 410.5 | 0.80 | 1 | 290 | 151.0 |
| 1.07 | 148 | 99.9 | 2.0 | 1462 | 353.8 | 0.83 | 1 | 296 | 156.9 |
| 1.08 | 134 | 93.0 | 2.2 | 1417 | 338.4 | 0.85 | 1 | — | — |
| **1.09** | **124** | **87.6** | 2.4 | 1382 | 332.9 | 0.87 | 1 | 230 | 127.4 |
| 1.10 | — | — | 2.6 | 1354 | 322.6 | 0.93 | 1 | — | — |
| 1.11 | — | — | 2.7 | — | — | 0.94 | 1 | 186 | 120.8 |
| 1.12 | — | — | **2.8** | **1329** | **314.0** | 0.95 | 1 | — | — |
| 1.13 | — | — | 2.9 | 1320 | 314.3 | **0.98** | **1** | **168** | **100.5** |
| 1.14 | — | — | 3.0 | — | — | 1.00 | 1 | — | — |

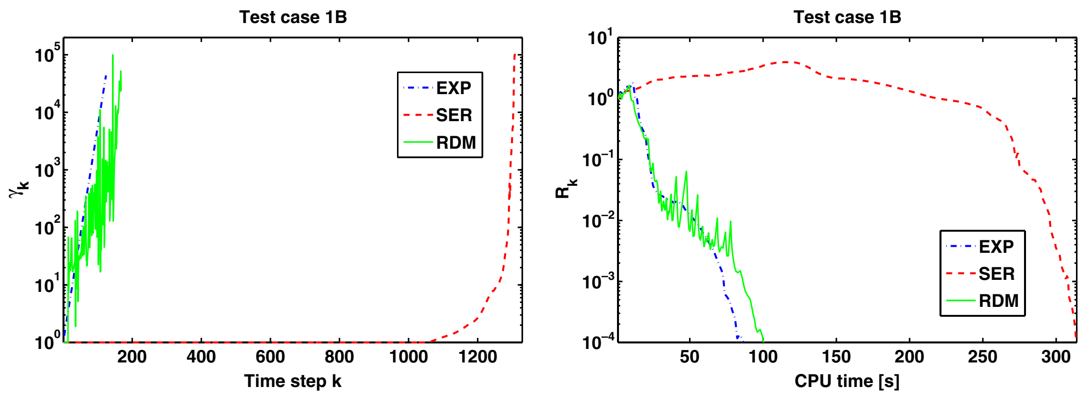

Figure 3. CFL numbers $\gamma_k$ selected by the three CFL evolution strategies (left) and the corresponding residual history (right) on the finest grid for test case 1B where the boldface values from Table II, $\gamma_{\text{EXP}} = 1.09$, $\alpha_{\text{SER}} = 2.8$, $\alpha_{\text{RDM}} = 0.98$, and $\gamma_{\text{RDM}} = 1.0$, are taken.

In the case of RDM, it is even harder to find feasible parameter values for $\alpha_{\text{RDM}}$ and $\gamma_{\text{RDM}}$ as many combinations yield a diverging iteration process. If, however, a feasible pair of values is found, e.g. $\alpha_{\text{RDM}} = 0.98$ and $\gamma_{\text{RDM}} = 1.0$, RDM is significantly faster than the SER strategy. For test case 1B, the selected CFL numbers for EXP, SER, and RDM, using their 'best' parameter values, and the corresponding residuals are plotted in Figure 3.

As a third example, consider the parameter study for test case 2 that is summarized in Table III. Again, it can be observed that the iteration process does not converge if the control parameters are not chosen carefully. The parameter values for EXP, SER, and RDM yielding the fastest iteration process in terms of CPU time are $\gamma_{\text{EXP}} = 1.2$, $\alpha_{\text{SER}} = 2.5$, $\alpha_{\text{RDM}} = 1.5$, and $\gamma_{\text{RDM}} = 1.0$. The selected CFL numbers using these parameter values and the corresponding residuals are plotted in Figure 4. Compared with SER and RDM, the EXP strategy with $\gamma_{\text{EXP}} = 1.2$ is significantly slower. While SER and RDM choose relatively large CFL numbers already in the early iterations, the EXP

Table III. Test case 2: time steps needed for convergence on the finest grid and the corresponding CPU times in seconds. Different values for the control parameters of the three CFL evolution strategies are compared.

| EXP | | | SER | | | RDM | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\gamma_{\text{EXP}}$ | # ts | CPU | $\alpha_{\text{SER}}$ | # ts | CPU | $\alpha_{\text{RDM}}$ | $\gamma_{\text{RDM}}$ | # ts | CPU |
| 1.05 | 183 | 328.9 | 1.2 | 80 | 252.8 | 1.0 | 1 | 70 | 210.0 |
| 1.1 | 104 | 226.0 | 1.4 | 51 | 188.6 | 1.0 | 2 | 47 | 158.2 |
| **1.2** | **62** | **169.2** | 1.5 | 42 | 165.5 | 1.0 | 5 | 41 | 152.8 |
| 1.3 | — | — | 1.6 | 36 | 151.5 | 1.1 | 1 | 56 | 192.3 |
| 1.4 | — | — | 1.7 | 32 | 149.5 | 1.1 | 2 | 41 | 161.3 |
| 1.5 | — | — | 1.8 | 28 | 128.4 | 1.1 | 5 | 32 | 121.9 |
| 1.6 | — | — | 1.9 | 26 | 122.2 | 1.4 | 1 | 32 | 135.4 |
| 1.7 | — | — | 2.0 | — | — | **1.5** | **1** | **26** | **111.1** |
| 1.8 | — | — | **2.5** | **19** | **104.9** | 1.5 | 2 | — | — |
| 1.9 | — | — | 3.0 | — | — | 1.6 | 1 | — | — |

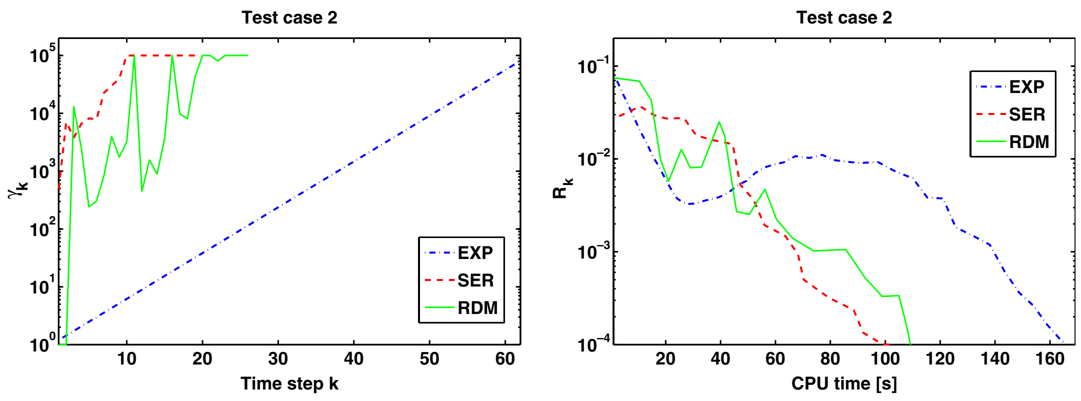

Figure 4. CFL numbers $\gamma_k$ selected by the three CFL evolution strategies (left) and the corresponding residual history (right) on the finest grid for test case 2 where the boldface values from Table III, $\gamma_{\text{EXP}} = 1.2$, $\alpha_{\text{SER}} = 2.5$, $\alpha_{\text{RDM}} = 1.5$, and $\gamma_{\text{RDM}} = 1.0$, are taken.

strategy has a constant increase factor of $\gamma_{\text{EXP}} = 1.2$. However, setting this factor to 1.3 or higher leads to a divergent iteration process, as shown in Table III. Using the 'best' parameter values stated above, the performance of SER and RDM is quite similar with a slight advantage of SER. Hence, the SER method yields the fastest iteration process, if the control parameter $\alpha_{\text{SER}}$ is chosen appropriately.

The SER strategy can be parametrized with the parameter $\ell$ as shown in (7). In the following, we investigate the impact of $\ell = 1, 2, 5, 10$ on the performance of SER. The corresponding results are presented in Table IV for test case 2. For fixed $\alpha_{\text{SER}}$, increasing $\ell$ can lead to faster convergence. On the other hand, divergence to a non-physical solution can occur, if $\ell$ is chosen too large. Computations with larger values for $\ell$ require smaller values for $\alpha_{\text{SER}}$. Therefore, the corresponding values for $\alpha_{\text{SER}}$ in Table IV are small compared with those in Table III. Thus, both parameters

Table IV. Test case 2: time steps needed for convergence on the finest grid and the corresponding CPU times in seconds. Different values for the control parameters $\alpha_{\text{SER}}$ and $\ell$ of the more general SER strategy (7) are compared.

| | SER ($\ell=1$) | | SER ($\ell=2$) | | SER ($\ell=5$) | | SER ($\ell=10$) | |
|---|---|---|---|---|---|---|---|---|
| $\alpha_{\text{SER}}$ | # ts | CPU | # ts | CPU | # ts | CPU | # ts | CPU |
| 0.2 | 8952 | 6039.0 | 2096 | 1889.4 | 152 | 366.2 | 34 | 141.6 |
| 0.4 | 2094 | 1868.1 | 297 | 566.0 | 28 | 124.2 | — | — |
| 0.6 | 698 | 922.2 | 83 | 255.4 | — | — | — | — |
| 0.8 | 295 | 564.1 | 38 | 163.1 | — | — | — | — |
| 1.0 | 144 | 365.3 | — | — | — | — | — | — |
| 1.2 | 80 | 252.8 | 20 | 100.5 | — | — | — | — |
| 1.4 | 51 | 188.6 | — | — | — | — | — | — |
| 1.6 | 36 | 151.5 | — | — | — | — | — | — |

$\alpha_{\text{SER}}$ and $\ell$ have similar effects on the CFL number. However, if the residuals almost stagnate, as in test case 1B, the selected CFL numbers tend to stay in the order of one. Therefore, more than 1200 time steps are required in test case 1B, independent of the choice of the parameter $\ell$.

It would be desirable to determine values for the control parameters in advance. However, looking closer at Table III, one can imagine that this may be hardly possible, because the relations of CPU time *versus* control parameter have no single minimum: For example, looking at the CPU time needed when applying the SER strategy with the control parameters $\alpha_{\text{SER}}=1.8, \alpha_{\text{SER}}=1.9$, and $\alpha_{\text{SER}}=2.0$, one might believe that $\alpha_{\text{SER}}=1.9$ is a good choice and all values greater than 2.0 would result in a divergent iteration process. Nevertheless, a better choice is obviously $\alpha_{\text{SER}}=2.5$. A similar behavior can be observed in Table IV for $\ell=2$.

## 5. LOCALLY OPTIMAL CFL NUMBERS

As there is no clear winner among the three strategies for CFL-control (in terms of CPU time or total iterations needed to converge), a different approach is presented in this section. From an abstract point of view, each time step $k$ could be considered as a function

$$R_k : \mathbb{R}_+ \to \mathbb{R}_+, \quad \gamma_k \mapsto R_k(\gamma_k) \tag{9}$$

mapping a CFL number $\gamma_k$ to the norm of the relative density residual that is obtained after performing this iteration using $\gamma_k$. Some typical plots of the function (9) are given in Figure 5. Apparently, the shape of the functions does not change much from time step to time step. The idea is to find the best CFL number in every iteration, i.e. finding a value for $\gamma_k$ such that the residual gets as small as possible:

$$R_k(\gamma_k) = \min_{\gamma \in \mathbb{R}_+} R_k(\gamma) \tag{10}$$

Note that the residual $R_k$ in the $k$th iteration depends not only on $\gamma_k$ but also on the CFL numbers $\gamma_1, \dots, \gamma_{k-1}$ used in the previous iterations.

In order to test this approach, we implemented a heuristic search strategy approximating $\gamma_k$, denoted by LOC in the sequel, where in each iteration several trial steps are carried out, using

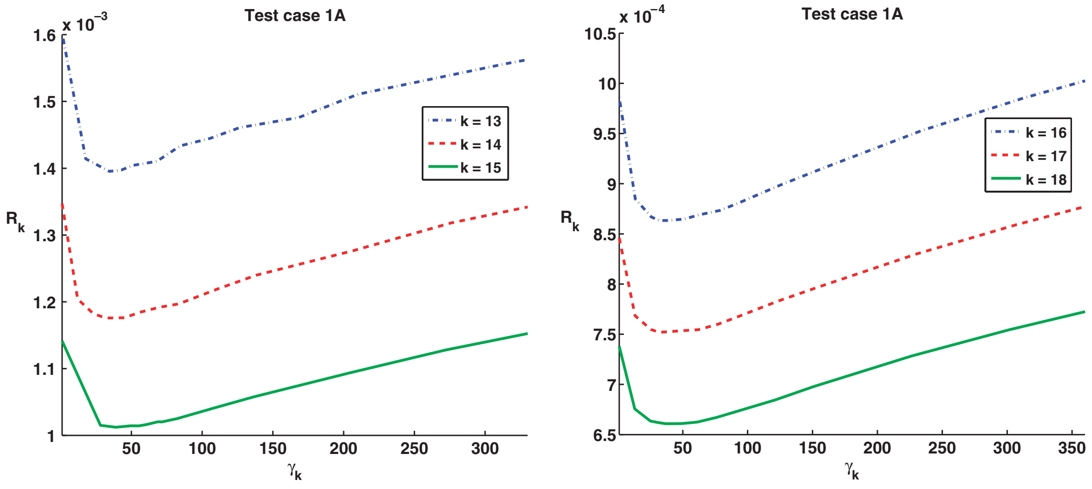

Figure 5. Relative residual of density $R_k$ for different values $\gamma_k$ for test case 1A on the finest grid. The residuals $R_k$ for time steps $k=13, 14, 15$ and for $k=16, 17, 18$ are shown in the left and right subplots, respectively. The CFL numbers $\gamma_1, \ldots, \gamma_{k-1}$ for the first $k-1$ time steps are selected by the LOC strategy, i.e. by approximating (10).

different values for the CFL number within a certain interval. In the neighborhood of the CFL value yielding a minimal residual, further trial steps are performed. From the set of CFL numbers tested during this heuristic search, the best CFL number, i.e. the value $\gamma_k$ that yields the smallest residual, is then employed to perform the actual iteration. The selected CFL numbers $\gamma_k$ for the computation corresponding to Figure 5 were gained using the LOC strategy. A clear decrease in the residuals $R_k$ can be observed in every time step.

However, it turns out that this method of choosing locally optimal CFL numbers does not decrease the total number of iterations. In fact, the total number of iterations needed for convergence is typically larger than if any of the other methods described in Section 3 were used. Taking test problem 1A from Section 4.1 as an example, applying the EXP strategy with parameters $[\gamma_{\min}, \gamma_{\max}]=[1, 1000]$, $\gamma_{\mathrm{EXP}}=1.1$, and $\gamma_0=1.0$ yields convergence within 159 iterations while the iteration process did not converge within 2000 iterations when the LOC strategy is employed. In additional experiments, a combination of EXP and LOC is used. More precisely, we carry out $n_{\mathrm{LOC}}-1$ iterations using the EXP strategy before we switch to LOC. We denote this strategy by LOC($n_{\mathrm{LOC}}$). The CFL numbers $\gamma_k$ used by the different strategies EXP and LOC($n_{\mathrm{LOC}}$) for $n_{\mathrm{LOC}} \in \{2, 20, 40, 80\}$ in each iteration $k$ are shown in the left subplot of Figure 6. The corresponding residuals $R_k$ are given in the right subplot of Figure 6. A closer look at Figure 6 reveals that, as soon as the LOC strategy is initiated, the relative density residual decreases quite fast. In subsequent iterations, the rate of decrease of the residual gets smaller such that almost no progress can be observed.

In the long run, the pure EXP strategy yields faster convergence although the residual actually *increases* during several iterations.

*Remark*
The implicit (e.g. Euler) time-stepping method and the Newton's method are in conflict in the following sense: It is obvious that for any implicit method the steady-state solution can be achieved

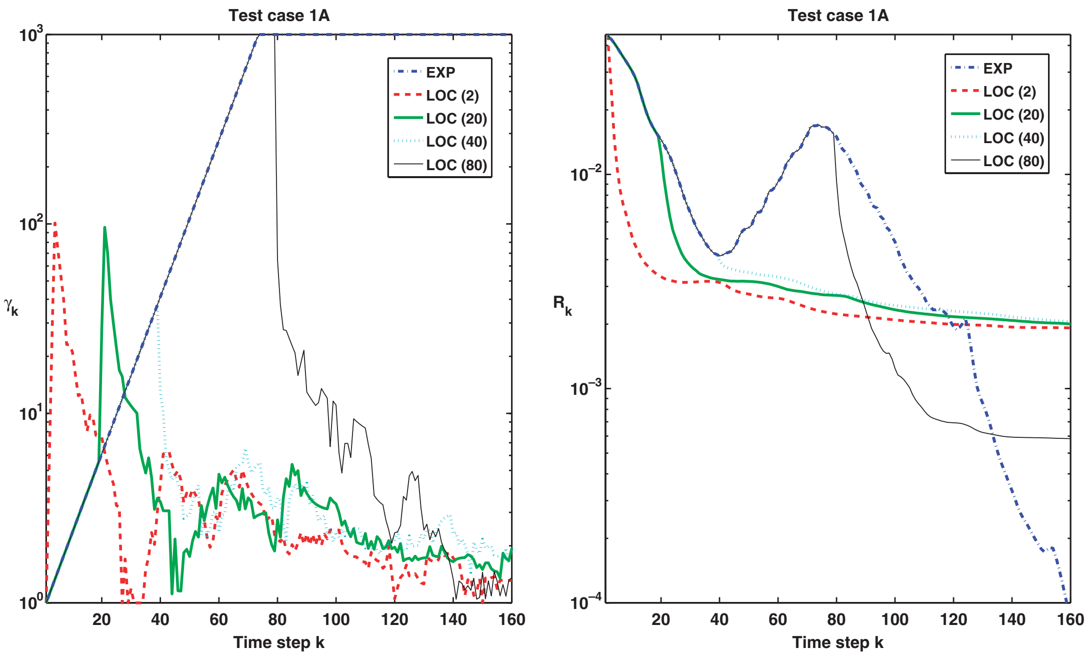

Figure 6. Test case 1A: CFL numbers $\gamma_k$ selected by EXP and LOC($n_{\text{LOC}}$), for various $n_{\text{LOC}}$ (left), and corresponding residual history $R_k$ (right).

faster by choosing larger time steps. On the other hand, Newton's method converges only locally and its rate of convergence increases with decreasing time steps.

Finding the balance between these two methods is the task of an adaptive CFL control strategy. This was in fact the motivation for the design of the LOC strategy, which decreases the density residual in every time step as much as possible. Although the residuals decrease rapidly in the beginning, the CFL numbers selected by LOC tend to become very small after a couple of time steps. Hence, the implicit time-stepping method requires significantly more iterations than if large CFL numbers, as e.g. determined by EXP, were used.

From Figure 6 it seems that a fast iteration process must allow an increase in the residual $R$. However, the LOC($n_{\text{LOC}}$) strategy eventually yields very small CFL numbers for all values of $n_{\text{LOC}}$.

Function (9) depends on many ingredients such as Newton's method, the implicit time integration method, the choices of the reconstruction scheme, the Riemann solver, the Krylov solver, and the limiter. Therefore, the residuals result from a multitude of ingredients. Thus, claiming a reduction in the residual in every time step might not be the fastest strategy. The main reason for the failure of the LOC strategy might be the time integration itself. Consider again test case 1B, cf. Section 4.1. Here, the calculation on the finest adaptation level was initialized with free instream conditions. Hence, in the first time step, the residual is zero in all cells except those cells that are adjacent to the profile. As the computation proceeds, the flow field evolves over the whole computational domain, starting from the profile. This naturally leads to an increase in the residual in the cells that are not adjacent to the profile. Therefore, an increase in the overall residual is not surprising for test case 1B. Even in an adaptive computation, such as test case 1A, the same effect can occur. Although we do not start with free instream conditions, but with an interpolated solution, there is always a region in which the grid has been changed by the adaptation and where even an increase in the residual from time step to time step may be advantageous for a fast convergence process.

## 6. SENSITIVITY ANALYSIS

In order to further investigate the relationship between the CFL number and the residual, we carry out a sensitivity analysis on the function $R_k$ given in (9). The derivative of the norm of the density residual $R_k$ w.r.t. the CFL number $\gamma_k$ is computed by automatic differentiation (AD). This term comprises a set of techniques for automatically augmenting a computer program with statements for the computation of derivatives. The AD technology is applicable whenever derivatives of functions given in the form of a high-level programming language, such as Fortran, C, or C++, are required. The reader is referred to the books [30, 31] and the proceedings of AD workshops [32–35] for details on this technique. In AD the program is treated as a—potentially very long—sequence of elementary operations such as addition or multiplication, for which the derivatives are known. Then the chain rule of differential calculus is applied repeatedly, combining these step-wise derivatives to yield the derivatives of the whole program. This mechanical process can be automated, and several AD tools are available for transforming a given code to the new *differentiated code*, also called *derivative code*; see `www.autodiff.org` for a recent list of AD tools. Thus, AD requires little human effort and produces derivatives without additional truncation error. That is, in contrast to numerical differentiation based on divided differencing, no evaluations with perturbed input are needed.

In this study, the differentiation procedure was not fully automatic because the parts of QUADFLOW that are relevant for this study consist of modules written in different programming languages, namely Fortran, C, and C++. In QUADFLOW, several low-level Fortran subroutines constitute the mathematical core of the flow solver. Higher-level routines calling these Fortran subroutines are implemented in C and C++ and are mainly responsible for the control flow and for the interfacing to the PETSc library [15, 16] written in C.

Since currently no AD tool is capable of augmenting mixed-language programs with derivatives, we followed a semi-automatic approach for this study: first, the low-level Fortran subroutines have been automatically transformed by ADIFOR [36]. Thereafter, the higher-level modules were modified manually such that subroutine invocations to the original low-level routines were replaced by corresponding calls to the differentiated versions of those Fortran routines. This also includes providing memory for the derivatives. When a system of linear equations is iteratively solved by the PETSc library, AD is not applied to the PETSC code but is handled in a hierarchical way as described in [37].

Employing the differentiated version of QUADFLOW, we are able to compute, in each iteration, $k$, not only the residual $R_k(\gamma_k)$ but also the derivative $\partial R_k(\gamma_k)/\partial\gamma_k$. The corresponding results for the three CFL evolution strategies EXP, SER, and RDM, using test case 1A, are presented in the first, second, and third rows of Figure 7, respectively. Each CFL evolution strategy is applied with three different sets of control parameters. The residuals $R_k(\gamma_k)$ are displayed in the plots on the left-hand side of Figure 7, whereas the plots on the right-hand side show the absolute values of the derivatives of $R_k(\gamma_k)$ with respect to $\gamma_k$. Note that the scales for $R_k(\gamma_k)$ and $|R_k(\gamma_k)/\partial\gamma_k|$ are logarithmic. In all cases, the absolute values of the derivatives are relatively large during the

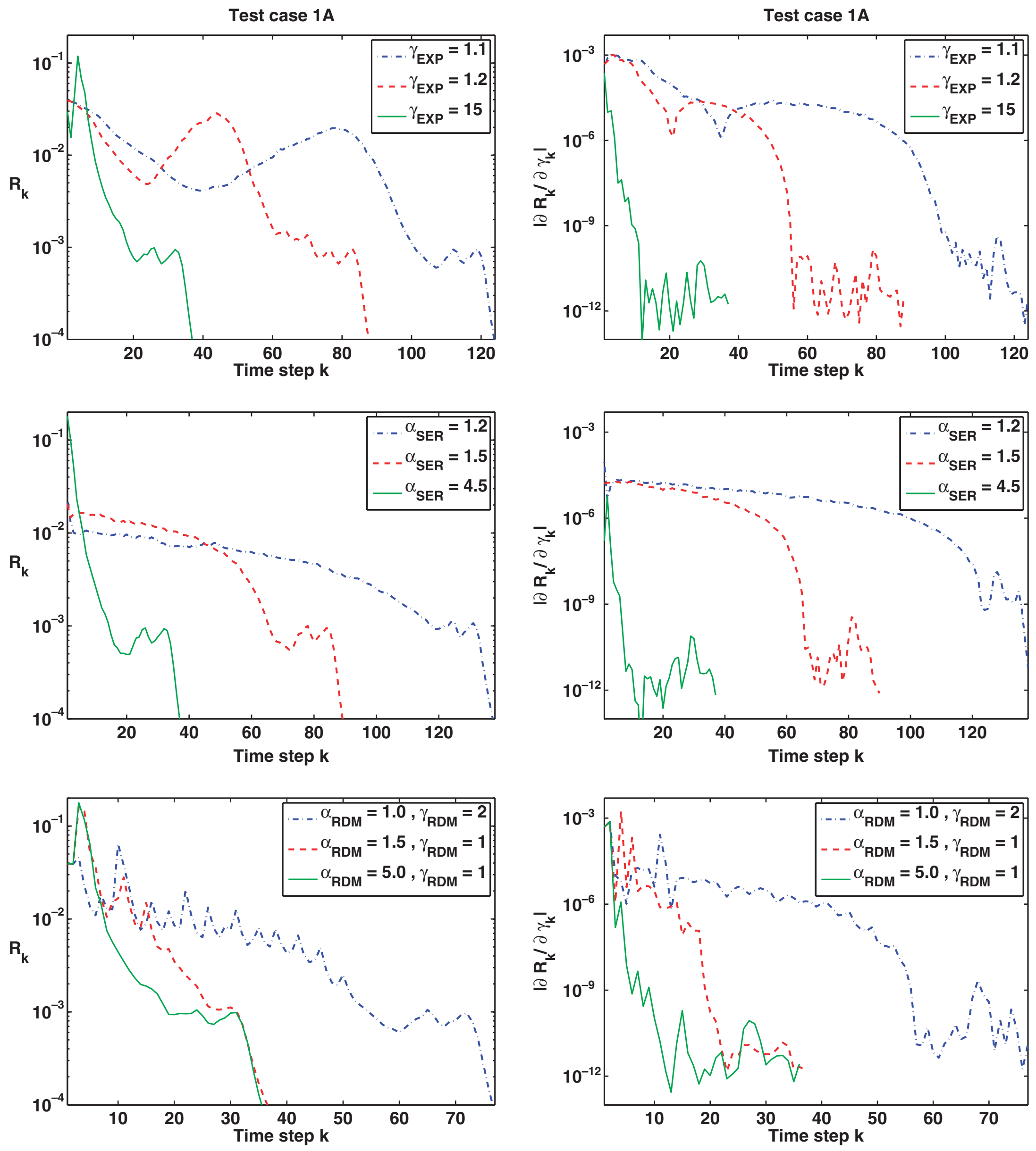

Figure 7. Residuals $R_k(\gamma_k)$ and absolute values of their sensitivities, $|\partial R_k(\gamma_k)/\partial\gamma_k|$, for the three CFL evolution strategies EXP (top row), SER (middle row), and RDM (bottom row), using various control parameters.

early time steps and then decrease as the residual values decrease. This indicates a larger impact of the selected CFL number $\gamma_k$ on the solution in the early iterations, which becomes smaller as the solution converges. This is expected because in the steady-state problems discussed here the choice of the CFL number has virtually no effect on an almost converged solution.

## 7. CONCLUSIONS

The aim of a good CFL evolution strategy is to find the balance between choosing large CFL numbers in order to achieve fast convergence of the implicit time-stepping method and selecting small time steps so that convergence of Newton's method can be guaranteed and all flow features can be resolved. The results of Section 5 show that the best strategy does not have to locally minimize the residuals as much as possible in every time step and that even an increase in the residual must be accepted in order to achieve rapid overall convergence.

A new CFL evolution strategy, called RDM, is introduced, and a comparison of RDM with the existing strategies EXP and SER shows that there is no clear winner. Using the different CFL evolution strategies within an expert system like advocated in [24] may improve this situation, but the feeling that CFL evolution can be improved will remain. Currently, optimal CFL evolution is still an open problem. Application-specific knowledge, intuition, and trial and error are still needed in order to determine appropriate values for the CFL control parameters.

To better understand the impact of the CFL numbers on the residuals, a sensitivity analysis is carried out. As, today, the CFL evolution is still not completely understood, involving numerous different numerical phenomena, a sensitivity analysis based on divided differencing would add another potential source of error. Therefore, we rely on automatic differentiation to evaluate sensitivities without additional truncation error. The analysis confirmed that CFL control is a subtle issue.

## ACKNOWLEDGEMENTS

This work has been performed with funding by the Deutsche Forschungsgemeinschaft (DFG) in the Collaborative Research Center SFB 401 'Flow Modulation and Fluid–Structure Interaction at Airplane Wings' of the RWTH Aachen University, Aachen, Germany. The authors acknowledge the fruitful collaboration with several members of the QUADFLOW research group.

## REFERENCES

1. Knoll DA, Keyes DE. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics* 2004; **193**(2):357–397.
2. Courant R, Friedrichs K. *Supersonic Flow and Shock Waves*. Springer: Berlin, 1978.
3. Brakhage K, Müller S. Algebraic-hyperbolic grid generation with precise control of intersection of angles. *International Journal for Numerical Methods in Fluids* 2000; **33**:89–123.
4. Bramkamp F, Gottschlich-Müller B, Hesse M, Lamby P, Müller S, Ballmann J, Brakhage K, Dahmen W. $H$-adaptive multiscale schemes for the compressible Navier–Stokes equations—polyhedral discretization, data compression and mesh generation. In *Flow Modulation and Fluid–Structure-Interaction at Airplane Wings*, Ballmann J (ed.). Notes on Numerical Fluid Mechanics and Multidisciplinary Design, vol. 84. Springer: Berlin, 2003; 125–204.
5. Bramkamp F, Lamby P, Müller S. An adaptive multiscale finite volume solver for unsteady and steady state flow computations. *Journal of Computational Physics* 2004; **197**(2):460–490.

6. Bramkamp FD. Unstructured $h$-adaptive finite–volume schemes for compressible viscous fluid flow. *Ph.D. Thesis*, RWTH Aachen University, 2003.
7. Luo H, Baum J, Löhner R. A fast, matrix-free implicit method for compressible flows on unstructured grids. *Journal of Computational Physics* 1998; **146**(2):664–690.
8. McHugh PR, Knoll DA. Comparison of standard and matrix-free implementations of several Newton–Krylov solvers. *AIAA Journal* 1994; **32**(12):394–400.
9. Meister A. Comparison of different Krylov subspace methods embedded in an implicit finite volume scheme for the computation of viscous and inviscid flow fields on unstructured grids. *Journal of Computational Physics* 1998; **140**(2):311–345.
10. Meister A, Sonar T. Finite-volume schemes for compressible fluid flow. *Surveys on Mathematics for Industry* 1998; **8**(1):1–36.
11. Meister A, Vömel C. Efficient preconditioning of linear systems arising from the discretization of hyperbolic conservation laws. *Advances in Computational Mathematics* 2001; **14**(1):49–73.
12. Venkatakrishnan V. Implicit schemes and parallel computing in unstructured grid CFD. *Technical Report 95-28*, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, 1995.
13. Moir IRM. Measurements on a two-dimensional aerofoil with high-lift devices. *AGARD-AR-303*: *A Selection of Experimental Test Cases for the Validation of CFD Codes*, vols 1 and 2. Advisory Group for Aerospace Research & Development, Neuilly-sur-Seine, France, 1994.
14. Kelley CT, Keyes DE. Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis* 1998; **35**(2):508–523.
15. Balay S, Buschelman K, Gropp WD, Kaushik D, Knepley MG, McInnes LC, Smith BF, Zhang H. PETSc Web page, 2001. http://www.mcs.anl.gov/petsc.
16. Balay S, Gropp WD, McInnes LC, Smith BF. Efficient management of parallelism in object oriented numerical software libraries. In *Modern Software Tools in Scientific Computing*, Arge E, Bruaset AM, Langtangen HP (eds). Birkhäuser: Basel, 1997; 163–202.
17. Hänel D, Schwane R. An implicit flux–vector splitting scheme for the computation of viscous hypersonic flow. *AIAA Paper 1989-0274*, 1989.
18. Venkatakrishnan V. Convergence to steady state solutions of the Euler equations on unstructured grids with limiters. *Journal of Computational Physics* 1995; **118**(1):120–130.
19. Batten P, Leschziner MA, Goldberg UC. Average-state Jacobians and implicit methods for compressible viscous and turbulent flows. *Journal of Computational Physics* 1997; **137**(1):38–78.
20. van der Vorst HA. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 1992; **13**(2):631–644.
21. Pollul B, Reusken A. Preconditioners for linearized discrete compressible Euler equations. *Technical Report 247*, IGPM, RWTH Aachen University, 2004.
22. Geuzaine P. An implicit upwind finite volume method for compressible turbulent flows on unstructured meshes. *Ph.D. Thesis*, Université de Liège, 1999.
23. Issman E, Degrez G, Deconinck H. Implicit upwind residual–distribution Euler and Navier–Stokes solver on unstructured meshes. *AIAA Journal* 1996; **34**(10):2021–2028.
24. Vanderstraeten D, Csík A, Rose D. An expert-system to control the CFL number of implicit upwind methods. *Technical Report TM 304*, Universiteit Leuven, 2000.
25. Mulder W, van Leer B. Experiments with implicit upwind methods for the Euler equations. *Journal of Computational Physics* 1985; **59**(2):232–246.
26. Delanaye M. Polynomial reconstruction finite volume schemes for the compressible Euler– and Navier–Stokes equations on unstructured adaptive grids. *Ph.D. Thesis*, Université de Liège, 1996.
27. Jones DJ. Reference test cases and contributors. *AGARD-AR-211*: *Test Cases for Inviscid Flow Field Methods*. Advisory Group for Aerospace Research & Development, Neuilly-sur-Seine, France, 1986.
28. Ballmann J. Flow modulation and fluid–structure-interaction at airplane wings—survey and results of the Collaborative Research Center SFB 401. *DGLR 2002-009*, 2002.
29. Ballmann J (ed.). *Flow Modulation and Fluid–Structure-Interaction at Airplane Wings*. Notes on Numerical Fluid Mechanics and Multidisciplinary Design, vol. 84. Springer: Berlin, 2003.
30. Griewank A. *Evaluating Derivatives*: *Principles and Techniques of Algorithmic Differentiation*. SIAM: Philadelphia, PA, 2000.
31. Rall LB. *Automatic Differentiation*: *Techniques and Applications*. Lecture Notes in Computer Science, vol. 120. Springer: Berlin, 1981.

32. Berz M, Bischof C, Corliss G, Griewank A (eds). *Computational Differentiation*: *Techniques*, *Applications*, *and Tools*. SIAM: Philadelphia, 1996.
33. Bücker HM, Corliss GF, Hovland PD, Naumann U, Norris B (eds). *Automatic Differentiation*: *Applications*, *Theory*, *and Implementations*. Lecture Notes in Computational Science and Engineering, vol. 50. Springer: Berlin, 2005.
34. Corliss G, Faure C, Griewank A, Hascoët L, Naumann U (eds). *Automatic Differentiation of Algorithms*: *From Simulation to Optimization*. Springer: New York, 2002.
35. Griewank A, Corliss G. *Automatic Differentiation of Algorithms*. SIAM: Philadelphia, 1991.
36. Bischof C, Carle A, Khademi P, Mauer A. Adifor 2.0: automatic differentiation of Fortran 77 programs. *IEEE Computational Science and Engineering* 1996; **3**(3):18–32.
37. Bischof CH, Bücker HM, Hovland PD. On combining computational differentiation and toolkits for parallel scientific computing. In *Euro-Par 2000—Parallel Processing*, *Proceedings of the 6th International Euro-Par Conference*, Bode A, Ludwig T, Karl W, Wismüller R (eds), Munich, Germany, August/September 2000. Lecture Notes in Computer Science, vol. 1900. Springer: Berlin, 2000; 86–94.