Journal of Computational Physics 229 (2010) 3938–3956

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A hybrid multilevel method for high-order discretization of the Euler equations on unstructured meshes

Georg May [a,*], Francesca Iacono [a], Antony Jameson [b]

[a] Aachen Institute for Advanced Study in Computational Engineering Science, RWTH Aachen University, Schinkelstr. 2, 52062 Aachen, Germany
[b] Department of Aeronautics and Astronautics, Stanford University, Durand Building, 496 Lomita Mall, Stanford, CA 94305-4035, USA

ARTICLE INFO

Article history:
Received 23 June 2009
Received in revised form 25 January 2010
Accepted 25 January 2010
Available online 4 February 2010

Keywords:
Multigrid methods
Newton–Krylov methods
Euler equations
Spectral Difference scheme
Nodal schemes

ABSTRACT

Higher order discretization has not been widely successful in industrial applications to compressible flow simulation. Among several reasons for this, one may identify the lack of tailor-suited, best-practice relaxation techniques that compare favorably to highly tuned lower order methods, such as finite-volume schemes. In this paper we investigate solution algorithms in conjunction with high-order Spectral Difference discretization for the Euler equations, using such techniques as multigrid and matrix-free implicit relaxation methods. In particular we present a novel hybrid multilevel relaxation method that combines (optionally matrix-free) implicit relaxation techniques with explicit multistage smoothing using geometric multigrid. Furthermore, we discuss efficient implementation of these concepts using such tools as automatic differentiation.

1. Introduction

Many have anticipated the arrival of high-order discretization as the CFD method of choice for compressible fluid flow. However, for industrial applications in external aerodynamics lower order methods, such as finite-volume schemes, are still by far more popular, while numerical schemes of third or higher spatial order are often not efficient enough. Among the reasons for this is the fact that for established CFD methodologies tailor-suited convergence acceleration techniques have emerged over the past decades [1–5]. High-order solvers thus compete with very mature technology, and novel discretization techniques have to be augmented by extremely efficient solutions algorithms. For compressible flow simulation using finite-volume methods two paradigms have become quite popular over the years. Firstly, nonlinear (FAS) geometric multigrid methods, based on mesh coarsening, have become a standard approach in CFD, often combined with multistage relaxation methods and nonlinear preconditioning techniques [1,3]. Secondly, implicit relaxation methods, often using Newton–Krylov methods, are frequently used. Often these approaches are viewed as complementary [6], but attempts have been made to combine the two, for example using the former to precondition the latter [5].

For higher order discretization techniques, the paradigms change. Stability restrictions become a major concern for multigrid methods using explicit multistage relaxation, even on non-stretched meshes, as permissible CFL numbers of high-order methods typically behave as CFL $\propto p^{-2}$, where p is the polynomial degree of approximation [7]. Furthermore, the direct extension of nonlinear multigrid methods to higher order schemes in a FAS p-multigrid approach has not always been found to be very efficient [8]. To understand this, one should recall that for convection-dominated problems, not only elimination of high-frequency error modes on successively coarser meshes plays a role for multigrid convergence, but also accelerated

* Corresponding author.
E-mail address: may@aices.rwth-aachen.de (G. May).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.01.036

propagation of error modes, and expulsion from the computational domain [1,3]. For the latter, nonlinear p-multigrid cannot contribute very much, due to lack of global coarsening, and only marginally increased permissible time steps on lower levels of approximation. Furthermore, the workload decreases much more slowly in p-multigrid, compared to mesh coarsening. On the other hand, Newton–Krylov methods suffer from drawbacks as well, such as excessive storage requirements for high-order of approximation. A possible remedy is given by matrix-free Newton–Krylov methods [6].

The aim of the present work is to produce an algorithm which is tailor-suited for nonlinear convection-dominated problems using higher order discretization methods. The main focus is on investigation of (optionally matrix-free) Newton–Krylov methods in combination with nonlinear geometric multigrid methods. Here we use the FAS multigrid method as an accelerator for propagation of error modes between Newton iterations acting only on the volume averages of the current solution. Alternatively this may be viewed as a preconditioning procedure for the nonlinear Newton method. The rationale is to maximize the benefit of using nonlinear multigrid algorithms for propagation of error modes with minimal computational overhead. Numerical experiments indicate that with this approach mesh-independent convergence can be obtained at finite CFL numbers and only marginally increased computational cost, compared to standard implicit relaxation methods.

The paper is organized as follows. In Section 2, we introduce the governing equations and high-order spatial discretization, which is implemented using the Spectral Difference scheme [9–11]. We consider two-dimensional compressible fluid flow on triangular unstructured grids governed by the Euler equations. In Section 3, we discuss relevant relaxation techniques. In particular, we introduce the proposed hybrid multilevel method using a combination of (matrix-free) Newton–Krylov methods with nonlinear geometric multigrid. Focus is also on efficient implementation of solution algorithms. In particular during development of new solution methodologies, one desires a flexible implementation that lets the user combine different building blocks, such as numerical flux functions, approximations to Jacobian matrices, and preconditioning, without extensive re-derivation or re-implementation of code. We use such tools as efficient numerical libraries and automatic differentiation (AD) to obtain a flexible platform for high-order CFD computations. This is discussed in Section 4. In Section 5, we present numerical experiments for applications in external aerodynamics for smooth compressible flow governed by the Euler equations. Extensions to non-smooth flow, and advection–diffusion systems will be considered in a future publication.

2. Governing equations and spatial discretization

For inviscid compressible flow one seeks solutions to the Euler equations satisfying

$$\frac{\partial w}{\partial t}+\sum_{j=1}^{d}\frac{\partial f_j}{\partial x_j}=0, \tag{1}$$

where for two-dimensional problems, i.e. $d=2$, the vector of conservative variables w and the flux vectors f_j are given by

$$w=\begin{pmatrix}\rho\\ \rho u_1\\ \rho u_2\\ E\end{pmatrix},\quad f_j=\begin{pmatrix}\rho u_j\\ \rho u_j u_1+p\delta_{j1}\\ \rho u_j u_2+p\delta_{j2}\\ \rho u_j H\end{pmatrix},\quad j=1,2. \tag{2}$$

Here ρ is the density, p is the pressure, E is the energy, and $H=(E+p)/\rho$ is the enthalpy. The fluid velocity vector is given by $u=(u_1,u_2)^T$. We shall assume a thermally and calorically perfect gas, and hence close the equations by the equation of state

$$p=(\gamma-1)\left(E-\frac{1}{2}\rho\|u\|^2\right), \tag{3}$$

where $\gamma=1.4$ is the ratio of specific heats for air.

We use the Spectral Difference scheme [9–13] to discretize the spatial operator of the governing equations (1). The Spectral Difference scheme has been proposed as a collocation-based method with the aim to achieve efficiency by avoiding volume and surface quadratures, while maintaining conservation, and extending tensor-product-based collocation approaches that had previously been formulated for quadrilateral meshes [14] to more general unstructured-grid elements.

Consider a triangulation $\mathcal{T}_h=\{T^{(i)},\ i=1,\ldots,N_T\}$ on some domain $\Omega\subset\mathbb{R}^2$, such that $\overline{\Omega}_h=\bigcup_{i=1}^{N_T}\overline{T^{(i)}}$. Throughout this paper we shall use bracketed superscripts to denote mesh element indices, except where otherwise noted. Assume that there exist mappings $\Phi^{(i)}:\xi\to x$, with nonsingular Jacobian $J^{(i)}=\partial x/\partial\xi$, such that each element in the triangulation can be mapped to a reference domain $T^{(i)}=\Phi^{(i)}(\widehat{T})$. For simplicity we consider straight-sided triangular elements in this section, although such a restriction is not necessary. (The implementation used in our solver allows for elements with curved edges.) Let α be a multi-index, and let $\mathcal{P}^p(\widehat{T})=\text{span}\{\xi^\alpha:\xi\in\widehat{T},\ \alpha_i\geqslant 0,\ |\alpha|\leqslant p\}$ be the space of polynomials of total degree p. For the Spectral Difference scheme we seek solutions w_h, such that for each triangle $w_h|_{T^{(i)}}\circ\Phi^{(i)}\in[\mathcal{P}^p(\widehat{T})]^4$. The dimension of the space $\mathcal{P}^p$ is given by

$$N_p=\frac{(p+1)(p+2)}{2}. \tag{4}$$

As a particular basis for $\mathcal{P}^p(\widehat{T})$ consider the fundamental polynomials L_k of multivariate Lagrangian interpolation, corresponding to a set of interpolation nodes $\mathcal{S}_p = \{\xi_j,\ j = 1, \ldots, N_p\}$, such that $L_k(\xi_j) = \delta_{jk}$ for $\xi_j \in \mathcal{S}_p$. The solution may be represented as

$$w_h|_{T^{(i)}} = \sum_{j=1}^{N_p} w_j^{(i)} L_j(\xi), \tag{5}$$

where $w_j^{(i)} := w\left(x_j^{(i)}\right)$, with $x_j^{(i)} := \Phi^{(i)}(\xi_j)$ and $\xi_j \in \mathcal{S}_p$. The representation (5) is understood to hold componentwise. Furthermore, the nonlinear fluxes in (1) are projected onto a similar space using interpolation of order $p+1$, and a nodal set $\mathcal{Q}_{p+1} = \{\xi_j,\ j = 1, \ldots, N_{p+1}\}$ with $\overset{\circ}{L}_j$ the corresponding fundamental polynomials, such that

$$\tilde{f}_h|_{T^{(i)}} = \sum_{j=1}^{N_{p+1}} \tilde{f}_j^{(i)} \overset{\circ}{L}_j(\xi). \tag{6}$$

The degrees of freedom are computed for straight-sided elements as

$$\tilde{f}_k^{(i)} = \begin{cases} [J^{(i)}]^{-1} f\left(w_k^{(i)}\right), & \xi_k \in \widehat{T}, \\ [J^{(i)}]^{-1} f^{num}, & \xi_k \in \partial\widehat{T}, \end{cases} \tag{7}$$

where $\xi_k \in \mathcal{Q}_{p+1}$. The coefficients f^{num} are chosen such that $f^{num} \cdot n = h$, where h is a standard numerical flux function. In our implementation the numerical flux function h is given by Jameson's CUSP flux [15]. This definition ensures conservativity of the scheme.

The scheme for the jth degree of freedom in the ith cell may be written

$$\frac{dw_j^{(i)}}{dt} + \sum_{k=1}^{N_{p+1}} (\nabla^{\xi} \overset{\circ}{L}_k)|_{\xi_j} \cdot \tilde{f}_k = 0. \tag{8}$$

We use Hesthaven's interpolation nodes [16] for the nodal set $\mathcal{Q}_{p+1}$. The scheme as described above is independent of the choice of solution nodes (cf. [13]), due to the fact, that the differentiated polynomial approximation of the flux function is by construction a polynomial of degree p, and hence represented exactly on any unisolvent set of interpolation nodes.

The flux-based formulation of the Spectral Difference method facilitates implementation of boundary conditions, in that standard ghost-cell methods may be used to satisfy boundary conditions that are frequently used in compressible CFD computations, such as wall and farfield boundary conditions [17]. The same algorithm for flux computation may thus be used for boundary faces and interior faces.

Again, the restriction to straight-sided elements is not necessary. Curved edges are usually permitted in the vicinity of curved boundaries of the domain Ω, in order to avoid spurious entropy generation by polygonal approximation. In our implementation we use cubic spline interpolation of curved boundaries, and define isoparametric elements based on that approximation of the surface [12]. For such elements, the Jacobian of the mapping from reference domain to the physical domain is not constant over a triangle anymore. All metric terms relating to the transformation $\Phi^{(i)}$ are evaluated, and need to be stored for each internal node in the triangle.

3. Relaxation techniques

We focus here on the solution of steady-state problems. By the spatial discretization outlined in Section 2 the discrete equations satisfy at steady state a set of coupled algebraic equations

$$R(w_h) = 0, \tag{9}$$

where $R(w_h)$ is the residual vector, with entries given by the second term in Eq. (8). The core of the solution methodology pursued here is a pseudo time-dependent relaxation, marching the field equations to a steady state in a method of lines approach. This means one considers the system of nonlinear ODE

$$\frac{dw_h}{d\tau} + R(w_h) = 0. \tag{10}$$

Obviously no time accuracy is required if one wishes to iterate toward the steady state, allowing such convergence acceleration techniques as local time stepping and multigrid methods.

Since in the following we only deal with the numerical solution w_h, we drop the subscript h by default, and use it only where necessary to distinguish between different mesh levels.

3.1. Explicit multistage methods

Explicit temporal discretization techniques of the Runge–Kutta type may be used for the ODE (10). Let w^n be the nth iterate of the solution, and consider a scheme of the form

Table 1
Linear stability limits for Spectral Difference (SD) and Discontinuous Galerkin (DG) Schemes with Runge–Kutta Time Stepping using the Shu RK3 scheme, and Jameson's four-stage scheme [1]. The right-most column displays the CFL number multiplied with the local degrees of freedom (DOF), which is appropriate when making comparison with standard finite difference schemes using the same number of total degrees of freedom.

Order	CFL	CFL · DOF
(a) SD/Jameson RK4		
1	0.696	1.392
2	0.363	1.089
3	0.226	0.904
4	0.156	0.780
5	0.115	0.690
6	0.089	0.623
(b) SD/Shu-RK3		
1	0.595	1.190
2	0.322	0.966
3	0.201	0.804
4	0.139	0.695
5	0.103	0.618
6	0.0799	0.559
(c) DG/Shu-RK3		
1	0.409	0.818
2	0.209	0.627
3	0.130	0.520
4	0.089	0.445
5	0.066	0.396
6	0.051	0.357

$$\begin{aligned} w^{(0)} &= w^n, \\ w^{(k)} &= \sum_{l=0}^{k-1}\{\alpha_{kl}w^{(l)} - \Delta\tau\beta_{kl}R^{(l)}\}, \quad k = 1,\ldots,K, \\ w^{n+1} &= w^{(K)}, \end{aligned} \tag{11}$$

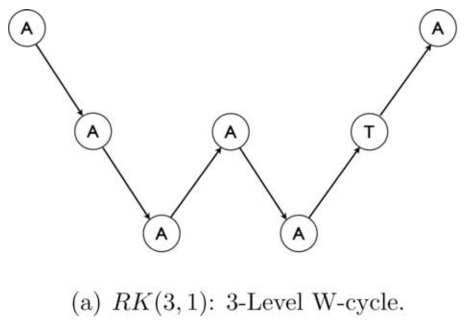
where $R^{(l)} := R(w^{(l)})$. (Here the bracketed superscripts denote stage iterates, not cell indices.) In particular we have used Shu's three-stage scheme [18], which has been designed to preserve TVD properties [19] of the spatial operator. Such TVD properties have been shown for the Spectral Difference scheme with explicit time stepping using standard limiting methods [12]. The coefficients of the scheme may be written, arranged in matrix form, as

$$\alpha = \begin{pmatrix} 1 & & \\ \frac{3}{4} & \frac{1}{4} & \\ \frac{1}{3} & 0 & \frac{2}{3} \end{pmatrix}, \quad \beta = \begin{pmatrix} 1 & & \\ 0 & \frac{1}{4} & \\ 0 & 0 & \frac{2}{3} \end{pmatrix}. \tag{12}$$

While explicit relaxation methods are attractive due to ease of implementation and parallelization, stability restrictions are a concern. Linear and nonlinear stability properties of the Spectral Difference scheme with explicit methods have been analyzed in [12,13]. The spectrum of the discrete differentiation operator is proportional to p^2, where p is the polynomial order of approximation [7], which suggests that stability for explicit methods necessitates CFL $\sim p^{-2}$. This has been confirmed for the one-dimensional linear advection equation, and permissible CFL numbers have been explicitly computed [12], see Table 1. Here the Gauss–Legendre quadrature nodes corresponding to polynomials of degree p were used for flux interpolation, augmented by the two interval endpoints. This allows flux interpolation of degree $p+1$, which leads to solution polynomials of degree p.[1]

It has to be stressed that many combinations of explicit time integration methods with higher order schemes, such as the Spectral Difference scheme, or standard RK Discontinuous Galerkin schemes [20] are not unconditionally linearly stable [12]. This point is sometimes masked by the more popular focus on nonlinear TVD stability estimates that require limiting techniques for stabilization, and thus may locally degrade the accuracy, even in smooth regions, if oscillations are generated by linear instability. For example, for the 1D Spectral Difference scheme, the popular Chebyshev–Lobatto nodes are not unconditionally stable [12,13] (and by extension tensor-products thereof). Linear instability has also been shown for the Spectral Difference scheme using different nodal sets for multivariate interpolation on triangular meshes [13]. In our experience, nonlinear equations can be stabilized quite effectively with standard filtering methods for nodal schemes [21], and limiting techniques [12], although in the present work we have focused on computations that do not require stabilization such as limiting or filtering. In any case, restrictive stability conditions of the order depicted in Table 1 always apply for explicit relaxation methods.

[1] Note that in [12] these nodes had been erroneously labelled Legendre–Lobatto nodes.



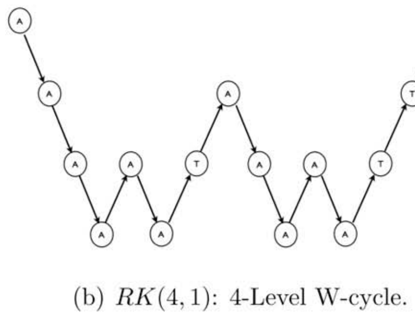


Fig. 1. Standard definition of *W*-cycle. The letter *A* stands for advancing the flow solution on a particular level, while the letter *T* stands for transfer of the solution to the next higher level. Concatenating multigrid cycles *n* times defines the *RK*(*m*,*n*) smoothing procedure for the current finest mesh, where *m* is the number of meshes.

3.2. Nonlinear multigrid methods

For the special case $w_h \in \mathcal{P}^0$, for which the spectral difference scheme reduces to a finite-volume method, one may, following Jameson [1], use geometric multigrid techniques combined with explicit Runge–Kutta methods, under the paradigm of the FAS methodology [22]. Assume that the equations have been iterated n steps on a given mesh of characteristic length h, the "fine" mesh, resulting in an approximation w_h^n, and residual $R_h^n = R_h(w_h^n)$. Using a suitable coarser mesh of characteristic length H, and defining appropriate transfer operators for the solution and residual, $I_h^H w_h^n$, and $\widetilde{I}_h^H R_h^n$, respectively, one may advance the solution on a coarse grid by the modified multistage scheme

$$w_H^{(k)} = \sum_{l=0}^{k-1} \left\{ \alpha_{kl} w_H^{(l)} + \Delta\tau \beta_{kl} \left(R_H^{(l)} + S_H \right) \right\}, \quad k = 1, \ldots, K, \tag{13}$$

where the additional defect correction term

$$S_H = \widetilde{I}_h^H R_h^n - R_H^{(0)} \tag{14}$$

appears on the right-hand side [1,5]. After relaxing on the coarse mesh for q iterations the corrected solution on the fine grid is computed as

$$w_h^{n+1} = w_h^n + I_H^h (w_H^q - w_H^0), \tag{15}$$

where I_H^h is an interpolation operator. Note that in the present work we always use $q = 1$: we advance the solution only once on each mesh level, before transfer takes place.

One uses recursive application of this concept to extend the method to more than two meshes, where we use *W* cycles following standard nomenclature, see e.g. [17]. These are defined by allowing transfer to the next higher level only, if the current coarser mesh has been already visited twice in the multigrid cycle. Fig. 1 shows a schematic depiction for 3 and 4 level multigrid. For easy reference we denote by $RK(m,n)$ the application of n concatenated *W* cycles using m mesh levels with Runge–Kutta smoothing. Sometimes we use the notation $w^n = RK(w^0; m, n)$ if explicit reference to the start value and the output value is required.

In the present work non-nested meshes have been used. Coarser meshes are always generated by new triangulation during the pre-processing stage, either externally by a third-party mesh generator, or internally within our solver.[2] The transfer operators for non-nested meshes given in [23] have been used.

Using this methodology with discretization of locally higher order, however, is not necessarily straight forward. As outlined in Section 1, for nonlinear hyperbolic equations, multigrid aids through two mechanisms: firstly, the elimination of high-frequency error modes on successively coarser meshes; secondly, the propagation of error modes, and expulsion from the computational domain [3]. Typically, analysis of multigrid schemes is done via Fourier methods for linear or linearized equations [24,25]. The convergence rates obtained are only asymptotic, and indeed for convection-dominated problems, early convection-dominated convergence rates typically exceed the asymptotic convergence rates dominated by high-frequency smoothing. Often one observes effectively converged output functionals, such as lift and drag coefficients at relatively high levels of residuals before asymptotic convergence rates are reached.

During the early transient, multigrid may be viewed primarily as an increase of the effective wave speed propagating these large-scale error modes, which is, however, dependent on global coarsening, such as geometric multigrid. Using

[2] Our solver is capable of generating triangular meshes for such shapes that can be mapped to a circle, e.g. common airfoils.

FAS-based nonlinear p-multigrid methods only provides local coarsening at marginally reduced computational expense. This serves as motivation for the multilevel strategy outlined below, where we only use geometric multigrid on the volume-averaged solution to accelerate convective convergence, wrapped inside a Newton–Krylov algorithm.

3.3. Implicit relaxation

A backward Euler temporal discretization of (10) may be written

$$(I + \Delta\tau D_w R|_{w^n})\Delta w^n = -\Delta\tau R(w^n), \tag{16}$$

where $\Delta w^n = w^{n+1} - w^n$ and D_w is the Jacobian matrix of the residual vector, i.e. the total differentiation of the residual vector R with respect to the state vector w. For $\Delta\tau \to \infty$ one obtains a Newton iteration, while finite time steps may be interpreted as damped Newton iterations, or in the context of pseudo-transient continuation methods [6]. Time steps are set proportional to an approximation to the spectral radius of the convective operator, where the constant of proportionality is the CFL number of the implicit scheme. For easy reference we define the application of n damped Newton iterations, Eq. (16), acting on a spatial discretization of polynomial order p, in conjunction with a GMRES solver for the linear system, as $NK(p,n)$, where NK stands for *Newton–Krylov*. We shall sometimes use the notation $w_h^n = NK(w_h^0; p, n)$ when we need explicit reference to the start value, and the output value.

For a sizable part of the transient finite CFL numbers have to be used. If no other start-up procedures are used, a simple ramping method may serve to reach a nominal CFL number:

$$\mathrm{CFL}(z) = \begin{cases} z^2(3-2z)\mathrm{CFL}_{nom}, & z < 1, \\ \mathrm{CFL}_{nom}, & \text{otherwise}, \end{cases} \tag{17}$$

with $z = \frac{N_{cyc}}{N_{start}}$, where N_{cyc} is the nonlinear iteration counter, and N_{start} is a fixed parameter, usually $N_{start} < 10$. Best-practice methods considered here, however, rely on a full multigrid approximation as a start-up method, requiring very little initial attenuation of the CFL number on the highest level of approximation, see Section 3.4. Once the nominal CFL number is reached, it may be further controlled by an adaptive procedure, and may ultimately be increased to infinity with the potential of reaching quadratic convergence if an exact Newton method is used. In this paper, however, we shall focus primarily on the transient solution, where finite CFL numbers are used, with the aim to demonstrate our goal of achieving mesh-independent convergence in this region through use of multigrid. Furthermore, we do not use exact Newton methods, as the computational expense of solving the arising linear systems exactly is typically not warranted if one tries to minimize time-to-solution rather than iteration count.

The key parameters in the implementation are the approximation and assembly of the Jacobian matrix D_wR, the solution methodology for the linear equations arising at each nonlinear iteration step n, and preconditioning of the system. For the linear solver we have used the GMRES method [26]. In particular the “flexible GMRES” formulation [27] has been implemented, which allows the preconditioning matrix to depend on the GMRES iteration number. The restarted flexible GMRES version with left-preconditioning approach may be written for a general linear system $Ax = b$ as follows:

Algorithm 3.1. Flexible GMRES(m) with Left Preconditioning

```
 1. Compute $s_0 = P_0^{-1}(b - Ax_0)$, $\beta = \|s_0\|_2$, and $v_1 = \frac{s_0}{\beta}$
 2. For $j = 1, \dots, m$, Do
 3.     Compute $z_j = Av_j$ and $w := P_j^{-1} z_j$
 4.     For $i = 1, \dots, j$, Do
 5.         $h_{i,j} := (w, v_i)$
 6.         $w := w - h_{i,j} v_i$
 7.     EndDo
 8.     Compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = \frac{w}{h_{j+1,j}}$
 9. EndDo
10. Define $V_m := [v_1, \dots, v_m]$, $\overline{H}_m = \{h_{i,j}\}_{1 \leqslant i \leqslant j+1, 1 \leqslant j \leqslant m}$
11. Compute $y_m = \arg\min_y \|\beta e_1 - \overline{H}_m y\|_2$ and $x_m = x_0 + V_m y_m$
12. If satisfied Stop, else set $x_0 := x_m$ and go to 1
```

The reason for this choice, besides favorable convergence properties, are twofold. Firstly, one needs the matrix only in products with known vectors. This facilitates the implementation of matrix-free methods considerably, cf. Section 3.3.2. Secondly, the flexible GMRES formulation also facilitates implementation of matrix-free preconditioning by allowing the preconditioning matrix to depend on the iteration index j. We return to the issue of preconditioning in Section 3.3.3.

For the present work, both matrix storage and matrix-free approaches are considered. We outline the former approach first.

3.3.1. Explicit assembly and storage of the exact Jacobian matrix

One may obtain the Jacobian of the residual by exact differentiation with respect to the solution vector $w \in \mathbb{R}^{N_{dof}}$ where $N_{dof} = N_w N_p N_e$ is the number of degrees of freedom. Here N_w is the number of equations ($N_w = d+2$, in the case of the d-dimensional Euler equations), N_p is the number of local degrees of freedom, see Eq. (4) for the case $d = 2$, and N_e is the number of mesh elements.

An advantage of the Spectral Difference scheme is a relatively simple structure of the Jacobian, which we illustrate here for general $d > 1$. The global system is very sparse, in a structure determined by the coupling of the mesh elements. Let $\mathbb{M}$ be the iteration matrix $I + \Delta\tau D_w R|_{w^n}$ in Eq. (16), which may be depicted schematically as

$$\mathbb{M} = I_{N_e \times N_p \times (d+2)} + \Delta\tau \left.\overbrace{\begin{pmatrix} \blacksquare & \square & \blacksquare & \square & \blacksquare & \square & \cdots & \square & \square & \square \\ \square & \blacksquare & \blacksquare & \square & \square & \square & \cdots & \blacksquare & \square & \square \\ \square & \square & \blacksquare & \square & \blacksquare & \square & \cdots & \square & \square & \square \\ \square & \square & \square & \blacksquare & \square & \square & \cdots & \square & \square & \square \\ \blacksquare & \square & \square & \square & \blacksquare & \square & \cdots & \blacksquare & \square & \square \\ \blacksquare & \square & \blacksquare & \square & \square & \blacksquare & \cdots & \square & \square & \square \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \square & \square & \blacksquare & \square & \blacksquare & \square & \cdots & \blacksquare & \square & \square \\ \square & \square & \square & \square & \blacksquare & \square & \cdots & \square & \blacksquare & \square \\ \square & \square & \square & \square & \square & \blacksquare & \cdots & \blacksquare & \square & \blacksquare \end{pmatrix}}^{N_e}\right\} N_e. \tag{18}$$

The symbols $\blacksquare$ are non-null block entries of the Jacobian $D_w R$, reflecting the coupling between mesh elements. In turn each such block is a dense matrix of size $N_p \cdot (d+2) \times N_p \cdot (d+2)$ representing the coupling of the degrees of freedom inside each cell. This may be written

$$\blacksquare^{(i,r)} = \left.\overbrace{\begin{pmatrix} \frac{\partial R_1^{(i)}}{\partial w_1^{(r)}} & \frac{\partial R_1^{(i)}}{\partial w_2^{(r)}} & \cdots & \frac{\partial R_1^{(i)}}{\partial w_{N_p}^{(r)}} \\ \frac{\partial R_2^{(i)}}{\partial w_1^{(r)}} & \frac{\partial R_2^{(i)}}{\partial w_2^{(r)}} & \cdots & \frac{\partial R_2^{(i)}}{\partial w_{N_p}^{(r)}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial R_{N_p}^{(i)}}{\partial w_1^{(r)}} & \frac{\partial R_{N_p}^{(i)}}{\partial w_2^{(r)}} & \cdots & \frac{\partial R_{N_p}^{(i)}}{\partial w_{N_p}^{(r)}} \end{pmatrix}}^{N_p}\right\} N_p, \tag{19}$$

where again superscripts refer to cell indices, while subscripts refer to local interpolation nodes. Each entry $\frac{\partial R_j^{(i)}}{\partial w_q^{(r)}}$ is a matrix of size $(d+2) \times (d+2)$. For arbitrary cell indices r and i one has

$$\frac{\partial R_j^{(i)}}{\partial w_q^{(r)}} = \begin{cases} \sum\limits_{l=1}^{d} \sum\limits_{k=1}^{N_{p+1}} d_{j,k,l} \mathbb{A}_{k,l}^{(i)} l_{k,q}, & \text{if } r = i, \\ \sum\limits_{l=1}^{d} \sum\limits_{k \in e_{ir}} d_{j,k,l} \mathbb{B}_{k,l}^{(i,r)} l_{k,q}, & \text{if } T^{(r)} \text{ is a neighbor of } T^{(i)}, \\ 0, & \text{otherwise}, \end{cases} \quad 1 \leqslant j,\ q \leqslant N_p \tag{20}$$

where e_{ir} denotes the set of indices of local flux collocation nodes in cell i that lie on the edge separating cell i from cell r. Each $d_{j,k,l}$ is an element of the local differentiation matrix $d_{j,k,l} = \partial_l \dot{L}_k|_{\xi_j}$, where $\xi_j \in \mathcal{S}_p$. (Refer to Section 2 for nomenclature.) The global reconstruction coefficients $l_{k,q}$ are needed to evaluate the solution at the flux collocation points $\xi_k \in \mathcal{Q}_{p+1}$, and are given by $l_{k,q} = L_q(\xi_k)$. The $(d+2) \times (d+2)$ matrices $\mathbb{A}_{k,l}^{(i)}$ and $\mathbb{B}_{k,l}^{(i,r)}$ are given by

$$\mathbb{A}_{k,l}^{(i)} := \frac{\partial \tilde{f}_{k,l}^{(i)}}{\partial w_k^{(i)}}, \quad l = 1, \ldots, d, \quad 1 \leqslant i \leqslant N_e, \tag{21}$$

$$\mathbb{B}_{k,l}^{(i,r)} := \frac{\partial \tilde{f}_{k,l}^{(i)}}{\partial w_k^{(r)}}, \quad l = 1, \ldots, d, \quad 1 \leqslant i,\ r \leqslant N_e \text{ with } r \neq i. \tag{22}$$

Thus each block entry of the iteration matrix may be computed as a simple superposition of local (numerical) flux Jacobians, using the local reconstruction and differentiation coefficients similarly to computing the residual. One can appreciate that the assembly of the Jacobian is relatively straightforward, with the only possible complication resulting from the differentiation of the numerical flux (or approximation thereto). This will be addressed in Section 4.

Storing all non-null blocks of the matrix $\mathbb{M}$ shall be referred to as "matrix-explicit" here. Computationally this is quite efficient, since the overhead of assembling the matrix is mitigated by firstly the efficiency of the sparse matrix–vector product using the stored matrix elements, and secondly the possibility of freezing the Jacobian for a number of Newton iterations. However, there is a considerable memory penalty. In the matrix-explicit approach the memory requirement grows as N_p^2. The local degrees of freedom grow as

$$N_p = \frac{\prod_{l=1}^{d}(p+l)}{d!} \sim p^d, \tag{23}$$

which is the d-dimensional analogue of Eq. (4). In two dimensions the overall storage requirement thus increases as p^4, whereas in three dimensions we have $N_p \propto p^3$, and thus the required memory grows as p^6. One can easily foresee situations where the memory requirement becomes prohibitive, or at least necessitates the use of massively parallel computations to distribute memory, when the problem could otherwise be handled with more moderate computational resources. This has motivated the use of matrix-free Krylov methods.

3.3.2. Numerical approximation of Krylov vectors

In order to produce a new Krylov vector in Algorithm 3.1, one needs to compute $(D_w R)v$, which is a projection of the total derivative of the residual onto the Krylov vector v. One may generate a numerical approximation to first order accuracy in a small parameter ε by writing

$$(D_w R)v \approx \frac{R(w+\varepsilon v) - R(w)}{\varepsilon}. \tag{24}$$

Here the cost of applying the matrix–vector product is the same as one residual evaluation. There is some freedom in the choice of ε. Several methods have been proposed in the literature to estimate the step size [6]. Out of these we adopt a rather simple method, which may be written, supposing normalized Krylov vectors:

$$\varepsilon = \sqrt{1+\|w\|}\varepsilon_{rel}, \tag{25}$$

where the parameter ε_{rel} should roughly represent the square-root of machine accuracy, and $\varepsilon_{rel} = 10^{-6}$ was used in our implementation. We shall see below that this simple choice yields good results.

3.3.3. Preconditioning

Fully matrix-explicit methods using GMRES to solve the linear systems may be preconditioned using incomplete LU factorization [28], denoted $ILU(n)$ where n stands for the level of additional fill allowed in the incomplete factorization. We have used $ILU(n)$ preconditioning for all matrix-explicit methods. (In this case the preconditioning matrix does not depend on the Krylov iteration number j.)

The primary obstruction against fully matrix-free implicit methods is the need to precondition the linear systems. The flexible GMRES formulation facilitates the implementation of matrix-free preconditioning. Note that the preconditioning step is contained in step 3 of Algorithm 3.1. One generates a preconditioned Krylov vector by solving the linear system

$$w = P_j^{-1} z_j, \tag{26}$$

where $z_j = Av_j$. Since the preconditioning matrix is allowed to depend on the GMRES iteration index, one may use iterative solvers as preconditioners. Noting that the preconditioning matrix P should approximate A in Algorithm 3.1, one may apply a few iteration steps with the same GMRES method to solve the preconditioning equation (26), i.e. apply Algorithm 3.1 recursively. In particular, this may be done using the same matrix-free approximation. This method, which we denote "squared preconditioning", due to the recursive application of the linear GMRES solver, is used here for all matrix-free computations. In principle this algorithm could be recursively applied even further, since Algorithm 3.1 always includes the preconditioning step 3. However, in the present work we do not use any preconditioning in the "inner" GMRES loop.

It should thus be pointed out that the method is completely matrix-free. This means that storage requirements grow linearly in the degrees of freedom, as opposed to quadratically, which led to the extreme asymptotic storage requirements outlined in Section 3.3.1. For the matrix-free variant memory requirements are now dominated by storage of the Krylov vectors, which in 3D is certainly still considerable, but manageable with modern computer architectures.

3.4. Hybrid multilevel schemes

It is certainly possible to use multigrid with implicit and explicit relaxation schemes on different mesh levels and with different levels of approximation. Depending on the constraints deemed important one may find very different "optimal" combinations. For example, in [29] a scheme is proposed that uses a Discontinuous Galerkin method with an explicit Runge–Kutta time stepping on the finest grid, and implicit schemes on the coarser level, with the primary concern being storage requirements on the finest grid. While storage requirements are certainly a very big concern, we are considering the exact opposite approach here. Explicit schemes are deemed ineffective for higher orders of accuracy, which motivates the use of implicit methods on the highest levels of approximation. We address storage concerns by including an optional matrix-free formulation. On the other hand, explicit geometric multigrid methods operating in a finite-volume context are very effective, and are popular choices in many solvers.

The main proposition of the present paper is a scheme that uses the damped Newton/GMRES iterator $NK(p,n)$ defined in Section 3.3 on the highest level of approximation, while using multistage Runge–Kutta finite-volume smoothing with geometric h-multigrid $RKMG(m,n)$, as defined in Section 3.2, between Newton iterations to accelerate the convection of the volume-averaged large-scale error modes. Subsequently the smoothed volume averages replace the volume averages of the high-order relaxation. The procedure may thus be viewed as preconditioning for the nonlinear Newton iteration. The rationale behind this is that experience indicates that error convection and expulsion is the primary mode of convergence, when considering integrated quantities, such as force coefficients, which can often be essentially converged at rather high levels of residuals, when high-frequency errors still persist. For the acceleration of this convergence mode, no higher order solution representation is needed, and may indeed be of hindrance. The method is completed by a full multigrid (FMG) finite-volume start-up procedure.

Algorithm 3.2 gives an example of a practical implementation of the overall approach. We identify mesh levels by an indexed characteristic length h_m, where the coarsest mesh is indexed with $m = 1$, while the finest available mesh is indexed $m = M$. We have used this for external aerodynamics computations, as shown in Section 5. For easy reference we have denoted volume averages by an overbar, i.e. $\bar{w}_h$, while w_h indicates the solution at the current high-order polynomial level p.

Algorithm 3.2. Hybrid multilevel with full multigrid

```
 1. Initialize $\bar{w}^0_{h_1}$ with free stream conditions
 2. For $m = 1, \ldots, M$, Do
 3.     $\bar{w}^n_{h_m} = RKMG\left(\bar{w}^0_{h_m}; m, n\right)$
 4.     if $(m = M)$ exit
 5.     $\bar{w}^0_{h_{m+1}} = I^{h_{m+1}}_{h_m} \bar{w}^n_{h_m}$
 6. EndDo
 7. $w^0_{h_M} = Inject\left(\bar{w}^n_{h_M}; p\right)$
 8. For $n = 0, \ldots, N_{cyc}$, Do
 9.     $w^+_{h_M} = NK\left(w^n_{h_M}; p, 1\right)$
10.     if (converged) exit
11.     $\bar{w}^0_{h_M} = \mathcal{V}\left(w^+_{h_M}\right)$
12.     $\bar{w}^+_{h_M} = RKMG\left(\bar{w}^0_{h_M}, M, n_{RK}\right)$
13.     $w^{n+1}_{h_M} = w^+_{h_M} - Inject\left(\bar{w}^0_{h_M} - \bar{w}^+_{h_M}; p\right)$
14. EndDo
```

The first loop over the meshes defines the start-up procedure. The computation starts on the coarsest grid using a finite-volume method with explicit multistage relaxation, and then proceeds up to the next finer grid when a sufficiently good approximation to the solution has been achieved. This is applied recursively, using all coarser meshes in a FAS multigrid method on each mesh level, until the finest mesh is reached. The number of multigrid cycles n should be enough to attain reasonable convergence of integrated quantities, such as lift and drag, on each mesh level. The cost of these explicit finite-volume iterations is only a fraction of the total iteration cost so that the exact number is not critical. Typically, we use $n \leqslant 100$ for all levels.

The results of the finite-volume relaxation procedure are used as initial guess for the damped Newton iteration acting on the high-order discretization in Algorithm 3.2, step 7. We define the injection operator $Inject(\overline{w}_h; p)$, which distributes the volume averages to all nodal degrees of freedom used for the current approximation order p. This operator is also used in the subsequent multilevel relaxation procedure. Because of the good start value provided by the initial full multigrid relaxation, only very mild ramping of the CFL number has to be used to avoid start-up problems resulting from the inclusion of the higher order spectrum on the highest level. We typically use $N_{start} = 2$ in Eq. (17).

The main loop is over the combined Newton/GMRES and explicit smoothing operators. First the implicit iterator $NK(p,n)$ is applied. Note that we usually choose $n = 1$, as shown in Algorithm 3.2, step 9. Subsequently the volume averages are extracted in step 13. This is done for the non-hierarchical Spectral Difference basis by (exact) numerical quadrature based on the solution nodes, denoted $\mathcal{V}(w_h)$. Finally, the explicit multigrid iterations are performed for the volume averages, which produces updated values that replace the previous ones. We typically use $n_{RK} = 20$ for the additional $RKMG$ smoothing steps between Newton iterations.

Since intermediate p-levels are not used in the nonlinear multigrid cycle the theory outlined in Section 3.2 for finite-volume methods applies. Note however, that we have the option of augmenting the finite-volume procedure with a piecewise linear reconstruction procedure for the flux computation, which we typically use on the finest mesh. The finite-volume solution, of course, is still defined by the volume averages. In particular, transfer operators, such as the interpolation in step 5, and those within the smoothing operator $RKMG(m,n)$, only have to be defined between meshes for the volume averages, so that standard linear finite-volume transfer operators for inviscid flow [23] are sufficient and have been used here. In principle, intermediate p levels may also be used within this framework through lower-order preconditioning for the linear systems. In the present work, however, we used $ILU(n)$ preconditioning in case of matrix-explicit methods, while squared preconditioning is used for all matrix-free methods.

4. Implementation

The implementation of the techniques described thus far can be tedious. Multilevel methods require extra data structure to accommodate storage and transfer of the solution on different mesh levels. For implicit relaxation various tasks need to be considered, including solution of the linear systems, along with suitable preconditioning techniques. We used the PETSc library [30] for the linear solves, i.e. driving the GMRES algorithm, and providing preconditioning for matrix-explicit methods. For the matrix-free methods, user-defined routines implementing the matrix–vector product Av have to be provided.

4.1. Using automatic differentiation

One would like generic assembly of the Jacobian in the sense that changing the parameters of the scheme, e.g. order of approximation, numerical flux function, boundary conditions, etc., is automatically incorporated with no hand-derivation required. For the reconstruction and differentiation matrices that appear in Eq. (20) this is certainly true, since they are computed during pre-processing for the computation of the residual. The only part of the system Jacobian that is not generic in this sense is the differentiation of the numerical flux function and boundary conditions, which are also implemented as a modified numerical flux. A change of these requires a non-trivial re-derivation. In particular during development and validation, where changes are likely to occur, this is a drawback.

In order to automate the construction of the Jacobian matrix we adopted the TAPENADE automatic differentiation (AD) engine, developed at the INRIA Sophia-Antipolis research centre,[3] which may be used as a command line tool, and can thus be included in a standard Makefile structure. Suppose as an example that the numerical flux is implemented in *FORTRAN*90 as a subroutine called *NUMERICAL_FLUX*, in a file of the same name. Furthermore, suppose the output is given by the dependent variables (i.e. the fluxes), which we want to differentiate, and the input are the independent variables (i.e. the conservative variables) on both sides of the interface, with respect to which differentiation is required. Including a command line of the following form

```
tapenade -tangent -head numerical_flux -outvars "left_flux right_flux" -vars "left_density
left_x_momentum left_y_momentum left_energy right_density right_x_momentum right_y_momentum
right_energy" -outputfiletype fortranplushtml numerical_flux.f90
```

in the Makefile for the solver as a target rule, allows to automatically update the source code for the differentiated numerical flux whenever it is changed, including modified fluxes for boundary conditions, so that no additional manual operations are required by the user.

For production runs, it may prove beneficial to hand-optimize the differentiation routines after the source code has been generated, incorporating reasonable simplifications of the Jacobian matrices. In our experience this is far less time consuming than derivation by hand or even with the aid of symbolic manipulation tools.

5. Numerical experiments

Our numerical experiments primarily aim to assess convergence properties, measuring both reduction of the nonlinear residual, for which we define the quantity Res_ρ as

$$Res_\rho = (\Delta\tau)^{-1}(\rho^{n+1} - \rho^n), \tag{27}$$

as well as output functionals of interest. For compressible flow problems this may be the drag or the lift of a body submerged in a flow field.

As outlined previously, we focus on the transient solution process, with the ultimate aim of achieving mesh independent convergence at finite CFL numbers. Hence we compare test runs at constant CFL number, which we do not bother to increase past a nominal value, after the start-up procedure outlined in Section 3.3. By default we use CFL = 550.

All computations have been carried out on an Apple MacPro architecture with 2.66 GHz Quad-core Intel Xeon processor and Intel Fortran Compiler (version 10.1). All results use sequential processing.

5.1. Inviscid flow over a bump

We first carry out a study using matrix-explicit and matrix-free implicit methods in a single-grid approach to isolate the effect of the matrix-free approximation. The relaxation procedure is given by the damped Newton relaxation $NK(p,n)$ de-

[3] TAPENADE can be downloaded at http://www-sop.inria.fr/tropics/tapenade.html.

fined in Section 3.3, using Eq. (16), with GMRES Algorithm 3.1 in either matrix-explicit or matrix-free formulation, but *without* use of the multilevel procedure in Section 3.4. Consider inviscid flow over a smooth bump at free-stream Mach number $M_\infty = 0.3$ (see Fig. 2). Computations were performed using the Spectral Difference scheme on a regular triangular mesh with 342 cells. Polynomial orders of $p = 2,\ p = 3$, and $p = 5$ have been used, leading to a maximum number of degrees of freedom of $N_{dof} = N_w N_p N_{elem} = 28,728$ for the case $p = 5$. A comparison of the convergence for various orders of approximation is shown in Fig. 3 for the matrix-explicit method. The Jacobian matrix of the system is based on an exact differentiation of the residual, including the convective flux function, in this case Jameson's CUSP flux [15], and is re-computed every three nonlinear iterations by default, which has been found to be a good compromise between better convergence properties in terms of nonlinear iteration steps, and cost-per-iteration. For the case $p = 2$ we use $ILU(3)$ preconditioning, while $ILU(4)$ is used for the other cases. By default GMRES is restarted after 30 iterations. Besides the convergence in terms of Newton iterations (Fig. 3(a)) we also plot the convergence against cumulative GMRES iterations, i.e. number of Krylov vectors generated. Note that the number of linear iterations shown in Fig. 3(b) is by itself not very meaningful, as it can be adjusted very easily through preconditioning. In particular, if an exact LU decomposition is used this number becomes equal to the number of nonlinear Newton iterations, at the expense of much increased computational cost. The GMRES iterations are shown here merely to give an idea of representative "best practice" results, and facilitate comparison between different approaches. Also note that the goal is not to minimize the nonlinear iteration count, which could be accomplished by using adaptive step sizes, and increasing the CFL number past its nominal value, but rather the comparison with the matrix-free approach outlined in Section 3.3.2 at finite CFL number, leading up to mesh refinement studies presented below. For the matrix-free

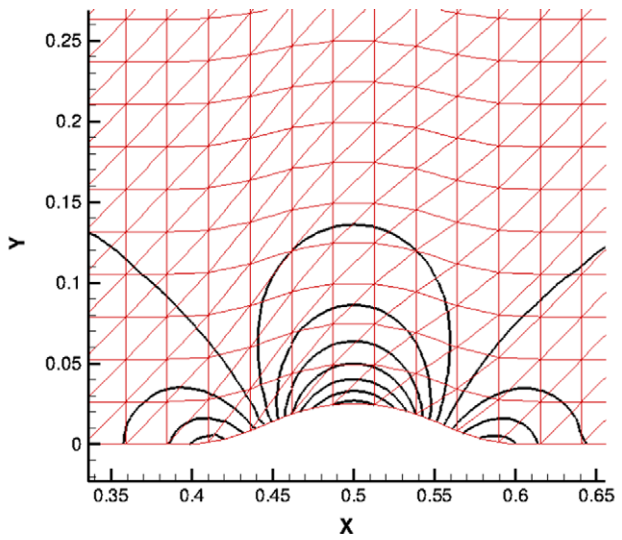


Fig. 2. Contours of constant Mach number for inviscid flow over a smooth bump at free-stream Mach number $M_\infty = 0.3$. Computation performed with the Spectral Difference scheme using cubic polynomials ($p = 3$).

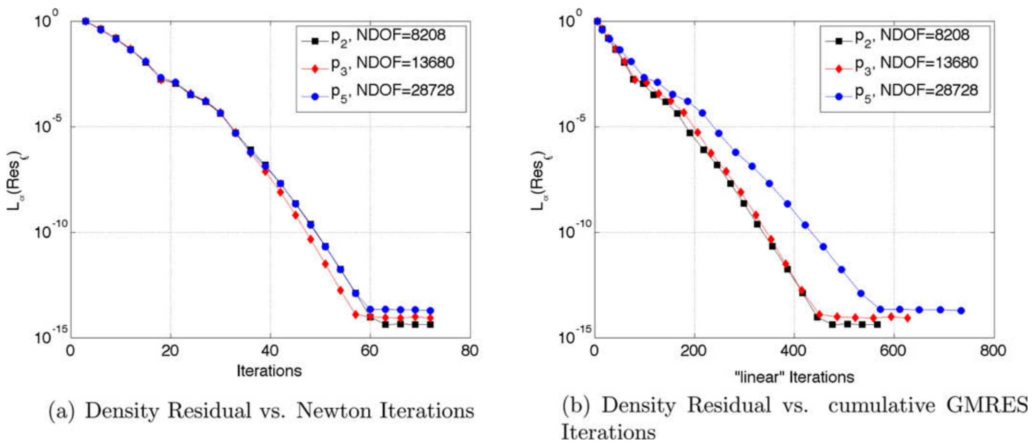


Fig. 3. Convergence of the matrix-explicit method for smooth inviscid flow around a bump at free-stream Mach number $M_\infty = 0.3$.

approach we use squared preconditioning as described in Section 3.3.3. The convergence is shown in Fig. 4. The number of Newton iterations needed to drive the solution to a steady state is not significantly affected by the matrix-free approach. This indicates that numerical approximation of the Krylov vector (24) yields consistently accurate results, despite the relatively simple step size estimation (25) in the difference approximation. In fact, compared with the best practice matrix-explicit approach the Newton iteration count is lower. This stems from the fact that for the matrix-explicit approach time-to-solution is usually reduced when saving the Jacobian for a number of iterations (3 in this case), while tolerating a somewhat higher nonlinear iteration count. If the matrix elements cannot be saved, the numerical approximation (24) is carried out each time a new Krylov vector is needed, which decreases the number of nonlinear iterations, because a more up-to-date approximation of the Jacobian is used, but also leads to more residual evaluations and increased computational expense. Furthermore, it is usually advisable to keep the preconditioning iterations relatively low for the matrix-free approach in order to avoid the associated computational overhead, tolerating higher number of GMRES iterations in each Newton cycle. Thus the convergence in terms of linear iterations is typically slower for the matrix-free approach.

Note also that the iterations stall at somewhat higher levels of residual compared to the matrix-explicit approach, which is likely due to the numerical approximation of the Jacobian matrix.

5.2. Inviscid flow around the NACA0012 profile

We now turn to numerical experiments for the hybrid multilevel method outlined in Section 3.4. We use a third order accurate method, i.e. quadratic polynomials, on the fine mesh for all remaining numerical tests. As stated in Section 3.1 it was a criterion to select cases where limiting and stabilization was not necessary. We consider stabilization in conjunction with relaxation important enough to focus on this in a separate publication. Thus we restrict ourselves to polynomial order $p = 2$ for the NACA test cases, and do not use higher orders, where stabilization issues become more prevalent. (The bump

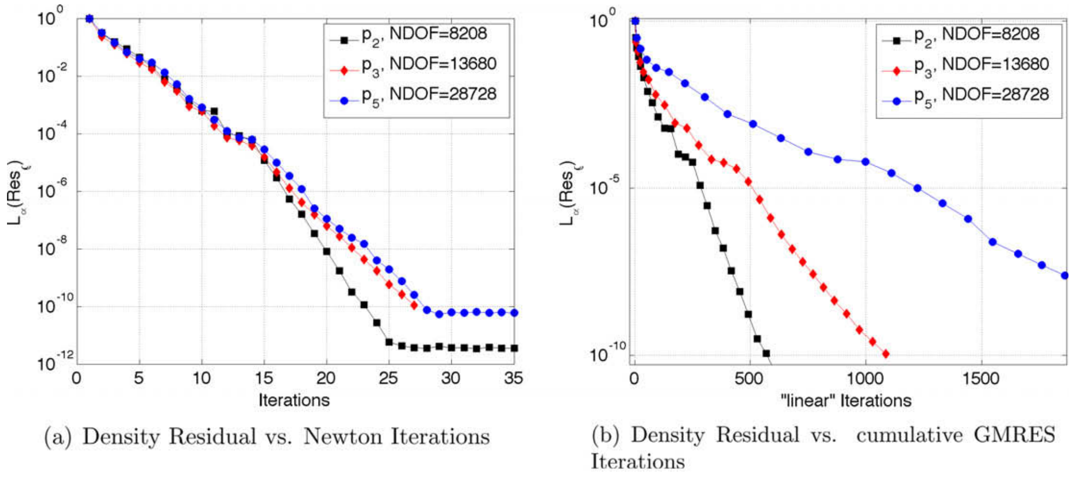


Fig. 4. Convergence of the matrix-free method for smooth inviscid flow around a bump at free-stream Mach number $M_\infty = 0.3$.

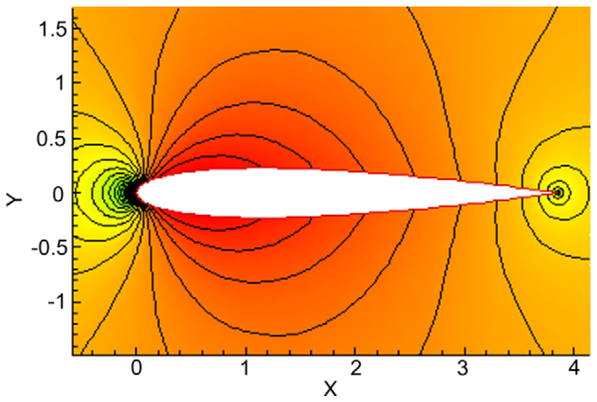


Fig. 5. Contours of constant Mach number for inviscid flow around the NACA0012 profile at free-stream Mach number $M_\infty = 0.3$ and angle-of-attack $\alpha = 0^\circ$.

test case is more benign, e.g. has no stagnation points, which we have found often to become problematic for higher order treatment, and necessitate procedures such as filtering or even limiting.) As an example consider smooth flow around the NACA0012 profile at free-stream Mach number $M_\infty = 0.3$, see Fig. 5.

Following the hybrid multilevel method, Algorithm 3.2 with $p = 2$, we first consider the matrix-explicit approach with GMRES(30) and $ILU(2)$ preconditioning. Between each Newton iterate we carry out 20 multigrid cycles with multistage smoothing ($n_{RK} = 20$ in Algorithm 3.2) to advance the volume averages. Again, due to the extremely low cost of the multistage smoothing iterations, the exact number is not critical, but it has been found that increasing the number of iterations beyond this point, does not improve performance much. Note that here we do *not* use the full multigrid start-up procedure, i.e. the first part of Algorithm 3.2, in order to better isolate the performance of the relaxation procedure. The meshes used for this test case are summarized in Table 2. Besides the number of elements and the total number of degrees of freedom we show the level of approximation and the CFL number used on each level.

The benefit of using multistage/multigrid smoothing between each Newton iteration comes from the fact that this requires little computational expense, compared to the implicit cycles, while aiding tremendously in speed-up of error convection. This is crucial for convergence of integrated quantities, even if the residual reduction rate is not greatly improved. This is demonstrated in Fig. 6, where the multilevel method is compared to an implicit method, which is lacking the multistage finite-volume smoothing, but is otherwise identical. For this mesh the residual reduction rate is almost unchanged, when plotted against execution time. Here the speed-up is roughly offset by the additional cost. However, the convergence of the drag is still noticeably improved. The strength of the approach will become more visible in mesh refinement, which is considered next.

The aim of geometric multigrid methods has traditionally been to achieve mesh-independent convergence rates. Here we consider three different meshes, which are summarized in Table 3. The coarsest of these is identical to the finest mesh in Table 2, and uses the same multigrid strategy, i.e. the one that produced Fig. 6. The multigrid strategy for a finer mesh includes the next coarser one recursively, so that the coarsest mesh is always the same (cf. Table 2), and the total number of meshes increases by one for every refinement. In order to assess the mesh dependence of the convergence at finite CFL numbers (nominal $CFL = 550$) we test the multilevel scheme first without the full multigrid start-up procedure, thus enlarging the transient solution region. Consider the matrix-explicit approach. We use $ILU(3)$ preconditioning for the implicit method on the highest level of approximation. The Jacobian is updated every other iteration. Fig. 7 shows the convergence of the drag coefficient vs. Newton iterations and cumulative GMRES iterations. It can be seen that in both measures the convergence of the multigrid method is very nearly mesh independent, while convergence of the single-grid implicit method degrades as the mesh is refined.

It should be noted that the benefit of the additional Runge–Kutta multigrid iterations ($n_{RK} = 20$ in Algorithm 3.2) comes at very modest computational expense (roughly 15% per nonlinear iteration), but aids tremendously in the convergence

Table 2
Degrees of freedom (DOF) used with the hybrid multilevel method. Note that the notation $p = 0(1)$ refers to a finite-volume method using piecewise linear reconstruction for the flux computation, as mentioned in Section 3.4.

Level	N_{dof}	p	Cells	CFL	Type
Hybrid multilevel					
4	61,440	2	2560	550	Implicit
3	10,240	0(1)	2560	6	Explicit
2	2560	0	640	6	Explicit
1	640	0	160	6	Explicit

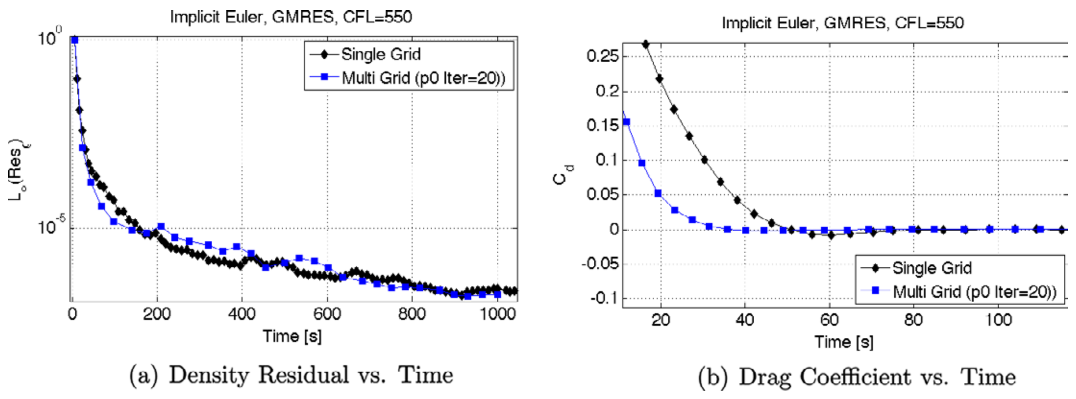


Fig. 6. Inviscid flow around the NACA0012 profile: comparison of implicit relaxation with and without additional geometric multigrid smoothing. Mesh with $N_{dof} = 61,440$.

Table 3
Meshes used for *h*-refinement study with hybrid multilevel method.

Level	N_{dof}	Elements
Fine	983,040	40,960
Medium	245,760	10,240
Coarse	61,440	2560

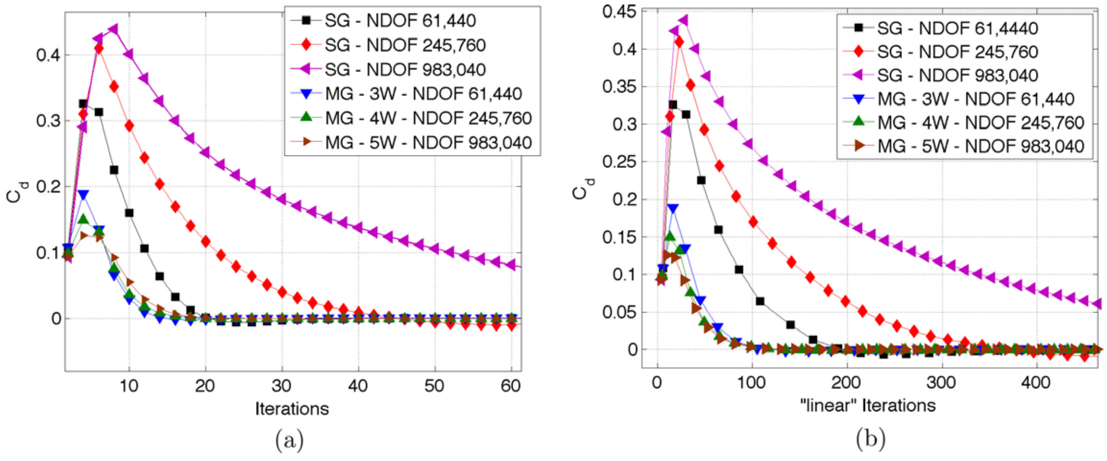


Fig. 7. Inviscid flow around the NACA0012 profile at free-stream Mach number $M_\infty = 0.3$, angle-of-attack $\alpha = 0^\circ$. Convergence of the matrix-explicit single-grid method and hybrid multilevel method (using 20 finite-volume multigrid cycles between Newton iterations).

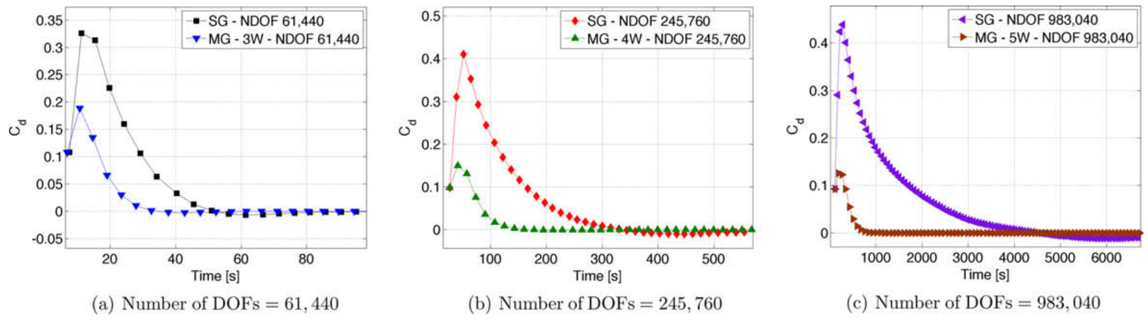


Fig. 8. Mesh refinement study. Inviscid flow around the NACA0012 profile at $M_\infty = 0.3$, $\alpha = 0^\circ$. Convergence of the drag coefficient. Comparison of implicit single-grid and hybrid multilevel methods in matrix-explicit formulation.

process. This is demonstrated in Fig. 8 where we compare the convergence of the single-grid method against the multilevel method in terms of CPU time. It can be seen that the speed-up becomes more dramatic as the mesh is refined. Note that, since we did not implement parallelization, we do not aim for 'realistic' timing estimates. Nevertheless, these estimates allow us to assess the mesh dependence of the convergence process.

The reduction rate of the residual is often used in theoretical analysis and numerical experiments to assess multigrid performance, where textbook multigrid is usually equated with mesh-independent convergence rate. Here we take a nonstandard approach with a hybrid method specifically addressing the pre-asymptotic convection-dominated convergence regime by operating on the volume averages to accelerate large-scale error convection and expulsion (cf. Section 3.4). It is interesting to examine the reduction rate of the residuals in this context. Fig. 9 shows the convergence of the density residual for the single-grid and the multilevel approach. It can be seen that for the multilevel method, the dependence of the convergence on the numerical mesh is greatly reduced. In particular in the pre-asymptotic phase, i.e. during the first 20 or so iterations, mesh-independent convergence is attained. This coincides with the convergence of large-scale error modes, indicated by the previously shown drag coefficient convergence (cf. Fig. 7), and nicely reflects the rationale outlined in Section 3.4.

It should also be noted that after convergence of large-scale modes, the global convergence region of a Newton method can be expected to have been reached. Increasing the CFL number may be the most effective way to control asymptotic

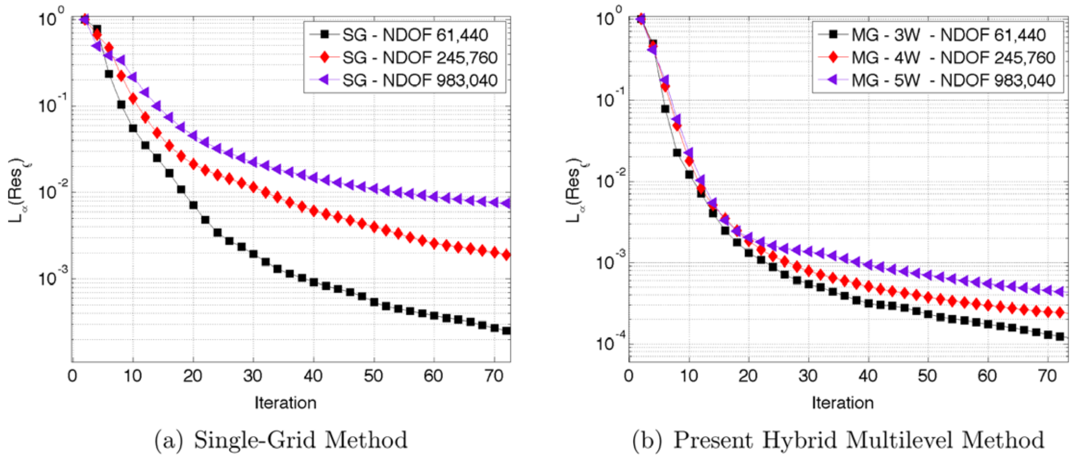


Fig. 9. Inviscid flow around the NACA0012 profile at free-stream Mach number $M_\infty = 0.3$, angle-of-attack $\alpha = 0^\circ$. Convergence of the density residual using matrix-explicit formulation.

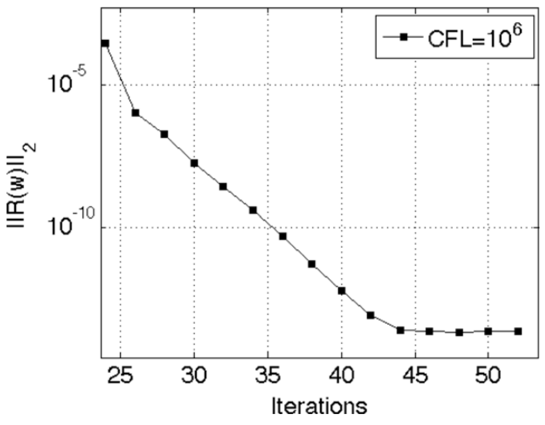


Fig. 10. Inviscid flow around the NACA0012 profile at free-stream Mach number $M_\infty = 0.3$, angle-of-attack $\alpha = 0^\circ$, $N_{dof} = 61,440$. Asymptotic convergence of the residual norm after increasing the CFL number to 10^6 at iteration number $n = 24$.

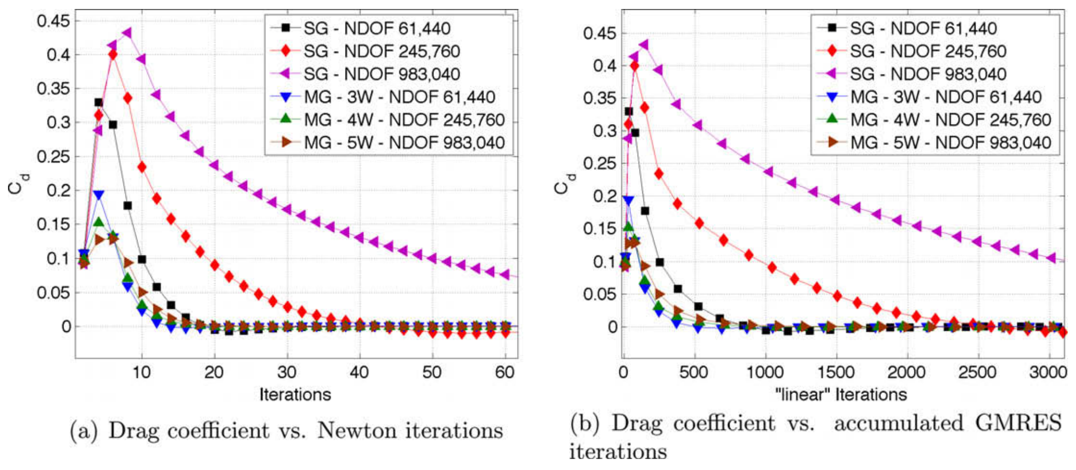


Fig. 11. Inviscid flow around the NACA0012 profile at free-stream Mach number $M_\infty = 0.3$, angle-of-attack $\alpha = 0^\circ$. Convergence of the matrix-free single-grid method and hybrid multilevel method (using 20 finite-volume multigrid cycles between Newton iterations).

convergence rates in this case. In experiments carried out for this test case, raising the CFL number to 10^6 after roughly 20 iterations proved quite effective (e.g. reducing the residuals to machine accuracy within 20 additional iterations for the coarse mesh case, see Fig. 10). However, since adaptive time stepping should be carefully integrated with the multilevel method, which is beyond the scope of this article, we shall not further pursue adaptive CFL numbers here.

Consider in comparison the matrix-free method applied to the same mesh refinement study. We replace the matrix storage of Section 3.3.1 with the matrix-free approximation of Section 3.3.2, and the ILU preconditioning with the squared preconditioning of Section 3.3.3. As shown in Fig. 11 the convergence is still nearly mesh-independent for the multilevel method with the matrix-free implementation. Again the number of Newton iterations is virtually unchanged, indicating good quality of numerical approximation of the Jacobian.

However, preconditioning becomes an issue, which can be seen from the number of linear iterations required to solve the system. Enforcing a low linear iteration count by increasing the number of inner iterations for the squared preconditioning method is not always advisable, since it increases time-to-solution. In our experience tolerating a somewhat higher linear iteration count is most beneficial in terms of runtime. An alternative to reducing the linear iteration count is increasing the GMRES restart level by quite a bit. It is felt that this is appropriate for the matrix-free method, since the incurred memory penalty is linear in the degrees of freedom, while storage of the Jacobian matrix requires memory proportional to $N_e N_p^2$. In this particular case we tolerate a restart value of 100, for which the memory saving is still quite dramatic, in particular for finer meshes, compared to explicit storage of the Jacobian matrix.

Again, the convergence benefits translate to improved runtime figures. Fig. 12 shows the convergence of the single-grid method against the multilevel method in terms of CPU time. It can be seen that the speed-up becomes more dramatic as the mesh is refined. Note that we have made no attempt to fully converge the single-grid method on the finest mesh, as the runtime becomes clearly prohibitive. Here the multigrid addition makes the computation of the testcase realistic with sequential processing.

As in the matrix-explicit case we also consider the convergence of the density residual. Fig. 13 confirms that again mesh independent convergence is reached primarily in the convection-dominated early convergence process. Overall, dependence of the convergence on the numerical mesh is greatly reduced.

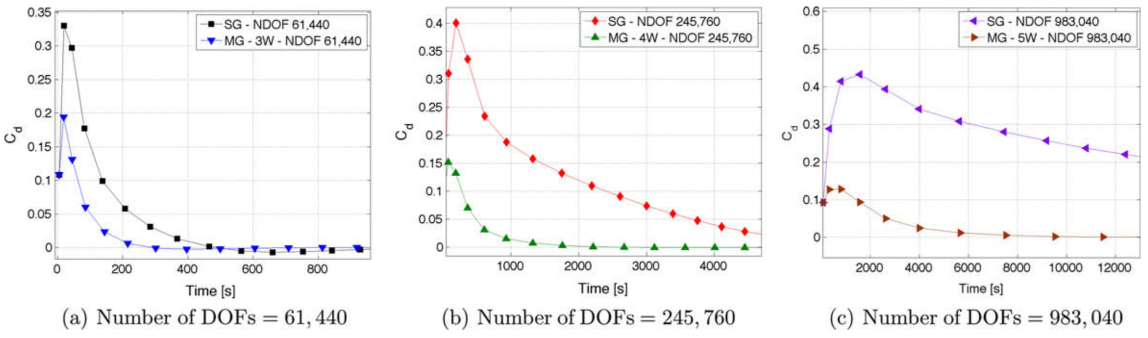


Fig. 12. Mesh refinement study. Inviscid flow around the NACA0012 profile at $M_\infty = 0.3,\ \alpha = 0^\circ$. Convergence of the drag coefficient. Comparison of implicit single-grid and hybrid multilevel methods in matrix-free formulation.

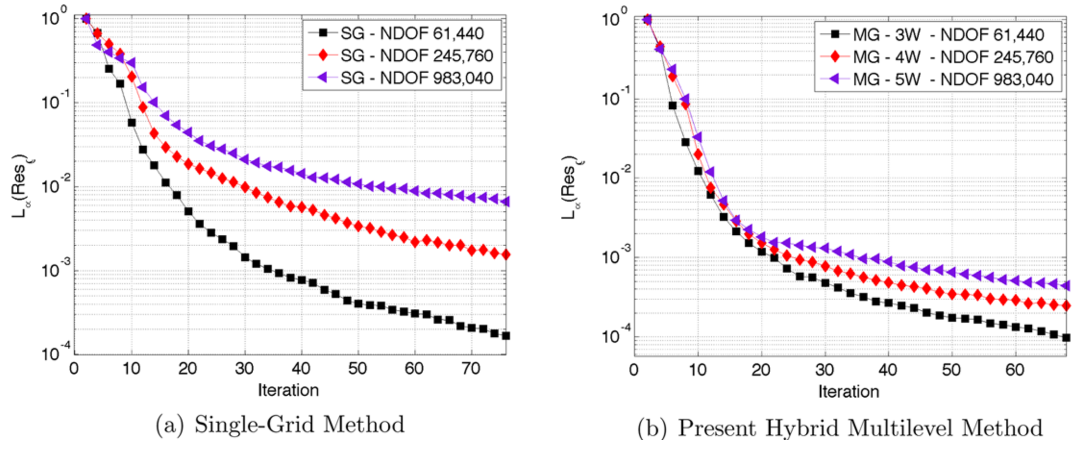


Fig. 13. Inviscid flow around the NACA0012 profile at free-stream Mach number $M_\infty = 0.3$, angle-of-attack $\alpha = 0^\circ$. Convergence of the density residual using matrix-free formulation (using 20 finite-volume multigrid cycles between Newton iterations).

Best-practice solver settings include a proper start-up procedure. So far in this section we started the iterative process on the highest level of approximation from a uniform flow approximation at free-stream conditions to better compare the properties of the relaxation procedures. In practice, however, we typically use a full multigrid approximation, where the solution process is started with a finite-volume method on the coarsest mesh used in the multigrid cycle, as shown in Algorithm 3.2. We repeat the NACA0012 testcase at $M = 0.3$ and $\alpha = 0^\circ$ using the matrix-explicit approach with the additional full multigrid approach included. Convergence is compared between the multilevel method with and without full multigrid, and a straight implicit Newton method for the medium mesh ($N_{dof} = 245,760$) in Fig. 14. The speed-up is dramatic, while the full multigrid start-up procedure typically incurs only a very small overhead. This will become evident below in timing measurements for mesh refinement.

Consider again the mesh refinement study carried out above with the full multigrid start-up included. Fig. 15 shows convergence of the drag coefficient. For clarity we omit the single-grid implicit method, and include only the two multilevel methods (with and without full multigrid start-up). It can be seen that convergence is nearly mesh-independent for both approaches at finite CFL numbers (nominal CFL = 550), with much improved start-up evident for the full multigrid approach. Finally, we compare convergence properties measured in CPU time in Fig. 16. It can be seen that the full multigrid method provides yet another noticeable improvement in turn-around-time compared to the multilevel procedure starting from the finest mesh at free-stream conditions. This represents currently our best-practice approach.

For a test involving lifting flow, we apply the same strategy to inviscid flow around the NACA0012 airfoil at flow conditions $M_\infty = 0.4$ and $\alpha = 5^\circ$. Contours of constant Mach number for this test case are shown in Fig. 17.

Fig. 18 shows the convergence history of the lift plotted against CPU time for the three different meshes summarized in Table 3, using the complete multilevel strategy described in Section 3.4 in mesh refinement. GMRES(30) was used for linear iterations along with *ILU*(3) preconditioning. It can be seen in Fig. 18 that quite rapid convergence is attained.

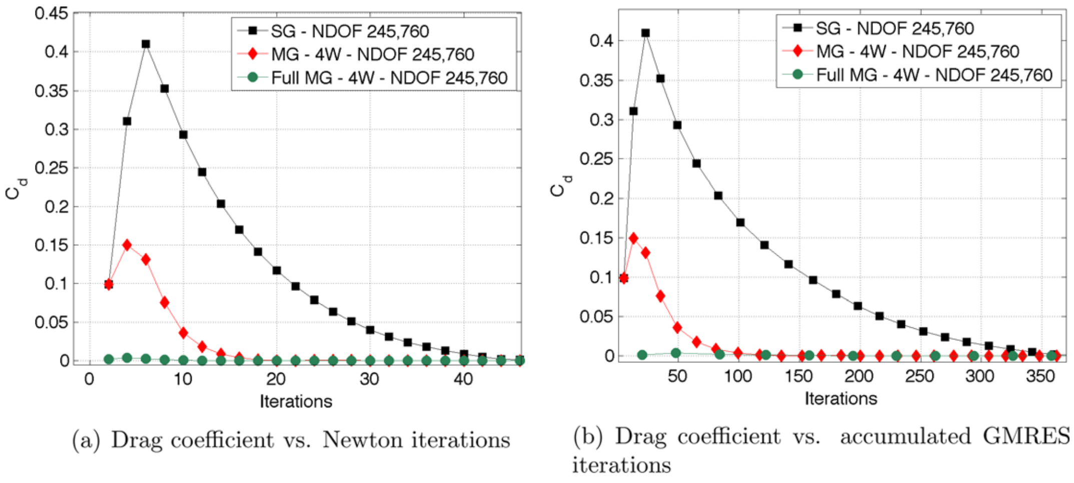


Fig. 14. Inviscid flow around the NACA0012 profile at $M_\infty = 0.3,\ \alpha = 0^\circ$. Comparison of relaxation procedures for the medium mesh ($N_{dof} = 245,760$). SG: Single-grid damped Newton method; MG: hybrid multilevel method; Full MG: hybrid multilevel with full multigrid, Algorithm 3.2.

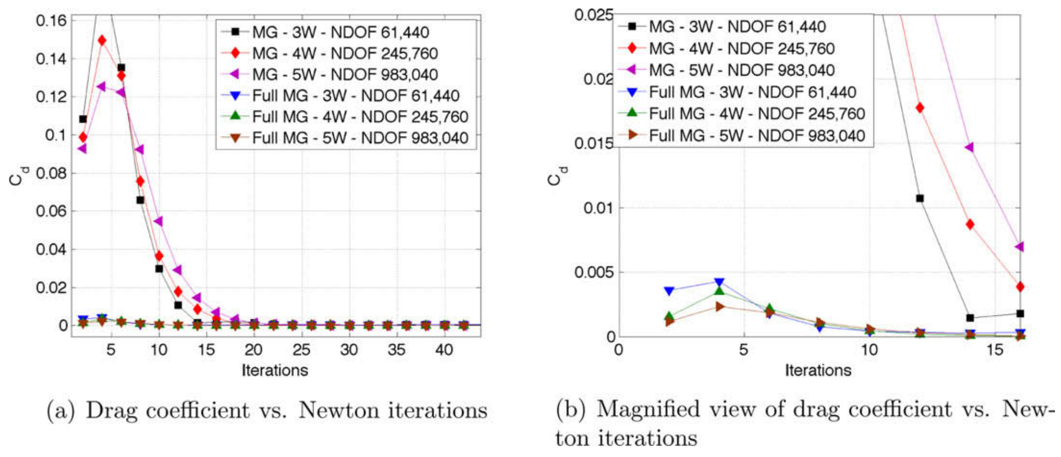


Fig. 15. Inviscid flow around the NACA0012 profile at $M_\infty = 0.3,\ \alpha = 0^\circ$. Comparison of relaxation procedures in mesh refinement. MG: Hybrid multilevel method; Full MG: hybrid multilevel with full multigrid.

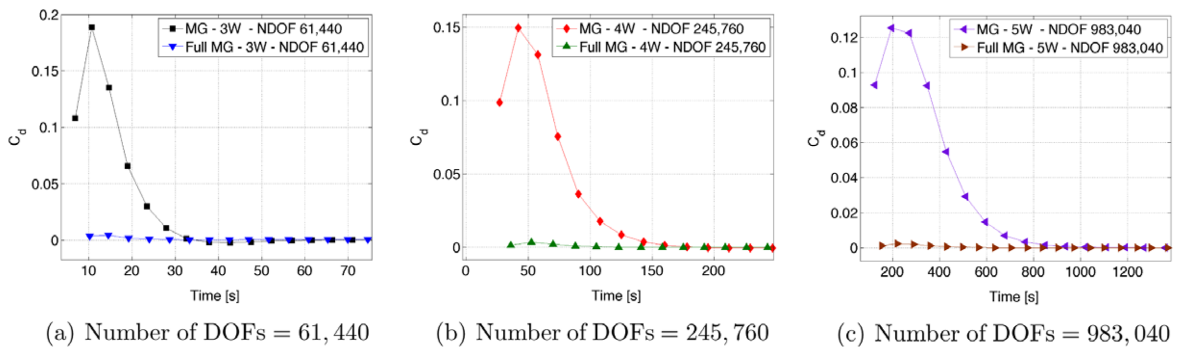


Fig. 16. Mesh refinement study. Inviscid flow around the NACA0012 profile at $M_\infty = 0.3,\ \alpha = 0^\circ$. Convergence properties of the hybrid multilevel method with and without full multigrid procedure.

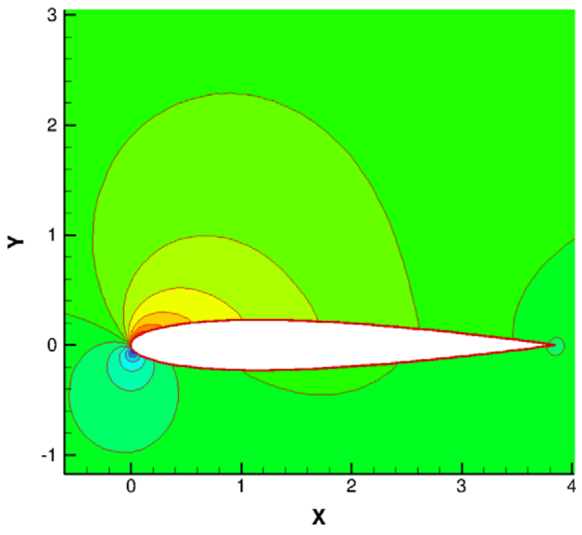


Fig. 17. Mach number contours for flow around the NACA0012 profile at $M_\infty = 0.4,\ \alpha = 5^\circ$.

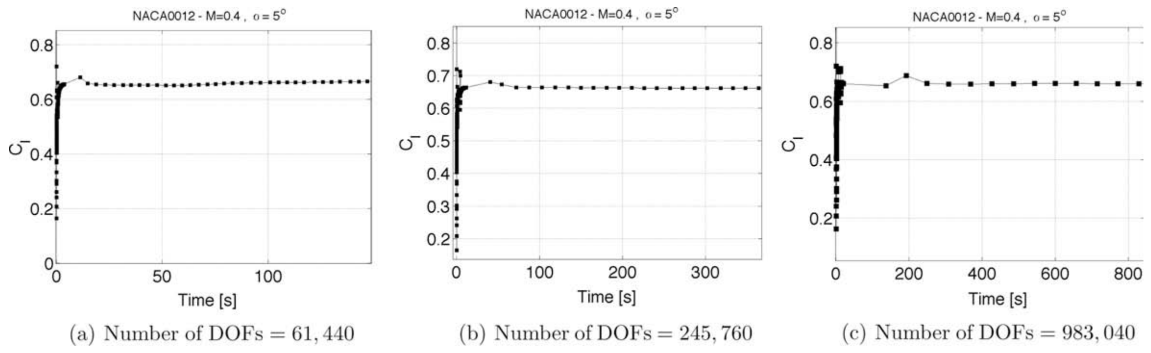


Fig. 18. Mesh refinement study. Inviscid flow around the NACA0012 profile at $M_\infty = 0.4,\ \alpha = 5^\circ$. Convergence of the lift coefficient.

6. Conclusion

We have proposed and validated a multilevel solution method for compressible fluid flow. The viability of the proposed method has been verified for 2D subsonic compressible flow simulations. In particular the feasibility of attaining mesh-independent convergence at finite CFL numbers, at only marginally increased per-iteration cost compared to a straight forward damped Newton relaxation, has been shown.

Optionally the method may be formulated in an entirely matrix-free fashion, which eliminates global matrix storage requirements, including preconditioning. This removes the immense storage requirements for implicit relaxation methods with higher order discretization, which often causes concern, in particular regarding the extension to 3D computations.

Two important future research directions may be identified. Firstly, the extension to three-dimensional test cases is the main goal. The method should be applicable to 3D flow without conceptual modifications, although the implementation effort for a viable 3D higher order solver, including a suitable multilevel data structure, is considerable. However, ingredients of the present approach may be added to such a baseline solver without much additional effort. For example, the system Jacobian can be rather easily computed, regardless of dimensionality using the automatic differentiation approach described in this article, while the GMRES solves can be easily done through a generic interface to a high-performance library. Matrix-free implementation requires only additional residual evaluations and recursive application of the GMRES solves. The second major goal for future investigation is the extension of the methodology to incorporate limiting or filtering procedures, and the application to shocked flow, such as transonic aerodynamics. In principle the automatic differentiation tools can be used via chain rule to include such stabilization methods in the derivation used here. On the other hand, identifying a viable approach regarding the interplay of the relaxation procedures with proper stabilization techniques, will very likely require carful investigation.

Acknowledgments

Financial support from the Deutsche Forschungsgemeinschaft (German Research Association) through grant GSC 111, and by the AirForce Office of Scientific Research, AirForce Materiel Command, USAF, under grant number FA8655-08-1-3060, is gratefully acknowledged. The US Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright notation thereon.

References

[1] A. Jameson, Solution of the Euler equations for two-dimensional transonic flow by a multigrid method, Appl. Math. Comput. 13 (1983) 327–356.
[2] A. Jameson, S. Yoon, Lower-upper implicit schemes with multiple grids for the Euler equations, AIAA J. 25 (7) (1987) 929–935.
[3] N.A. Pierce, M.B. Giles, Preconditioned multigrid methods for compressible flow calculations on stretched meshes, J. Comput. Phys. 136 (1997) 425–445.
[4] A. Jameson, D.A. Caughey, How many steps are required to solve the Euler equations of steady compressible flow: in search of a fast solution algorithm, AIAA Paper 01-2673, American Institute of Aeronautics and Astronautics, 2001.
[5] D.J. Mavriplis, An assessment of linear versus nonlinear multigrid methods for unstructured mesh solvers, J. Comput. Phys. 175 (1) (2002) 302–325.
[6] D.A. Knoll, D.E. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.
[7] J.S. Hesthaven, D. Gottlieb, Stable spectral methods for conservation laws on triangles with unstructured grids, Comput. Meth. Appl. Mech. Eng. 175 (1999) 361–381.
[8] C.R. Nastase, D.J. Mavriplis, High-order Discontinuous Galerkin methods using an *hp*-multigrid approach, J. Comput. Phys. 213 (2006) 330–357.
[9] Y. Liu, M. Vinokur, Z.J. Wang, Discontinuous spectral difference method for conservation laws on unstructured grids, in: Proceedings of the Third International Conference on Computational Fluid Dynamics, July 12–16, 2004, Toronto, Canada, Springer, 2004.
[10] Y. Liu, M. Vinokur, Z.J. Wang, Spectral Difference method for unstructured grids. I: Basic formulation, J. Comput. Phys. 216 (2) (2006) 780–801.
[11] Z.J. Wang, Y. Liu, G. May, A. Jameson, Spectral Difference method for unstructured grids. II: Extension to the Euler equations, J. Sci. Comput. 32 (1) (2007) 54–71.
[12] G. May, A kinetic scheme for the Navier–Stokes equations and high-order methods for hyperbolic conservation laws, Ph.D. Thesis, Stanford University, Stanford, CA 94305, 2006.
[13] K. van den Abeele, C. Lacor, Z.J. Wang, On the stability and accuracy of the spectral difference method, J. Sci. Comput. 37 (2) (2008) 162–188.
[14] D.A. Kopriva, J.H. Kolias, A conservative staggered-grid Chebyshev multidomain method for compressible flows, J. Comput. Phys. 125 (1996) 244–261.
[15] A. Jameson, Analysis and design of numerical schemes for gas dynamics. 2: Artificial diffusion and discrete shock structure, Int. J. Comput. Fluid Dynam. 5 (1995) 1–38.
[16] J.S. Hesthaven, From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex, SIAM J. Numer. Anal. 35 (2) (1998) 655–676.
[17] A. Jameson, Aerodynamics, in: E. Stein, R. De Borst, T.J.R. Hughes (Eds.), Encyclopedia of Computational Mechanics, vol. 3, Wiley, 2004 (Ch. 11).
[18] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock capturing schemes, J. Comput. Phys. 77 (1988) 439–471.
[19] A. Harten, High-resolution schemes for hyperbolic conservation laws, J. Comput. Phys. 49 (3) (1983) 357–393.
[20] B. Cockburn, C.W. Shu, Runge–Kutta Discontinuous Galerkin methods for convection-dominated problems, J. Sci. Comput. 16 (3) (2001) 173–261.
[21] J.S. Hesthaven, T. Warburton, Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications, Texts in Applied Mathematics, vol. 54, Springer-Verlag, 2007.
[22] A. Brandt, Multi-level adaptive solutions to boundary-value problems, Math. Comput. 31 (138) (1977) 333–390.
[23] A. Jameson, J.C. Vassberg, A vertex-centroid (V-C) scheme for the gas-dynamics equations, in: N. Satofuka (Ed.), Computational Fluid Dynamics 2000: Proceedings of the First International Conference on Computational Fluid Dynamics, Springer-Verlag, Kyoto, Japan, 2001, pp. 37–52.
[24] W.A. Mulder, A new multigrid approach to convection problems, J. Comput. Phys. (83) (1989) 303–323.
[25] W. Hackbusch, Multi-Grid Methods and Applications, Springer Series in Computational Mathematics, second ed., vol. 4, Springer-Verlag, 2003.
[26] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[27] A. Soulaimani, N.B. Salah, Y. Saad, Enhanced GMRES acceleration techniques for some CFD problems, Int. J. Comput. Fluid Dynam. 16 (1) (2002) 1–20.
[28] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., Society for Industrial and Applied Mathematics, 2003.
[29] H. Luo, J.D. Baum, R. Löhner, A *p*-multigrid Discontinuous Galerkin method for the Euler equations on unstructured grids, J. Comput. Phys. 211 (2006) 767–783.
[30] S. Balay, K. Buschelman, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, H. Zhang, PETSc users manual, Tech. Rep. ANL-95/11 – Revision 2.1.5, Argonne National Laboratory, 2004.