Journal of Computational Physics 228 (2009) 7917–7940

Contents lists available at ScienceDirect

Journal of Computational Physics

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

Multigrid algorithms for high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations

Khosro Shahbazi [a,*], Dimitri J. Mavriplis [b], Nicholas K. Burgess [b]

[a] *Division of Applied Mathematics, Brown University, 182 George Street, Providence, RI 02912, United States*
[b] *Department of Mechanical Engineering, University of Wyoming, 1000 E. University Avenue, Laramie, WY 82071, United States*

ARTICLE INFO

Article history:
Received 22 October 2008
Received in revised form 3 April 2009
Accepted 15 July 2009
Available online 24 July 2009

Keywords:
Discontinuous Galerkin methods
High-order methods
Compressible Navier–Stokes equations
Multigrid methods
Krylov subspace methods
Preconditioning

ABSTRACT

Multigrid algorithms are developed for systems arising from high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations on unstructured meshes. The algorithms are based on coupling both *p*- and *h*-multigrid (*ph*-multigrid) methods which are used in nonlinear or linear forms, and either directly as solvers or as preconditioners to a Newton–Krylov method.

The performance of the algorithms are examined in solving the laminar flow over an airfoil configuration. It is shown that the choice of the cycling strategy is crucial in achieving efficient and scalable solvers. For the multigrid solvers, while the order-independent convergence rate is obtained with a proper cycle type, the mesh-independent performance is achieved only if the coarsest problem is solved to a sufficient accuracy. On the other hand, the multigrid preconditioned Newton–GMRES solver appears to be insensitive to this condition and mesh-independent convergence is achieved under the desirable condition that the coarsest problem is solved using a fixed number of multigrid cycles regardless of the size of the problem.

It is concluded that the Newton–GMRES solver with the multigrid preconditioning yields the most efficient and robust algorithm among those studied.

1. Introduction

The inadequacy of current production-level computational fluid dynamics codes in delivering sufficient accuracy in numerical flow simulations, as well as in resolving a wide range of turbulence scales for reliable large eddy simulations have been widely realized over the past decade. Since these codes are typically based on low-order finite-volume methods, high-order methods such as discontinuous Galerkin (DG) methods have been advocated as alternative discretization techniques. DG methods are weighted residual methods with discontinuous approximate solution spaces typically consisting of polynomials of degree p defined on each element of the geometry triangulation. The inter-element connectivities are enforced through a proper definition of numerical fluxes along the shared boundaries between elements. High-order DG methods have advantages over continuous Galerkin methods in capturing features of convection-dominated flows, facilitating hp-adaptivity, ease of parallelization, and in the effectiveness of block-diagonal iterative solvers. For these advantages to be realized in industrial simulations, efficient solution strategies should be developed for systems arising from high-order DG discretizations. The importance of having efficient solution algorithms for DG methods is further appreciated when realizing that for a fixed mesh and a fixed approximation order (low to moderately high-orders), the DG discretization yields larger number of degrees of freedom compared to the continuous formulation.

* Corresponding author. Tel.: +1 4018632250; fax: +1 4018631355.
E-mail addresses: shahbazi@brown.edu (K. Shahbazi), mavripl@uwyo.edu (D.J. Mavriplis), nburges2@uwyo.edu (N.K. Burgess).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.07.013

In this work, we consider the solution of the steady compressible Navier–Stokes equations using multigrid algorithms on unstructured (possibly anisotropic) meshes. Multigrid methods have been proved to be highly efficient in iteratively solving the system arising from implicit treatment of a wide range of physical problems including elliptic and hyperbolic equations [10,17,18,24].

For low-order discretizations (e.g. low-order finite-volume or element method), multigrid algorithms have been traditionally involved constructing a sequence of increasingly coarser meshes, and then using a fixed discretization technique on these coarser grids to form coarse level approximations (h-multigrid). One approach to obtain coarser meshes is the so-called agglomeration process involving merging together neighboring elements to obtain coarser grids. First-order accurate ($p = 0$) agglomeration multigrid methods for unstructured meshes are well established and deliver near optimal convergence rates (h-independence convergence) [8,9].

On the other hand, for high-order finite element methods, a natural choice for constructing coarse level approximations is to hold the computational mesh fixed and discretize the equations using a lower approximation order within each element (p-multigrid). This approach has been advocated and used in the context of spectral element methods for the last few decades [23,22]. More recently, two-level and p-multigrid algorithms have been studied for DG methods [29,16,13]. Persson and Peraire developed a two-grid preconditioner for the Newton–GMRES solution of the system arising from the high-order DG discretization of the compressible Navier–Stokes equations [29]. They considered the block Jacobi, block Gauss–Seidel and block ILU smoothers. The coarse level was either the $p = 0$ approximation or the $p = 1$, which was solved exactly using a direct solver. They concluded that the ILU preconditioner performs better than other preconditioners studied. Fidkowski et al. developed a p-multigrid algorithm for the high-order DG discretization of the compressible Navier–Stokes equations [13]. Their algorithm employed an element line Jacobi smoother in which lines of elements are formed using coupling based on the $p = 0$ discretization of the scalar convection–diffusion equation. They used a V-cycle multigrid with the coarsest level formed with the $p = 0$ approximation, and solved using a large number of smoothing iterations. Using the element line Jacobi smoother, they reported order-independent (p-independent) convergence rates, up to $p = 3$. However, they observed some h-dependence; that is, the convergence rate degrades as the mesh size grows.

The performance of the p-multigrid (and two-grid) solvers may degrade if the coarsest level problem is not solved efficiently. For solving the coarsest problem, although either a direct solver or a large number of smoothing iterations can be employed (as was employed in [29,13], respectively), they both yield poor asymptotic scaling with the mesh size. One remedy to this problem is to increase the total number of coarse levels through coupling both p- and h-multigrid (ph-multigrid) to more efficiently damp the large wave length error modes. This strategy was indeed used by Nastase and Mavriplis for the solution of Euler equations in two and three dimensions on unstructured triangular meshes [20,21] demonstrating p-independent and nearly h-independent convergence rates.

We herein extend the idea of coupling p- and h-multigrid algorithms to the high-order DG solution of the compressible Navier–Stokes equations using triangular meshes. One of the distinctive features of this work, compared to [20], is the investigation of two different cycling strategies for the ph-multigrid solution of the Navier–Stokes equations. Specifically, we show that unlike for the Euler equations, having the $p = 0$ approximation for the coarsest level does not yield optimal convergence for the Navier–Stokes equations, and we require the use of the $p = 1$ approximation as the coarsest level problem. We suggest solving the $p = 1$ problem itself using several cycles of a V-cycle ph-multigrid scheme. We also investigate the use of the ph-multigrid algorithm as a preconditioner within the context of a Newton–GMRES solver.

The remainder of the paper is organized as follows. In the next section, we introduce the steady compressible Navier–Stokes equations, before describing the DG discretization in Section 3. We then present the single-grid solver and the proposed multigrid algorithms and the preconditioned Newton–GMRES solver in Sections 4–6, respectively. In the final two sections, we present the numerical results and conclude the work.

2. Compressible Navier–Stokes equations

The steady compressible Navier–Stokes equations are written in vector form as

$$\frac{\partial}{\partial x_i}\left(\mathbf{f}_i^c(\mathbf{u}) - \mathbf{f}_i^{v}(\mathbf{u}, \nabla\mathbf{u})\right) = 0 \text{ in } \Omega(i = 1, \dots, d), \tag{1}$$

where Ω is a bounded domain in d space dimensions with $d = 2$ or 3. The conservative state vectors are $\mathbf{u} \equiv [u_1, \dots, u_{d+2}]^T = [\rho, \rho v_1, \dots, \rho v_d, \rho E]^T$ with $\rho, \mathbf{v} = [v_1, \dots, v_d]$ and E representing the density, velocity vector and total energy, respectively.

The convective and viscous fluxes are, respectively, defined as

$$\mathbf{f}_i^c(\mathbf{u}) = \begin{bmatrix} \rho v_i \\ \rho v_1 v_i + p\delta_{1i} \\ \vdots \\ \rho v_d v_i + p\delta_{di} \\ \rho H v_i \end{bmatrix}, \quad \mathbf{f}_i^{v}(\mathbf{u}, \nabla\mathbf{u}) = \begin{bmatrix} 0 \\ \tau_{1i} \\ \vdots \\ \tau_{di} \\ \tau_{ij} v_j + \mathcal{K}\frac{\partial T}{\partial x_i} \end{bmatrix}, \quad (i = 1, \dots, d), \tag{2}$$

where δ_{ij} is the Kronecker delta, p the pressure, T the temperature, $\mathcal{K}$ the thermal conductivity and H the total enthalpy defined as

$$H = E + \frac{p}{\rho} = e + \frac{1}{2}\mathbf{v}^2 + \frac{p}{\rho}, \tag{3}$$

with e being the static internal energy. The pressure is determined by the equation of state for an ideal gas

$$p = (\gamma - 1)\rho e, \tag{4}$$

where γ is the ratio of specific heat capacities at a constant pressure and a constant volume. For dry air, $\gamma = 1.4$. The viscous stress tensor is defined as

$$\tau = \mu\left(\nabla\mathbf{v} + (\nabla\mathbf{v})^T - \frac{2}{3}(\nabla\cdot\mathbf{v}I)\right), \tag{5}$$

where μ is the dynamic viscosity. The temperature is related to the total energy and velocity as

$$\mathcal{K}T = \frac{\mu\gamma}{Pr}\left(E - \frac{1}{2}\mathbf{v}^2\right), \tag{6}$$

with $Pr = 0.72$ being the Prandtl number.

For the discretization purpose, we rewrite the governing equations in the following equivalent form:

$$\frac{\partial}{\partial x_i}\left(\mathbf{f}_i^c(\mathbf{u}) - G_{ij}(\mathbf{u})\frac{\partial\mathbf{u}}{\partial x_j}\right) = 0 \quad \text{in} \quad \Omega, (i,j = 1,\dots,d), \tag{7}$$

where the matrices $G_{ij} = \partial\mathbf{f}_i^v/\partial\left(\frac{\partial\mathbf{u}}{\partial x_j}\right)$ in two and three space dimensions are given in [1,2], respectively. The matrices G_{ij} are nonzero, nonsymmetric and positive semi-definite.

The governing Eq. (1) must be supplemented with appropriate boundary conditions. Let Γ denote the boundary of the domain Ω. We assume only three types of boundary conditions, subsonic inflow, subsonic outflow and adiabatic wall boundary conditions. The corresponding parts of the boundary Γ are denoted with $\Gamma_{in}, \Gamma_{out}$ and Γ_{wall}, where $\Gamma = \Gamma_{in} \cup \Gamma_{out} \cup \Gamma_{wall}$. For each boundary condition type, we define the boundary operator $\mathbf{u}_\Gamma$ as follows:

$$\mathbf{u}_\Gamma(\mathbf{u}) = \left((g_D)_1, (g_D)_2, \dots, (g_D)_{d+1}, \frac{p(\mathbf{u})}{\gamma - 1} + \frac{(g_D)_2^2 + \cdots + (g_D)_{d+1}^2}{2(g_D)_1}\right) \text{ on } \Gamma_{in}, \tag{8a}$$

$$\mathbf{u}_\Gamma(\mathbf{u}) = \left(u_1, u_2, \dots, u_{d+1}, \frac{p_\infty}{\gamma - 1} + \frac{u_2^2 + \cdots + u_{d+1}^2}{2u_1}\right) \text{ on } \Gamma_{out}, \tag{8b}$$

$$\mathbf{u}_\Gamma(\mathbf{u}) = (u_1, 0, \dots, 0, u_4) \text{ on } \Gamma_{wall}, \tag{8c}$$

where $(g_D)_1, \dots, (g_D)_{d+1}$ are the given Dirichlet values for the conservative variables $u_1, u_2, \dots, u_{d+1}$, and p_∞ is the farfield pressure value.

3. Discontinuous Galerkin discretization

Let Ω be divided into a set of shape-regular elements, $\mathcal{T}_h = \{K_i | i = 1, \dots, N\}$, consisting of triangles or tetrahedrons in two or three space dimensions, respectively. For each K, we denote by $\mathbf{n}_K$ the unit outward normal vector. Let K^+ and K^- be two adjacent elements of $\mathcal{T}_h$ and $\mathbf{x}$ be an arbitrary point on the interior face[1] $e = \partial K^+ \cap \partial K^-$. Moreover, let $\mathbf{v}$ and $\underline{\tau}$ be a vector- and a matrix-valued function, respectively, and let $(\mathbf{v}^\pm, \underline{\tau}^\pm)$ denote traces of $(\mathbf{v}, \underline{\tau})$ on e within the interior of $K^\pm$. At $\mathbf{x} \in e$, we then define the average operators as $\{\mathbf{v}\} = (\mathbf{v}^+ + \mathbf{v}^-)/2$ and $\{\underline{\tau}\} = (\underline{\tau}^+ + \underline{\tau}^-)/2$. The jump operator is defined as $\underline{[\![\mathbf{v}]\!]} = \mathbf{v}^+ \otimes \mathbf{n}_{K^+} + \mathbf{v}^- \otimes \mathbf{n}_{K^-}$, where the Kronecker product of vectors $\mathbf{v} \in \mathbb{R}^m$ and $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{v} \otimes \mathbf{w} \in \mathbb{R}^{m\times n}$, is defined as $(\mathbf{v} \otimes \mathbf{w})_{ij} = v_i w_j$. For a boundary face $e \in \Gamma$, we set $\{\mathbf{v}\} = \mathbf{v}, \{\underline{\tau}\} = \underline{\tau}$, and $\underline{[\![\mathbf{v}]\!]} = \mathbf{v} \otimes \mathbf{n}_K$. Finally, we define the Frobenius product of two matrices $\underline{\sigma}, \underline{\tau} \in \mathbb{R}^{m\times n}$ as $\underline{\sigma} : \underline{\tau} = \sum_{i=1}^m \sum_{j=1}^n \sigma_{ij}\tau_{ij}$.

The discontinuous approximate space we use is

$$\mathbf{V}_h := \{\mathbf{v} \in [L^2(\Omega)]^{d+2} : \mathbf{v}|_K \in P_p(K), \forall K \in \mathcal{T}_h\}, \tag{9}$$

where $P_p(K)$ is the set of polynomials of total degree at most p on K with $p \geqslant 1$. For $P_p(K)$, we choose a hierarchical basis presented in [26] and used in [20].

With $\mathbf{u}$ being approximated by $\mathbf{u}_h \in \mathbf{V}_h$, the DG discretization of Eq. (1) can be written as

$$\mathcal{N}(\mathbf{u}_h, \mathbf{v}_h) \equiv \mathcal{N}^c(\mathbf{u}_h, \mathbf{v}_h) + \mathcal{N}^v(\mathbf{u}_h, \mathbf{v}_h) = 0 \quad \forall \mathbf{v}_h \in \mathbf{V}_h, \tag{10}$$

[1] The terms “face” and “surface integral” denote edge and line integral in two space dimensions as well.

with $\mathcal{N}^c(\mathbf{u}_h, \mathbf{v}_h)$ and $\mathcal{N}^v(\mathbf{u}_h, \mathbf{v}_h)$ representing the discretization of the convective and viscous terms, respectively. The discretization of the convective term can be performed with different numerical fluxes such as Rusanov [3], Roe [4], HLL [5] and HLLC [6,7] fluxes. We use the discretization based on the Rusanov fluxes which is given as

$$\mathcal{N}^c(\mathbf{u}_h, \mathbf{v}_h) \equiv -\int_\Omega \underline{\mathcal{F}}^c(\mathbf{u}_h) : \nabla_h \mathbf{v}_h d\mathbf{x} + \int_{\Gamma_I} \{\underline{\mathcal{F}}^c(\mathbf{u}_h)\} : \underline{[\![\mathbf{v}_h]\!]} ds + \sum_K \int_{\partial K \setminus \Gamma} \frac{1}{2} \Lambda(\mathbf{u}^+, \mathbf{u}^-)(\mathbf{u}^+ - \mathbf{u}^-) \cdot \mathbf{v}^+ ds$$
$$+ \int_\Gamma \frac{1}{2} (\underline{\mathcal{F}}^c(\mathbf{u}_h^+) + \underline{\mathcal{F}}^c(\mathbf{u}_\Gamma(\mathbf{u}_\mathbf{h}^+))) : \underline{[\![\mathbf{v}_h]\!]} ds + \int_\Gamma \frac{1}{2} \Lambda_\Gamma(\mathbf{u})(\mathbf{u} - \mathbf{u}_\Gamma(\mathbf{u})) \cdot \mathbf{v} ds, \tag{11}$$

where Γ_I is the set of all interior faces,

$$\underline{\mathcal{F}}^c(\mathbf{u}_h) \equiv \mathbf{f}_1^c(\mathbf{u}), \ldots, \mathbf{f}_d^c(\mathbf{u})), \tag{12}$$

and

$$\Lambda(\mathbf{u}^+, \mathbf{u}^-) = \max(|\mathbf{u}^+ \cdot \mathbf{n}^+| + C(\mathbf{u}^+), |\mathbf{u}^- \cdot \mathbf{n}^-| + C(\mathbf{u}^-)), \tag{13a}$$
$$\Lambda_\Gamma(\mathbf{u}) = \max(|\mathbf{u} \cdot \mathbf{n}| + C(\mathbf{u}), |\mathbf{u}_\Gamma(\mathbf{u}) \cdot \mathbf{n}| + C(\mathbf{u}_\Gamma(\mathbf{u}))), \tag{13b}$$

for an interior and a boundary face, respectively. $C(\mathbf{u})$ is the speed of sound and is determined as $C(\mathbf{u}) = (\gamma p(\mathbf{u})/u_1)^{1/2}$.

The discretization of the viscous term is based on the interior penalty discretization of the linear Laplacian operator originally introduced in [27]. It has the form

$$\mathcal{N}^v(\mathbf{u}_h, \mathbf{v}_h) \equiv \int_\Omega \underline{\mathcal{F}}^v(\mathbf{u}_h, \nabla \mathbf{u}_h) : \nabla_h \mathbf{v}_h d\mathbf{x} - \int_{\Gamma_I} \{\underline{\mathcal{F}}^v(\mathbf{u}_h, \nabla \mathbf{u}_h)\} : \underline{[\![\mathbf{v}_h]\!]} ds - \int_{\Gamma_I} \{\underline{\mathcal{F}}^{v,T}(\mathbf{u}_h, \nabla \mathbf{v}_h)\} : \underline{[\![\mathbf{u}_h]\!]} ds$$
$$+ \int_{\Gamma_I} \nu \{G_{ii}\} \underline{[\![\mathbf{u}_h]\!]} : \underline{[\![\mathbf{v}_h]\!]} ds - \int_{\Gamma_{in}} \{\underline{\mathcal{F}}^v(\mathbf{u}_h, \nabla \mathbf{u}_h)\} : \underline{[\![\mathbf{v}_h]\!]} ds - \int_{\Gamma_{out}} \{\underline{\mathcal{F}}^{v,out}(\mathbf{u}_h, \nabla \mathbf{u}_h)\} : \underline{[\![\mathbf{v}_h]\!]} ds$$
$$- \int_{\Gamma_{wall}} \{\underline{\mathcal{F}}^{v,wall}(\mathbf{u}_h, \nabla \mathbf{u}_h)\} : \underline{[\![\mathbf{v}_h]\!]} ds - \int_\Gamma \{\underline{\mathcal{F}}^{v,T}(\mathbf{u}_h, \nabla \mathbf{v}_h)\} : (\mathbf{u}_h - \mathbf{u}_\Gamma(\mathbf{u}_h)) \otimes \mathbf{n} ds$$
$$+ \int_\Gamma \nu \{G_{ii}\} (\mathbf{u}_h - \mathbf{u}_\Gamma(\mathbf{u}_h)) \cdot \mathbf{v}_h ds (i = 1, \ldots, d), \tag{14}$$

where

$$\underline{\mathcal{F}}^v(\mathbf{u}_h, \nabla \mathbf{v}_h) \equiv \left(G_{1i}(\mathbf{u}_h) \frac{\partial_h \mathbf{v}_h}{\partial x_i}, \ldots, G_{di}(\mathbf{u}_h) \frac{\partial_h \mathbf{v}_h}{\partial x_i} \right) \quad (i = 1, \ldots, d) \tag{15a}$$

$$\underline{\mathcal{F}}^{v,T}(\mathbf{u}_h, \nabla \mathbf{v}_h) \equiv \left(G^T_{1i}(\mathbf{u}_h) \frac{\partial_h \mathbf{v}_h}{\partial x_i}, \ldots, G^T_{di}(\mathbf{u}_h) \frac{\partial_h \mathbf{v}_h}{\partial x_i} \right) \quad (i = 1, \ldots, d) \tag{15b}$$

$$\underline{\mathcal{F}}^{v,out}(\mathbf{u}_h, \nabla \mathbf{u}_h)_{ij} = 0 \quad (i = 1, \ldots, d+1 \text{ and } j = 1, \ldots, d), \tag{16a}$$
$$\underline{\mathcal{F}}^{v,out}(\mathbf{u}_h, \nabla \mathbf{u}_h)_{ij} = \underline{\mathcal{F}}^v(\mathbf{u}_h, \nabla \mathbf{u}_h)_{ij}, \quad (i = d+2 \text{ and } j = 1, \ldots, d), \tag{16b}$$

$$\underline{\mathcal{F}}^{v,wall}(\mathbf{u}_h, \nabla \mathbf{u}_h)_{ij} = \underline{\mathcal{F}}^v(\mathbf{u}_h, \nabla \mathbf{u}_h)_{ij} \quad (i = 2, \ldots, d+1 \text{ and } j = 1, \ldots, d), \tag{17a}$$
$$\underline{\mathcal{F}}^{v,wall}(\mathbf{u}_h, \nabla \mathbf{u}_h)_{ij} = 0 \quad (i = 1, i = d+2 \text{ and } j = 1, \ldots, d). \tag{17b}$$

The fourth and last integral of Eq. (14), referred to as penalty terms, are added to enforce the coercivity of the discretization. ν, in the penalty terms, is called the penalty parameter. The penalty parameter must be chosen sufficiently large to ensure the coercivity of the discretization of the viscous term. While it has been known that the value of ν depends on the underlying mesh geometry and the approximation order, an explicit expression for the minimum acceptable value of the penalty parameter was lacking and ν was only defined within a multiplicative constant [27]. However, recently, an explicit expression for the penalty parameter of the linear Poisson equation was derived [25], removing the undesirable empiricism in defining the penalty parameter. For the Navier–Stokes equations, we use the same expression. Specifically, for a face e shared with two elements K^+ and K^-, ν is given as

$$\nu = \frac{(p+1)(p+d)}{2d} \max\left(\frac{S_{K^+}}{V_{K^+}}, \frac{S_{K^-}}{V_{K^-}} \right), \tag{18}$$

where S_K and V_K represent the surface area (perimeter in two dimensions) and volume (area in two dimensions) of the element K, respectively. For an outer Dirichlet boundary face $e \in \partial K$, the penalty parameter is

$$\nu = \frac{(p+1)(p+d)}{d} \left(\frac{S_K}{V_K} \right). \tag{19}$$

Remark 1. Slightly different formulation of the viscous term was used by Hartmann and Houston [1]. In their formulation, the penalty term lacked $\{G_{ii}\}$ and they chose the penalty parameter to be sufficiently large to ensure the coercivity of the discretization.

Remark 2. The above discretization is only valid for $p \geqslant 1$. However, for the multigrid solver presented in Section 4, we require to have the DG discretization for the piecewise constant space, $p = 0$. While the consistent discretization of the convection terms for $p = 0$ is achieved by setting $\mathbf{v}_h = 1$ and $\nabla_h \mathbf{v}_h = 0$ in (11), a $p = 0$ consistent discretization of the viscous term is not readily available due to the cross derivatives in the compressible viscous operator. We use the following "inconsistent" form for the $p = 0$ discretization:

$$\mathcal{N}^{v}(\mathbf{u}_h, \mathbf{v}_h) \equiv \int_{\Gamma_I} v_0 \{G_{ii}\} \underline{[\![\mathbf{u}_h]\!]} : \underline{[\![\mathbf{v}_h]\!]} ds + \int_{\Gamma} v_0 \{G_{ii}\} (\mathbf{u}_h - \mathbf{u}_{\Gamma}(\mathbf{u}_h)) \cdot \mathbf{v}_h, ds \tag{20}$$

where $\mathbf{v}_h$ is the unit constant vector. v_0 for $e \in \partial K^+ \cap \partial K^-$ is defined as the summation of the normal distance of the centroid of elements K^+ and K^- from face e. If e is an outer boundary face, $e \in \partial K$, v_0 is set as the normal distance of the centroid of the element K from e. As will be shown in the verification section, this discretization is effective when used on the $p = 0$ coarse level approximation of the multigrid sequence.

4. Single-grid solvers

Let us rewrite the nonlinear system arising from the DG discretization of equation (10) as

$$\mathcal{R}(\mathbf{u}_h) = 0. \tag{21}$$

To solve this equation, we linearize the equation using Newton's method to obtain the following iterative scheme:

$$\left[\frac{\partial \mathcal{R}}{\partial \mathbf{u}_h}\right]^n \Delta \mathbf{u}_h^{n+1} = -\mathcal{R}(\mathbf{u}_h^n), \tag{22a}$$
$$\mathbf{u}_h^{n+1} = \mathbf{u}_h^n + \alpha \Delta \mathbf{u}_h^{n+1}, \tag{22b}$$

where $\mathbf{u}_h^n$ is the approximation of $\mathbf{u}_h$ at step n and α is a damping factor to confine the growth in $\Delta \mathbf{u}_h^{n+1}$. For instance, α can be chosen as

$$\alpha = \min\left(0.1 \| \frac{\mathbf{u}_h^n}{\Delta \mathbf{u}_h^{n+1}} \|_{L_\infty}, 1\right). \tag{23}$$

Solving the linear system (22a) requires the inversion of the Jacobian matrix $\left[\frac{\partial \mathcal{R}}{\partial \mathbf{u}_h}\right]^n$ which is prohibitively expensive for large problems. An efficient strategy is the so-called nonlinear element Jacobi (NEJ) approach in which the full Jacobian matrix is approximated with its diagonal blocks corresponding to each element of the triangulation that is

$$\left[\frac{\partial \mathcal{R}}{\partial \mathbf{u}_h}\right]^n \approx [D]^n = \begin{bmatrix} d_1^n & & & & \\ & \cdot & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & d_N^n \end{bmatrix},$$

where N is the total number of elements and d_i^n represents the black-diagonal matrix of the Jacobian matrix corresponding to the degrees of freedoms of the element i. The arising system is then solved iteratively as

$$[D]^n \Delta \mathbf{u}_h^{n+1} = -\mathcal{R}(\mathbf{u}_h^n), \tag{24a}$$
$$\mathbf{u}_h^{n+1} = \mathbf{u}_h^n + \alpha \Delta \mathbf{u}_h^{n+1}. \tag{24b}$$

Note that the above procedure is implemented in the element-by-element basis, and the matrix $[D]^n$ for each element is easily invertible.

A more efficient scheme, namely the quasi nonlinear element Jacobi (qNEJ) approach, can be derived by evaluating $[D]^n$ only after every m steps, where $m > 1$. This leads to the following solution procedure:

$$[D]^n \Delta \mathbf{u}_h^{k+1} = -\mathcal{R}(\mathbf{u}_h^k), \tag{25a}$$
$$\mathbf{u}_h^{k+1} = \mathbf{u}_h^k + \alpha \Delta \mathbf{u}_h^{k+1}, \tag{25b}$$
$$k = n, \dots, n + m.$$

Another efficient solution strategy for system (22a) is formed by decomposing the Jacobian matrix for each element into two sub-matrices, namely block-diagonal and off-diagonal matrices as

$$\left[\frac{\partial \mathcal{R}}{\partial \mathbf{u}_h}\right]^n \equiv [D]^n + [O]^n, \tag{26}$$

and then carrying out the following iterations:

$$[D]^n \Delta \mathbf{u}_h^{k+1} = -\mathcal{R}(\mathbf{u}_h^n) - [O]^n \Delta \mathbf{u}_h^k, \quad k = 1, \ldots, m. \tag{27}$$

This scheme is referred to as the linearized element Jacobi (LEJ) method, and involves a dual iteration strategy, where for each nth outer iteration, m inner iterations are required with the fixed Jacobian sub-matrices and the fixed residual. This infrequent evaluation of the expensive nonlinear residuals leads to a significant reduction in the required total computational time. Further refinement of the LEJ scheme can be achieved by replacing $\Delta \mathbf{u}_h^k$ on the right-hand side of Eq. (27) with the most recent updates $(\Delta \mathbf{u}_h^*)$. This last method is called the linearized Gauss–Seidel (LGS) method and reads

$$[D]^n \Delta \mathbf{u}_h^{k+1} = -\mathcal{R}(\mathbf{u}_h^n) - [O]^n \Delta \mathbf{u}_h^*, \quad k = 1, \ldots, m. \tag{28}$$

For LGS, the grid elements must be swept (numbered) in an orderly fashion. In this work, we employ a frontal sweep along the elements which begins near the inner boundary and proceeds toward the outer boundary [8].

Although the above elemental iterative methods are effective for isotropic meshes, their performance degrades dramatically for highly anisotropic meshes commonly used in resolving thin boundary layers at high-Reynolds-number flow computations. To alleviate the stiffness associated with stretched grids, we therefore require to treat implicitly clusters of elements with strongest coupling, namely to use an elemental-line-implicit solver. Variants of line-implicit relaxation methods have been successfully used in finite-volume solvers [19] as well as in the DG setting [13]. We herein follow the algorithm described in [19] to construct lines of elements, and we use a block variant of Thomas algorithm to efficiently solve the resulting local systems [31].

5. Multigrid solver

While the above iterative solvers are highly effective in damping the non-smooth or rough part of the error, they are not efficient in reducing the smooth (long wavelength) part of the error. The classical remedy to this problem is to devise a multigrid solver in which the smooth part of the error is approximated on coarser grids. The non-smooth part is then reduced with small numbers of iterations on the fine grid.

Any of the above mentioned iterative methods can be used as a smoothing operation. However, it is well-known that, when a point smoother (e.g. point Jacobi) is applied in the multigrid solution of elliptic problems, a damping factor may be required in order to achieve good high frequency damping properties [11]. Our numerical experiments with the element Jacobi solvers (NEJ, qNEJ and LEJ) also verify the need for a damping factor. Thus, when the NEJ or qNEJ is used as a smoother, we use a damping factor of 0.85 which appears to yield optimal performance. Note that unlike the Jacobi scheme, the Gauss–Seidel method does not require any damping factor when used as a smoother.

5.1. Coarse grid approximation

For the high-order DG discretization of the Navier–Stokes equations, coarse grid approximation spaces are formed by incrementally reducing the approximation order while maintaining the same computational mesh. The resulting coarse levels and multigrid are referred to as p-levels and p-multigrid, respectively. Upon reaching the $p = 0$ approximation, further coarse spaces (h-levels) are obtained by coarsening the mesh, but holding the approximation order $p = 0$ fixed (h-multigrid). The mesh coarsening is performed through an agglomeration technique which makes use of an automatically generated sequence of coarser level meshes, formed by merging together neighboring fine grid elements [8]. We denote the total number of h-levels by H, the total number of p-levels by $p + 1$, and the total number of both h-level and p-levels by $L = p + H + 1$.

5.2. Nonlinear and linear formulations

For solving a nonlinear problem using multigrid, two formulations are possible. The multigrid algorithm can be applied to the nonlinear system directly, or it can be used on the linearized form of the nonlinear system. The former is referred to as the nonlinear multigrid, while the latter is called the linear multigrid scheme. In this work, we consider both formulations. A description of the nonlinear and linear two-grid formulations is presented in Algorithms 1 and 2, respectively. In Algorithms 1 and 2, the superscript (or subscript) l signifies the approximation level. $\mathcal{S}\left(\mathbf{u}_h^l, \tilde{\mathbf{u}}_h^l, \frac{\partial \mathcal{R}^l(\tilde{\mathbf{u}}_h^l)}{\partial \tilde{\mathbf{u}}_h^l}, \mathcal{R}^l(\tilde{\mathbf{u}}_h^l), \mathbf{f}^l, \beta^l\right)$ represents the smoothing algorithm to obtain $\mathbf{u}^l$ from an initial guess $\tilde{\mathbf{u}}_h^l$ using β^l number of smoothing iterations with $\mathbf{f}^l$ being the source term.

Algorithm 1. Nonlinear two-grid formulation

```
1:  for $i = 1$ to # of nonlinear cycles do
2:    Evaluate $\mathcal{R}^l(\tilde{\mathbf{u}}_h^l)$
3:    Evaluate $\frac{\partial \mathcal{R}^l(\tilde{\mathbf{u}}_h^l)}{\partial \tilde{\mathbf{u}}_h^l}$.
4:    $\mathcal{S}\left(\mathbf{u}_h^l, \tilde{\mathbf{u}}_h^l, \frac{\partial \mathcal{R}^l(\tilde{\mathbf{u}}_h^l)}{\partial \tilde{\mathbf{u}}_h^l}, \mathcal{R}^l(\tilde{\mathbf{u}}_h^l), \mathbf{f}^l, v^l\right)$
5:    $\mathbf{r}^l = \mathbf{f}^l - \mathcal{R}^l(\mathbf{u}_h^l)$
6:    $\tilde{\mathbf{u}}_h^{l-1} = \mathbf{I}_l^{l-1}\mathbf{u}_h^l$
7:    Evaluate $\mathcal{R}^{l-1}(\tilde{\mathbf{u}}_h^{l-1})$
8:    Evaluate $\frac{\partial \mathcal{R}^{l-1}(\tilde{\mathbf{u}}_h^{l-1})}{\partial \tilde{\mathbf{u}}_h^{l-1}}$.
9:    $\mathbf{f}^{l-1} = \mathcal{R}^{l-1}(\tilde{\mathbf{u}}_h^{l-1}) + \tilde{\mathbf{I}}_l^{l-1}\mathbf{r}^l$
10:   $\mathcal{S}\left(\mathbf{u}_h^{l-1}, \tilde{\mathbf{u}}_h^{l-1}, \frac{\partial \mathcal{R}^{l-1}(\tilde{\mathbf{u}}_h^{l-1})}{\partial \tilde{\mathbf{u}}_h^{l-1}}, \mathcal{R}^{l-1}(\tilde{\mathbf{u}}_h^{l-1}), \mathbf{f}^{l-1}, v^{l-1}\right)$
11:   $\mathbf{u}_h^l = \mathbf{u}_h^l + \mathbf{I}_{l-1}^l(\mathbf{u}_h^{l-1} - \tilde{\mathbf{u}}_h^{l-1})$
12: end for
```

Algorithm 2. Linear two-grid formulation

```
1:  for$i = 1$ to # of nonlinear cycles do
2:    Evaluate $\mathcal{R}^l(\tilde{\mathbf{u}}_h^l)$
3:    Evaluate $\frac{\partial \mathcal{R}^l(\tilde{\mathbf{u}}_h^l)}{\partial \tilde{\mathbf{u}}_h^l}$
4:    $\tilde{\mathbf{u}}_h^{l-1} = \mathbf{I}_l^{l-1}\tilde{\mathbf{u}}_h^l$
5:    Evaluate $\frac{\partial \mathcal{R}^{l-1}(\tilde{\mathbf{u}}_h^{l-1})}{\partial \tilde{\mathbf{u}}_h^{l-1}}$
6:    for $j = 1$ to # of linear cycles do
7:      $\mathcal{S}\left(\Delta\mathbf{u}_h^l, \Delta\tilde{\mathbf{u}}_h^l, \frac{\partial \mathcal{R}^l(\tilde{\mathbf{u}}_h^l)}{\partial \tilde{\mathbf{u}}_h^l}, \mathcal{R}^l(\tilde{\mathbf{u}}_h^l), \mathbf{f}^l, \beta^l\right)$
8:      $\mathbf{r}^l = \mathbf{f}^l - \mathcal{R}^l(\tilde{\mathbf{u}}_h^l) - \frac{\partial \mathcal{R}^l(\tilde{\mathbf{u}}_h^l)}{\partial \tilde{\mathbf{u}}_h^l}\Delta\mathbf{u}_h^l$
9:      $\mathbf{f}^{l-1} = \tilde{\mathbf{I}}_l^{l-1}\mathbf{r}^l$
10:     $\mathcal{S}\left(\Delta\mathbf{u}_h^{l-1}, 0, \frac{\partial \mathcal{R}^{l-1}(\tilde{\mathbf{u}}_h^{l-1})}{\partial \tilde{\mathbf{u}}_h^{l-1}}, 0, \mathbf{f}^{l-1}, \beta^{l-1}\right)$
11:     $\Delta\mathbf{u}_h^l = \Delta\mathbf{u}_h^l + \mathbf{I}_{l-1}^l\Delta\mathbf{u}_h^{l-1}$
12:   end for
13:   $\mathbf{u}_h^l = \tilde{\mathbf{u}}_h^l + \Delta\mathbf{u}_h^l$
14: end for
```

$\mathbf{I}_{l-1}^l$ represents a prolongation operator which interpolates the coarse $(l-1)$ grid values to the fine (l) grid. On the other hand, $\mathbf{I}_l^{l-1}$ and $\tilde{\mathbf{I}}_l^{l-1}$ signify restriction operators which map the fine (l) grid values and residual to the coarse grid $(l-1)$. Since we use a hierarchical basis for $p \geqslant 1$ approximation orders, restriction and prolongation operators for $p \geqslant 1$ are highly simplified. Let a coarse level have N_p modal coefficients. Then the restriction from a fine to a coarse level is obtained by injecting the first N_p modal coefficients of the fine level to the coarse level. Similarly, the prolongation from a coarse to a fine level is obtained by injecting the first N_p modal coefficients exactly and setting the remaining high order modes to zero. We choose $\mathbf{I}_l^{l-1}$ and $\tilde{\mathbf{I}}_l^{l-1}$ to be identical for spaces with $p \geqslant 1$. The prolongation from a $p = 0$ level is simply an injection operator. The restriction to a $p = 0$ level is the volume-based weighted average of all finer level values for $\mathbf{I}_l^{l-1}$ and the summation of all the finer level residuals for $\tilde{\mathbf{I}}_l^{l-1}$ [20].

While the coarse level Jacobian in the nonlinear multigrid (Algorithm 1, line 8) is obtained from the direct discretization on the coarse space, in the linear multigrid formulation, it (Algorithm2, line 3) is obtained through the Galerkin projection as

$$\frac{\partial \mathcal{R}^{L-i}(\tilde{\mathbf{u}}_h^{L-i})}{\partial \tilde{\mathbf{u}}_h^{L-i}} \approx \tilde{\mathbf{I}}_L^{L-i}\frac{\partial \mathcal{R}^L(\tilde{\mathbf{u}}_h^L)}{\partial \tilde{\mathbf{u}}_h^L}\mathbf{I}_{L-i}^L \quad L > H+1, i < p. \tag{29}$$

Since we use a hierarchical basis, the Galerkin projection, Eq. (29), is carried out without any operation counts or memory requirements. Specifically, assuming the coarse space has N_p dimensions, the coarse level Jacobian matrix is chosen as the sub-matrix of the first N_p modes of the finest level Jacobian matrix.

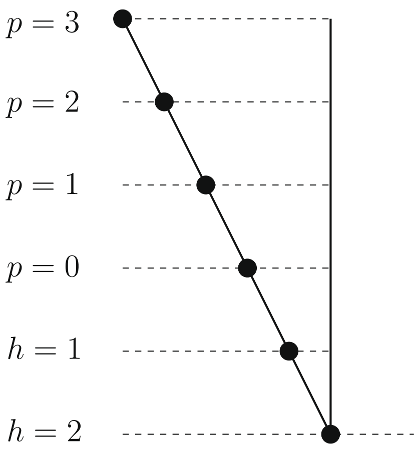

Fig. 1. A saw-tooth V-cycle with four p-levels and two h-levels. A bold dot represents a smoothing operation.

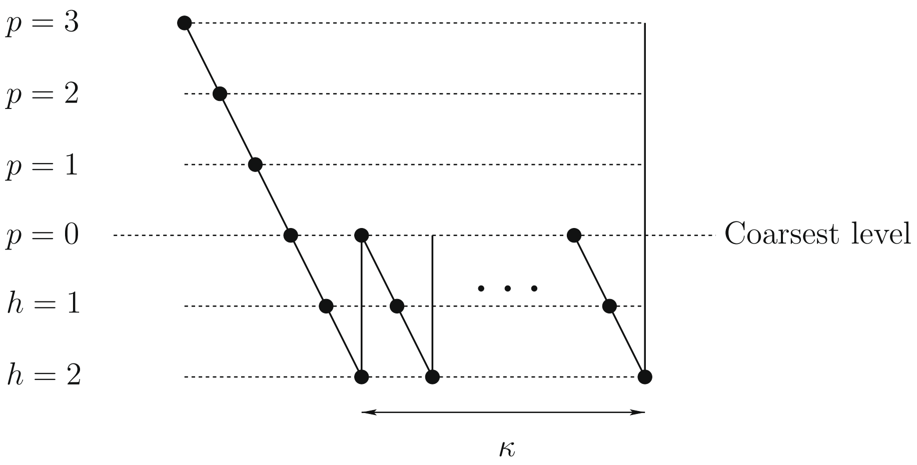

Fig. 2. The first cycling strategy, a dual saw-tooth V-cycle formed by a V-cycle over the $p = 3$ approximation space to the $p = 0$ and κ V-cycles associated with solving the $p = 0$ problem with two coarse h-levels. A bold dot represents a smoothing operation.

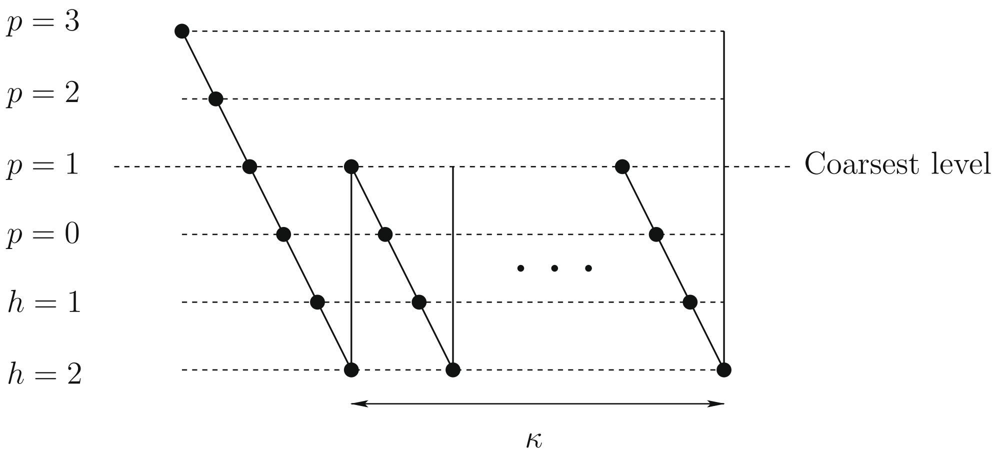

Fig. 3. The second cycling strategy, a dual saw-tooth V-cycle formed by a V-cycle over the $p = 3$ approximation space to the $p = 1$ and κ V-cycles associated with solving the $p = 1$ problem with two coarse h-levels. A bold dot represents a smoothing operation.

In Algorithm 2, each nonlinear cycle for the linear multigrid formulation requires inner iterations for solving the linear system. If the number of inner iterations is chosen sufficiently large (the linear system is solved to sufficient accuracy), the quadratic rate of convergence of the Newton method can be observed. This can substantially reduce the number of evaluations of the expensive residual and Jacobian matrix, compared to the nonlinear multigrid formulation. On the other hand, solving the linear system to the machine zero at each nonlinear cycle leads to a less than optimal scheme. Thus, provided a suitable linear system tolerance is used, the linear multigrid formulation can be more efficient than the nonlinear counterpart. This will be confirmed with the numerical experiments in the result section.

5.3. Multigrid cycle

We intend to couple both p- and h-level spaces to construct a ph-multigrid cycle, and we require the coupling be performed in a fashion that yields an optimal, or a near optimal solver. There are several cycling strategies that can be used in a ph-multigrid cycle. A simple strategy is to use a saw-tooth V-cycle over all p- and h-levels (Fig. 1). In the saw-tooth V-cycle, a fixed (small) number of smoothing iterations is performed on each approximation level, before restricting the problem on the next coarse level (pre-smoothing). The smoothing operations after fine level prolongations (post-smoothing) are deleted [11].

Although this cycling strategy can lead to optimal convergence rates for hyperbolic problems [12], it is less than optimal for elliptic problems. Here optimality refers to the ability to maintain p- and h-independent convergence rates. For elliptic

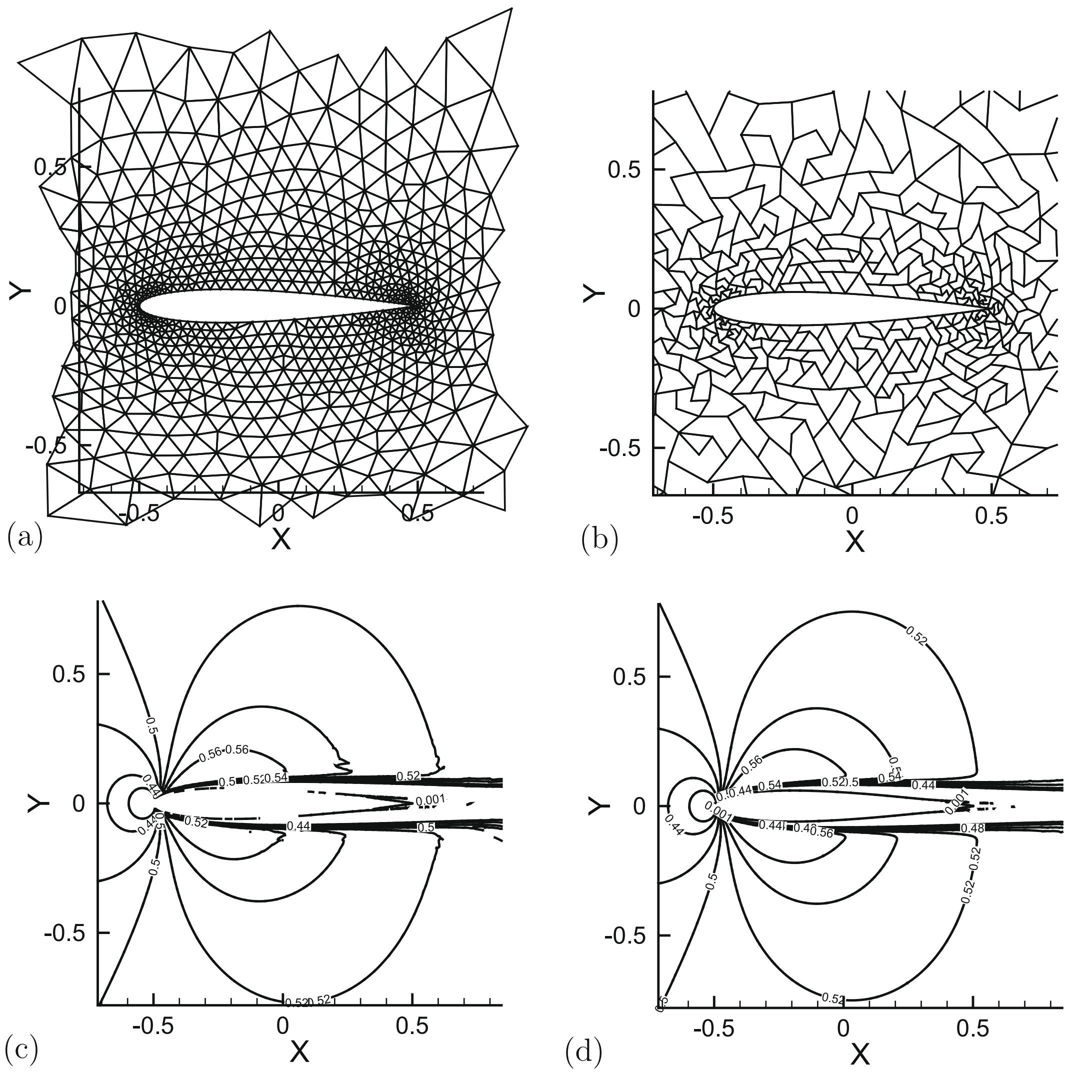

Fig. 4. (a) A view of the isotropic mesh with $N = 1822$ triangles for the NACA0012 problem; (b) a view of the one level agglomeration of the same mesh; (c) and (d) the Mach number contour lines for flow around the NACA0012 airfoil at $Re = 5000$ and $M = 0.5$ for approximation orders $p = 3$ and 5, respectively.

problems, the coarsest level problem must be solved to sufficient accuracy to yield optimal convergence rates. Thus, we propose a cycle in which a saw-tooth V-cycle is formed from the finest p-level to the $p = 0$ approximation level on the original fine mesh, and this coarse problem itself is solved using several cycles of an h-multigrid saw-tooth V-cycle as shown in Fig. 2. This cycle is referred to as the first cycle type (the first cycling strategy), hereafter. As will be shown in the verification section, this cycling strategy is not robust for high-Reynolds-number flows. Thus, we propose an alternative cycle which is similar to the second cycle, but the coarsest level now is the $p = 1$ approximation space which itself is solved using several cycles of a ph-multigrid cycle as shown in Fig. 3. This last variant is called the second cycle type (or the second cycling strategy), hereafter, and shown to result in superior performance in the result section.

6. Newton–GMRES solver

An alternative approach for solving the linear system arising from the Newton iterations is to use a Krylov subspace method. Since the system arising from the Navier–Stokes discretization is nonsymmetric, we choose to use the Generalized Minimal Residual method (GMRES) [14]. To accelerate the convergence of the GMRES solver, a preconditioned variant is often used. In this work, we use the linear multigrid algorithm described earlier as a preconditioner for the GMRES solver. A brief description of the method is as follows.

Let us rewrite the linearized system (22a) as

$$Ax = b, \tag{30}$$

where A denotes the Jacobian matrix, x a vector of unknowns and b a known right-hand-side vector. In a right preconditioned system, one effectively solves the following system:

$$AM^{-1}y = b, \quad x = M^{-1}y, \tag{31}$$

where M is a preconditioner matrix which should be easily invertible. In the $m - 1$ iteration of the GMRES method, the vector y is approximated using the orthonormal basis of the m dimensional Krylov subspace

$$\mathcal{K} = \text{Span}\{r_0, AM^{-1}r_0, \ldots, (AM^{-1})^{m-1}r_0\}, \tag{32}$$

such that the L^2 norm of the residual $(b - AM^{-1}y)$ is minimized. r_0 represents the initial residual associated with an initial guess for the unknown vector.Computing the Krylov subspace basis involves solving a linear system (preconditioner system) of the form

$$Mz = c \tag{33}$$

at each GMRES iteration. We propose to solve such a system approximately using a few cycles of the multigrid algorithms introduced in the previous section. Since the preconditioner system is not solved exactly, the preconditioner matrix is not strictly fixed for different GMRES iterations, and we require the use of the so-called flexible implementation of GMRES (fGMRES) [15].

Note that the GMRES algorithm requires additional storage for the Krylov basis compared to pure multigrid solvers. However, this is not a very restrictive issue, since our multigrid algorithm with the second cycle type constitutes a highly effective preconditioner for the GMRES solver, yielding the convergence of GMRES solutions in a moderately small number of itera-

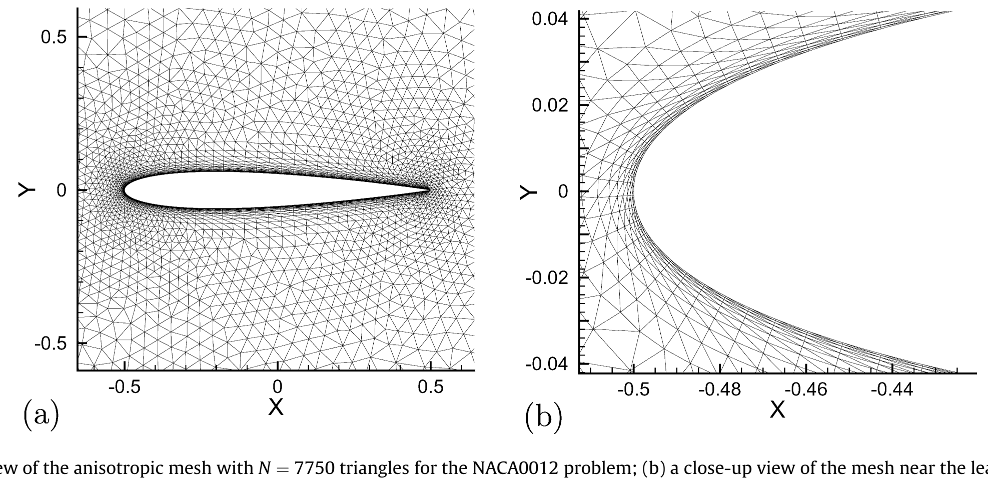

Fig. 5. (a) A view of the anisotropic mesh with $N = 7750$ triangles for the NACA0012 problem; (b) a close-up view of the mesh near the leading edge of the airfoil.

tions; that is, the size of the Krylov basis remains small and the memory requirement remains low. This is verified in the next section.

7. Numerical results

We examined the performance of the above described algorithms in simulating the two-dimensional symmetric laminar flow around the NACA0012 airfoil with a freestream Mach number of $M = 0.5$, a Reynolds number of $Re = 5000$ (based on the free stream velocity and the airfoil chord length of unity) and zero degrees incidence. The geometry of the airfoil was defined based on an analytic function.

We considered both isotropic and anisotropic meshes. For the isotropic case, we used an isotropic mesh with $N = 1822$ triangles, its approximately one level refinement $N = 7701$ and an isotropic mesh with $N = 16061$ triangles. For anisotropic meshes, we chose two meshes with $N = 2250$ and 7750 with maximum aspect ratios of 82 and 235, respectively. A view of the mesh with $N = 1822$ triangles and its one level agglomeration are shown in Fig. 4(a) and (b), respectively. Fig. 4(c) and (d) shows the Mach contour lines for the 1822 triangular grid for approximation orders $p = 3$ and 5, respectively. Fig. 4(c) and (d) show that as the approximation order increases, smoother contour lines, or equivalently more accurate results, are obtained. Views of the anisotropic mesh with $N = 7750$ are shown in Fig. 5(a) and (b).

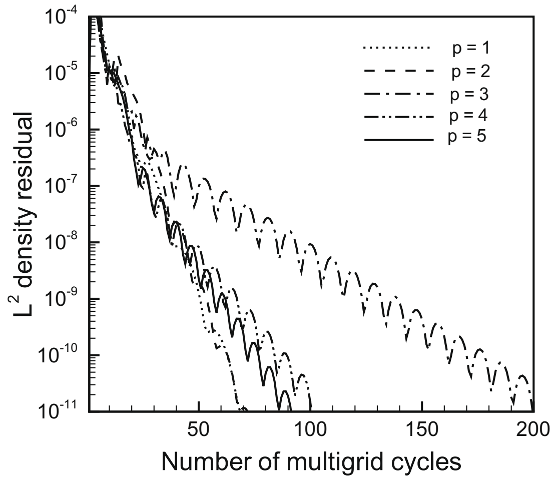

Fig. 6. p-Convergence of the nonlinear multigrid formulation with the first cycling strategy in solving the Navier–Stokes equations for the NACA0012 airfoil at $Re = 5000$. L^2 norm of the density equation residual vs. the number of nonlinear multigrid cycles for approximation orders $p = 1, \ldots, 5$ for the mesh with $N = 1822$ triangles.

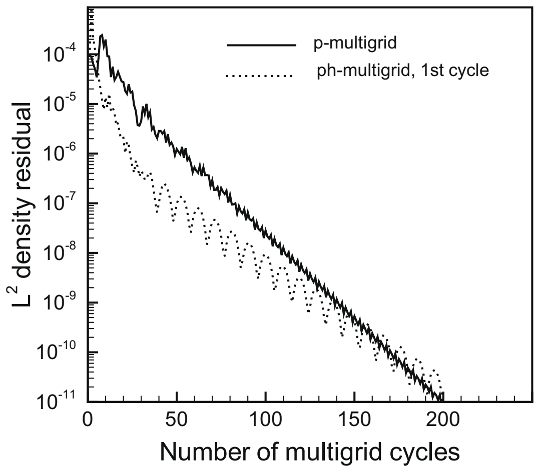

Fig. 7. Convergence comparison of p-multigrid and ph-multigrid with the first cycle type in simulating flow around NACA0012 airfoil at $Re = 5000$. L^2 norm of the density equation residual vs. the number of nonlinear multigrid cycles for the approximation order $p = 3$ and the mesh with $N = 1822$ triangles.

Note that all of the below numerical experiments were carried out using the constant farfield data as the initial solution. In order to provide an improved initial guess, we augment the solution process with a full multigrid (FMG) technique [11]. The use of FMG is particularly critical in the case of linear solvers (the linear multigrid or the Newton–GMRES solver) for it is known that the Newton iterations diverge if the initial guess is not sufficiently close to the initial solution. In FMG, the coarse level problems are solved approximately using several iterations/cycles prior to iterating on the fine level problem, thereby increasing the robustness by improving the starting solution of the finer level approximation. We chose the $p = 1$ approximation as the coarsest level of FMG. We use twenty iterations/cycles on the all coarse FMG levels when using the single-grid solver, and two iterations/cycles on all FMG levels for the nonlinear multigrid algorithms. For the linear multigrid or the Newton–GMRES solves, we used ten iterations on the $p = 1$ approximation level and two multigrid cycles on all other higher levels. Also note that all numerical tests were performed using ten smoothing iterations. In the *ph*-multigrid algorithm, the total number of *h*-levels was chosen $H = 3$ and 4 for the two meshes with 1822 ($N = 2250$) and 7701 (7750) triangles, respectively. For the mesh with $N = 16061$ triangles, $H = 5$.

For the isotropic meshes, only results using the element Gauss–Seidel smoother are reported, since this smoother performs better than the element Jacobi smoother [20]. For the anisotropic meshes, the results are based on the

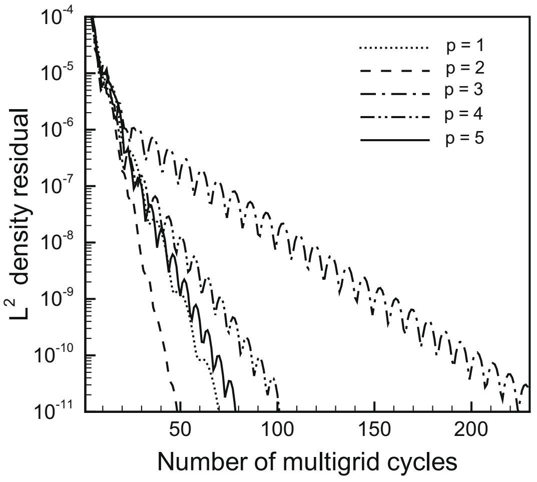

Fig. 8. *p*-Convergence of the nonlinear multigrid formulation with the first cycling strategy in solving the "modified" Navier–Stokes equations (the viscous terms replaced with the Laplacian) for the NACA0012 airfoil at $Re = 5000$. L^2 norm of the density equation residual vs. the number nonlinear multigrid cycles for approximation orders $p = 1, \dots, 5$ and the mesh with $N = 1822$ triangles.

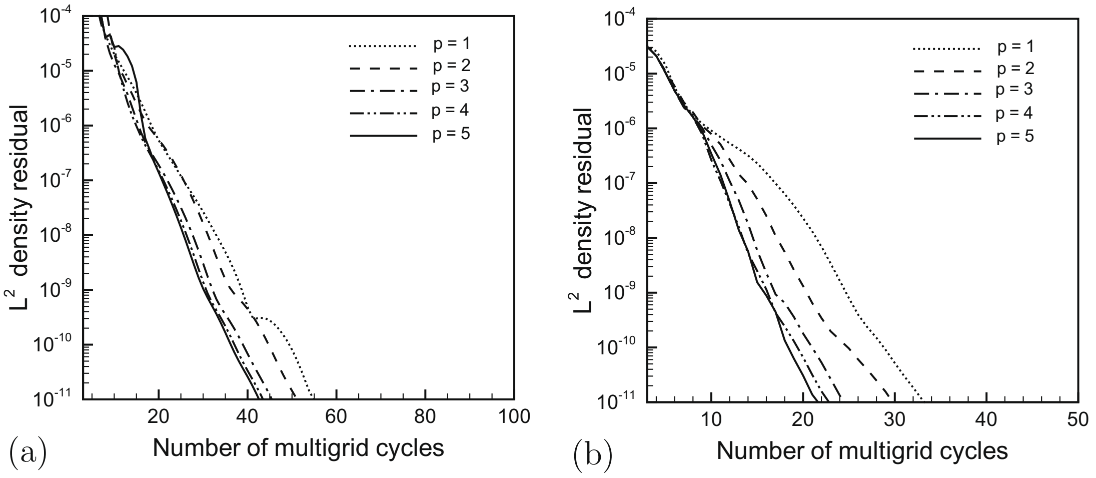

Fig. 9. *p*-Convergence of the nonlinear multigrid formulation with the first cycling strategy. L^2 norm of the density equation residual vs. the number of nonlinear multigrid cycles for approximation orders $p = 1, \dots, 5$ and the mesh with $N = 1822$ triangles. (a) Navier–Stokes equations with $Re = 500$ and no-slip wall boundary conditions; (b) Navier–Stokes equations with $Re = 5000$ and Neumann wall boundary conditions.

elemental-line-implicit smoother in the anisotropic region of the mesh (near the wall) and element Jacobi smoother in the isotropic region.

7.1. Nonlinear multigrid results

For the nonlinear multigrid formulation, we first show that the first cycling strategy (Fig. 2) yields sub-optimal p-convergence rates. We then explore the cause of this problem and examine the performance of the alternative cycling strategy, the second cycle type (Fig. 3), with respect to approximation orders and mesh sizes.

We first solved the Navier–Stokes equations using the nonlinear ph-multigrid formulation with the first cycle type (Fig. 2) for the NACA0012 airfoil problem at $Re = 5000$. Fig. 6 shows the p-convergence for the isotropic mesh with 1822 triangular elements and a range of approximation orders $p = 1, \ldots, 5$. As is clear from the figure, similar convergence rates are observed for all orders except for $p = 3$. Note that using an approximately fourfold finer mesh ($N = 7701$) yields analogous results. To investigate the cause of the large discrepancy in the convergence behaviors, we compared the ph-multigrid with pure p-multigrid with $p = 1$ as the coarsest level for the same test case, and for $p = 3$. The results are depicted in Fig. 7. It is evident that adding the h-levels to the p-multigrid not only did not reduce the number of iterations, but it even worsened the asymptotic convergence rate. One may then suspect that inconsistency of the $p = 0$ discretization for the viscous term might have

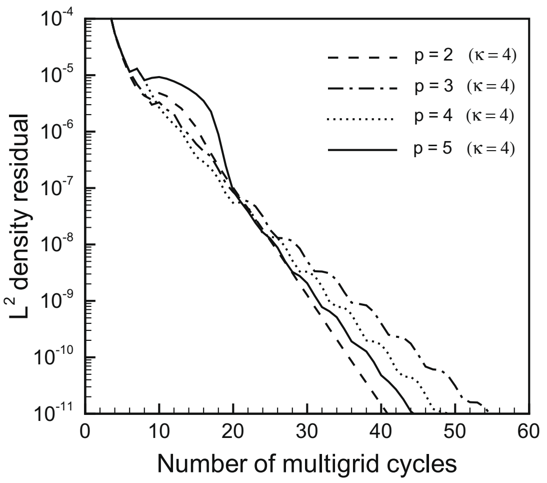

Fig. 10. p-Convergence of the nonlinear multigrid formulation with the second cycling strategy in simulating flow around the NACA0012 airfoil $Re = 5000$. L^2 norm of the density equation residual vs. the number of nonlinear multigrid cycles for approximation orders $p = 2, \ldots, 5$ and the mesh with 1822 triangles.

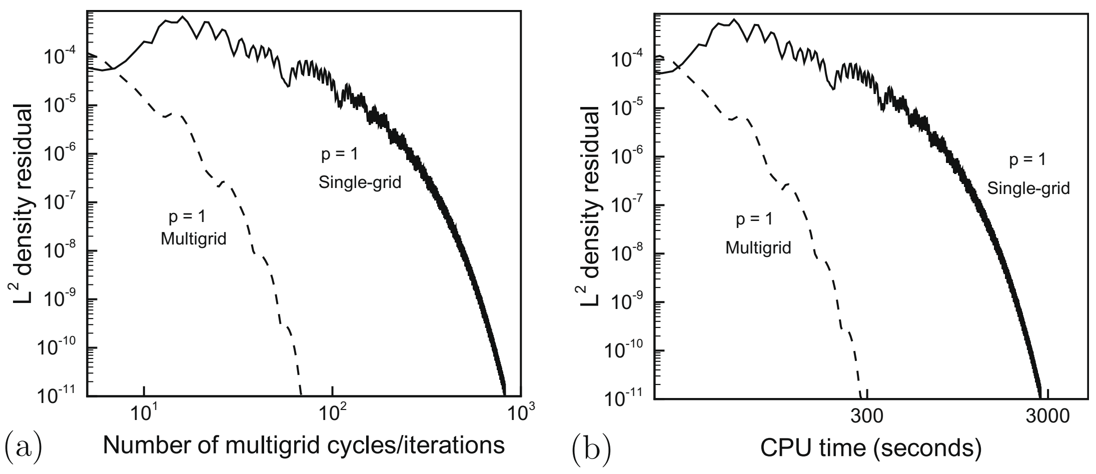

Fig. 11. Efficiency of the ph-multigrid over the single-grid solver for $p = 1$. (a) L^2 norm of the density equation residual vs. the number of nonlinear multigrid cycles or the number of iterations; (b) L^2 norm of the density equation residual vs. the CPU time (s).

caused the lower convergence rate. To examine this possibility, we replaced the viscous term in the Navier–Stokes equations with the Laplacian operator for all conservative variables, which has a consistent $p = 0$ discretization, and repeated the p-convergence experiment. The results are plotted in Fig. 8, showing the same non-uniform convergence rates as that of solving the original Navier–Stokes equations (1).

Having confirmed that the inconsistent $p = 0$ discretization of the Navier–Stokes equations is not the reason for the undesirable convergence rate, we examined the possibility that large directional gradients due to the thin boundary layers at high Reynolds numbers are the cause of this non-uniform convergence behavior. To investigate the effect of boundary layers on the convergence behavior, we carried out two tests. First, we ran the *ph*-multigrid solver for the same test, but at a lower Reynolds number of $Re = 500$ (which features smaller gradients or thicker boundary layers compared to $Re = 5000$). Secondly, we kept $Re = 5000$, but instead of the no-slip boundary conditions (8c) at the wall, we imposed Neumann boundary conditions to artificially remove the boundary layer. Results for both tests are displayed in Fig. 9(a) and (b), respectively. Unlike the former cases, almost uniform convergence rates were observed for all polynomial degrees in both tests, implying that the undesirable convergence behavior appears at high-Reynolds-number cases featuring large directional gradients due to the thin boundary layers.

Having realized that the $p = 0$ approximation as the coarsest level of the multigrid for the high-Reynolds-number flow solutions can yield less than optimal performance, we decided to use the second cycling strategy (Fig. 3), to use the $p = 1$ approximation as the coarsest level and solve this coarse problem itself using *ph*-multigrid.

The p-convergence of the *ph*-multigrid with the second cycle type for $p = 2, \ldots, 5$ and the 1822 triangular mesh is shown in Fig. 10. In contrast to the first cycle type, almost uniform convergence rates are observed; the L^2 norm of density residual of 1.0^{-11} is obtained in 40–55 multigrid cycles for all values of p.

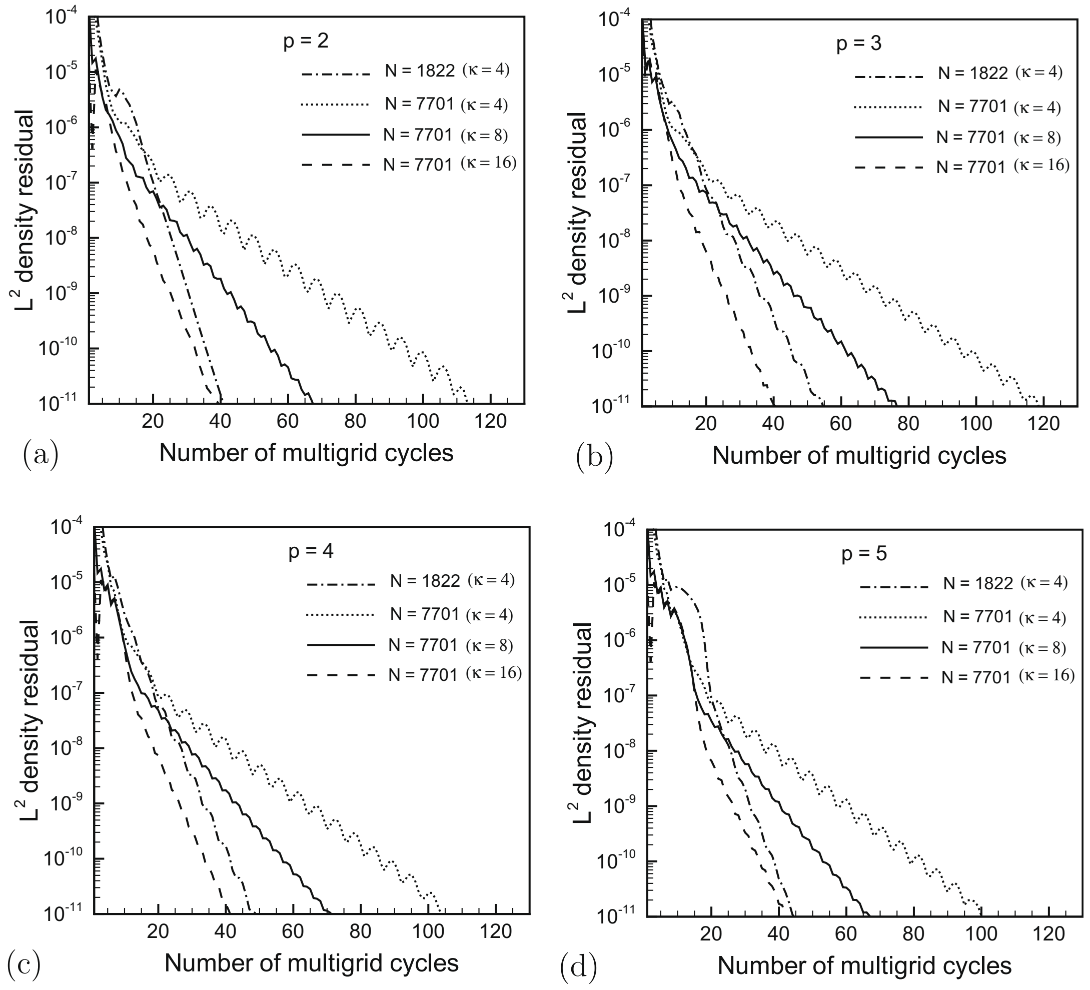

Fig. 12. *h*-dependence of the nonlinear *ph*-multigrid solver (the second cycling strategy). The L^2 norm of the density equation residual vs. the number of multigrid cycles in solving the NACA0012 problem for two isotropic meshes with 1822 and 7701 triangles and for approximation orders $p = 2, \ldots, 5$ in (a)–(d). For the 1822 element mesh, $\kappa = 4$ (the number of multigrid subcycles for solving the $p = 1$ problem) and for the 7701 element mesh, the results for three values of $\kappa = 4$, 8 and 16 are shown.

The efficiency of solving the $p = 1$ problem using the multigrid solver is confirmed when compared with the single-grid solver (Fig. 11(a) and (b)). It is clear that the multigrid solver yields an order of magnitude improvement in both the number of iterations and the total CPU time compared to the single-grid solver.

We now examine the *h*-convergence (convergence behavior with varying mesh sizes) of the *ph*-multigrid with the second cycle type. Fig. 12(a)–(d) demonstrate the convergence history as a function of the number of multigrid cycles for $p = 2, \ldots, 5$ and two different meshes, the mesh with 1822 triangles and the approximately fourfold finer mesh ($N = 7701$). For the coarser mesh, the data for four numbers of multigrid subcycles ($\kappa = 4$) for solving the coarsest level problem, $p = 1$, is reported,

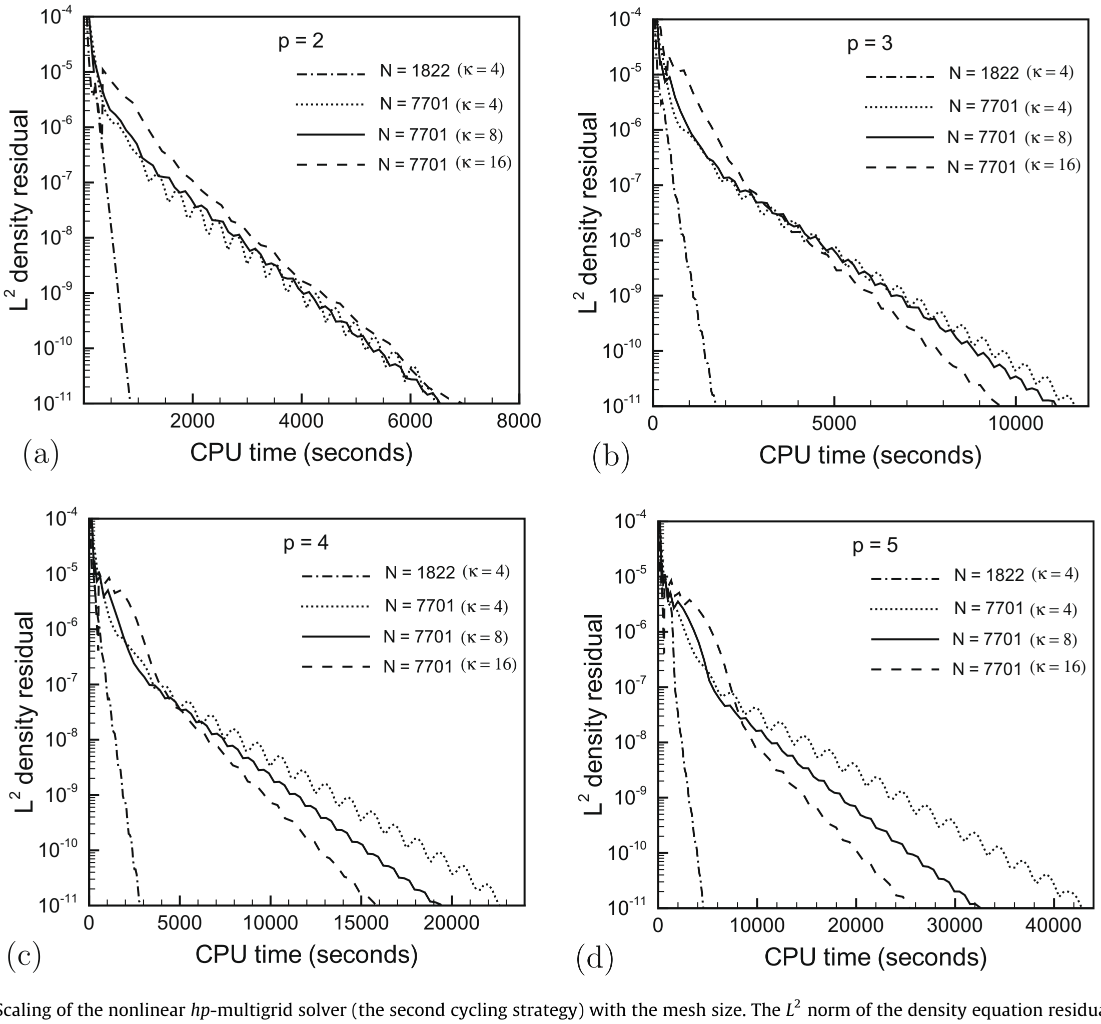

Fig. 13. Scaling of the nonlinear *hp*-multigrid solver (the second cycling strategy) with the mesh size. The L^2 norm of the density equation residual vs. the total CPU time in solving the NACA0012 problem at $Re = 5000$ for two isotropic meshes with $N = 1822$ and 7701 triangles and for approximation orders $p = 2, \ldots, 5$ in (a)–(d). For the $N = 1822$ element mesh, $\kappa = 4$ and for 7701 element mesh, the results for three values of $\kappa = 4$, 8 and 16 are shown.

Table 1
The total number of linear iterations for the linear multigrid formulation with the second cycle type in simulating the flow around NACA0012 at $Re = 5000$ for polynomial degrees $p = 2, \ldots, 5$ and for two isotropic meshes with $N = 1822$ and 7701. For both meshes, the data for four multigrid subcycles for solving $p = 1$ problem, $\kappa = 4$, are given, and for $N = 7701$, the data $\kappa = 8$ are also listed. The linear iteration count starts from the coarsest level solve ($p = 1$) in the full multigrid and stops when the L^2 norm of the total residual drops below 10^{-11} on the finest level.

p	$N = 1822, \kappa = 4$	$N = 7701, \kappa = 4$	$N = 7701, \kappa = 8$
	Total number of linear multigrid iterations (isotropic meshes)		
2	85	208	117
3	120	223	147
4	121	213	144
5	137	203	116

Table 2
The total number of linear iterations for the Newton–Krylov formulation using linear multigrid with the second cycle type as a preconditioner in simulating flow around NACA0012 at $Re = 5000$ for polynomial degrees $p = 2, \ldots, 5$ for three isotropic meshes with $N = 1822$, 7701 and 16061. For all meshes, the data for four multigrid subcycles for solving $p = 1$ problem, $\kappa = 4$, are given.

p	$N = 1822, \kappa = 4$	$N = 7701, \kappa = 4$	$N = 16061, \kappa = 4$
	Total number of linear preconditioned GMRES iterations (isotropic meshes)		
2	52	72	94
3	65	69	85
4	71	76	84
5	84	81	92

Table 3
The total number of linear iterations for the Newton–Krylov formulation using linear multigrid with the second cycle type as a preconditioner in simulating flow around NACA0012 at $Re = 5000$ for polynomial degrees $p = 2, \ldots, 5$ for two anisotropic meshes with $N = 2211$ and 7750. For both meshes, the data for four multigrid subcycles for solving $p = 1$ problem, $\kappa = 4$, are given.

p	$N = 2211, \kappa = 4$	$N = 7750, \kappa = 4$
	Total number of linear preconditioned GMRES iterations (anisotropic meshes)	
2	68	70
3	56	80
4	64	78
5	90	104

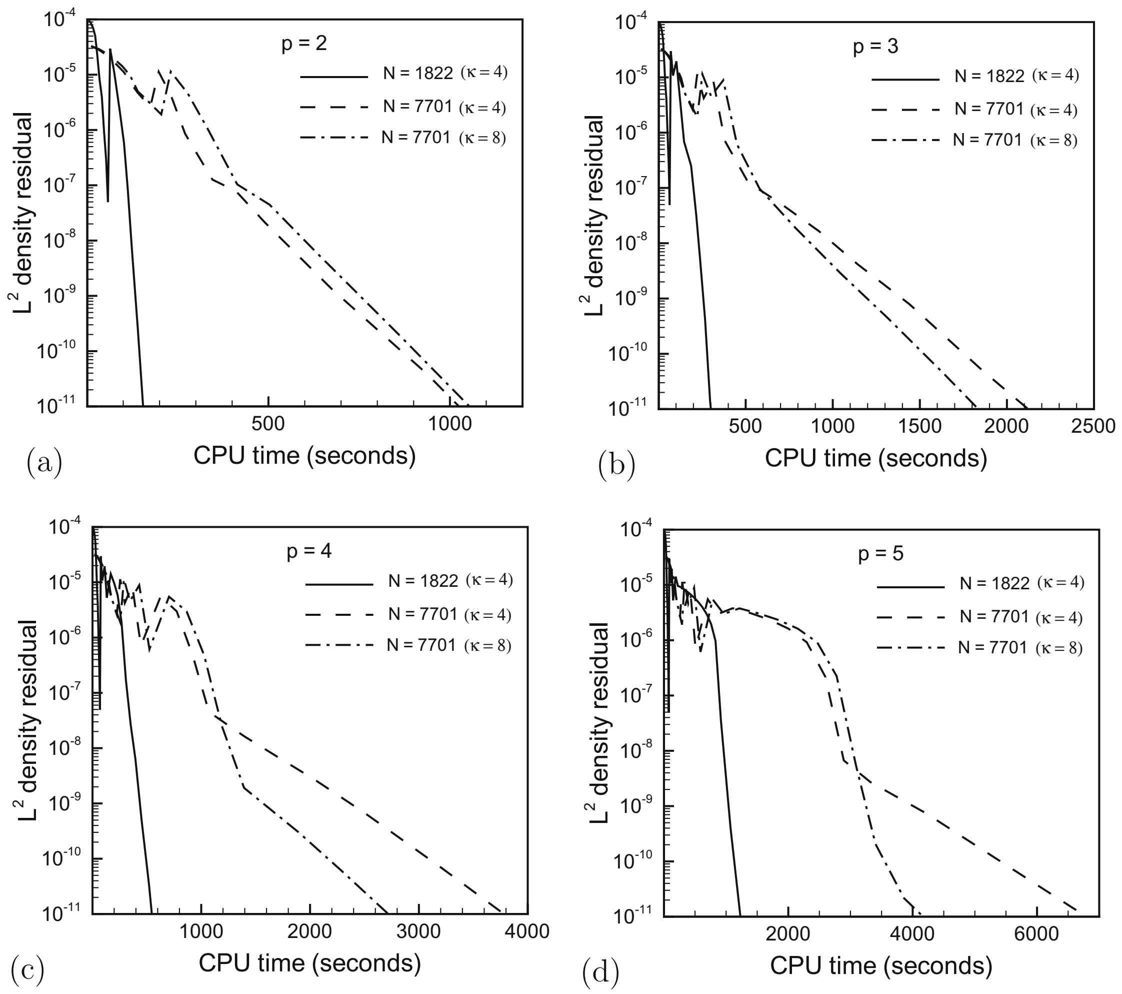

Fig. 14. Scaling of the linear multigrid solver with the second cycling strategy with the mesh size. The L^2 norm of the density equation residual vs. the total CPU time in solving the NACA0012 problem for two different meshes with 1822 and 7701 triangles and for approximation orders $p = 2, \ldots, 5$ in (a)–(d). For the 1822 element mesh, $\kappa = 4$ and for the 7701 element mesh, the results for two values of $\kappa = 4$ and 8 are shown.

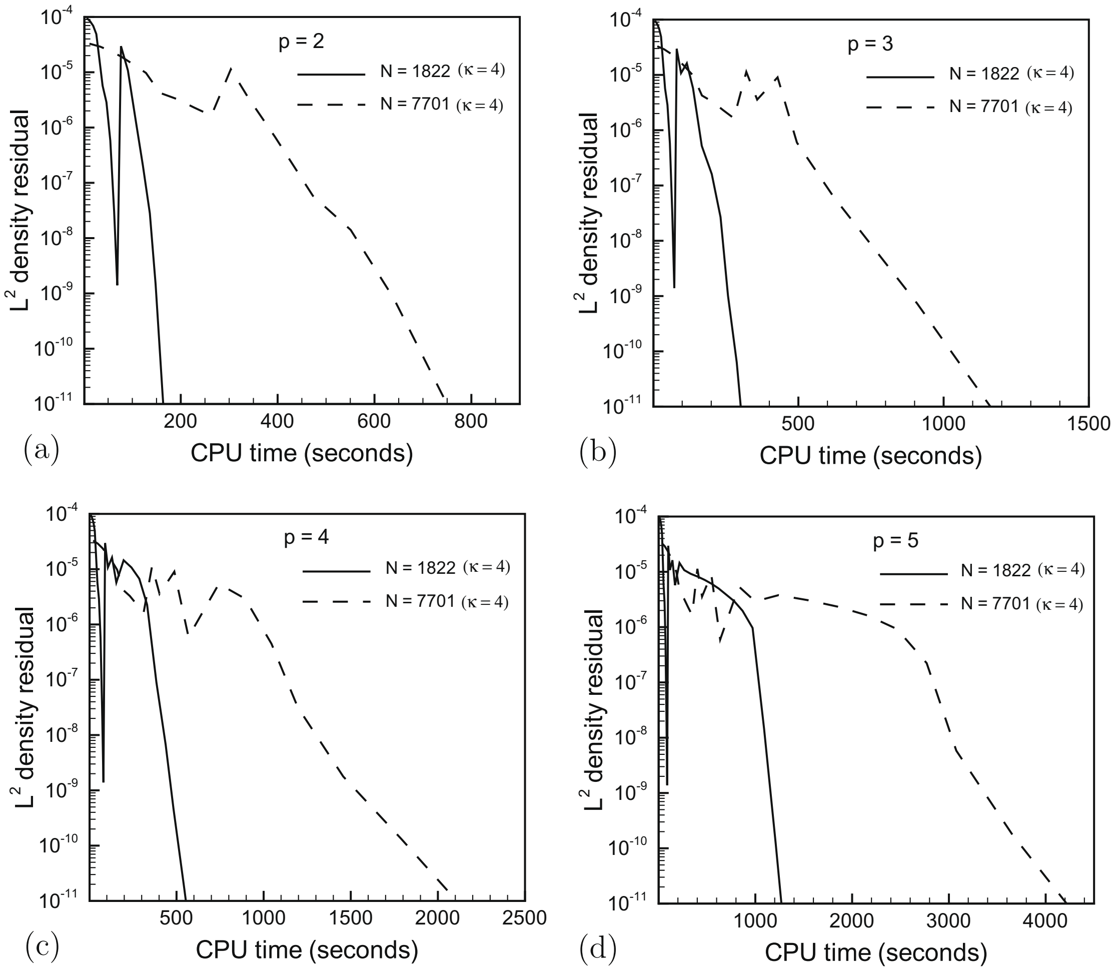

Fig. 15. Scaling of the Newton–GMRES solver with the linear multigrid solver with the second cycling strategy as a preconditioner with the mesh size. The L^2 norm of the density equation residual vs. the total CPU time in solving the NACA0012 problem for two different meshes with 1822 and 7701 triangles for approximation orders $p = 2, \ldots, 5$ in (a)–(d). The results is for $\kappa = 4$.

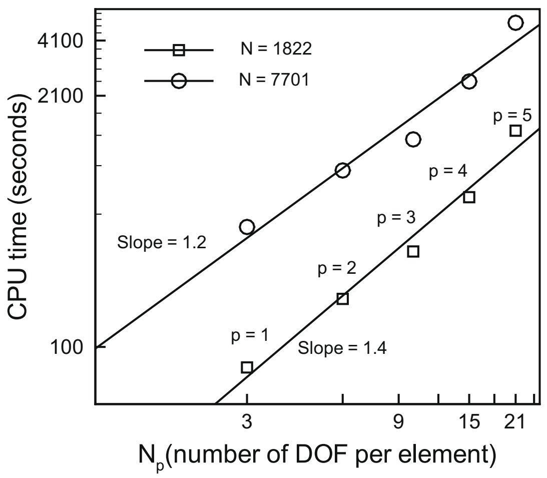

Fig. 16. Scaling of the multigrid preconditioned Newton–GMRES solver with approximation orders. Total CPU time vs. the number of DOF per element, N_p in solving the NACA0012 problem for approximation orders $p = 1, \ldots, 5$ and for two meshes with 1822 and 7701 triangles. For each data set, the best linear fit and its slope are also shown. "DOF" signifies "degrees of freedom".

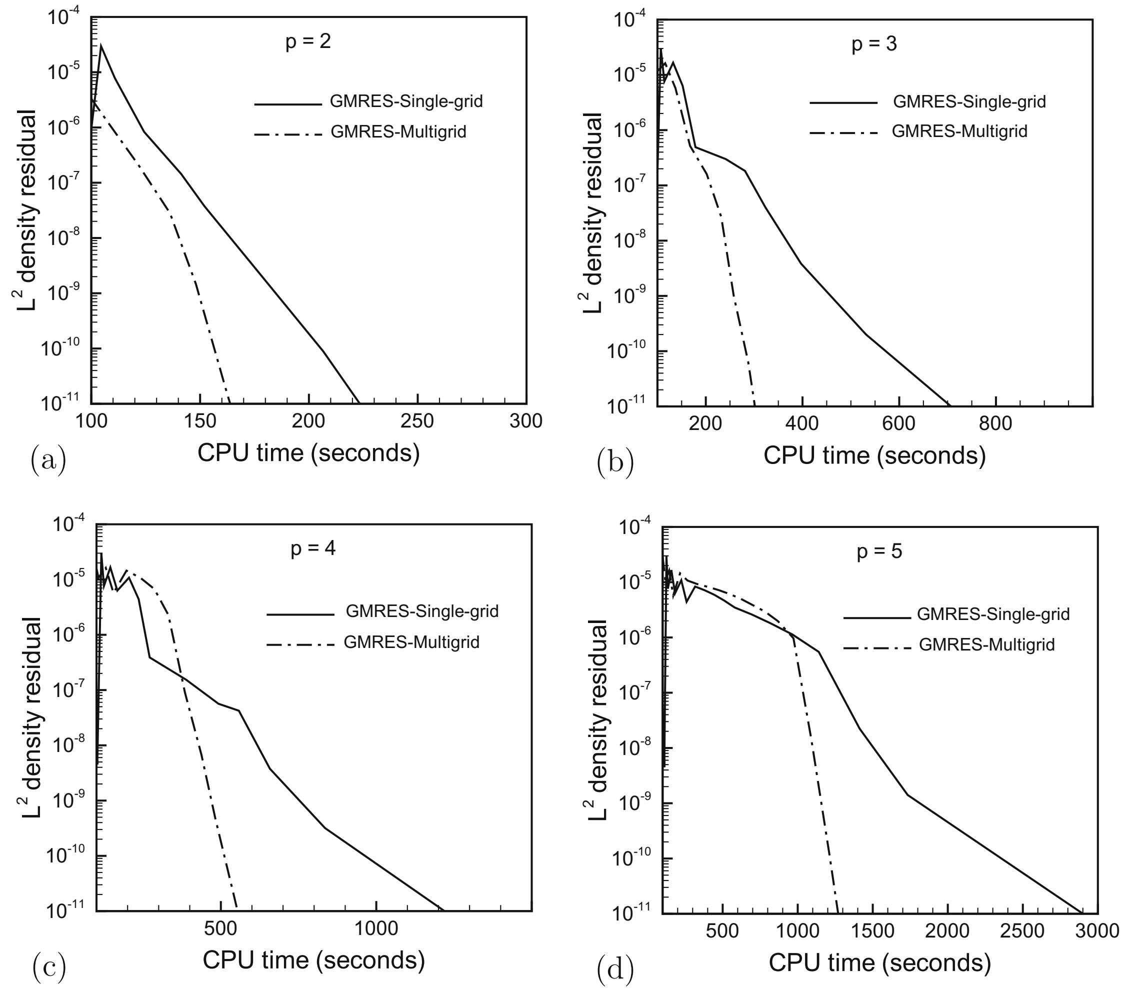

Fig. 17. Efficiency comparison of the Newton–GMRES solver with single-grid and *ph*-multigrid preconditioners. The L^2 norm of the density equation residual vs. the total CPU time in solving the NACA0012 problem on a mesh with $N = 1822$ for approximation orders $p = 2, \ldots, 5$ in (a)–(d). The results for the *ph*-multigrid is for the second cycle type and $\kappa = 4$.

while for the finer mesh, the data for $\kappa = 4$, 8 and 16 are presented. Three points are notable about the results. First, at the fixed value of $\kappa = 4$, the number of multigrid cycles required to achieve residual tolerance of 10^{-11} is more than twofold higher for the finer mesh size ($N = 7701$) than those of the coarser mesh ($N = 1822$), implying mesh-dependent convergence rates. Secondly, increasing the number of multigrid subcycles (e.g. increasing $\kappa = 4$ to $\kappa = 16$) on the finer mesh problem yields smaller number of multigrid cycles close to those on the coarser mesh. Finally, while for low values of κ on the finer mesh problem, the convergence is clearly non-monotonic, a sufficiently large value of κ leads to almost monotonic convergence rates (compare the curve for $\kappa = 4$ and that for $\kappa = 16$).

For the same test setting, in Fig. 13(a)–(d), we also plot the L^2 norm of the density equation residual vs. the total CPU time. It is clear that for low values of κ, linear scaling is not observed; to reach to certain residual values, the approximately fourfold finer mesh problem requires more than a sixfold higher CPU time than those of the coarser mesh problem. However, except for the $p = 2$ approximation order, for all other orders, the situation improves as the number of subcycles (κ) increases on the finer mesh. The lack of overall efficiency improvements with the increased κ observed for $p = 2$ is due to the fact that the cost of solving the $p = 2$ approximation is the closest to that of the $p = 1$ level among all orders. This implies that the efficiency gain due to the improved convergence rate while increasing κ is compensated with the increased cost associated with the more accurate solution of the $p = 1$ level.

7.2. Linear multigrid results

In this subsection, we study the performance of the linear multigrid method both as a solver and as a preconditioner for the Newton–Krylov method. For these two variants, we show the *p*- and *h*-convergence characteristics of the solvers. In

addition to results for the isotropic meshes, we present data for the anisotropic cases. We next present the scaling of the linear multigrid solver and the preconditioned Newton–Krylov method with mesh sizes. Finally, we report the linear scaling for the preconditioned Newton–Krylov method with approximation order.

For all linear solvers, the linear system arising from the Newton iterations was solved approximately. It is known that in the early stages of the nonlinear iterations, where the solution is far from the exact solution, the linear system only needs to be solved to a limited tolerance (see for instance [28]). On the other hand, in the later stages, the linear system must be solved sufficiently accurately to obtain super-linear convergence for the Newton method. We empirically found that the

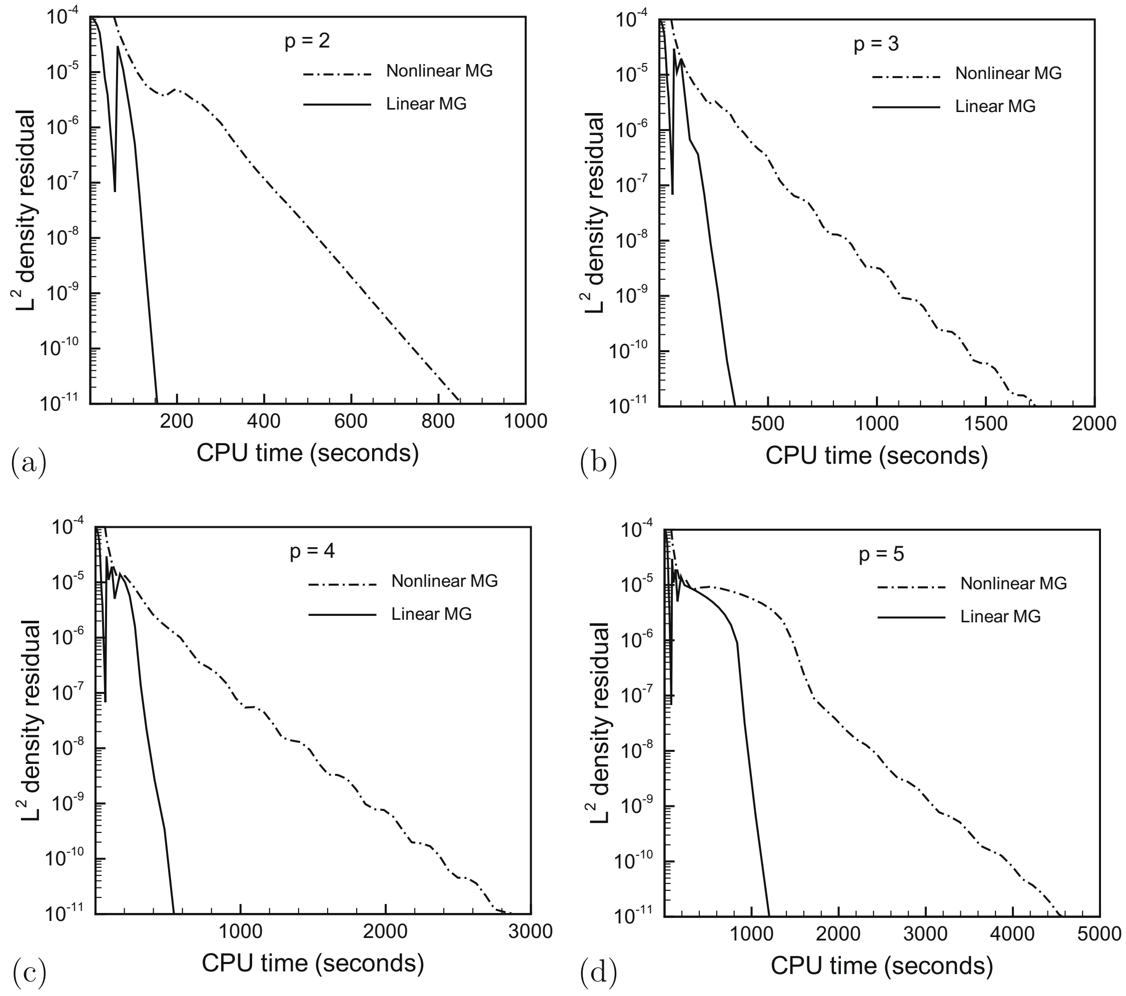

Fig. 20. Efficiency comparison of the nonlinear and linear *ph*-multigrids with the second cycle type. The L^2 norm of the density equation residual vs. the total CPU time (s) in solving the NACA0012 problem the mesh with $N = 1822$ and for approximation orders $p = 2, \ldots, 5$ in (a)–(d). The results is for $\kappa = 4$.

Table 4
The total number of nonlinear iterations for the nonlinear and linear multigrid (using the second cycle type) solution of the NACA0012 problem for polynomial degrees $p = 1, \ldots, 5$ for a mesh size of $N = 7701$. For $p = 2, \ldots, 5$, four multigrid subcycles ($\kappa = 4$) was used for solving the coarsest problem, $p = 1$.

p	Nonlinear multigrid	Linear multigrid
	Total number of nonlinear iterations	
1	100	15
2	104	19
3	109	16
4	94	20
5	90	23

following strategy for solving the linear system yields near optimal results. At the $n+1$ nonlinear iteration, we continue the iteration on the linear system until the L^2 norm of the total linear residual $(b - Ax)$ drops below ϵ^{n+1} defined as

$$\epsilon^{n+1} = \max\left(\min\left(\frac{||\mathcal{R}^n||}{2.5^{(n-n_c-p)}}, 0.1||\mathcal{R}^n||\right), 10^{-14}\right), \tag{34}$$

where $||\mathcal{R}^n||$ represents the L^2 norm of the nonlinear residual at iteration n and $n_c = 8, 10, 12, 14$ and 16 for $p = 1, \ldots, 5$, respectively. Note that for $p = 2, \ldots, 5$, the values of n_c correspond to the total numbers of cycles on the coarse levels of the full multigrid procedure.

Table 1 lists the total numbers of linear iterations to achieve L^2 norm of the total residual of 10^{-11} for the linear multigrid solver for 1822 and 7701 triangular meshes and a range of approximation orders $p = 2, \ldots, 5$. The results correspond to the

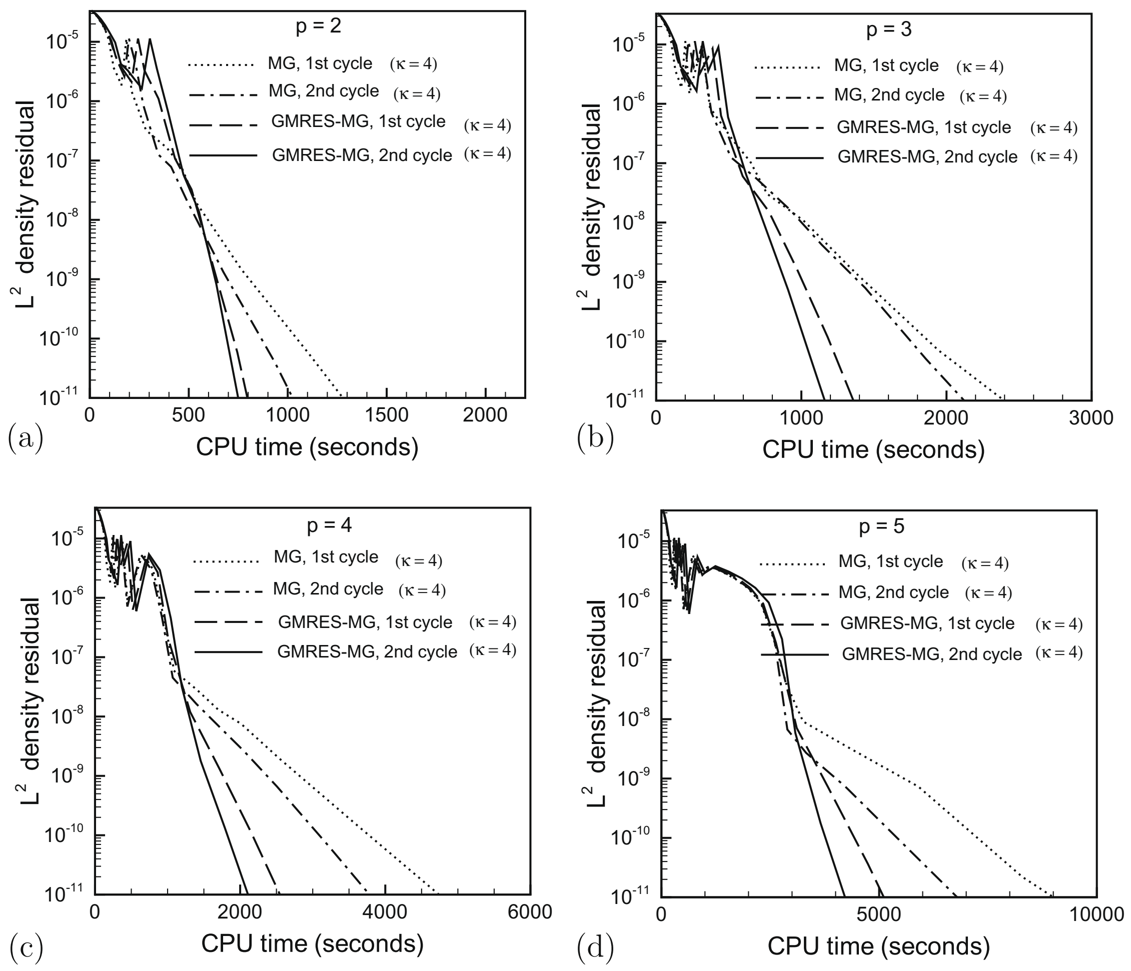

Fig. 21. Efficiency comparison of different solvers, linear multigrid with the first and second cycling strategies (MG, 1st cycle and MG, 2nd cycle) and Newton–GMRES with the second and third cycling strategies (GMRES–MG, second cycle and GMRES–MG, third cycle). The L^2 norm of the density equation residual vs. the total CPU time (s) in solving the NACA0012 problem for a mesh with $N = 7701$ and for approximation orders $p = 2, \ldots, 5$ in (a)–(d). The results is for $\kappa = 4$.

Table 5
Speedup of multigrid preconditioned GMRES solver with the second cycle type over that of the first cycle for $p = 2, \ldots, 5$ for two mesh size of $N = 7701$ and $N = 16061$. Four multigrid subcycles ($\kappa = 4$) was used for solving the coarsest problem.

p	$N = 7701 (\kappa = 4)$ (%)	$N = 16061 (\kappa = 4)$ (%)
	Speedup of the second cycle over the first (preconditioned GMRES solver)	
2	4	0
3	32	30
4	20	53
5	14	32

second cycle type with $\kappa = 4$ for the coarse mesh and $\kappa = 4$ and 8 for the fine mesh. For the p-convergence, similar to the nonlinear multigrid solver studies in the preceding section, almost uniform numbers of iterations for different orders are observed for both $N = 1822$ and 7701 cases, implying p-independent convergence. For the h-convergence, the total number of iterations are increased as the mesh is refined from 1822 triangles to 7701 triangles with a fixed value of $\kappa = 4$. This undesirable h-dependence improves as κ grows on the fine mesh, similar to the nonlinear multigrid formulation.

Table 2 shows the results for the same setting, but now the Newton–GMRES solver was used with linear multigrid with the second cycle type as a preconditioner. Contrary to the previous case, both p- and h-independent convergence rates are observed. Moreover, unlike the nonlinear and linear multigrid solvers where h-independence was achieved by increasing the number of subcycles (κ) on the finer mesh, the h-independence for the preconditioned Newton–GMRES was obtained for both mesh sizes at the fixed value of $\kappa = 4$. We further confirm this optimal performance of the preconditioned Newton–GMRES solver for highly anisotropic meshes. The results for two anisotropic meshes with $N = 2211$ and 7750 (Fig. 5), having maximum aspect ratios of approximately 82 and 235, respectively, are shown in Table 3, demonstrating the (almost) p- and h-independent convergence rates. We reiterate that these results of the anisotropic meshes were obtained using the elemental-line-implicit smoother in the region of grid anisotropy, since using the Gauss–Seidel smoother would significantly degrade the convergence.

Similar outcomes are expected when considering the time scaling for both variants of the linear multigrids (as a solver and as a preconditioner). We plot the L^2 norm of the residual vs. the total CPU time for the same two meshes as before and $p = 2, \ldots, 5$ in Fig. 14(a)–(d) and Fig. 15(a)–(d) for the linear multigrid solver and the preconditioned Newton–GMRES solver, respectively. As Fig. 14(a)–(d) illustrate, for the linear multigrid solver, an approximately fourfold increase in the mesh size with a fixed value of $\kappa = 4$, yields significantly higher than fourfold (approximately eight-fold) increases in the total CPU time. Similar results are observed when a higher value of $\kappa = 8$ was used, except for the $p = 5$ case which yields almost linear scaling. On the other hand, very different behavior is observed when the linear multigrid algorithm was used as a preconditioner for the Newton–Krylov method (Fig. 15(a)–(d)). For all approximation orders, an approximately fourfold increase in the mesh size yields an approximately fourfold higher CPU time, implying linear scaling of the solver with mesh sizes.

The good performance of the preconditioner Newton–GMRES solver is further verified by plotting the total CPU time vs. the number of DOF per element for $p = 1, \ldots, 5$ and the two meshes with $N = 1822$ and 7701 in Fig. 16. For each data set, the best linear fit and its slope are also shown. For the linear fits, slopes of 1.4 and 1.2 are obtained for the coarse and fine meshes, respectively, confirming the almost linear scaling of the solver with approximation orders.

7.3. *Efficiency comparison*

In this subsection, we first compare the ph-multigrid method with the single-grid method both used as preconditioners for the Newton–GMRES solver for both isotropic and anisotropic meshes. We then show the efficiency of the linear multigrid over the nonlinear variant, before comparing the performance of the linear multigrid as a solver and as a preconditioner for the Newton–GMRES solver.

We first plot the L^2 norm of the density equation residual vs. the total CPU time for the isotropic mesh with 1822 triangles and $p = 2, \ldots, 5$ for the single-grid and the ph-multigrid preconditioners in Fig. 17(a)–(d). For the multigrid case, the results for the second cycle type with $\kappa = 4$ are presented. The figures show that multigrid preconditioner outperforms the single-grid preconditioner for all approximation orders. Moreover, as the problem size grows so does the efficiency of the multigrid method over the single-grid method. This is evident from Fig. 18(a)–(d), where the same measures are plotted for the larger mesh with $N = 7701$. The significant efficiency of the multigrid preconditioner over the single-grid preconditioner also extends to the anisotropic case as is seen in Fig. 19(a) and (b). For the anisotropic mesh, the multigrid preconditioner appears to be approximately one hundredfold faster than the single-grid.

In Fig. 20(a)–(d), we plot the same measures for the same mesh and approximation orders as the former case, but for the nonlinear and linear ph-multigrid solvers using the second cycle type with $\kappa = 4$. It is evident that the linear multigrid is at least fourfold faster that the nonlinear case. As mentioned earlier, this is attributed to the fewer number of nonlinear residual and Jacobian evaluations in the case of the linear multigrid. To confirm this, for $p = 1, \ldots, 5$ and $N = 7701$, the total number of nonlinear iterations to achieve the L^2 norm of the density residual of 10^{-11} for both the nonlinear and linear ph-multigrid with the second cycle type are listed in Table 4. For all approximation orders, the total number of nonlinear iterations for the linear case are approximately fivefold lower than that for the nonlinear case.

In a final set of figures, Fig. 21(a)–(d), we compare the performance of linear multigrid as a solver with that of the multigrid preconditioned Newton–GMRES. Specifically, the L^2 norm of the density equation residual is depicted vs. the total CPU time for four different solver variants: the linear multigrid using the first or second cycle types, and the Newton–GMRES solver using the linear multigrid method, with the first or second cycle type, as a preconditioner. The results are reported for the 7701 triangular mesh, $p = 2, \ldots, 5$ and $\kappa = 4$. The figures demonstrate that multigrid Newton–GMRES solver with either first or second cycle type is faster than the linear multigrid. Moreover, the solvers with the second cycle type are more efficient that those with the first cycle. Corresponding speedups for the multigrid preconditioned Newton–GMRES solvers are listed in Table 5 for two meshes $N = 7701$ and 16061. It is notable that as the mesh size grows, the solver with second cycle type becomes increasingly more efficient.

Finally, we note that the Newton–GMRES solver with the multigrid preconditioner using the second cycle type, besides being faster, has another advantage over the Newton–GMRES solver with the multigrid preconditioner using the first cycle

Table 6
The total number of maximum inner iterations for the multigrid preconditioned Newton–GMRES formulation with the first and second cycle types as well as the percentage of memory increase compared to the linear multigrid solver in solving the NACA0012 problem for polynomial degrees $p = 2, \dots, 5$ and the mesh with $N = 7701$. For the first cycle type, data for $p = 1$ is also shown.

p	First cycle		Second cycle	
	Inner iterations	Memory increase (%)	Inner iterations	Memory increase (%)
	Multigrid preconditioned GMRES			
1	17	121	–	–
2	21	80.0	11	42.3
3	19	45.2	11	26.2
4	17	27.4	9	14.5
5	18	20.9	9	10.5

type. The former requires fewer GMRES inner iterations, or equivalently fewer number of the Krylov basis; thus, a lower memory storage. This is verified by the data in Table 6, where the maximum number of the inner GMRES iterations (maximum number of Krylov basis) measured over the whole solution procedure is listed for both the first and second cycle types for the 7701 triangular mesh and $p = 1, \dots, 5$. Also listed is the percentage of the memory requirement increase compared to the pure linear multigrid solver. The solver with the second cycle type requires approximately half the number of inner iterations and thus a twofold lower increase in the memory requirements. Also notable from the table is that as approximation order increases the percentage of the memory requirement increase declines. For instance, while for $p = 2$, the multigrid preconditioned Newton–GMRES with the second cycle type requires the 42.3% higher memory storage than the corresponding linear multigrid solver, for $p = 5$, the memory requirement increase drops to only 10.5%.

8. Conclusions

We have developed linear and nonlinear multigrid algorithms for systems arising from high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations on unstructured meshes.

We show that the choice of cycling strategy is crucial in achieving efficient and scalable solvers. While the first cycling strategy yields non-optimal convergence rates for high-Reynolds number flows, the second cycling strategy leads to order-independent convergence rates. Mesh-independent convergence rates for the solver with the second cycling strategy are achieved provided the coarsest problem is solved to sufficient accuracy. On the other hand, the Newton–GMRES solver using multigrid with the second cycle type as a preconditioner appears to be insensitive to this condition, and both order- and mesh-independent convergence are obtained at a fixed number of multigrid subcycles for solving the coarsest problem. This results hold true for both isotropic and highly anisotropic meshes provided, in the region of mesh anisotropy, instead of pure elemental smoother, an elemental-line-implicit smoother is used.

In comparing the total CPU times, we have verified that multigrid preconditioners outperform the single-grid preconditioner for both isotropic and anisotropic meshes, the linear multigrid approach is approximately fourfold faster than the nonlinear multigrid, and the multigrid preconditioned Newton–GMRES solver using the second cycling strategy is the fastest overall scheme. We conclude that the multigrid preconditioned Newton–GMRES solver with the second cycle type yields the most efficient and scalable algorithm.

For comparison, the production-optimized unstructured mesh multigrid second-order finite-volume Navier–Stokes solver NSU2D [32] achieves the same level of convergence for a similar test case with approximately 50,000 degrees of freedom (per equation) in 720 s on the same hardware, while a test case with 150,000 degrees of freedom requires approximately 3000 s to fully converge. This compares favorably with the best timings for the $p = 2$ discretization on the $N = 7701$ mesh, which from Fig. 21 is seen to require approximately 750 s, and corresponds to 46206 degrees of freedom, while the $p = 5$ discretization on the $N = 7701$ mesh, which corresponds to 161721 degrees of freedom, required just under 4000 s to converge, as seen in the same figure. Although the finite-volume code timings are slightly faster for the same number of degrees of freedom, the DG results are much more accurate than the NSU2D results, particularly at the higher p values. A fully quantitative comparison of the efficiency of DG vs. traditional finite-volume methods would require the establishment of cost for a given level of accuracy, and is beyond the scope of this paper, although given these qualitative results, the DG approach at high p-orders can be expected to be competitive on a cost vs. accuracy level.

Acknowledgments

The first author acknowledges fruitful conversations with Dr. Christian Nastase, University of Wyoming. This work was financially supported by NASA, under Contract NNXO7AC31A.

References

[1] R. Hartmann, P. Houston, Symmetric interior penalty DG methods for the compressible Navier–Stokes equations I: method formulations, Int. J. Numer. Anal. Model. 3 (2006) 1–20.

[2] C.M. Klaij, J.J.W. van der Vegt, H. van der Ven, Space-time discontinuous Galerkin method for the compressible Navier–Stokes equations, J. Comput. Phys. 217 (2006) 589–611.
[3] S.F. Davis, Simplified second-order Godunov-type methods, SIAM J. Sci. Stat. Comput. 9 (1988) 445–473.
[4] P.L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys. 43 (1981) 357–372.
[5] A. Harten, P.D. Lax, B. Van Leer, On upstream differencing and Godunov-type schemes for hyperbolic conservation laws, SIAM Rev. 25 (1983) 35–61.
[6] F.E. Toro, Riemann solvers and numerical methods for fluid dynamics, Appl. Mech., Springer, New York, NY, 1999.
[7] P. Batten, N. Clarke, C. Lambert, D.M. Causon, On the choice of wavespeeds for the HLLC Riemann solver, SIAM J. Sci. Comput. 18 (1997) 1553–1570.
[8] D.J. Mavriplis, Unstructured mesh generation and adaptivity, in: VKI Lecture Series VKI-LS 1995-02, 1995.
[9] D.J. Mavriplis, V. Venkatakrishnan, Agglomeration multigrid for two-dimensional viscous flows, Comput. Fluids 24 (5) (1995) 553–570.
[10] D.J. Mavriplis, An assessment of linear vs. non-linear multigrid methods for unstructured mesh solvers, J. Comput. Phys. 175 (2002) 302–325.
[11] P. Wesselling, An introduction to multigrid methods, Willy, New York, 1992. p. 284.
[12] P. Lötstedt, Grid independent convergence of the multigrid method for first-order equations, J. SIAM Numer. Anal. 29 (1992) 1370–1394.
[13] K.J. Fidkowski, T.A. Oliver, J. Lu, D.L. Darmofal, *p*-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations, J. Comput. Phys. 207 (2005) 92–113.
[14] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[15] Y. Saad, Iterative Methods for Sparse Linear System, PWS Press, New York, 1996.
[16] B.T. Helenbrook, D.J. Mavriplis, H.A. Atkins, Analysis of *p*-multigrid for continuous and discontinuous finite element discretizations, AIAA Paper 2003-3989, 2003.
[17] A. Jameson, Solution of the Euler equations for two-dimensional transonic flow by a multigrid method, Appl. Math. Comput. 13 (1983) 327–356.
[18] D.J. Mavriplis, Multigrid solution of the 2-D Euler equations on unstructured triangular meshes, AIAA J. 26 (1988) 824–831.
[19] D.J. Mavriplis, Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes, J. Comput. Phys. 145 (1998) 141–165.
[20] C. Nastase, D.J. Mavriplis, High-order discontinuous Galerkin methods using an *hp*-multigrid approach, J. Comput. Phys. 213 (2006) 330–357.
[21] C. Nastase, D.J. Mavriplis, A parallel *hp*-Multigrid solver for three-dimensional discontinuous Galerkin discretizations of the Euler equations, AIAA Paper 2007-512, 2007.
[22] J.W. Lottes, P.F. Fischer, Hybrid multigrid/Schwarz algorithms for the spectral element method, J. Sci. Comput. 24 (2005) 45–78.
[23] E.M. Rönquist, A.T. Patera, Spectral element multigrid. I. Formulation and numerical results, J. Sci. Comput. 2 (1987) 389–406.
[24] W. Hackbusch, Multigrid Methods and Applications, Springer, Berlin, 1995.
[25] K. Shahbazi, An explicit expression for the penalty parameter of the interior penalty method, J. Comput. Phys. 205 (2005) 401–407.
[26] P. Solin, P. Segeth, I. Zel, High-order finite element methods, Studies in Advanced Mathematics, Chapman and Hall, London, 2003.
[27] D. Arnold, An interior penalty finite element method with discontinuous elements, SIAM J. Numer. Anal. 19 (1982) 742–760.
[28] C.T. Kelly, D.E. Keyes, Convergence analysis of pseudo-transient continuation, SIAM J. Numer. Anal. 35 (1998) 508–523.
[29] P.O. Persson, J. Peraire, Newton–GMRES preconditioning for discontinuous Galerkin discretizations of the Navier–Stokes equations, SIAM J. Sci. Comput. 30 (2008) 2709–2733.
[31] L.W. Thomas, Numerical Partial Differential Equations, Springer, 1995.
[32] W.O. Valarezo, D.J. Mavriplis, Navier–Stokes applications to high-lift airfoil analysis, AIAA J. Aircraft 23 (1995) 457–688.