Journal of Computational Physics 231 (2012) 7739–7755

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

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

Efficient evaluation of the direct and adjoint linearized dynamics from compressible flow solvers

Miguel Fosas de Pando [a,*], Denis Sipp [b], Peter J. Schmid [a]

[a] Laboratoire d'Hydrodynamique (LadHyX), CNRS-Ecole Polytechnique, 91128 Palaiseau, France
[b] ONERA/DAFE, 8 rue des Vertugadins, 92190 Meudon, France

ARTICLE INFO

Article history:
Received 17 December 2011
Received in revised form 26 June 2012
Accepted 27 June 2012
Available online 20 July 2012

Keywords:
Adjoint methods
Matrix-free framework
Compressible flows
Hydrodynamic stability

ABSTRACT

The direct and adjoint operators play an undeniably important role in a vast number of theoretical and practical studies that range from linear stability to flow control and nonlinear optimization. Based on an existing nonlinear flow solver, the design of efficient and straightforward procedures to access these operators is thus highly desirable. In the case of compressible solvers, the use of high-order numerical schemes combined with complicated governing equations makes the derivation of efficient procedures a challenging and often tedious undertaking. In this work, a novel technique for the evaluation of the direct and adjoint operators directly from compressible flow solvers is presented and extended to include nonlinear differentiation schemes and turbulence models. The application to the incompressible counterpart is also discussed. The presented method requires minimal additional programming effort and automatically takes into account subsequent modifications in the governing equations and boundary conditions. The introduced methodology is demonstrated on existing numerical codes, and direct and adjoint global modes are calculated for three typical flow configurations. Implementation issues and the performance measures are also discussed. The proposed algorithm presents an easy-to-implement and efficient technique to extract valuable information for the quantitative analysis of complex flows.

1. Introduction

Even though most physical processes in fluid flows are most aptly described by a nonlinear mathematical model, our tools for analyzing them often rely on a linear approximation of the underlying dynamics [9]. For this reason, the Jacobian matrix $\mathbf{A}$ (or direct operator) of a temporal evolution process together with its adjoint $\mathbf{A}^*$ play an undeniably important role in theoretical and practical studies [37]. The direct operator describes the dynamics of small perturbations around an equilibrium state; the associated adjoint operator contains gradient information about changes in the system and their influence on its dynamics [15,45,11,26]. Together, they lay a theoretical foundation for stability analyses and are integral parts in many gradient-based numerical methods, e.g. Newton's method [21], pseudo-arc-length continuation [39], steepest-descent optimization [17,31], nonlinear adjoint looping [47,16], and many more. In a more physical context, since the Jacobian represents an approximation of the nonlinear dynamics around an equilibrium state, it plays a central part in the extraction of physical mechanisms for instabilities and, in the case of fluid dynamics, in the study of transition scenarios. In the latter case, theoretical studies within the linear framework — such as global stability [4] and transient growth analysis [36] — have

* Corresponding author.
E-mail address: miguel@ladhyx.polytechnique.fr (M. Fosas de Pando).

0021-9991/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcp.2012.06.038

demonstrated their usefulness in the physical description of the onset of unsteadiness and the determination of its cause. Furthermore, weakly nonlinear analyses [41] and many efficient techniques for active and passive flow control [19] also depend on the linear dynamics.

Gradient-based analysis of fluid systems have concentrated predominantly on incompressible flow configurations because of the relative ease in deriving efficient procedures for the evaluation and extraction of the Jacobian. Despite some commendable attempts [7,2,27,25], many interesting issues of compressible flows are yet to be explored: for example, linear stability theory could shed some light on the mechanisms for sound generation and their role in the onset of instabilities [8,2,29]. Although the structure of the compressible Navier–Stokes equations is algorithmically simpler because of the absence of the incompressibility constraint, the derivation of the direct linear operator and its adjoint for general compressible flows can be an arduous task and susceptible to mistakes due to a significantly larger number of terms. This situation is aggravated by the necessity of more complex boundary conditions [33,23]. An approach that has been successfully applied to spatial schemes with relatively short stencils is the explicit extraction of the linearized operator $\mathbf{A}$ by a quasi-linearization technique using a sequence of unit vectors $\mathbf{e}_j$, thus yielding, column by column, the Jacobian $\mathbf{A}$. In this case, given a base state $\mathbf{V}$, we have

$$\mathbf{A}_{ij} \approx \frac{\mathbf{F}_i(\mathbf{V} + \epsilon \mathbf{e}_j) - \mathbf{F}_i(\mathbf{V})}{\epsilon} \tag{1}$$

with $\mathbf{F}_i$ denoting the ith component of the right-hand-side of the nonlinear equations and ϵ given as a small parameter. If the stencil is short and the grid ordering is known, this approach constitutes a very efficient method. This *a priori* knowledge will then allow a tiling-technique that determines multiple columns of $\mathbf{A}$ with one evaluation.

However, the disparity of length and time scales (such as in sound generation problems) makes the use of high-order spatial schemes a convenient approach to achieve high resolution [6,32], leading to low-sparsity or dense matrices whose explicit construction for its use in practical calculations is intractable, even for configurations with small numbers of degrees of freedom. Numerical methods relying on matrix-free evaluations of $\mathbf{Av}$ and $\mathbf{A}^*\mathbf{w}$ for linear stability appear to be the best option to tackle this problem. This approach is reflected in the success of Krylov subspace techniques for large-scale linear algebra problems [34,35]. A first attempt to overcome the above-mentioned sparsity problem is the numerical evaluation of the direct operator $\mathbf{A}$ by quasi-linearization, a technique that has been widely applied (see [24] for an example of compressible flows). We have

$$\widetilde{\mathbf{Av}} = \frac{\mathbf{F}(\mathbf{V} + \epsilon \mathbf{v}) - \mathbf{F}(\mathbf{V})}{\epsilon} \tag{2}$$

which provides an approximation for the action of the Jacobian (direct operator) $\mathbf{A}$ on a given vector $\mathbf{v}$. This technique, however, fails to provide direct access to the adjoint operator $\mathbf{A}^*$.

The derivation of the direct and adjoint operators can follow two different strategies. In the *continuous* approach, the direct operator is derived from the continuous form of the nonlinear equations and implemented in a numerical code; the adjoint operator is then derived from the continuous form of the direct operator before it is discretized appropriately and implemented in a numerical code. In the *discrete* approach, the direct operator is obtained by linearization of the already discretized equations and then implemented; the discrete adjoint is then easily written in terms of the transconjugate of the discretized direct operator. In the context of flow stability studies, the continuous approach is more common. On the contrary, abundant examples of the discrete approach can be found in the optimization community. It is important to note that these two strategies do not yield the same operators, since the discretization and linearization steps generally do not commute [42]. On one hand, the continuous approach provides the numerical approximation of the sensitivity of the continuous equations; on the other hand, the discrete approach represents the sensitivity of the discretized approximation of the nonlinear eqations.

The discrete approach offers important advantages over the continuous one, both from a practical and numerical point of view. The most relevant property is that, given the inner product $\langle \cdot, \cdot \rangle$, $\mathbf{A}$ and $\mathbf{A}^*$ satisfy the fundamental relation $\langle \mathbf{w}, \mathbf{Av} \rangle = \langle \mathbf{A}^*\mathbf{w}, \mathbf{v} \rangle$ up to machine precision rather than up to the discretization error of the numerical scheme on a given mesh, which is the case for the continuous approach. This property is very desirable as it avoids convergence problems of iterative gradient-based algorithms due to amplification of errors stemming from this discrepancy. In addition, the above-mentioned relation applied to the continuous case generates boundary terms, commonly by integration by parts, that have to be dealt with separately. For compressible flow simulations, this is not a trivial task. On the contrary, in the case of the discrete operators, if the discretized nonlinear equations together with a well-posed set of boundary conditions are available, then the linearized operators are in principle straightforward to derive and no additional thought has to be given to discretizations or boundary conditions for either operator, which includes, for example, the use of artificial numerical techniques such as sponge layers [43]. This feature is also central to automatic differentiation (AD) [3,13,12] which has become a powerful tool for the derivation of direct and adjoint code; the nonlinear source code is interpreted in a non-standard fashion to automatically derive new source code for the evaluation of the direct and adjoint operators. This technique has been successfully applied to a wide range of applications; however, issues related to parallelization or efficiency remain an active field of research [28].

In this article, we present a novel technique for the efficient evaluation of the linearized direct operator and its adjoint *directly* from a nonlinear compressible flow solver using the discrete approach. This technique benefits from all the advantages of the discrete approach while circumventing most of the difficulties mentioned above. Most importantly, it can be

classified as a matrix-free algorithm since the operators are evaluated directly from a nonlinear solver without explicitly forming the resulting matrices. Moreover, it requires minimal storage and can be applied to a general class of spatial discretizations. We further contend that black-box strategies may not necessarily lead to optimal procedures; we thus prefer customization (without loss of generality), allowing for potential code reuse and avoiding restrictions to a specific geometry or set of boundary conditions. Only minor assumptions about the code, in particular its modularity, will be necessary; these assumptions should readily be met by exercising good programming practices and standards. In particular, it is important to notice that only spatial differentiation couples the temporal evolution of flow variables at each grid point to the flow variables at neighboring grid points. As will be shown below, this fact can be exploited, and the general structure of the discrete direct and adjoint operators can be determined.

Even though our technique will be demonstrated on an explicit compressible code, we also mention generalizations of the algorithm to incompressible flow solvers. Once the direct and adjoint operations have been extracted from the nonlinear simulation code, they can be easily incorporated into any quantitative study that requires them. We present (in Section 4) a representative application of our technique: we will illustrate the efficiency of the algorithm by computing direct and adjoint global modes for a spatially developing compressible boundary layer and for flow around an airfoil.

2. Direct and adjoint linear operators from a nonlinear code: demonstration on the compressible Navier–Stokes equations

Before launching into a generalized formulation of the extraction of direct and adjoint operators from a modular nonlinear code, we will focus on a specific case and present a step-by-step algorithm based on a typical setup.

2.1. Direct numerical simulation (DNS) code

A compressible flow solver developed by the authors will be used as a demonstration example. In particular, we will only discuss details of the code as they pertain to the extraction technique. The program is based on the three-dimensional Navier–Stokes equations for compressible flow, formulated in terms of the pressure p, entropy s and velocity field $\mathbf{u}$. The equations are augmented by a material law, specifying the heat flux $\mathbf{q}$, the viscous stress tensor $\bar{\tau}$ and the state equation for an ideal gas, given by $p = \rho R_g T$. The heat capacity at constant volume c_v, the thermal conductivity λ and the viscosities μ and μ_v are taken as constants but a dependence on arbitrary state variables, such as the temperature T, can be readily considered. We have

$$\mathbf{q} = -\lambda \nabla T, \tag{3a}$$

$$\bar{\boldsymbol{\tau}} = \mu\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right) + \left(\mu_v - \frac{2}{3}\mu\right)(\nabla \cdot \mathbf{u})\mathrm{I}, \tag{3b}$$

$$\frac{\partial p}{\partial t} + \mathbf{u} \cdot \nabla p + \rho c^2 \nabla \cdot \mathbf{u} = \frac{p}{c_v}\left(\frac{\partial s}{\partial t} + \mathbf{u} \cdot \nabla s\right), \tag{4a}$$

$$\frac{\partial s}{\partial t} + \mathbf{u} \cdot \nabla s = \frac{1}{\rho T}(-\nabla \cdot \mathbf{q} + \bar{\boldsymbol{\tau}} : \nabla \mathbf{u}), \tag{4b}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\nabla \cdot \bar{\boldsymbol{\tau}}. \tag{4c}$$

Typically, time integration of this set of equations can be performed using the method of lines. An appropriate spatial discretization is applied, together with the treatment of the boundary conditions for the computational domain. The implementation of the boundary conditions can be thought of as the substitution of the governing equations at boundary grid points to impose a desired behavior, i.e. inflow, outflow, solid wall, etc. A common procedure consists in the decomposition of the inviscid part of the equations into incoming and outgoing *characteristics* and specifying the value of the ingoing characteristics and viscous fluxes [33,23]. Sometimes it is preferred to implement Neumann boundary conditions or symmetry conditions directly in the differentiation schemes. In either case, the problem is reduced to the temporal integration of a system of nonlinear ordinary differential equations, symbolically written as

$$\frac{d\mathbf{v}}{dt} = \mathbf{F}(\mathbf{v}), \tag{5}$$

where $\mathbf{v}$ represents the state vector containing all the variables at every grid point, and the right-hand-side $\mathbf{F}(\mathbf{v})$ comprises the governing equations, including the boundary conditions. We will use hereafter $\mathbf{v}$ indistinctly for the state vector and for the small perturbations around a base state $\mathbf{V}$ when there is no ambiguity. A suitable temporal integration scheme is used, starting from a specified initial field.

2.2. Practical implementation using a modular code structure

The evaluation of the right-hand-side $\mathbf{F}(\mathbf{v})$ of Eq. (5) can be performed by algorithmically processing the following steps:

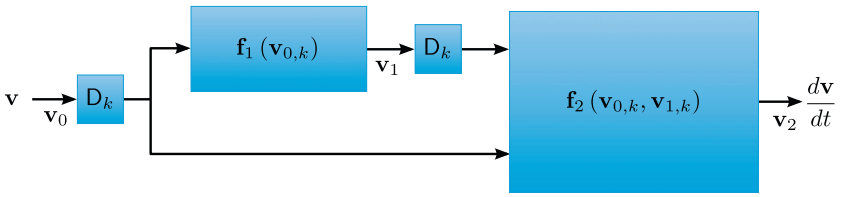

Fig. 1. Block diagram for the evaluation of the right-hand side of the compressible Navier–Stokes equations.

1. Computation of the spatial derivatives of the state vector $\mathbf{v}_0 = [p \quad s \quad \mathbf{u}]^T$; the result will be called $\mathbf{v}_{0,k}$, where k stands for the respective spatial derivatives and can take on the values $k \in \{x, y, 0\}$ with $_0$ denoting no differentiation.
2. Evaluation of $\mathbf{f}_1(\mathbf{v}_{0,k})$, the discrete form of Eq. (3) including boundary conditions.
3. Computation of the spatial derivatives of $\mathbf{v}_1 = [\mathbf{q} \quad \bar{\tau}]$; the result will be called $\mathbf{v}_{1,k}$.
4. Evaluation of $\mathbf{f}_2(\mathbf{v}_{0,k}, \mathbf{v}_{1,k})$, the discrete form of Eq. (4), including boundary conditions; finally, the result ($\mathbf{v}_2$) is identical to the time derivative of $\mathbf{v}$.

In the above, $\mathbf{v}_0$ and $\mathbf{v}$ are used, for convenience and indistinctly, as the state vector containing all the flow variables that are integrated in time. This four-step procedure is sketched in form of a block diagram in Fig. 1. In general, the access to each step is available by practicing a modular layout of the code. The implementation of the boundary conditions fits this layout, as they are commonly enforced by replacing the governing equations on the domain boundaries and, most commonly, can be expressed in terms of the variables introduced above (see [33,23]). In the above diagram the differentiation routine, denoted by D_k, is used as a short form to obtain all the necessary derivatives of the state-vector $\mathbf{v}$, with the notational convention that $\mathbf{v}_{,x} = \mathsf{D}_x\mathbf{v}, \mathbf{v}_{,y} = \mathsf{D}_y\mathbf{v}$, and $\mathbf{v}_{,0} = \mathsf{D}_0\mathbf{v} = \mathbf{v}$. The subscript k thus denotes differentiation with respect to x and y, or (in the case of 0) no differentiation at all (see Fig. 5). The extension to three dimensions is obvious.

The modular structure of the code suggests the introduction of auxiliary variables, which will aid in the description of the subsequent linearization algorithm, as well as in the derivation of the adjoint operator. A new variable for each spatial derivative of the state vector is introduced. We recall that $\mathbf{v}_0 = [p \quad s \quad \mathbf{u}]^T$ is the vector state and $\mathbf{v}_1 = [\mathbf{q} \quad \bar{\tau}]^T$ is an auxiliary variable. Additionally, we have $\mathbf{v}_2 = \frac{\partial \mathbf{v}}{\partial t}$. We can then write symbolically

$$\begin{aligned} &\mathbf{v}_0 && \rightarrow && \mathbf{v}_{0,k} = \mathsf{D}_k\mathbf{v}_0, \\ &\mathbf{v}_1 = \mathbf{f}_1(\mathbf{v}_{0,k}) && \rightarrow && \mathbf{v}_{1,k} = \mathsf{D}_k\mathbf{v}_1, \\ &\mathbf{v}_2 = \mathbf{f}_2(\mathbf{v}_{0,k}, \mathbf{v}_{1,k}) && \rightarrow && \mathbf{v}_2 = \tfrac{d\mathbf{v}}{dt}. \end{aligned} \tag{6}$$

In analogy to the block diagram (Fig. 1), this procedure can be interpreted as two distinct steps. First, the viscous tensor and the heat flux ($\mathbf{v}_1$) are computed from the state vector, together with its derivatives ($\mathbf{v}_{0,k}$). Second, the temporal derivative of the state vector ($\mathbf{v}_2$) is calculated from the flow variables and their derivatives ($\mathbf{v}_{0,k}$), as well as from the heat flux and viscous tensor and their derivatives ($\mathbf{v}_{1,k}$). The proper boundary conditions are incorporated into each respective step.

2.3. The linearization step

It is important to realize that, in the above scheme, only the calculation of the spatial derivatives links the flow field variables at a given point to their neighboring points; all other operations are local (i.e., grid-point-wise). This fact can be exploited to derive a computationally efficient algorithm for the extraction and evaluation of the linearized dynamics.

To illustrate this point, we consider typical nonlinear advection terms $(\mathbf{u} \cdot \nabla)\mathbf{u}$ of a two-dimensional model problem. Following the above-mentioned procedure, we consider the spatial derivatives as independent variables and thus introduce a function $\mathbf{f}(\mathbf{u}, \mathbf{r}, \mathbf{s}) = [\mathbf{r}|\mathbf{s}]\mathbf{u}$ with $\mathbf{r} = \mathsf{D}_x\mathbf{u}$ and $\mathbf{s} = \mathsf{D}_y\mathbf{u}$. This puts further emphasis on the fact that spatial derivatives are calculated separately from the linearization procedure. Continuous differentiation is a linear operation and, for the time being, we will assume that the discrete differentiation scheme is also linear (see below for extensions to nonlinear differentiation schemes). Consequently, only the nonlinear (grid-local) terms need to be linearized. In our particular example of the nonlinear advection term, we have

$$\mathbf{f}(\mathbf{u}, \mathbf{r}, \mathbf{s}) = \begin{pmatrix} u_1 r_1 + u_2 s_1 \\ u_1 r_2 + u_2 s_2 \end{pmatrix} \tag{7}$$

where the subscripts indicate the respective components of the variables. The linearization of this expression has the following structure

$$\left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_0 \mathbf{u} + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{r}}\right|_0 \mathbf{r} + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{s}}\right|_0 \mathbf{s} = \mathbf{A}_u\mathbf{u} + \mathbf{A}_r\mathbf{r} + \mathbf{A}_s\mathbf{s} \tag{8}$$

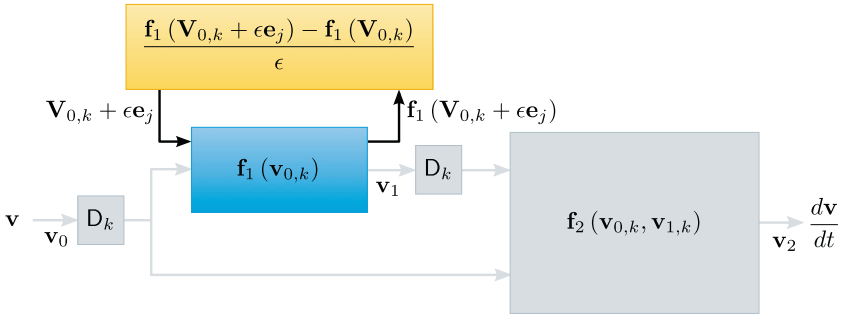

Fig. 2. Block diagram demonstrating the linearization step for a selected module of the code.

where $\mathbf{A}_u, \mathbf{A}_r$ and $\mathbf{A}_s$ are block-diagonal matrices depending only on the base-flow quantities of the associated variables $\mathbf{u}, \mathbf{r}$ and $\mathbf{s}$ (denoted by $\mathbf{U}, \mathbf{R}$ and $\mathbf{S}$, respectively) and the subscript $_0$ denotes evaluation at the base state. Although in this simple example the coefficients can be readily obtained, they can be evaluated in a more systematic way. For instance, the matrix $\mathbf{A}_r$ may be obtained numerically from a nonlinear code by two evaluations of our function $\mathbf{f}(\mathbf{u}, \mathbf{r}, \mathbf{s})$ according to

$$\mathbf{A}_r = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{r}}\right|_0 \approx \left[\frac{\mathbf{f}(\mathbf{U}, \mathbf{R}+\epsilon\mathbf{e}_1, \mathbf{S}) - \mathbf{f}(\mathbf{U}, \mathbf{R}, \mathbf{S})}{\epsilon} \middle| \frac{\mathbf{f}(\mathbf{U}, \mathbf{R}+\epsilon\mathbf{e}_2, \mathbf{S}) - \mathbf{f}(\mathbf{U}, \mathbf{R}, \mathbf{S})}{\epsilon}\right] \tag{9}$$

up to an accuracy of $\mathcal{O}(\sqrt{\epsilon_m})$ (with ϵ_m as the machine precision) which suffices for most applications [21]. Other high-order approximations could be used without additional effort, however, attention must be paid to avoid cancellation errors. The same procedure applies for the determination of $\mathbf{A}_u$ and $\mathbf{A}_s$. In summary, the complete linearization of the nonlinear advection term produces the following three matrices

$$\mathbf{A}_u = \begin{pmatrix} R_1 & S_1 \\ R_2 & S_2 \end{pmatrix} \qquad \mathbf{A}_r = \begin{pmatrix} U_1 & 0 \\ 0 & U_1 \end{pmatrix} \qquad \mathbf{A}_s = \begin{pmatrix} U_2 & 0 \\ 0 & U_2 \end{pmatrix}. \tag{10}$$

The exact linearization of the discretized equation is recovered by replacing $\mathbf{r}$ and $\mathbf{s}$ by $\mathsf{D}_x\mathbf{u}$ and $\mathsf{D}_y\mathbf{u}$, respectively. This process can be extrapolated to the case of the compressible Navier–Stokes equations, and it is illustrated in Fig. 2: the coefficient matrices $\mathbf{A}_{i,j,k}$ symbolize the linearized functions according to

$$\mathbf{A}_{i,j,k} = \left.\frac{\partial \mathbf{f}_i}{\partial \mathbf{v}_{j,k}}\right|_0. \tag{11}$$

In the above expression, the indices i and j represent components of the nonlinear function $\mathbf{f}$ or the variable $\mathbf{v}$, while the index k indicates the derivative (i.e., $k \in \{x, y, 0\}$).

2.4. The direct operator

After each module of the nonlinear code has been linearized following the previous section, we can now assemble the various parts to explicitly derive the algorithm for the evaluation of the linearized direct operator. The linearization leads to

$$\begin{aligned} \mathbf{v}_0 &= \mathbf{v} \\ \mathbf{v}_1 &= \mathbf{A}_{1,0,k}\mathsf{D}_k\mathbf{v}_0 \\ \mathbf{v}_2 &= \mathbf{A}_{2,0,k}\mathsf{D}_k\mathbf{v}_0 + \mathbf{A}_{2,1,k}\mathsf{D}_k\mathbf{v}_1 \rightarrow \frac{d\mathbf{v}}{dt} = \mathbf{v}_2 \end{aligned} \tag{12}$$

where Einstein's summation convention applies for the repeated subscript $_k$. By combining the various terms, a composite linearized operator $\mathbf{A}$ can finally be determined and the temporal evolution of the field $\mathbf{v}$ reads

$$\frac{d\mathbf{v}}{dt} = \underbrace{\left(\mathbf{A}_{2,0,k}\mathsf{D}_k + \mathbf{A}_{2,1,k}\mathsf{D}_k\mathbf{A}_{1,0,l}\mathsf{D}_l\right)}_{\mathbf{A}}\mathbf{v}. \tag{13}$$

This operator then forms the basis for inquiries into the linearized dynamics of the flow, such as its global stability properties or its response to external forcing or noise. It further includes the linearization of the boundary conditions implemented in the nonlinear code, and its form does not depend of any specific geometry. The algorithm for the evaluation of the linearized dynamics has exactly the same structure as the nonlinear code, but replaces the nonlinear blocks by the linearized ones. We can thus benefit from substantial code reuse and parallelization efforts already implemented in the nonlinear code.

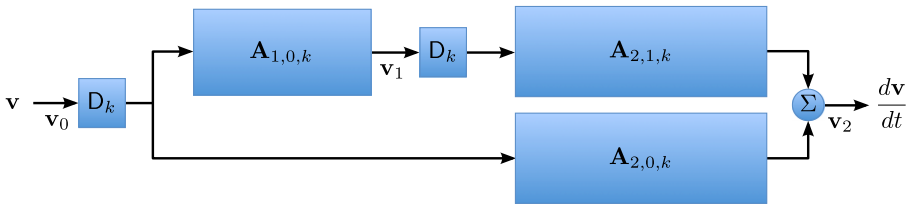

Fig. 3. Block diagram of the fully linearized direct operation.

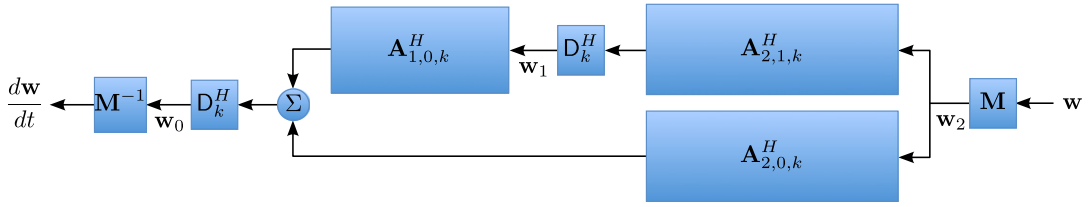

Fig. 4. Block diagram of the fully linearized adjoint operation.

2.5. The adjoint operator

Substantially more information about the linearized dynamics of the flow can be gained by considering the adjoint linearized operator or products of direct and adjoint variables. It is thus desirable to extract this operator from the nonlinear code. We first introduce the inner product $\langle \mathbf{w}, \mathbf{v} \rangle = \mathbf{w}^H \mathbf{M} \mathbf{v}$, with a Hermitian and positive-definite weight matrix $\mathbf{M}$. The matrix $\mathbf{M}$ represents the discretization of the continuous inner product (containing, e.g., metric terms stemming from a non-uniform grid), but can also be used to give more weight to prescribed regions of the computational domain or selected components of the state vector. The adjoint operator is then trivially obtained from the definition $\langle \mathbf{w}, \mathbf{A}\mathbf{v} \rangle = \langle \mathbf{A}^* \mathbf{w}, \mathbf{v} \rangle$, leading to $\mathbf{A}^* = \mathbf{M}^{-1} \mathbf{A}^H \mathbf{M}$. The temporal evolution of the adjoint field $\mathbf{w}$ reads

$$\frac{d\mathbf{w}}{dt} = \underbrace{\mathbf{M}^{-1} \left(\mathsf{D}_k^H \mathbf{A}_{2,0,k}^H + \mathsf{D}_l^H \mathbf{A}_{1,0,l}^H \mathsf{D}_k^H \mathbf{A}_{2,1,k}^H \right) \mathbf{M}}_{\mathbf{A}^*} \mathbf{w}. \tag{14}$$

Due to the modular nature of the extraction procedure for the direct operator, the adjoint operator can be computed by reversing the procedure in Fig. 3 and by conjugate transposing all involved linear modules. A detailed mathematical proof of this procedure is given in appendix A. It shall suffice here to demonstrate the reversal of the direct methodology based on the block-diagram, which is given in Fig. 4. Starting at the right-hand side of the diagram with the variable $\mathbf{w}$, which is adjoint to the direct variable $\mathbf{v}$, we determine in a similar modular fashion the time-derivative of the adjoint variable.

$$\begin{aligned} \mathbf{w}_2 &= \mathbf{M}\mathbf{w} \\ \mathbf{w}_1 &= \mathsf{D}_k^H \mathbf{A}_{2,1,k}^H \mathbf{w}_2 \\ \mathbf{w}_0 &= \mathsf{D}_l^H \mathbf{A}_{1,0,l}^H \mathbf{w}_1 + \mathsf{D}_k^H \mathbf{A}_{2,0,k}^H \mathbf{w}_2 \rightarrow \frac{d\mathbf{w}}{dt} = \mathbf{M}^{-1} \mathbf{w}_0 \end{aligned} \tag{15}$$

The various matrices $\mathbf{A}_{i,j,k}$ which have been determined during the linearization step for the direct operator are simply transconjugated (symbolized by ^{H}). The differentiation matrices D_k need to be transconjugated as well. In one dimension, most of the spatial-derivative approximations using finite differences can symbolically be written as $\mathsf{M}_L \mathbf{v}_{,x} = \mathsf{M}_R \mathbf{v}$ leading to $\mathsf{D} = \mathsf{M}_L^{-1} \mathsf{M}_R$. The matrix–vector multiplication $\mathsf{D}^H \mathbf{w}$ is simple to implement since it only needs the transconjugation of the matrices M_L and M_R and its respective application in reverse order. The extension to spatial derivatives in two and three dimensions is straightforward, since derivatives in higher dimensions can easily be written as Kronecker products of one-dimensional differentiation operators. For instance, considering two dimensions and lexicographical ordering of the grid, the spatial derivatives along x and y are $\mathsf{D}_x = \mathsf{I} \otimes \mathsf{D}$ and $\mathsf{D}_y = \mathsf{D} \otimes \mathsf{I}$, respectively.

The adjoint of the differentiation matrices D_k is displayed in Fig. 5 and involves an additional summation over all components. On one hand, the forward operation D_k takes a single variable $\mathbf{v}$ and produces multiple spatial derivatives ($\mathbf{v}_{,k}$) which are subsequently used and summed in a linearized function block (note that $\mathbf{f}_2$ has been replaced by *two* matrices $\mathbf{A}_{2,1,k}, \mathbf{A}_{2,0,k}$); on the other hand, by reversing this sequence, the adjoint of the function block produces multiple variables from a single-variable input which are in turn "adjoint differentiated" and summed in the D_k^H-block. This is consistent with the summation of repeated indices in the above equations. Adjoint first-order differentiation can be thought of as negative differentiation, since for continuous derivatives we have $(\partial/\partial x)^* = -\partial/\partial x$. The incorporation of boundary closures into the discrete differentiation matrices D_k, however, makes this analogy only true in an approximate or interpretive sense. We shall remark here that the linearized boundary conditions were automatically included in the direct operator, and therefore they are also taken into account in the adjoint operator without any additional effort.

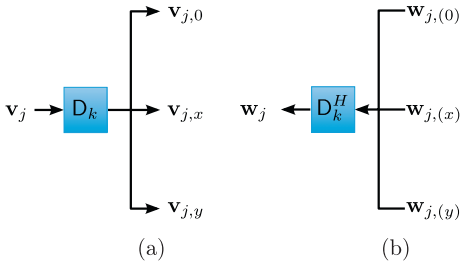

Fig. 5. Differentiation module (a) and its transconjugate (b).

By transconjugation of the procedural algorithm for the evaluation of the direct operator, we have thus obtained the adjoint linearized operator directly from our nonlinear code. We can then use both the direct and adjoint linearized operator to address and quantify various issues related to the linear dynamics of our flow.

3. A general framework for the evaluation of direct and adjoint operators

In this section we extend the evaluation technique of the direct and adjoint operators to an arbitrarily complex nonlinear code that uses an explicit discretization in time. At the end of the section, potential extensions of this procedure to take into account incompressibility or nonlinear differentiation schemes are analyzed.

3.1. Generalization

The state vector $\mathbf{v}$ that describes the evolution of a physical state is supposed to be governed by a system of partial differential equations, nonlinear and first order in time, which can be stated as

$$\frac{\partial \mathbf{v}}{\partial t} = \mathbf{f}(\mathbf{v}) \tag{16}$$

with appropriate initial and boundary conditions. In this equation, an explicit dependence on the spatial coordinates $\mathbf{x}$ and time t will be excluded, even though it could be incorporated in the formulation that follows. When discretized explicitly in time, we arrive at a system of autonomous ordinary differential equations via the method of lines, and a numerical code that implements the right-hand side and advances the state vector in time is assumed to be available. In a continuous formulation, the right-hand side $\mathbf{f}(\mathbf{v})$ is generally a complicated function of not only the state vector $\mathbf{v}$ but also of its spatial derivatives (up to arbitrary order). The link between changes in one of the state variables at a given grid point and associated changes in neighboring grid points is caused by spatial derivatives.

We reconsider the continuous version of the equations and introduce auxiliary variables and local functions. In the discretized equations, nonlinearity arises from expressions involving those auxiliary variables. This latter fact is easily exploited as has been shown before. Following the procedure introduced in the previous section, Eq. (16) can be split into as many auxiliary functions as the order of the highest spatial derivative of the state vector in $\mathbf{f}(\mathbf{v})$. More formally, this can be written as

$$\begin{aligned}
&\mathbf{v}_0 = \mathbf{v} && \rightarrow \quad \mathbf{v}_{0,k} = \tfrac{\partial \mathbf{v}_0}{\partial x_k},\\
&\mathbf{v}_1 = \mathbf{f}_1(\mathbf{v}_{0,k}) && \rightarrow \quad \mathbf{v}_{1,k} = \tfrac{\partial \mathbf{v}_1}{\partial x_k},\\
&\vdots && \quad \vdots\\
&\mathbf{v}_i = \mathbf{f}_i(\mathbf{v}_{0,k},\ldots,\mathbf{v}_{i-1,k}) && \rightarrow \quad \mathbf{v}_{i,k} = \tfrac{\partial \mathbf{v}_i}{\partial x_k},\\
&\vdots\\
&\mathbf{v}_n = \mathbf{f}_n(\mathbf{v}_{0,k},\ldots,\mathbf{v}_{n-1,k}) && \rightarrow \quad \mathbf{v}_n = \tfrac{\partial \mathbf{v}}{\partial t}.
\end{aligned} \tag{17}$$

As before, it is convenient to define $\mathbf{v}_{j,0} = \mathbf{v}_j$. Since our attention is only focused on the extraction of the linearized dynamics from already discretized nonlinear equations, only the discrete formulation will be considered. The structure of the above equations imposes weak constraints on the manner in which expressions are implemented; but a sufficiently well-written simulation code easily satisfies these constraints.

3.2. Linearization

The linearization of an arbitrary auxiliary function $\mathbf{f}_i$ is carried out by introducing a base state $\mathbf{V}$ plus a small perturbation $\mathbf{v}'$, i.e. $\mathbf{v} = \mathbf{V} + \epsilon\mathbf{v}'$. The values of the auxiliary variables at the base state can be obtained by subsequently evaluating $\mathbf{f}_i(\mathbf{V}_{j,k})$ using the nonlinear code. The equation for the perturbation is then

$$\mathbf{v}'_i = \sum_{j=0}^{i-1}\sum_k \left.\frac{\partial \mathbf{f}_i}{\partial \mathbf{v}_{j,k}}\right|_0 \mathbf{v}'_{j,k} \tag{18}$$

where the subscript $_0$ stands for an evaluation at the base state. As before, the submatrices of the Jacobian of $\mathbf{f}_i$ can be determined numerically choosing a sufficiently small value ϵ and evaluating

$$\left[\mathbf{A}_{i,j,k}\right]_{lm} = \left.\frac{\partial \mathbf{f}_i}{\partial \mathbf{v}_{j,k}}\right|_0 \approx \left[\frac{\mathbf{f}_i(\mathbf{V}_{0,k},\ldots,\mathbf{V}_{j,k}+\epsilon\mathbf{e}_m,\ldots,\mathbf{V}_{i-1,k}) - \mathbf{f}_i(\mathbf{V}_{0,k},\ldots,\mathbf{V}_{i-1,k})}{\epsilon}\right]_l. \tag{19}$$

This expression presumes that $\mathbf{f}_i(\mathbf{v}_{j,k})$ is easily accessible. Since all the elements of $\mathbf{A}_{i,j,k}$ can be obtained in very few operations (the auxiliary variables are defined at a grid point), the value of ϵ can be chosen adaptively in order to ensure that the numerical differentiation remains within an acceptable error tolerance. Furthermore, the primes (indicating the linearized variables) are omitted hereafter.

3.3. Direct and adjoint operator evaluation

The linearized version of the discretized system is easily determined by replacing the nonlinear functions by their linearized counterparts. The adjoint Eq. (21) is obtained by explicitly forming the direct operator Eq. (20) that stems from this derivation and applying the definition of the adjoint operator. Details of the derivation of the adjoint operator are given in appendix A, but the main results are included below.

$$\begin{aligned}
&\mathbf{v}_0 = \mathbf{v}\\
&\vdots\\
&\mathbf{v}_i = \sum_{j=0}^{i-1}\sum_k \mathbf{A}_{i,j,k}\mathsf{D}_k\mathbf{v}_j\\
&\vdots\\
&\mathbf{v}_n = \sum_{j=0}^{n-1}\sum_k \mathbf{A}_{n,j,k}\mathsf{D}_k\mathbf{v}_j \rightarrow \frac{d\mathbf{v}}{dt} = \mathbf{v}_n
\end{aligned} \tag{20}$$

$$\begin{aligned}
&\mathbf{w}_n = \mathbf{M}\mathbf{w}\\
&\vdots\\
&\mathbf{w}_i = \sum_{j=i+1}^{n}\sum_k \mathsf{D}_k^H\mathbf{A}_{j,i,k}^H\mathbf{w}_j\\
&\vdots\\
&\mathbf{w}_0 = \sum_{j=1}^{n}\sum_k \mathsf{D}_k^H\mathbf{A}_{j,n,k}^H\mathbf{w}_j \rightarrow \frac{d\mathbf{w}}{dt} = \mathbf{M}^{-1}\mathbf{w}_0
\end{aligned} \tag{21}$$

We see that the procedure follows the outline introduced previously for the specific case of the compressible Navier–Stokes equations. In particular, we recognize the reversed processing and transconjugation of the linearized modules for the adjoint operation.

3.4. Extensions

The above methodology is, so far, applicable to any flow solver provided that the differentiation schemes are linear and the temporal advancement is performed explicitly. In this section we relax these limitations and consider several extensions such as the use of nonlinear differentiation schemes (upwinding, WENO) and mixed discretizations, as is the case for incompressible flow solvers.

3.4.1. Nonlinear differentiation schemes

Weighted essentially non-oscillatory (WENO) schemes. In WENO schemes [40] the differentiation operator is nonlinear and must be included in the linearization procedure. Several stencils are computed using different sets of neighboring points, and a convex nonlinear combination of them is formed to avoid spurious oscillations close to discontinuities. The differentiation operator can be expressed as $\mathsf{D}(\mathbf{v})$, where the value of the derivative at x_i has compact support, i.e. it only depends on the values at $x_{i\pm k}$, where the largest value of k determines the overall stencil width which is commonly very small. If the stencil width is known *a priori*, an efficient linearization can be performed using a tiling technique. In this case, the linearized operator can be computed using $2k_{\max}$ function calls and can be stored in sparse format.

Upwinding. In upwind schemes, two different stencils are considered for the computation of the spatial derivatives and the appropriate one is chosen depending on the sign of the advection velocity. The choice of spatial derivatives using upwinded stencils are $\mathbf{v}^+_{j,k} = \mathsf{D}^+_k\mathbf{v}_j$ and $\mathbf{v}^-_{j,k} = \mathsf{D}^-_k\mathbf{v}_j$, and a decision is made based on a generally nonlinear function which can be written as $\mathbf{f}_i\left(\mathbf{v}^+_{j,k}, \mathbf{v}^-_{j,k}\right)$. If the latter function is differentiable, one can proceed according to the method above; however, this function is usually non-differentiable and thus the linearization has to be performed cautiously. For instance, it is common to switch from one derivative ($^+$) to the other ($^-$) using the sign of the advection speed. In numerical implementations this amounts to an *if-statement* which renders the function non-differentiable and the linearized operator ill-defined. Nevertheless, the *if-statement* can be replaced by a smooth representation such as the logistic function $L(u) \equiv 1/\left(1 + e^{-u/\delta}\right)$, where δ is chosen sufficiently small to achieve a fast transition but sufficiently large to avoid numerical difficulties during the linearization. In practice, a choice of $\delta \sim O(\sqrt{\epsilon_m})$ (with ϵ_m as the machine precision) is typically satisfactory. If we reconsider the example given in Section 2.3 and introduce upwinding in the differentiation along the x direction, we obtain

$$\mathbf{f}(\mathbf{u}, \mathbf{r}^+, \mathbf{r}^-, \ldots) = H(u_1)u_1\mathbf{r}^+ + H(-u_1)u_1\mathbf{r}^- + \cdots \approx \frac{u_1}{1 + e^{-u_1/\delta}}\mathbf{r}^+ + \frac{u_1}{1 + e^{u_1/\delta}}\mathbf{r}^- + \cdots \tag{22}$$

with $H(\cdot)$ denoting the Heaviside function. Performing a linear stability analysis of the resulting discretization, it can be easily verified that such a modification does not introduce numerical instabilities or artifacts. Once the above function is linearized around a base state, the corresponding linearized equation reads

$$\mathbf{A}^+_r\mathbf{r}^+ + \mathbf{A}^-_r\mathbf{r}^- + \cdots \tag{23}$$

and the thus linearized module can be incorporated in the overall procedure for the direct and adjoint operator. In summary, the above procedure to treat upwind schemes aims at avoiding problems arising from numerical differentiation.

3.4.2. Turbulence models

Turbulence models such as Spalart–Allmaras, k-ϵ, k-ω, Reynolds stress model (RSM) and their variants add one or several differential evolution equations for the modeled variables such as turbulent eddy viscosity, turbulent kinetic energy, dissipation, etc., to our governing equations. These new variables and their associated equations can be readily accounted for by expanding the state vector to include the respective quantities. However, it is important to verify that the chosen turbulence model is differentiable or can be regularized similar to the procedure given in the previous section. Also noteworthy is the fact that the linearized system includes the effect of small perturbations in the resolved structures on the modeled turbulent quantities [7].

3.4.3. Incompressible flow solvers

As a further extension, we next consider the incompressible Navier–Stokes equations which, written in terms of primitive variables $(p, \mathbf{u})$, read

$$\nabla \cdot \mathbf{u} = 0, \tag{24}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \nu\nabla^2\mathbf{u}. \tag{25}$$

When integrating the above set of equations in time, it is common practice to consider implicit time-stepping for the diffusion term in order to achieve a reasonable time step. This fact, together with the incompressibility constraint, complicates the straightforward application of our linearization technique. For instance, the time evolution can no longer be written in the form $\frac{d\mathbf{v}}{dt} = \mathbf{F}(\mathbf{v})$ since the incompressibility constraint leads to a separate Poisson equation for the pressure. Nevertheless, we observe that the system of equations resulting from the implicit treatment of diffusion and the incompressibility constraint is already linear and thus does not need to be linearized. Moreover, the coefficient matrix stemming from the discretization of the Laplacian is typically symmetric or can be transconjugated easily to form the corresponding adjoint operation.

There exists a great variety of numerical methods to integrate the above system of equations in time, and the derivation of a general linearization strategy for all of them is beyond the scope of this article. Rather, we consider an extension of our technique to the pressure-less fractional-step method [20]. The discretization is performed on a staggered-grid, where the pressure is defined in the cell centers and the components of the velocity field $\mathbf{u}$ at the cell edges. The advancement over one time step can be decomposed into the following steps:

1. nonlinear advection at interior grid points

$$\frac{\mathbf{u}^a - \mathbf{u}^n}{\Delta t} + (\mathbf{u}^n \cdot \nabla)\mathbf{u}^n = 0, \tag{26}$$

2. diffusion using backward Euler

$$\frac{\mathbf{u}^{ad} - \mathbf{u}^a}{\Delta t} = \nu \nabla^2 \mathbf{u}^{ad} \quad \text{with } \mathbf{u}^{ad}_{\partial\Omega} = \mathbf{u}^n_{\partial\Omega}, \tag{27}$$

3. computation of ϕ and update of the final velocity field

$$\nabla^2 \phi^{n+1} = \frac{1}{\Delta t} \nabla \cdot \mathbf{u}^{ad} \quad \text{with } \frac{\partial \phi^{n+1}}{\partial n} = 0, \tag{28}$$

$$\mathbf{u}^{n+1} = \mathbf{u}^{ad} - \Delta t \nabla \phi^{n+1}. \tag{29}$$

Due to the mixed discretization in space and time, it is desirable to derive a linearization procedure for the operator that propagates a given flow field over one time step, i.e. $\mathbf{v}^{n+1} = \mathbf{A}_{\Delta t}\mathbf{v}^n$, with $\mathbf{v} = [u_i, \phi]^T$. We consider hereafter the discretized equations and introduce the parameter $\alpha = \nu \Delta t$. The temporal advancement of the Stokes operator is readily identified in steps 2–3 and denoted by the linear operator $\mathbf{A}_{\mathcal{S}}$. As far as our linearization procedure is concerned, the Stokes operator $\mathbf{A}_{\mathcal{S}}$ plays a role similar to spatial differentiation and thus can be treated analogous to the latter. The propagation over one time step can now be stated as

$$\begin{aligned} \mathbf{v}_0 = \mathbf{v}^n \quad &\rightarrow \quad \mathbf{v}_{0,k} = \tfrac{\partial \mathbf{v}_0}{\partial x_k} \\ \mathbf{v}_1 = \mathbf{f}_1(\mathbf{v}_{0,k}) \quad &\rightarrow \quad \mathbf{v}^{n+1} = \mathbf{A}_{\mathcal{S}}\mathbf{v}_1, \end{aligned} \tag{30}$$

where $\mathbf{v}_1 = [u_i^a]^T$. Only the function $\mathbf{f}_1(\mathbf{v}_{0,k})$ needs to be linearized. As before, this can be performed using the numerical approximation of quasi-linearization. Once the matrices $\mathbf{A}_{1,0,k}$ are determined, the direct and/or adjoint operators for the propagation over one time step read

$$\begin{aligned} &\mathbf{v}_0 = \mathbf{v}^n \\ &\mathbf{v}_1 = \mathbf{A}_{1,0,k}\mathsf{D}_k\mathbf{v}_0 \rightarrow \mathbf{v}^{n+1} = \mathbf{A}_{\mathcal{S}}\mathbf{v}_1, \end{aligned} \tag{31}$$

$$\begin{aligned} &\mathbf{w}_1 = \mathbf{A}_{\mathcal{S}}^H \mathbf{M}\mathbf{w}^n \\ &\mathbf{w}_0 = \mathsf{D}_k^H \mathbf{A}_{1,0,k}^H \mathbf{w}_1 \rightarrow \mathbf{w}^{n+1} = \mathbf{M}^{-1}\mathbf{w}_0. \end{aligned} \tag{32}$$

What remains is the determination of the explicit expression for the transconjugate of the Stokes operator $\mathbf{A}_{\mathcal{S}} = \mathbf{A}_{\mathcal{P}}\mathbf{A}_{\mathcal{D}}$ (steps 2 and 3). In the latter expression, $\mathbf{A}_{\mathcal{D}}$ is the operator for the diffusion (step 2) and $\mathbf{A}_{\mathcal{P}}$ the computation of the divergence-free velocity field (step 3):

$$\mathbf{A}_{\mathcal{D}} = (\mathsf{I} - \alpha \mathsf{L}_d)^{-1}, \tag{33}$$

$$\mathbf{A}_{\mathcal{P}} = \left[\mathsf{I} - \mathsf{G}\mathsf{L}_n^{-1}\mathsf{D}\right], \tag{34}$$

where L stands for the discretized Laplacian, D for the discretized divergence and G for the discretized gradient. Subscripts $_d$ and $_n$ refer to Dirichlet and homogeneous Neumann boundary conditions, respectively. Using the previous relations we can write the transconjugate of $\mathbf{A}_{\mathcal{S}}$ as $\mathbf{A}_{\mathcal{S}}^H = \mathbf{A}_{\mathcal{D}}^H \mathbf{A}_{\mathcal{P}}^H$ with

$$\mathbf{A}_{\mathcal{D}}^H = (\mathsf{I} - \alpha \mathsf{L}_d)^{-1}, \tag{35}$$

$$\mathbf{A}_{\mathcal{P}}^H = \left[\mathsf{I} - \mathsf{D}^H \mathsf{L}_n^{-1} \mathsf{G}^H\right]. \tag{36}$$

Although the above derivation of the discrete adjoint operator for the linearized incompressible solver is rather simple and can be performed systematically, it may not be as straightforward to implement as for the compressible flow solver. The continuous Stokes operator is self-adjoint, but the use of the pressure projection and the discretization render this property only approximately true for the numerical implementation. For this reason, the transconjugate of the involved operators needs to be performed judiciously. Nevertheless, most of these operators are simple to transconjugate; for instance, L_n and L_d are usually symmetric, and, in the case considered here, D^H and G^H are equal to $-\mathsf{G}$ and $-\mathsf{D}$, respectively, which renders the projection operator $\mathbf{A}_{\mathcal{P}}$ and the diffusion operator $\mathbf{A}_{\mathcal{D}}$ symmetric.

4. Applications

In this section, we focus our attention on the application to compressible and incompressible flow solvers to demonstrate the efficiency and capabilities of the method introduced above. More precisely, the computation of direct and adjoint global modes – a central component in many quantitative flow analyses – will be performed on several flow configurations, namely a spatially developing compressible boundary layer, the compressible flow around an airfoil, and the incompressible flow in a lid-driven cavity. Implementation details and the obtained performances will be discussed in particular.

4.1. Implementation and performance

The compressible flow solver under consideration implements the three-dimensional Navier–Stokes equations in simple multiblock-structured grids using the so-called *pseudo-characteristics* formulation [38] in curvilinear coordinates. The numerical code can be used to address typical flow configurations in aeroacoustics, and, consequently, high-order numerical methods have been chosen in order to accurately resolve all flow features at a reasonable computational cost. In particular, compact upwind low-dissipative (CULD) schemes [1] are used for the advection terms and central compact schemes [22] for the computation of viscous and heat fluxes. The temporal advancement is carried out using a 4th-order low-storage Runge–Kutta scheme [18], and appropriate boundary conditions are implemented by extending the Navier–Stokes characteristics boundary conditions, or NSCBC [33,23], to curvilinear coordinates.

The nonlinear solver is written in C++ and conveniently parallelized using the message-passing library MPI. The evaluation of the direct and adjoint operators is implemented in two distinct modules: (a) the numerical linearization of the auxiliary nonlinear functions around a base state, Eq. (19), and (b) the evaluation of the linearized direct and adjoint operators, Eqs. (12) and (15), respectively. Both modules can straightforwardly be incorporated into quantitative investigations that rely on them. It is worth pointing out that, in the case under consideration, both modules represent only 9% of the lines-of-code of the entire program: 4% for the linearization module and 5% for the evaluation module. The latter module shares most of the routines with its nonlinear counterpart and thus inherits its performance characteristics and reuses already parallelized code. Moreover, the linearization module automatically takes into account modifications in the boundary conditions and even in the governing equations.

More precisely, the linearization process is performed by evaluating Eq. (19) with a value of ϵ sufficiently small, typically $\sqrt{\epsilon_m}$, where ϵ_m is the machine precision. The optimal value can be determined by repeated evaluation using progressively smaller values of ϵ. This procedure yields the coefficients of $\mathbf{A}_{i,j,k}$ at an expected convergence rate, e.g., $\mathcal{O}(\epsilon)$ for our first-order Jacobian approximation. By further reducing ϵ, cancellation errors begin to dominate until they contaminate the approximate coefficients. The memory cost associated with the storage of the matrices $\mathbf{A}_{i,j,k}$ is usually small. For example, in our case, the storage of over one hundred coefficients per grid point are typically required for two-dimensional problems.

In order to verify the accuracy of the linearization procedure and the implementation of the direct operator, the norm of the difference between the linearized direct operator obtained using Eq. (2) and our technique is presented in Fig. 6. A local minimum in relative error between the two techniques is reached for a parameter $\epsilon \approx 4 \cdot 10^{-8}$; for values above, monotonic first-order convergence is observed, whereas for values below, round-off errors start to dominate. The implementation of the adjoint operator has been verified using the identity $\langle \mathbf{w}, \mathbf{A}\mathbf{v} \rangle = \langle \mathbf{A}^* \mathbf{w}, \mathbf{v} \rangle$, which for the cases presented here, is satisfied up to round-off error.

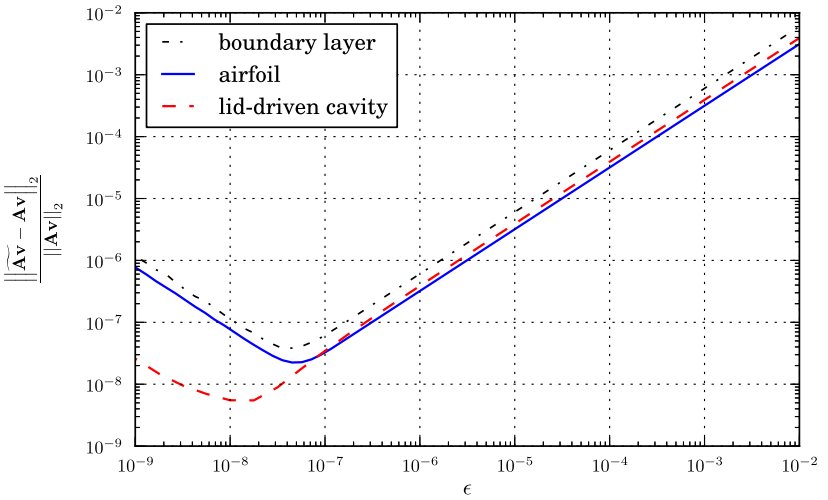

Fig. 6. Relative residual for the linearized direct operator for the application cases.

Table 1
Comparison between the time spent on the evaluation of the different operators and the linearization procedure. The ratio to the time spent on the nonlinear operator evaluation, shown in bold, is also presented.

Case	Nonlinear		Linearization		Direct		Adjoint	
	Time (s)	**Ratio**	Time (s)	**Ratio**	Time (s)	**Ratio**	Time (s)	**Ratio**
Boundary layer	0.0096	**1.0**	0.1575	**16.4**	0.0103	**1.07**	0.0109	**1.17**
Airfoil	0.0428	**1.0**	0.4690	**11.0**	0.0604	**1.41**	0.0624	**1.45**

A comparison between the time spent in the evaluation of the nonlinear and linearized operators, together with the time spent in the linearization process, is shown in Table 1. As expected, the nonlinear and linearized direct and adjoint codes show nearly identical performances; the time spent in the linearization module is comparable to the time it takes to perform one evaluation, since the computation of matrix coefficients can be performed very efficiently.

The minimal additional programming effort that is required for the implementation of this technique, specially in the case of compressible flow solvers, and its performance result in a substantial advantage over linearized codes obtained from automatic differentiation. In the latter case, the *differentiated* code is usually as long as the original code, and sustained performance can only be achieved via careful optimization.

The above-mentioned modules have also been implemented for an incompressible flow solver. The advancement in time of the Stokes operator is performed using the pressure-less fractional-step method described in [20]. The nonlinear advection terms are discretized in space using upwinded, central finite differences and advanced in time using the forward Euler method.

Finally, we illustrate the potential applications of this technique by computing direct and adjoint global modes of several flow configurations. The temporal advancement of the linearized operators [10] has been coupled with the eigenvalue solver SLEPc [14]. The calculations have been performed using the Krylov–Schur algorithm [44], together with the harmonic-extraction method [30] to select different parts of interest of the spectrum.

4.2. Spatially developing compressible boundary layer

We first consider a two-dimensional compressible boundary layer over a flat plate. The boundary-layer displacement thickness δ_1 at the inlet of the domain is taken as the reference length; the Reynolds number is $\mathrm{Re}_{\delta_1} = 1000$ and the Mach number is $M = 0.8$. The domain extends over 800 and 40 unit lengths along the tangential and wall normal directions, respectively. The numerical grid is refined in the vertical direction in the vicinity of the wall. The velocity and entropy profiles obtained from the self-similar solution of the compressible boundary-layer equations are imposed at the inlet of the domain using the characteristics-based approximate non-reflecting inflow boundary condition given in [23]. A no-slip adiabatic boundary condition is imposed at the wall, and a characteristics-based approximate non-reflecting outflow boundary condition is implemented at the free-stream boundary and at the outlet of the domain. For this first example demonstrating the matrix extraction technique, we will concentrate on a classical global stability analysis, determine the direct spectrum and focus on the least stable global modes.

Given a suitable initial condition, the flow is advanced in time until the norm of the time derivative falls below 10^{-8}. The linearization of the equations around this steady solution is performed, and the temporal propagator related to the linearized

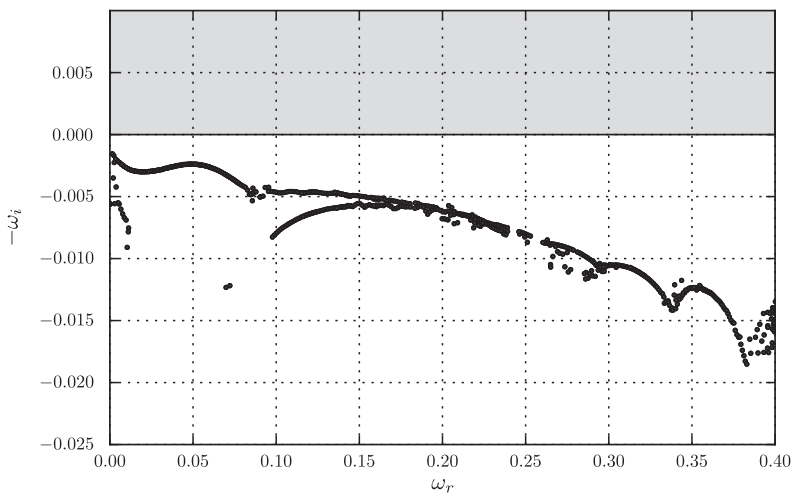

Fig. 7. Global spectrum of compressible boundary layer flow for $\mathrm{Re}_{\delta_1} = 1000$ and M = 0.8, displaying three families of spectral branches.

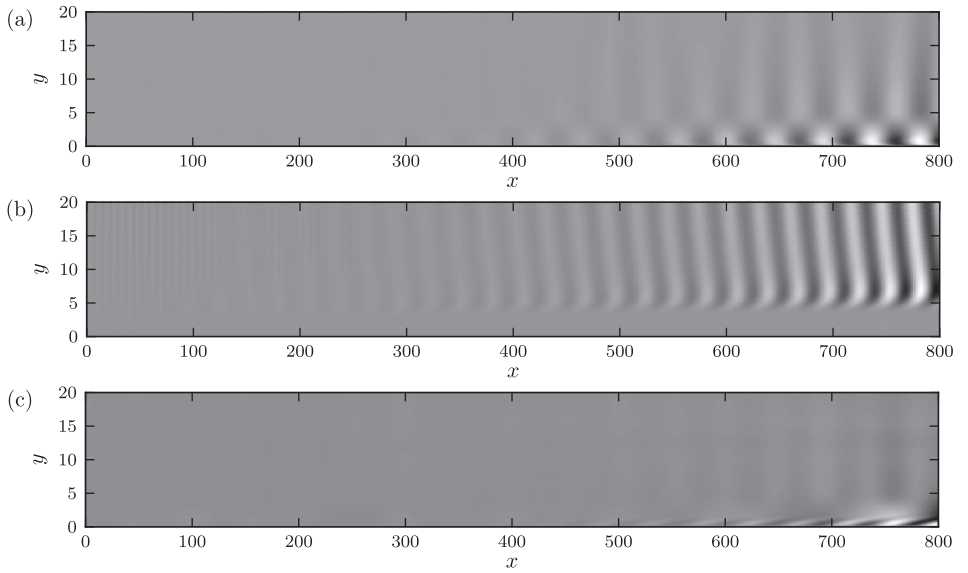

Fig. 8. Three different types of modes selected from the global spectrum for the compressible boundary layer with $\mathrm{Re}_{\delta_1} = 1000$ and M = 0.8. (a) Tollmien–Schlichting (TS) waves, (b) free-stream modes and (c) Orr modes.

direct problem is used to obtain the global modes. In the global spectrum (Fig. 7), three types of modal branches can be recognized, linked to Tollmien–Schlichting waves, Orr modes and free-stream modes.

Selected eigenfunctions corresponding to each of these branches are displayed in Fig. 8. The Tollmien-Schlichting branch is characterized by low phase velocities on the order of 30% of the free-stream value. Its associated structures show compact support inside the boundary layer and exponential decay towards the free-stream (see Fig. 8(a)). Exponential spatial growth in the streamwise direction is also observed for this type of modes. The disturbance dynamics in the free-stream is represented by the free-stream modes (see Fig. 8(b)) which show a phase velocity comparable to the free-stream value. We notice

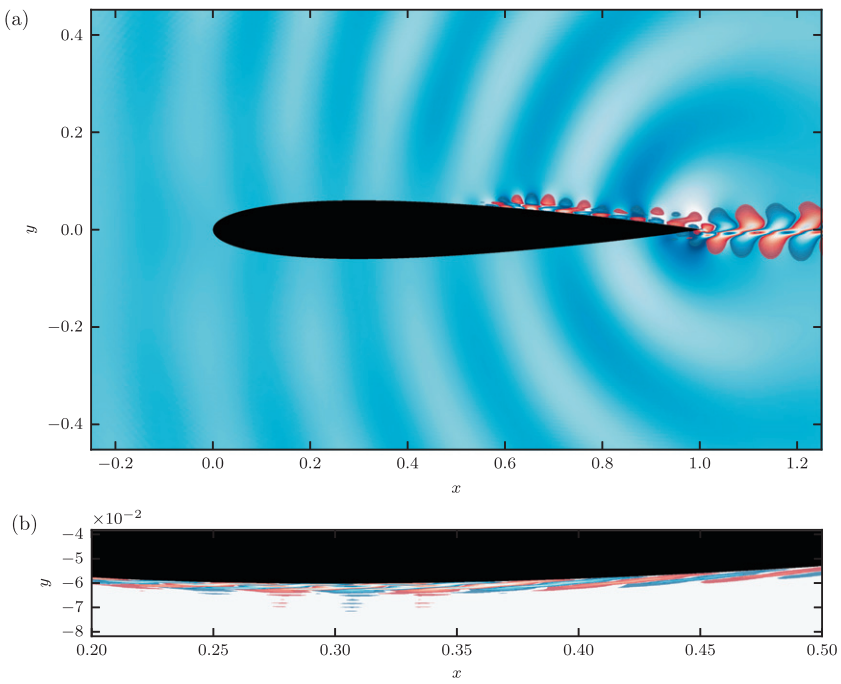

Fig. 9. Leading direct (a) and adjoint (b) global modes ($\lambda \approx 43.8i$) for compressible flow around an NACA0012 airfoil at 2° angle of attack, $\mathrm{Re} = 2 \cdot 10^5$ and M = 0.4. In both cases pressure field and streamwise velocity contours are shown.

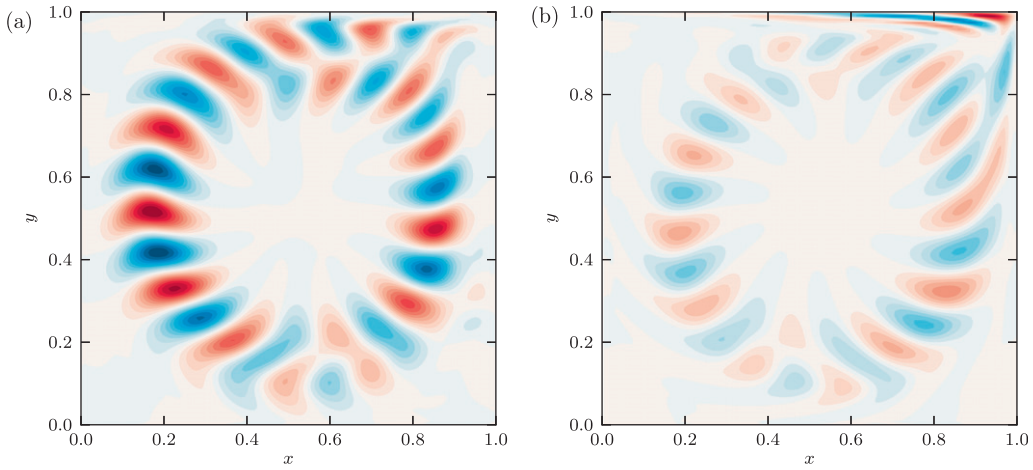

Fig. 10. Representative direct (a) and adjoint (b) global mode in a lid-driven square cavity at Re = 10000.

the typical weak exponential decay towards the free-stream as well as substantially stronger exponential decay towards the wall inside the boundary layer. These types of modes are particularly important for receptivity studies which address the transfer of disturbance energy between the free-stream and the boundary layer. The final type of mode is known as the Orr modes which commonly occur in predominantly two-dimensional configurations. They represent a specific dynamics whereby structures extract energy from the background base-flow via a tilting process induced by the mean shear. Similar to the Tollmien–Schlichting waves, Orr modes are confined to the boundary layer.

4.3. Sound generation by an airfoil

In this example, the compressible flow around a NACA0012 airfoil at 2° angle of attack is computed. The Reynolds number based on the chord length is $\mathrm{Re} = 2 \cdot 10^5$ and the Mach number is M = 0.4. The Navier–Stokes equations are solved on a C-grid that extends over 7 chord lengths along the wake and wall-normal direction. The numerical grid consists of three curvilinear blocks with a total of 3840×384 points. In the chosen parameter regime, the flow exhibits a substantial level of acoustic noise that appears as a sharp peak in the frequency spectrum. Experimental and theoretical studies show that the generation of this type of sound can be linked to the linear stability of the flow. In order to avoid contamination of the acoustic field with spurious reflections caused by vortical structures as they leave the domain, a sponge layer is added at the outlet downstream of the airfoil.

A nonlinear simulation of this configuration is conducted until the flow reaches a quasi-periodic state, and the linearization of the governing equations is performed around the mean flow. Fig. 9 presents the most unstable direct and adjoint global mode extracted by our algorithm. Each panel depicts the pressure field and the streamwise velocity field. For the direct mode we observe a strong spatial growth in the chordwise direction that culminates near the trailing edge. In addition, amplified structures are visible near the separation bubble on the suction (upper) side. The pressure field displays the associated sound field scattered by the fluid structures. Again, the strongest emission of sound stems from the trailing edge of the airfoil. The corresponding adjoint global mode identifies structures on the pressure (lower) side of the airfoil and can be used to identify regions where the associated direct global mode (Fig. 9(a)) is particularly sensitive to perturbations. Both direct and adjoint modes play an important role in pinpointing localized areas that may be the source of self-sustained oscillations.

4.4. Lid-driven cavity flow

We conclude this section by presenting results obtained using the incompressible flow solver. The flow in a lid-driven square cavity at $\mathrm{Re} = 10,000$ is considered. A typical high-order cavity mode is displayed in Fig. 10. It shows a typical vortical structure superimposed on the mean-flow cavity vortex. The adjoint mode (Fig. 10(b)) shows additional features near the downstream edge of the lid which indicates increased sensitivity in this region for the excitation of the associated direct mode (Fig. 10(a)).

This display of modes has been included here for demonstration purposes only, to illustrate the capability of the matrix-extraction technique even for incompressible flow solvers. In true stability calculations, a less dissipative spatial discretization scheme has to be chosen in order to isolate the physical dissipation from the dissipation of the numerical scheme. In this sense, the upwind scheme chosen for this demonstration is unsuitable to determine proper growth rates even though the qualitative features of the modes are readily captured.

5. Summary and conclusions

In this article, we have presented a novel technique for the efficient evaluation of the linearized discrete direct and adjoint operators directly from existing compressible nonlinear simulation codes. The discrete approach contains several advantages over the continuous one: if the nonlinear solver is assumed to accurately describe the underlying physics, the associated linearized operators will consequently represent the linear dynamics and characterize their sensitivities. The derivation of these linearized operators can be obtained in a systematic manner without giving specific thoughts to boundary conditions or numerical discretizations. Nevertheless, the derivation of efficient procedures for discrete linearized operators, especially for the compressible case, can be cumbersome, error-prone and challenging. However, the apparent difficulties can be overcome by carefully analyzing the structure of the discretized governing equations. A crucial observation is the fact that (i) variables at different grid points are only linked by spatial discretization, which is commonly a linear operation, and (ii) nonlinearities arise from expressions that relate local quantities at a single grid point. This observation also holds for the boundary conditions considered in this article. By introducing auxiliary variables and assuming a modular structure of the code, the nonlinear modules for the evaluation of the nonlinear terms can be linearized with a small number of function evaluations and minimal memory requirements. The adjoint of the linearized direct operator can be obtained by transconjugating the direct operator, or equivalently, *reversing* the block-diagram that represents the evaluation of direct operator. The evaluation technique consists of two steps: first, the numerical linearization of the nonlinear local expressions and, second, the evaluation of the direct and adjoint operators using a sequence of all linearizations. We have shown that the structure of the linearized direct and adjoint operators allows for potential code reuse and parallelization efforts from the nonlinear solver. For the compressible case shown here, the total added code represents only 9% of the entire program. The routines for the evaluation of the linearized operators inherit the performance characteristics of the nonlinear code. The evaluation technique has been derived and demonstrated first to a compressible flow solver and has then been extended to also treat nonlinear differentiation schemes, turbulence models and incompressible flow solvers. The presented algorithm has been applied to and showcased on three selected flow configurations: a compressible spatially growing boundary layer, compressible flow around an airfoil and the incompressible flow in a lid-driven square cavity. Global spectra and global direct and adjoint eigenfunctions have been presented and discussed.

In particular, in the field of computational aeroacoustics (CAA) this technique could have substantial benefits in the analysis, optimization and control of noise-generating mechanisms. Even though earlier attempts at direct-adjoint optimizations have been made using a continuous formulation [5,46], the above technique based on the discrete adjoint offers additional advantages.

The efficient extraction of direct and adjoint operators has numerous and important applications in the quantitative analyses of complex fluid flows. Modal solutions of the direct operator give insight into stability properties, receptivity characteristics, and physical transition mechanisms. Combined with the associated adjoint modes, they allow for the quantification and localization of sensitivity measures, feedback mechanisms, and gradient information for optimization schemes. Moreover, direct and adjoint information is prevalent – but, in general, difficult to obtain – in the design of active and passive control strategies and in the reduction of high-dimensional models. For this reason, an efficient technique that provides this necessary information directly from nonlinear simulation codes with a moderate amount of effort is valuable and welcome addition to the currently available methods to analyze complex configurations arising in multi-physics and multi-scales flow applications.

Acknowledgments

The first author wishes to thank Xavier Garnaud for fruitful discussions. This work was performed using HPC resources from GENCI-CINES (Grant 2011-026451).

Appendix A. Derivation of the adjoint

In order to proceed with the derivation of the adjoint of Eq. (20), we seek an expression for the time derivative of the state vector $\mathbf{v}$, denoted by $\mathbf{v}_n$ in terms of $\mathbf{v}$. We start by expressing the dependence of an intermediate variable $\mathbf{v}_i$ on $\mathbf{v}_0,\dots,\mathbf{v}_{i-1}$ in matrix form:

$$\underbrace{\begin{bmatrix}\mathbf{v}_1\\ \mathbf{v}_2\\ \vdots\\ \mathbf{v}_{i-1}\\ \mathbf{v}_i\end{bmatrix}}_{\mathbf{v}_{1:i}} = \underbrace{\begin{bmatrix}0 & 0 & \cdots & 0 & 0\\ \mathbf{A}_{2,1,k}\mathrm{D}_k & 0 & \cdots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots\\ \mathbf{A}_{i-1,1,k}\mathrm{D}_k & \mathbf{A}_{i-1,2,k}\mathrm{D}_k & \cdots & 0 & 0\\ \mathbf{A}_{i,1,k}\mathrm{D}_k & \mathbf{A}_{i,2,k}\mathrm{D}_k & \cdots & \mathbf{A}_{i,i-1,k}\mathrm{D}_k & 0\end{bmatrix}}_{\mathbf{L}_i} \underbrace{\begin{bmatrix}\mathbf{v}_1\\ \mathbf{v}_2\\ \vdots\\ \mathbf{v}_{i-1}\\ \mathbf{v}_i\end{bmatrix}}_{\mathbf{v}_{1:i}} + \underbrace{\begin{bmatrix}\mathbf{A}_{1,0,k}\mathrm{D}_k\\ \mathbf{A}_{2,0,k}\mathrm{D}_k\\ \vdots\\ \mathbf{A}_{i-1,0,k}\mathrm{D}_k\\ \mathbf{A}_{i,0,k}\mathrm{D}_k\end{bmatrix}}_{\mathbf{I}_{1:i,0}} \mathbf{v}_0, \tag{A.1}$$

where $\mathbf{v}_{1:i}$ is the composite vector containing the intermediate variables $\mathbf{v}_1,\dots,\mathbf{v}_i$. The lower-triangular matrix $\mathbf{L}_i$ and the matrix $\mathbf{I}_{1:i,0}$ describe in shorthand the relation between the auxiliary variables and the vector state $\mathbf{v}$. Using the above definitions, we write

$$\begin{bmatrix} \mathbf{I}-\mathbf{L}_{n-1} & 0 \\ -\mathbf{l}_{n,1:n-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{v}_{1:n-1} \\ \mathbf{v}_n \end{bmatrix} = \begin{bmatrix} \mathbf{l}_{1:n-1,0} \\ \mathbf{l}_{n,0} \end{bmatrix} \mathbf{v}_0. \tag{A.2}$$

This system of equations can readily be reformulated by eliminating the composite vector $\mathbf{v}_{1:n-1}$. We obtain

$$\frac{d\mathbf{v}}{dt} = \mathbf{v}_n = \left(\mathbf{l}_{n,1:n-1} (\mathbf{I}-\mathbf{L}_{n-1})^{-1} \mathbf{l}_{1:n-1,0} + \mathbf{l}_{n,0} \right) \mathbf{v} \tag{A.3}$$

and, given the scalar product $\langle \mathbf{w}, \mathbf{v} \rangle = \mathbf{w}^H \mathbf{M} \mathbf{v}$. The equation adjoint the one above reads

$$\frac{d\mathbf{w}}{dt} = \mathbf{M}^{-1} \left(\mathbf{l}^H_{1:n-1,0} \left(\mathbf{I}-\mathbf{L}^H_{n-1} \right)^{-1} \mathbf{l}^H_{n,1:n-1} + \mathbf{l}^H_{n,0} \right) \mathbf{M} \mathbf{w}. \tag{A.4}$$

The evaluation of the above adjoint can be expressed in an equivalent form to Eq. (20), which is more amenable for practical implementation. After introducing the auxiliary adjoint variables $\mathbf{w}_n = \mathbf{M}\mathbf{w}, \mathbf{w}_{1:n-1} = \left(\mathbf{I}-\mathbf{L}^H_{n-1} \right)^{-1} \mathbf{l}^H_{n,1:n-1} \mathbf{w}_n$, we have $\frac{d\mathbf{w}}{dt} = \mathbf{M}^{-1}\mathbf{w}_0$. Combining these definitions, we arrive at an expression analogous to Eq. (A.2). It reads

$$\left[\begin{array}{c|c} \mathbf{I} & -\mathbf{l}^H_{1:n-1,0} \\ \hline 0 & \mathbf{I}-\mathbf{L}^H_{n-1} \end{array} \right] \left[\begin{array}{c} \mathbf{w}_0 \\ \hline \mathbf{w}_{1:n-1} \end{array} \right] = \left[\begin{array}{c} \mathbf{l}^H_{n,0} \\ \hline \mathbf{l}^H_{n,1:n-1} \end{array} \right] \mathbf{w}_n. \tag{A.5}$$

Finally we obtain the familiar sequential procedure given by

$$\begin{aligned} &\mathbf{w}_n = \mathbf{M}\mathbf{w} \\ &\vdots \\ &\mathbf{w}_i = \sum_{j=i+1}^{n} \sum_k \mathsf{D}^H_k \mathbf{A}^H_{j,i,k} \mathbf{w}_j \\ &\vdots \\ &\mathbf{w}_0 = \sum_{j=1}^{n} \sum_k \mathsf{D}^H_k \mathbf{A}^H_{j,0,k} \mathbf{w}_j \rightarrow \frac{d\mathbf{w}}{dt} = \mathbf{M}^{-1}\mathbf{w}_0. \end{aligned} \tag{A.6}$$

References

[1] N.A. Adams, K. Shariff, A high-resolution hybrid compact-ENO scheme for shock-turbulence interaction problems, J. Comput. Phys. 127 (1) (1996) 27–51.
[2] G.A. Brès, T. Colonius, Three-dimensional instabilities in compressible flow over open cavities, J. Fluid Mech. 599 (2008) 309–339.
[3] A. Carle, M. Fagan, ADIFOR 3.0 overview. Tech. Rep. CAAM-TR-00-02, Department of Computational and Applied Mathematics, Rice University, 2000.
[4] J.M. Chomaz, Global instabilities in spatially developing flows: non-normality and nonlinearity, Annu. Rev. Fluid Mech. 37 (2005) 357–392.
[5] S.S. Collis, K. Ghayour, M. Heinkenschloss, Optimal transpiration boundary control for aeroacoustics, AIAA J. 41 (7) (2003) 1257–1270.
[6] T. Colonius, S.K. Lele, Computational aeroacoustics: progress on nonlinear problems of sound generation, Prog. Aerosp. Sci. 40 (6) (2004) 345–416.
[7] J.D. Crouch, A. Garbaruk, D. Magidov, Predicting the onset of flow unsteadiness based on global instability, J. Comput. Phys. 224 (2) (2007) 924–940.
[8] G. Desquesnes, M. Terracol, P. Sagaut, Numerical investigation of the tone noise mechanism over laminar airfoils, J. Fluid Mech. 591 (2007) 155–182.
[9] P.G. Drazin, W. Reid, Hydrodynamic Stability, 2nd Edition., Cambridge University Press, Cambridge Mathematical Library, 2004.
[10] W.S. Edwards, L.S. Tuckerman, R.A. Friesner, D.C. Sorensen, Krylov methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 110 (1994) 82–102.
[11] F. Giannetti, P. Luchini, Structural sensitivity of the first instability of the cylinder wake, J. Fluid Mech. 581 (2007) 167–197.
[12] M. Giles, N. Pierce, An introduction to the adjoint approach to design, Flow Turbul. Combust. 65 (3) (2000) 393–415.
[13] L. Hascoët, V. Pascual, TAPENADE 2.1 user's guide. Rapport technique 300, INRIA, Sophia Antipolis, 2004.
[14] V. Hernandez, J.E. Roman, V. Vidal, SLEPc: a scalable and flexible toolkit for the solution of eigenvalue problems, ACM T. Math. Software 31 (3) (2005) 351–362.
[15] D.C. Hill, A theoretical approach for analyzing the restabilization of wakes. Tech. Rep. 103858, NASA, 1992.
[16] M.P. Juniper, Triggering in the horizontal rijke tube: non-normality, transient growth and bypass transition, J. Fluid Mech. 667 (2011) 272–308.
[17] C.T. Kelley, Iterative Methods for Optimization, Frontiers in Applied Mathematics, 18, Society for Industrial and Applied Mathematics, Philadelphia, 1999.
[18] C.A. Kennedy, M.H. Carpenter, R.M. Lewis, Low-storage, explicit Runge–Kutta schemes for the compressible Navier–Stokes equations, Appl. Numer. Math. 35 (3) (2000) 177–219.
[19] J. Kim, T.R. Bewley, A linear systems approach to flow control, Annu. Rev. Fluid Mech. 39 (2006) 383–417.
[20] J. Kim, P. Moin, Application of a fractional-step method to incompressible Navier–Stokes equations, J. Comput. Phys. 59 (2) (1985) 308–323.
[21] D.A. Knoll, D.E. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2) (2004) 357–397.
[22] S.K. Lele, Compact finite difference schemes with spectral-like resolution, J. Comput. Phys. 103 (1) (1992) 16–42.
[23] G. Lodato, P. Domingo, L. Vervisch, Three-dimensional boundary conditions for direct and large-eddy simulation of compressible viscous flows, J. Comput. Phys. 227 (10) (2008) 5105–5143.
[24] C.J. Mack, P.J. Schmid, A preconditioned krylov technique for global hydrodynamic stability analysis of large-scale compressible flows, J. Comput. Phys. 229 (3) (2010) 541–560.
[25] C.J. Mack, P.J. Schmid, Global stability of swept flow around a parabolic body: the neutral curve, J. Fluid Mech. 678 (2011) 589–599.
[26] O. Marquet, D. Sipp, L. Jacquin, Sensitivity analysis and passive control of cylinder flow, J. Fluid Mech. 615 (2008) 221–252.
[27] P. Meliga, D. Sipp, J. Chomaz, Effect of compressibility on the global stability of axisymmetric wake flows, J. Fluid Mech. 660 (2010) 499–526.
[28] J. Müller, P. Cusdin, On the performance of discrete adjoint CFD codes using automatic differentiation, Int. J. Numer. Methods Fl. 47 (8–9) (2005) 939–945.

[29] K. Mohseni, T. Colonius, J.B. Freund, An evaluation of linear instability waves as sources of sound in a supersonic turbulent jet, Phys. Fluids 14 (10) (2002) 3593.
[30] R.B. Morgan, M. Zeng, A harmonic restarted arnoldi algorithm for calculating eigenvalues and determining multiplicity, Linear Algebra Appl. 415 (1) (2006) 96–113.
[31] J. Nocedal, S.J. Wright, Numerical Optimization, 2nd Edition., Springer, 2006.
[32] S. Pirozzoli, Numerical methods for High-Speed flows, Annu. Rev. Fluid Mech. 43 (1) (2011) 163–194.
[33] T.J. Poinsot, S.K. Lele, Boundary conditions for direct simulations of compressible viscous flows, J. Comput. Phys. 101 (1) (1992) 104–129.
[34] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd Edition., Society for Industrial and Applied Mathematics, 2003.
[35] Y. Saad, Numerical Methods for Large Eigenvalue Problems, revised edition., Society for Industrial and Applied Mathematics, 2011. Edition.
[36] P.J. Schmid, Nonmodal stability theory, Annu. Rev. Fluid Mech. 39 (2007) 129–162.
[37] P.J. Schmid, D.S. Henningson, Stability and Transition in Shear Flows, Applied Mathematics Sciences, 142, Springer, 2001.
[38] J. Sesterhenn, A characteristic-type formulation of the Navier–Stokes equations for high order upwind schemes, Comput. Fluids 30 (1) (2000) 37–67.
[39] R. Seydel, Practical Bifurcation and Stability Analysis, Interdisciplinary Applied Mathematics, 3rd Edition., Springer, 2010.
[40] C.W. Shu, Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. Tech. Rep. NASA/CR-97-206253, NASA 1998.
[41] D. Sipp, A. Lebedev, Global stability of base and mean flows: a general approach and its applications to cylinder and open cavity flows, J. Fluid Mech. 593 (2007) 333–358.
[42] Z. Sirkes, E. Tziperman, Finite difference of adjoint or adjoint of finite difference?, Mon Weather Rev. 125 (12) (1997) 3373–3378.
[43] B. Spagnoli, C. Airiau, Adjoint analysis for noise control in a two-dimensional compressible mixing layer, Comput. Fluids 37 (4) (2008) 475–486.
[44] G.W. Stewart, A Krylov–Schur algorithm for large eigenproblems, SIAM. J. Matrix Anal. Appl. 23 (3) (2002) 601–614.
[45] P.J. Strykowski, K.R. Sreenivasan, On the formation and suppression of vortex 'shedding' at low reynolds numbers, J. Fluid Mech. 218 (1990) 71–107.
[46] M. Wei, J.B. Freund, A noise-controlled free shear flow, J. Fluid Mech. 546 (2005) 123–152.
[47] S. Zuccher, A. Bottaro, P. Luchini, Algebraic growth in a blasius boundary layer: nonlinear optimal disturbances, Eur. J. Mech. B Fluid 25 (1) (2006) 1–17.