Journal of Computational Physics 229 (2010) 5394–5407

Contents lists available at ScienceDirect

Journal of Computational Physics

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

Local-in-time adjoint-based method for design optimization of unsteady flows

Nail K. Yamaleev [a,*], Boris Diskin [b,d], Eric J. Nielsen [c]

[a] Department of Mathematics, North Carolina A&T State University, Greensboro, NC 27411, USA
[b] National Institute of Aerospace, Hampton, VA 23666, USA
[c] Computational AeroSciences Branch, NASA Langley Research Center, Hampton, VA 23681, USA
[d] Department of Mechanical and Aerospace Engineering, University of Virginia, Charlottesville, VA 22904, USA

ARTICLE INFO

Article history:
Received 20 May 2009
Received in revised form 1 February 2010
Accepted 29 March 2010
Available online 4 April 2010

Keywords:
Time-dependent optimization
Discrete adjoint equations
Gradient methods
Design optimization
Euler equations

ABSTRACT

We present a new local-in-time discrete adjoint-based methodology for solving design optimization problems arising in unsteady aerodynamic applications. The new methodology circumvents storage requirements associated with the straightforward implementation of a global adjoint-based optimization method that stores the entire flow solution history for all time levels. This storage cost may quickly become prohibitive for large-scale applications. The key idea of the local-in-time method is to divide the entire time interval into several subintervals and to approximate the solution of the unsteady adjoint equations and the sensitivity derivative as a combination of the corresponding local quantities computed on each time subinterval. Since each subinterval contains relatively few time levels, the storage cost of the local-in-time method is much lower than that of the global methods, thus making the time-dependent adjoint optimization feasible for practical applications. Another attractive feature of the new technique is that the converged solution obtained with the local-in-time method is a local extremum of the original optimization problem. The new method carries no computational overhead as compared with the global implementation of adjoint-based methods. The paper presents a detailed comparison of the global- and local-in-time adjoint-based methods for design optimization problems governed by the unsteady compressible 2-D Euler equations.

1. Introduction

The continuous growth of computer power and the development of efficient and accurate computational tools now attract more attention to design optimization of unsteady flows. The time-dependent optimization problems arise in many aerodynamic applications including optimal design of helicopter rotors and turbomachinery blades, flutter and vibration control, noise reduction, active and passive flow control, etc. These problems can be formulated as minimization/maximization of appropriate cost functionals (e.g., lift, drag, torque, etc.) and can be solved by utilizing optimal control theory.

Among various optimization techniques available in the literature, adjoint-based gradient methods have recently grown in popularity, rapidly becoming one of the most widely used algorithms for solving a variety of steady and unsteady optimization problems. The adjoint methodology is particularly attractive for aerodynamic shape/design optimization problems that are characterized by the presence of a large number of design variables, yet relatively few constraints. In contrast to a classical forward mode differentiation approach whose computational cost is directly proportional to the number of design

* Corresponding author. Tel.: +1 336 285 2093; fax: +1 336 256 0876.
E-mail address: nkyamale@ncat.edu (N.K. Yamaleev).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.03.045

variables, the adjoint methodology has the advantage of computing the cost functional gradients at a fixed expense independent of the number of design variables. Although the adjoint-based methods have been successfully used for problems of optimal design within the steady-state aerodynamics [1–4], applications of the adjoint formulation to time-dependent optimal design problems are still lacking. One of the main reasons why the time-dependent optimization has not been practically used in real-life aerodynamic applications is the storage cost involved. Straightforward global implementations of the discrete unsteady adjoint formulation require that the entire flow solution history should be available during the reverse time integration of the adjoint equations. For realistic 3-D design optimization problems, these storage requirements can quickly become prohibitive. For example, the storage cost of a typical discretization of the 3-D unsteady Reynolds Averaged Navier–Stokes (URANS) equations on a grid with 10^5 points per processor, which are integrated over 1000 time steps, is of the order of $O(10)$ Gb. Note that the storage cost may be significantly higher if a finer grid and more time levels are required to resolve the unsteady flow dynamics, and one stores not only the flow variables and grid coordinates, but also the grid velocities, face normals, control volumes, etc.

Several strategies aimed at circumventing these storage requirements have been developed and reported in the literature. All these methods can be divided into two groups. The first group of methods is "exact" in the sense that the primal and adjoint solutions computed using these methods exactly satisfy the corresponding equations of the original adjoint formulation. The most straightforward exact approach is to store the entire flow solution history to a hard disk (e.g., see [5–7]) and then use it during the reverse time integration of the adjoint equations. Note that for large-scale problems that are nonperiodic in time and require a very large number of time steps to integrate the governing equations, the storage and input/output costs may become prohibitively expensive. Another technique that provides a partial remedy to the storage problem is based on various checkpointing procedures which are performed either statically [8] or dynamically [9]. For this class of methods, the flow variables are stored only at so-called checkpoints whose number is much smaller than the total number of time steps required for integration of the primal and adjoint equations. During the backward-in-time integration of the adjoint equations, the required flow solution on each time subinterval between $(k-1)$th and kth checkpoints is recomputed by using the previously stored flow solution at the $(k-1)$th checkpoint as an initial condition. As a result, the flow solution should be stored only over a small time subinterval $[T_{k-1}, T_k]$ and at all checkpoints, thus significantly reducing the overall storage cost. However, as mentioned in [8,9], the computational cost increases by a factor of 2–3 because of the additional solves of the primal equations.

The key idea of the second group of methods is to reduce the storage cost by constructing sufficiently accurate approximations of either the original optimization problem or the corresponding governing equations. As a result, a solution obtained using these approximate techniques is suboptimal, i.e., not necessarily an extremum of the original time-dependent optimization problem. Among various suboptimal techniques, we would like to mention receding horizon control [10–12], system reduction [13–17], and nonlinear frequency domain methods [18,19]. The receding horizon techniques replace the original time-dependent optimization problem formulated on the entire time interval (the full time horizon) with a sequence of local optimal control problems defined on each time subinterval. Each of the subinterval problems, which are solved sequentially, consists of only a few (possibly one) time steps, so that its storage cost is much lower than that of the original unsteady optimization problem. This approach has been successfully used for optimal control problems governed by the 2-D incompressible Navier–Stokes equations. In [10], the receding horizon method is used for controlling the unsteady flow around a cylinder. Bewley et al. [11] use the receding horizon technique to re-laminarize the turbulent flow in a channel. In [12], Hou and Yan prove that the receding horizon method with distributed controls is stable for problems with a tracking-type functional governed by the 2-D incompressible Navier–Stokes equations. Note that the receding horizon techniques cannot be directly used for solving shape/design optimization problems. These methods compute only the local sensitivity derivative, while the global sensitivity derivative over the entire time interval of interest, which is required for solving the optimal design problems, is not available.

Another suboptimal approach that can significantly reduce the storage and computation costs is based on reduced-order or low-dimensional models of the original high-fidelity approximation of the Euler/Navier–Stokes equations. In [13], Tang et al. use a proper orthogonal decomposition (POD) reduced-order model based on a snapshot basis to control the unsteady wake flow around a cylinder. Hinze and Kunisch [14] present a POD-based boundary control technique that iteratively updates the low-order model and apply it to control the unsteady flow near a cylinder. In [15], two POD-based design optimization methods are used for inverse design of various airfoil shapes. POD modes and their Lagrangian sensitivities with respect to the shape variables are used to derive the POD basis to approximate a class of solutions over a range of design parameter values in [16]. This POD-based methodology is then applied to solving the two-dimensional flow past a square over a range of incidence angles. Modifications to the conventional POD procedure based on nonlinear projection for computing flow solutions are presented and demonstrated on several inverse design problems in [17]. Though POD-based reduced-order models can in principle drastically reduce the overall storage and CPU costs, their accuracy and consequently efficiency strongly depend on how well the POD basis represents the designed set of solutions. This problem associated with a proper selection of snapshots becomes a real challenge for essentially nonlinear compressible flows with shocks and contact discontinuities.

For periodic or quasi-periodic flows, the dimensionality of the corresponding unsteady discrete optimization problem can be reduced by expanding the flow solution in a Fourier series in time, thus reformulating the original optimization problem in the frequency domain. In [18], a gradient method based on the discrete adjoint equations and the corresponding boundary conditions in the frequency domain has been developed. This approach significantly reduces the storage and computation costs of the shape optimization of a 3-D wing oscillating at a constant frequency. An adjoint-based optimization procedure

based on the time-spectral formulation is developed and used for the analysis and shape design of helicopter rotors in forward flight in [19]. Similar to optimization techniques based on the POD reduced-order models, the time-spectral methods are suboptimal. Moreover, these methods are applicable only to time-periodic problems, and their efficiency strongly depends on the number of Fourier modes required to accurately approximate the solution of the unsteady governing equations.

In this paper, we present a new local-in-time discrete adjoint-based optimization methodology that combines the best features of both groups of methods outlined above. Similar to the suboptimal techniques, the new methodology tremendously reduces the overall storage cost by approximating the original adjoint equations on a set of local time subintervals, so that each subinterval involves only a few (possibly one) time steps. The distinctive features of the new local-in-time adjoint-based optimization algorithm are (1) the ability of the new method to converge to a local minimum of the original unsteady optimization problem; and (2) the fact that there is no additional computational overhead as compared with the global-in-time methods. Furthermore, since the global sensitivity derivative is evaluated at each optimization iteration of the new technique, it can be directly used for solving both optimal control and design optimization problems.

The rest of the paper is organized as follows. In Section 2, we present the discrete time-dependent optimization problem. Section 3 presents the conventional global-in-time adjoint-based method. The new local-in-time adjoint-based optimization method is introduced in Section 4. In Section 5, we validate the proposed time-dependent optimization methodology and evaluate its efficiency for three design optimization problems governed by the 2-D compressible Euler equations. We draw conclusions in Section 6.

2. Discrete design optimization problem

We consider a class of time-dependent design optimization problems governed by discretized unsteady flow equations written in the following form:

$$\frac{\mathbf{Q}^n - \mathbf{Q}^{n-1}}{\Delta t} + \mathbf{R}^n = \mathbf{0}, \tag{1}$$

where $\mathbf{Q} = \int_V \mathbf{U} dV$, $\mathbf{U}$ is a vector of the conserved variables, V is a control volume, $\mathbf{R}$ is the spatial undivided (by volume) flux residual, Δt is a time step, and superscript n denotes a time level number. The above discrete formulation (1) is very general and can be directly applied to the unsteady Euler or Reynolds-averaged Navier–Stokes equations [7]. In Eq. (1), the time derivative is approximated by using the implicit first-order backward-difference (BDF-1) formula; 2nd- and 3rd-order BDF formulae can also be used in the present formulation with minor modifications [7]. The governing (1) are discretized on a mesh which is given by the following equation:

$$\mathbf{G}(\mathbf{X}^n, \mathbf{D}) = \mathbf{0}, \tag{2}$$

where $\mathbf{X}^n$ is a mesh at time level n and $\mathbf{D}$ is a vector of the design variables. This time-dependent grid equation can easily adopt static, rigidly moving, and deforming meshes. For static grids considered in this paper, the grid $\mathbf{X}$ in Eq. (2) is independent of time, and the same grid equation is used for all time levels.

The discrete time-dependent optimization problem is formulated as follows:

$$\begin{cases} \min\limits_{\mathbf{D}\in\mathcal{D}_a} F_{\text{obj}}(\mathbf{D}), \quad F_{\text{obj}}(\mathbf{D}) = \sum\limits_{n=1}^{N} f^n(\mathbf{D}, \mathbf{Q}^n, \mathbf{X}^n)\Delta t, \\ \text{subject to Eqs.(1) and (2)}, \end{cases} \tag{3}$$

where $\mathbf{D}$ is a vector of the design variables, $\mathcal{D}_a$ is a set of admissible design parameters, which depends on specifics of the target physical system and ensures the existence of a solution of the optimization problem, N is the total number of time steps, $\mathbf{Q}$ is the solution of the unsteady flow Eq. (1), F_{obj} is an objective functional. The minimization problem (3) is very general and directly applicable to both active flow control and aerodynamic design optimization of unsteady flows.

To reduce the complexity of the optimization problem (3), without loss of generality, it is assumed that the objective functional F_{obj} is a scalar quantity. In the present analysis, f^n in Eq. (3) is defined as follows:

$$f^n = \sum_{j\in\Gamma_c} \left[C_j^n - \left(C_j^{\text{target}} \right)^n \right]^2, \tag{4}$$

where C_j is an aerodynamic quantity such as the lift or the pressure coefficient on a controlled boundary surface Γ_c, C_j^{target} is a given target value of C_j. Thus, F_{obj} given by Eqs. (3) and (4) is a matching-type functional.

3. Global-in-time adjoint-based optimization method

The discrete time-dependent optimization problem (3) is solved by the method of Lagrange multipliers which is used to enforce the governing Eq. (1) as constraints. The discrete Lagrangian functional is defined as follows:

$$L(\mathbf{D}, \mathbf{Q}, \mathbf{X}, \boldsymbol{\Lambda}_f, \boldsymbol{\Lambda}_g) = \sum_{n=0}^{N} f^n \Delta t + \sum_{n=1}^{N} \left[\boldsymbol{\Lambda}_f^n\right]^T \left(\frac{\mathbf{Q}^n - \mathbf{Q}^{n-1}}{\Delta t} + \mathbf{R}^n \right) \Delta t + \left[\boldsymbol{\Lambda}_f^0\right]^T (\mathbf{Q}^0 - \mathbf{Q}^{\text{in}}) + \sum_{n=0}^{N} \left[\boldsymbol{\Lambda}_g^n\right]^T \mathbf{G}^n \Delta t, \tag{5}$$

where $\mathbf{\Lambda}_f$ and $\mathbf{\Lambda}_g$ are flow and grid Lagrange multipliers (adjoint variables), respectively, time levels $n = 0$ and $n = N$ correspond to times $t = 0$ and $t = T_{\text{final}}$, $\mathbf{Q}^{\text{in}}$ is an initial condition for the flow (1), f^n is given by Eq. (4), and $\mathbf{R}^n = \mathbf{R}(\mathbf{Q}^n, \mathbf{X}^n, \mathbf{D})$ is the spatial undivided residual.

The sensitivity derivative is obtained by differentiating the Lagrangian with respect to $\mathbf{D}$, which yields

$$\begin{aligned}\frac{dL}{d\mathbf{D}} = &\sum_{n=0}^{N}\frac{\partial f^n}{\partial \mathbf{D}}\Delta t + \sum_{n=1}^{N}[\mathbf{\Lambda}_f^n]^T\frac{\partial \mathbf{R}^n}{\partial \mathbf{D}}\Delta t + \sum_{n=1}^{N}\left(\frac{[\mathbf{\Lambda}_f^n]^T - [\mathbf{\Lambda}_f^{n+1}]^T}{\Delta t} + [\mathbf{\Lambda}_f^n]^T\frac{\partial \mathbf{R}^n}{\partial \mathbf{Q}^n} + \frac{\partial f^n}{\partial \mathbf{Q}^n}\right)\frac{\partial \mathbf{Q}^n}{\partial \mathbf{D}}\Delta t \\ &+ \left(\frac{\left[\mathbf{\Lambda}_f^0\right]^T - \left[\mathbf{\Lambda}_f^1\right]^T}{\Delta t} + \frac{\partial f^0}{\partial \mathbf{Q}^0}\right)\frac{\partial \mathbf{Q}^0}{\partial \mathbf{D}}\Delta t - \left[\mathbf{\Lambda}_f^0\right]^T\frac{\partial \mathbf{Q}^{\text{in}}}{\partial \mathbf{D}} + \sum_{n=1}^{N}\left(\frac{\partial f^n}{\partial \mathbf{X}^n} + \left[\mathbf{\Lambda}_f^n\right]^T\frac{\partial \mathbf{R}^n}{\partial \mathbf{X}^n} + \left[\mathbf{\Lambda}_g^n\right]^T\frac{\partial \mathbf{G}^n}{\partial \mathbf{X}^n}\right)\frac{\partial \mathbf{X}^n}{\partial \mathbf{D}}\Delta t \\ &+ \sum_{n=0}^{N}[\mathbf{\Lambda}_g^n]^T\frac{\partial \mathbf{G}^n}{\partial \mathbf{D}}\Delta t + \left(\frac{\partial f^0}{\partial \mathbf{X}^0} + \left[\mathbf{\Lambda}_g^0\right]^T\frac{\partial \mathbf{G}^0}{\partial \mathbf{X}^0}\right)\frac{\partial \mathbf{X}^0}{\partial \mathbf{D}}\Delta t, \end{aligned} \tag{6}$$

where $\mathbf{\Lambda}^{N+1} = 0$. In the above equation and throughout the paper, we use the following notations. The derivative of a scalar $c \in R$ with respect to a column vector $\mathbf{a} \in R^m$, $\partial c/\partial \mathbf{a}$, is the row vector: $\left[\frac{\partial c}{\partial a_1}, \ldots, \frac{\partial c}{\partial a_m}\right]$, and the derivative of a column vector $\mathbf{b} \in R^l$ with respect to a column vector $\mathbf{a} \in R^m$ is the $l \times m$ matrix:

$$\frac{\partial \mathbf{b}}{\partial \mathbf{a}} = \begin{bmatrix} \frac{\partial b_1}{\partial a_1} & \cdots & \frac{\partial b_1}{\partial a_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial b_l}{\partial a_1} & \cdots & \frac{\partial b_l}{\partial a_m} \end{bmatrix}.$$

For aerodynamic design optimization problems, the number of design variables is typically very large. Therefore, the computation of $\partial \mathbf{Q}^n/\partial \mathbf{D}$ and $\partial \mathbf{X}^n/\partial \mathbf{D}$ is extremely expensive in terms of the CPU time, because it requires as many solves of the flow and grid equations as the total number of the design variables involved. To eliminate the $\partial \mathbf{Q}^n/\partial \mathbf{D}$ and $\partial \mathbf{X}^n/\partial \mathbf{D}$ terms from the sensitivity derivative, their coefficients on the right-hand side of Eq. (6) are set equal to zero, thus leading to the following adjoint equations for determining the flow adjoint variables:

$$\begin{cases} \frac{1}{\Delta t}\mathbf{\Lambda}_f^N + \left[\frac{\partial \mathbf{R}^N}{\partial \mathbf{Q}^N}\right]^T\mathbf{\Lambda}_f^N = -\left[\frac{\partial f^N}{\partial \mathbf{Q}^N}\right]^T, & \text{for } n = N, \\ \frac{1}{\Delta t}\left(\mathbf{\Lambda}_f^n - \mathbf{\Lambda}_f^{n+1}\right) + \left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{Q}^n}\right]^T\mathbf{\Lambda}_f^n = -\left[\frac{\partial f^n}{\partial \mathbf{Q}^n}\right]^T, & \text{for } 2 \leqslant n \leqslant N-1, \\ \frac{1}{\Delta t}\left(\mathbf{\Lambda}_f^0 - \mathbf{\Lambda}_f^1\right) = -\left[\frac{\partial f^0}{\partial \mathbf{Q}^0}\right]^T, & \text{for } n = 1, \end{cases} \tag{7}$$

and the grid adjoint variables:

$$\begin{cases} \left[\frac{\partial \mathbf{G}^n}{\partial \mathbf{X}^n}\right]^T\mathbf{\Lambda}_g^n = -\left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{X}^n}\right]^T\mathbf{\Lambda}_f^n - \left[\frac{\partial f^n}{\partial \mathbf{X}^n}\right]^T, & \text{for } 1 \leqslant n \leqslant N, \\ \left[\frac{\partial \mathbf{G}^0}{\partial \mathbf{X}^0}\right]^T\mathbf{\Lambda}_g^0 = -\left[\frac{\partial f^0}{\partial \mathbf{X}^0}\right]^T, & \text{for } n = 0. \end{cases} \tag{8}$$

The main advantage of the adjoint formulation is that at each optimization iteration, the adjoint Eqs. (7) and (8) are independent of $\mathbf{D}$ and should be solved once regardless of the number of the design variables. Equations (7) and (8) represent linear systems of equations for the flow and grid adjoint variables, respectively. The flow adjoint equations do not depend on $\mathbf{\Lambda}_g$. Therefore, the systems of Eqs. (7) and (8) are weakly coupled and can be solved sequentially. Once the solution of the flow adjoint equations at the nth time level is available, then $\mathbf{\Lambda}_f^n$ is substituted into Eq. (8) which is solved to determine the grid adjoint variables $\mathbf{\Lambda}_g^n$ at the same time level.

In contrast to the primal flow Eq. (1), the first term in each Eq. (7) approximates the negative time derivative, thus indicating that the unsteady flow adjoint equations have to be integrated backward in time. Therefore, the flow solution $\mathbf{Q}^n$, which is used for computing the matrix $\left[\frac{\partial \mathbf{R}^n}{\partial \mathbf{Q}^n}\right]^T$ and the vector $\left[\frac{\partial f^n}{\partial \mathbf{Q}^n}\right]^T$ in Eq. (7), must be available for all time levels during the backward-in-time integration of the flow adjoint equations. For the global time-dependent adjoint-based method, the entire flow solution history for all time levels is stored during the forward sweep in time. As a result, the storage cost of the time-dependent adjoint formulation is much higher than that of the steady state adjoint formulation.

With the flow and grid adjoint variables found from Eqs. (7) and (8), the sensitivity derivative is calculated as follows:

$$\frac{dL}{d\mathbf{D}} = \sum_{n=0}^{N}\frac{\partial f^n}{\partial \mathbf{D}}\Delta t + \sum_{n=1}^{N}\left[\mathbf{\Lambda}_f^n\right]^T\frac{\partial \mathbf{R}^n}{\partial \mathbf{D}}\Delta t + \sum_{n=0}^{N}\left[\mathbf{\Lambda}_g^n\right]^T\frac{\partial \mathbf{G}^n}{\partial \mathbf{D}}\Delta t - \left[\mathbf{\Lambda}_f^0\right]^T\frac{\partial \mathbf{Q}^{\text{in}}}{\partial \mathbf{D}}. \tag{9}$$

A minimum of the functional given by Eq. (5) is found by the steepest descent method in which each step of the optimization cycle is taken in the negative gradient direction

$$\mathbf{D}_{i+1} = \mathbf{D}_i - \delta_i \left[\frac{dL}{d\mathbf{D}}\right]_i^T, \tag{10}$$

where δ_i is an optimization step size which is chosen adaptively [22], i is a steepest descent iteration number, $\mathbf{D}$ is a vector of the design variables. The sensitivity derivative $dL/d\mathbf{D}$ in Eq. (10) is computed using Eq. (9) which requires the solution of the flow and grid adjoint Eqs. (7) and (8). When the flow and grid adjoint equations are integrated backward in time, the sensitivity derivative at each time step is computed and added to its value at the previous time step. At $n = 0$, the complete sensitivity derivative vector is available and used in Eq. (10) for updating the vector of design variable $\mathbf{D}_{i+1}$. Then, the entire optimization cycle is repeated until either $|F_{\text{obj}}^{i+1} - F_{\text{obj}}^{i}| < \epsilon_1$ or $\|dL/\mathbf{D}_i\| < \epsilon_2$, where ϵ_1 and ϵ_2 are given tolerances and $\|\cdot\|$is an appropriate norm. The above procedure can be summarized in the form of the following global-in-time (GT) adjoint-based algorithm:

Algorithm 1. Global-in-time (GT) adjoint-based method

(1) Choose $\mathbf{D}_1$ and set $i = 1$.
(2) Solve Eq. (1) forward in time for $\mathbf{Q}^0, \ldots, \mathbf{Q}^N$ and store $\mathbf{Q}^n$, $1 \leqslant n \leqslant N$.
(3) Solve Eqs. (7) and (8) backward in time for $\mathbf{\Lambda}_f^n$ and $\mathbf{\Lambda}_g^n$, $1 \leqslant n \leqslant N$.
(4) Evaluate $\frac{dL}{d\mathbf{D}}$ using Eq. (9).
(5) Choose δ_i and update $\mathbf{D}_{i+1}$ using Eq. (10).
(6) If $|F_{\text{obj}}^{i+1} - F_{\text{obj}}^{i}| > \epsilon_1$ and $\|dL/\mathbf{D}_i\| > \epsilon_2$, set $i = i + 1$ and go to step 2; otherwise stop.

This GT algorithm possesses the following property. Namely, if the objective functional is defined to be zero on the entire time interval of interest except the final time level, i.e., $F_{\text{obj}} = f^N \Delta t$, then the corresponding flow and grid adjoint variables exponentially decay to zero in reverse time. This property is a direct consequence of a similarity between the homogeneous flow adjoint Eq. (7) and error equations. Indeed, assuming that $\mathbf{Q}$ is the exact solution of the semi-discrete flow equations $\mathbf{Q}_t + \mathbf{R}(\mathbf{Q}) = 0$ and $\mathbf{e}$ is a solution error caused by a small perturbation of the initial condition, we have

$$\frac{\partial(\mathbf{Q} + \mathbf{e})}{\partial t} + \mathbf{R}(\mathbf{Q} + \mathbf{e}) = 0. \tag{11}$$

Linearizing the above equation with respect to $\mathbf{Q}$ yields

$$\frac{\partial \mathbf{e}}{\partial t} + \frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\mathbf{e} = 0. \tag{12}$$

The homogeneous flow adjoint equations obtained from Eq. (7) by setting $\partial f^n/\partial \mathbf{Q}^n = 0$ for all $n \leqslant N - 1$ are similar to a first-order approximation of the transposed error equation (12). The linear Eqs. (7) and (12) can be integrated in time, thus leading to the following matrix exponential solutions: $\exp(-[\partial \mathbf{R}/\partial \mathbf{Q}]t)\mathbf{e}_0$ and $\exp(-[\partial \mathbf{R}/\partial \mathbf{Q}]^T(T_{\text{final}} - t))\mathbf{\Lambda}_f^N$ for the error and flow adjoint vectors, respectively. For strongly stable numerical schemes, all eigenvalues of the Jacobian matrix $-\partial \mathbf{R}/\partial \mathbf{Q}$ and its transpose $-[\partial \mathbf{R}/\partial \mathbf{Q}]^T$ are located in the left half of the complex plane. Therefore, the numerical error and the flow adjoints exponentially decay in forward and reverse times, respectively. For the flow adjoints, the decay is expected to be strong, because the time derivative of the flow adjoint vector approaches zero at $t \to 0$, as follows from the last equation in Eq. (7) with $\partial f^0/\partial \mathbf{Q}^0 = 0$. From Eq. (8) with $\partial f^n/\partial \mathbf{X}^n = 0$ for $1 \leqslant n \leqslant N - 1$ it follows that the exponential decay of the flow adjoint vector to zero in reverse time results in a similar decay of the grid adjoint vector. Thus, the major contributions of the flow and grid adjoints to the sensitivity derivative come from the final time levels, dominating contributions from intermediate and initial time levels. Numerical results corroborating the above estimates are presented in Section 5.

4. Local-in-time adjoint-based optimization method

As has been mentioned in the foregoing section, at each iteration of the GT method, the flow equations are integrated forward in time while the adjoint equations are integrated backward in time over the entire time interval considered. Since the adjoint operators in Eqs. (7) and (8) depend on $\mathbf{Q}^n$ and $\mathbf{X}^n$, the solution of the flow problem and the corresponding computational grid in the GT algorithm are stored for all time levels over which the optimization problem is solved. For realistic 3-D optimization problems, these storage requirements can quickly become prohibitive. This motivates us to consider local-in-time strategies to reduce the storage cost of the GT method presented in Section 3.

We begin by dividing the entire time interval into K subintervals such that $0 = T_0 < \cdots < T_K = N\Delta t = T_{\text{final}}$, where $T_k = \Delta t\, N_k$, $K \leqslant N$, and Δt is a constant time step used for integrating the primal and adjoint equations. In general, this partitioning can be chosen so that each subinterval contains one or several time steps of the time-marching scheme used for solving the governing equations. The main idea of the proposed strategy is based on the observation that the global sensitivity derivative given by Eq. (9) can be represented as a sum of local sensitivity derivatives defined on each time subinterval. That is

$$\frac{dL}{d\mathbf{D}} = \sum_{k=1}^{K} \frac{dL^k}{d\mathbf{D}}, \tag{13}$$

where the local Lagrangian functionals are given by

$$L^k = \begin{cases} \sum\limits_{n=N_{k-1}+1}^{N_k} f^n \Delta t + \sum\limits_{n=N_{k-1}+1}^{N_k} \left[\Lambda_f^n\right]^T \left(\frac{\mathbf{Q}^n - \mathbf{Q}^{n-1}}{\Delta t} + \mathbf{R}^n\right)\Delta t, & \text{for } 2 \leqslant k \leqslant K, \\ \sum\limits_{n=0}^{N_1} f^n \Delta t + \sum\limits_{n=1}^{N_1} \left[\Lambda_f^n\right]^T \left(\frac{\mathbf{Q}^n - \mathbf{Q}^{n-1}}{\Delta t} + \mathbf{R}^n\right)\Delta t + \left[\Lambda_f^0\right]^T \left(\mathbf{Q}^0 - \mathbf{Q}^{\text{in}}\right), & \text{for } k = 1. \end{cases} \tag{14}$$

In this section, without loss of generality, the grid terms are omitted for simplicity.

The adjoint equations corresponding to the local Lagrangian functionals, L^k for $1 \leqslant k \leqslant K$, can be derived by using the same adjoint-based approach described in the foregoing section. Differentiating each local Lagrangian, L^k, with respect to $\mathbf{D}$ and taking into account the contribution from L^{k+1} yields the following flow adjoint equations on subinterval $(T_{k-1}, T_k]$:

$$\begin{cases} \frac{1}{\Delta t}\left(\Lambda_f^{N_k} - \Lambda_f^{N_k+1}\right) + \left[\frac{\partial \mathbf{R}^{N_k}}{\partial \mathbf{Q}^{N_k}}\right]^T \Lambda_f^{N_k} = -\left[\frac{\partial f^{N_k}}{\partial \mathbf{Q}^{N_k}}\right]^T, & \text{for } n = N_k, \\ \frac{1}{\Delta t}\left(\Lambda_f^{n} - \Lambda_f^{n+1}\right) + \left[\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{Q}^{n}}\right]^T \Lambda_f^{n} = -\left[\frac{\partial f^{n}}{\partial \mathbf{Q}^{n}}\right]^T, & \text{for } N_{k-1}+1 \leqslant n \leqslant N_k - 1, \\ \frac{1}{\Delta t}\left(\Lambda_f^{0} - \Lambda_f^{1}\right) = -\left[\frac{\partial f^{0}}{\partial \mathbf{Q}^{0}}\right]^T, & \text{for } n = 0, \end{cases} \tag{15}$$

where Λ_f^n is the solution of the flow adjoint equations defined for $N_{k-1} < n \leqslant N_k$. The presence of the $\Lambda_f^{N_k+1}$ term in Eq. (15) indicates that the system of adjoint equations on subinterval $(T_{k-1}, T_k]$ is coupled with the system of adjoint equations defined on the next subinterval $(T_k, T_{k+1}]$. In fact, Eq. (15) for $1 \leqslant k \leqslant K$ represents a set of coupled systems of adjoint equations on the entire time interval $[0, T_{\text{final}}]$, which is equivalent to the original adjoint Eq. (5). As a result, the flow solution for all time levels has to be available when these adjoint Eq. (15) for $1 \leqslant k \leqslant K$ are integrated backward in time.

To reduce the storage cost, we decouple the set of (15) for $1 \leqslant k \leqslant K$ by approximating $\Lambda_f^{N_k+1}$ as $\widetilde{\Lambda}_f$, thus leading to the following local-in-time adjoint equations defined on $(T_{k-1}, T_k]$:

$$\begin{cases} \frac{1}{\Delta t}\left(\widehat{\Lambda}_f^{N_k} - \widetilde{\Lambda}_f\right) + \left[\frac{\partial \mathbf{R}^{N_k}}{\partial \mathbf{Q}^{N_k}}\right]^T \widehat{\Lambda}_f^{N_k} = -\left[\frac{\partial f^{N_k}}{\partial \mathbf{Q}^{N_k}}\right]^T, & \text{for } n = N_k, \\ \frac{1}{\Delta t}\left(\widehat{\Lambda}_f^{n} - \widehat{\Lambda}_f^{n+1}\right) + \left[\frac{\partial \mathbf{R}^{n}}{\partial \mathbf{Q}^{n}}\right]^T \widehat{\Lambda}_f^{n} = -\left[\frac{\partial f^{n}}{\partial \mathbf{Q}^{n}}\right]^T, & \text{for } N_{k-1}+1 \leqslant n \leqslant N_k - 1, \\ \frac{1}{\Delta t}\left(\widehat{\Lambda}_f^{0} - \widehat{\Lambda}_f^{1}\right) = -\left[\frac{\partial f^{0}}{\partial \mathbf{Q}^{0}}\right]^T, & \text{for } n = 0, \end{cases} \tag{16}$$

where $\widehat{\Lambda}_f^n$ is an approximation of the corresponding adjoint solution Λ_f^n. The last equation in Eq. (16) is used only on the first subinterval $[T_0, T_1]$ corresponding to $k = 1$.

It should be noted that the partitioning of the entire time interval into subintervals does not alter the solution of the flow equations. Indeed, the flow equations are integrated forward in time beginning from $n = 0$ that corresponds to the initial condition of the original flow problem. The flow solution obtained at the end of the first time subinterval, $\mathbf{Q}^{N_1}$, is used as an initial condition for the second subinterval, and so on. In the case that a second- or higher order backward difference (BDF) scheme is employed for discretization of the time derivative, flow solutions at the corresponding number (depending on the BDF scheme used) of time levels of the previous subinterval are employed to continue the integration of the governing equations on the current time subinterval. The result is that the flow solution obtained in this manner is identical to that computed on the entire time interval $[0, T_{\text{final}}]$ by using a single sweep in time.

With the local flow adjoint variables $\widehat{\Lambda}_f^n$ satisfying Eq. (16), the local sensitivity derivative on each subinterval $(T_{k-1}, T_k]$ is calculated as follows:

$$\frac{d\widehat{L}^k}{d\widehat{\mathbf{D}}} = \begin{cases} \sum\limits_{n=N_{k-1}+1}^{N_k} \frac{\partial f^n}{\partial \widehat{\mathbf{D}}} \Delta t + \sum\limits_{n=N_{k-1}+1}^{N_k} \left[\widehat{\Lambda}_f^n\right]^T \frac{\partial \mathbf{R}^n}{\partial \widehat{\mathbf{D}}} \Delta t, & \text{for } 2 \leqslant k \leqslant K, \\ \sum\limits_{n=0}^{N_1} \frac{\partial f^n}{\partial \widehat{\mathbf{D}}} \Delta t + \sum\limits_{n=1}^{N_1} \left[\widehat{\Lambda}_f^n\right]^T \frac{\partial \mathbf{R}^n}{\partial \widehat{\mathbf{D}}} \Delta t - \left[\widehat{\Lambda}_f^0\right]^T \frac{\partial \mathbf{Q}^{\text{in}}}{\partial \widehat{\mathbf{D}}}, & \text{for } k = 1. \end{cases} \tag{17}$$

By analogy with Eq. (13), the approximate global sensitivity derivative, $\frac{d\widehat{L}}{d\widehat{\mathbf{D}}}$, is computed as

$$\frac{d\widehat{L}}{d\widehat{\mathbf{D}}} = \sum_{k=1}^{K} \frac{d\widehat{L}^k}{d\widehat{\mathbf{D}}}. \tag{18}$$

Once the global sensitivity derivative is available at the last Kth time subinterval, the vector of design variables is updated by using the steepest descent method

$$\widehat{\mathbf{D}}_{i+1} = \widehat{\mathbf{D}}_i - \delta_i \left[\frac{d\widehat{L}}{d\widehat{\mathbf{D}}}\right]_i^T. \tag{19}$$

Similar to the GT method, the steepest descent iterations are repeated until either $\left|F_{\text{obj}}^{i+1} - F_{\text{obj}}^{i}\right| < \epsilon_1$ or $\|d\widehat{L}/\widehat{\mathbf{D}}_i\| < \epsilon_2$, where ϵ_1 and ϵ_2 are user-specified tolerances and $\|\cdot\|$ is an appropriate norm.

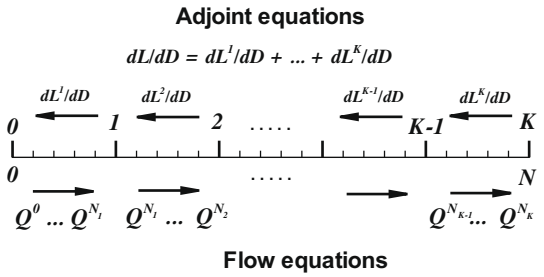

Fig. 1. A sketch of the local-in-time adjoint-based algorithm.

Comparing Eqs. (15) and (16), the following observation can be made. If $\widetilde{\mathbf{\Lambda}}_f$ in Eq. (16) is set equal to zero, then each local system of adjoint Eq. (16) defined on a given time subinterval $(T_{k-1}, T_k]$ is independent of the other adjoint equations defined on $[0, T_{k-1}] \cup (T_k, T_{\text{final}}]$. Thus, the local systems of adjoint Eq. (16) can be solved sequentially starting from the first time subinterval ($k = 1$) and marching *forward* one subinterval by another up to $k = K$. Within each subinterval $(T_{k-1}, T_k]$, the local adjoint equations (16) are integrated backward in time. Although the systems of local adjoint equations (16) defined on k and $k + 1$ time subintervals are decoupled if $\widetilde{\mathbf{\Lambda}}_f = 0$, they cannot be solved simultaneously because each system of adjoint equations requires the flow solutions to be available on the same time subinterval. The local sensitivity derivatives calculated on each subinterval using Eq. (17) are then summed up to give the global sensitivity derivative on the entire time interval $[0, T_{\text{final}}]$, as shown in Fig. 1. Note that the flow adjoints obtained with the local Eq. (16) for $1 \leqslant k \leqslant K$ and the corresponding approximate total sensitivity derivative given by Eq. (18) are, in principle, not equal to those given by Eqs. (7) and (9), i.e., $\widehat{\mathbf{\Lambda}}_f \neq \mathbf{\Lambda}_f$ and $d\widehat{L}/d\widehat{\mathbf{D}} \neq dL/d\mathbf{D}$ on $(T_{k-1}, T_k]$, where $\mathbf{\Lambda}_f$ denotes the solution of the global flow adjoint Eq. (7). Though $d\widehat{L}/d\widehat{\mathbf{D}}$ is only an approximation to $dL/d\mathbf{D}$ given by Eq. (9), this approach reduces the storage cost by a factor of K as compared with the GT algorithm. Indeed, since the local adjoint equations on each time subinterval $(T_k, T_{k+1}]$ can be solved independently of the adjoint equations defined on the other subintervals, only the flow solutions for the current subinterval, $\mathbf{Q}^{N_{k-1}+1}, \ldots, \mathbf{Q}^{N_k}$, have to be stored, thus drastically reducing the storage cost. Further in the paper, this algorithm with $\widetilde{\mathbf{\Lambda}}_f = \mathbf{0}$ is referred as a simplified local-in-time (SLT) method.

Another observation based on the comparative analysis of Eqs. (7), (9) and (16), (17) is that the entire set of systems of local adjoint Eq. (16) for $1 \leqslant k \leqslant K$ is identical to the global adjoint Eq. (15) and consequently to Eq. (7), if $\widetilde{\mathbf{\Lambda}}_f$ in Eq. (16) is set to be $\mathbf{\Lambda}_f^{N_k+1}$. In spite of the fact that this approach provides complete consistency of the local and global adjoint equations, it destroys the locality of the adjoint Eq. (16) and therefore requires the same full storage as the GT method.

These considerations suggest that $\widetilde{\mathbf{\Lambda}}_f$ in Eq. (16) should be chosen such that it preserves the locality of each system of adjoint equations defined on subinterval $(T_{k-1}, T_k]$ and provides a good approximation of $\mathbf{\Lambda}_f^{N_k+1}$. To satisfy these constraints, we propose to choose $\widetilde{\mathbf{\Lambda}}_f$ as

$$(\widetilde{\mathbf{\Lambda}}_f)_i = \left(\mathbf{\Lambda}_f^{N_k+1}\right)_{i-1}, \tag{20}$$

where i is a design iteration number. In other words, the required vector of adjoint variables at time level $N_k + 1$ is taken from the previous iteration of the steepest descent method (19). This local-in-time (LT) adjoint-based strategy for solving the minimization problem (3) and (4) is summarized in the form of the following algorithm:

Algorithm 2. Local-in-time (LT) adjoint-based method

(1) Choose $\widehat{\mathbf{D}}_1$, and K; set $k = 1$, $i = 1$, $\left(\widehat{\mathbf{\Lambda}}_f^{N_k+1}\right)_0 = 0$ for $1 \leqslant k \leqslant K$, and $\frac{\widehat{dL}}{d\mathbf{D}} = 0$.
(2) Solve Eq. (1) for $\mathbf{Q}^{N_{k-1}+1}, \ldots, \mathbf{Q}^{N_k}$ forward in time on $(T_{k-1}, T_k]$; store $\mathbf{Q}^n$ for $N_{k-1} + 1 \leqslant n \leqslant N_k$.
(3) If $i \leqslant i_s$ set $\widetilde{\mathbf{\Lambda}}_f = \mathbf{0}$, otherwise $\widetilde{\mathbf{\Lambda}}_f = \left(\widehat{\mathbf{\Lambda}}_f^{N_k+1}\right)_{i-1}$, where i_s is a user-defined number of iterations.
(4) Solve Eq. (16) backward in time for $\widehat{\mathbf{\Lambda}}_f^{N_{k-1}+1}, \ldots, \widehat{\mathbf{\Lambda}}_f^{N_k}$; store $\widehat{\mathbf{\Lambda}}_f^{N_{k-1}+1}$.
(5) Evaluate $\frac{\widehat{dL^k}}{d\mathbf{D}}$ by using Eq. (17).
(6) Set $\frac{\widehat{dL}}{d\mathbf{D}} = \frac{\widehat{dL}}{d\mathbf{D}} + \frac{\widehat{dL^k}}{d\mathbf{D}}$.
(7) Set $k = k + 1$, if $k \leqslant K$ go to step 2; otherwise continue.
(8) Calculate $\widehat{\mathbf{D}}_{i+1}$ using Eq. (19).
(9) If $|F_{\text{obj}}^{i+1} - F_{\text{obj}}^{i}| > \epsilon_1$ and $\|d\widehat{L}/d\widehat{\mathbf{D}}_i\| > \epsilon_2$ set $k = 1$, $i = i + 1$, $\frac{\widehat{dL}}{d\mathbf{D}} = 0$ and go to step 2; otherwise stop.

The above choice of $\widetilde{\mathbf{\Lambda}}_f$ given by Eq. (20) significantly reduces the storage cost as compared to the GT method. Indeed, for the LT method, the flow solution should be stored only at those time levels that belong to the current time subinterval

$(T_{k-1},T_k]$. In addition, flow adjoint solutions at $K-1$ time levels from the previous optimization cycle, $\left(\widehat{\mathbf{\Lambda}}_f^{N_k+1}\right)_{i-1}$ for $1 \leqslant k \leqslant K-1$, should also be stored, as follows from Eq. (20). Therefore, the overall storage cost of the LT algorithm is $O(K+N/K)$ flow variables versus $O(N)$ flow variables required for the GT method. Since $K+N/K$ achieves its minimum value at $K=\sqrt{N}$, the storage cost of the LT algorithm can be minimized if the number of time subintervals K is set equal to $\sqrt{N}$, where N is the total number of time levels. For $K=\sqrt{N}$, the total storage cost of the LT algorithm is $\sqrt{N}/2$ times less than that of the GT algorithm. The savings are even more significant when dynamic grids are involved.

In addition to the significant storage savings, another key advantage of the LT algorithm is that upon convergence, the set of local-in-time adjoint equations becomes identical to the original adjoint Eq. (7), thus providing full consistency between the local and global methods. In other words, the converged solution obtained with LT method is a local minimum of the original optimization problem (3). Indeed, assuming that for all time levels n the LT method convergences to the machine zero after i^* iterations, one can immediately conclude that $\left(\widehat{\mathbf{\Lambda}}_f^n\right)_{i_*-1}=\left(\widehat{\mathbf{\Lambda}}_f^n\right)_{i_*}=\widehat{\mathbf{\Lambda}}_f^n$, thus leading to $\widetilde{\mathbf{\Lambda}}_f=\left(\widehat{\mathbf{\Lambda}}_f^{N_{k+1}}\right)_{i_*-1}=\left(\widehat{\mathbf{\Lambda}}_f^{N_{k+1}}\right)_{i_*}=\widehat{\mathbf{\Lambda}}_f^{N_{k+1}}$ for $n=N_{k+1}$. Since the term $\widetilde{\mathbf{\Lambda}}_f$ in Eq. (16) converges to $\widehat{\mathbf{\Lambda}}_f^{N_{k+1}}$ for $1 \leqslant k \leqslant K-1$, the result is that the set of the local adjoint equations defined on each time subinterval converges to the original system of adjoint Eq. (7), provided that the adjoint operators in both systems are the same. Note that if initial values of the design variables for the GT algorithm are set equal to the converged optimal values obtained with the LT method, then the adjoint operators in Eq. (7) are identical to those in Eq. (16). Therefore, the local and global adjoint equations at the extremum point are identical to each other, and one can immediately conclude that the solution of the local-in-time adjoint Eq. (16) is equal to the solution of the global adjoint Eq. (7), thus leading to the equivalence of the corresponding sensitivity derivatives. Taking into account the fact that at the extremum obtained with the LT method, $d\widehat{L}/d\widehat{\mathbf{D}}$ vanishes, the true sensitivity derivative, $dL/d\mathbf{D}$, evaluated at the same point in the design space by using the GT algorithm is equal to $d\widehat{L}/d\widehat{\mathbf{D}}$ and therefore vanishes as well. It implies that the solution obtained with the LT method is optimal with respect to the original optimization problem Eq. (3). Note that in principle, the GT and LT algorithms may converge to different local extrema of the optimization problem Eq. (3). What is important, however, that the solutions computed with both the GT and LT algorithms are local minima of the original optimization problem. It should also be noted that for all test problems presented in the next section, the LT method converges to the same solution obtained with the GT counterpart.

Another attractive feature of the new LT algorithm is that it has the same complexity per optimization cycle as the GT method. Indeed, for the LT algorithm, the flow equations and the corresponding adjoint equations on $(T_{k-1},T_k]$, $1 \leqslant k \leqslant K$ at each optimization iteration are solved only once. Since there is no overlap between time subintervals, the total number of time steps, over which the LT equations are integrated, is equal to that used for integration of the original adjoint Eq. (7) in the GT algorithm.

The LT algorithm can be directly used for solving both time-dependent optimal control problems whose control variables depend on time and design optimization problems whose design variables are independent of time. This is one of the main advantages of the LT method over the receding horizon technique and its variants (e.g., see [10–12,14]) which are applicable only to optimal control problems, but cannot be directly used for design optimization. Another principle difference between these techniques is that the solution computed with the LT method is a local minimum of the optimization problem (3), while the corresponding solution obtained with any receding horizon technique is only suboptimal with respect to the original minimization problem.

5. Numerical results

We consider design optimization problems governed by the 2-D unsteady Euler equations for supersonic flows in a channel with a bump to evaluate the performance of the new local-in-time method. For all test problems considered, the final time, T_{final}, is set to be 1, and the freestream Mach number is given by

$$M(t)=2+0.1\cos(17\pi t/9). \tag{21}$$

Since the freestream Mach number oscillates in time, the entire flowfield is unsteady. The aerodynamic coefficient in Eq. (4) is chosen to be the time-dependent pressure coefficient at the lower boundary of the computational domain. The bump shape is described by the following equation:

$$y=d_1\psi_1(x)+d_2\psi_2(x)+d_3\psi_3(x),$$

where $\psi_l(x)$, $1 \leqslant l \leqslant 3$ are given polynomials satisfying the requirement that the leading and trailing edges of the bump continuously meet the straight lower wall on either side of the bump. Three coefficients d_1, d_2, and d_3 are design variables, i.e., $\mathbf{D}=[d_1,d_2,d_3]^T$.

The governing equations are discretized by using a first-order, node-centered, finite-volume scheme [20] on structured quadrilateral grids. The inviscid fluxes at cell interfaces are computed using the upwind scheme of Roe [21]. At each time step, the nonlinear discrete flow equations are solved by Newton's method. For each test, the residuals of the 2-D Euler equations and the corresponding adjoint equations are driven below 10^{-12}. The governing equations are integrated over 9 time steps with the nondimensional time step equal to 1/9. Along with the LT method, the SLT version of this algorithm with

$\widetilde{\mathbf{\Lambda}}_f = 0$ in Eq. (16) is also considered in the present analysis. For all test problems considered, the number of time subintervals used in the LT and SLT algorithms is set equal to 3 and 9, respectively, and the parameter i_s in the LT method is set to 3. For the SLT algorithm, only the local unsteady flow solution on a current time subinterval is held in the operating memory. For the LT method, in addition to the flow solution on the current subinterval, the adjoint variables $\widetilde{\mathbf{\Lambda}}_f^{N_k+1}$ for $1 \leqslant k \leqslant K-1$ are also held in the operating memory, while for the GT method, the entire flow solution history for all time levels is stored. The derivatives of $\mathbf{R}$ and f with respect to $\mathbf{Q}$ and $\mathbf{D}$, which are required to form the adjoint equations and the sensitivity derivative, are calculated by using the complex variable technique developed by Lyness and Moler [23].

First, we validate the implementation of the GT method and compare sensitivity derivatives obtained with the GT algorithm and a forward mode differentiation based on the complex variables approach [23]. The key advantage of the complex variables technique is that for sufficiently small values of the complex step size, this method provides the sensitivity derivative with the machine accuracy, which can be used for validation of the adjoint formulation. For the forward mode differentiation, the complex step size is chosen to be 10^{-10}. Note that at each optimization cycle, the forward mode differentiation technique solves the flow problem as many times as the total number of design variables, while the adjoint-based method requires one solve of the Euler and corresponding adjoint equations per optimization cycle, regardless of the number of the design variables. Table 1 shows the sensitivity derivatives computed with the forward mode differentiation and adjoint methods. As expected, the discrepancy is of the order of round-off error, thus validating the implementation and accuracy of the GT method.

Next, we evaluate the performance of the GT, LT, and SLT methods for the time-dependent design optimization problem (3) and (4) when the target flow is feasible. The feasibility of the target flow implies that there exists a set of design variables in the design space, that recovers the target flow precisely. Note that the value of the objective functional at the extremum is zero, and the optimal design variables are expected to be equal to their exact target values. This problem is well suited for evaluation of the performance of optimization methods, because the exact solution is known and the objective functional vanishes at the extremum. The target pressure coefficient is obtained by solving the unsteady 2-D Euler equations with the design variables chosen to be $d_1 = 0.05$, $d_2 = 0.03$, and $d_3 = 0.01$. The initial value of each design variable is set to be zero, thus initially, there is no bump on the lower wall. The optimization is stopped when the absolute value of the objective functional becomes smaller than 10^{-5}.

Convergence histories of the objective functional obtained with all three algorithms are presented in Fig. 2. Overall, the GT, LT, and SLT methods demonstrate very similar convergence rates. For each method, the value of the objective functional rapidly decreases over the first five iterations, dropping down by almost two orders of magnitude. Then, the convergence rate slows down, and the objective functional gradually decreases until it becomes less than the specified tolerance. Fig. 3 shows convergence histories of all three design variables during the optimization. The most important conclusion that can be drawn from this comparison is that the GT, LT, and SLT methods converge to the same solution. From this standpoint, the solutions obtained with LT and SLT algorithms are optimal with respect to the original optimization problem (3). It should also be noted that all the design variables converge to their target values. From the comparisons presented above it follows that the LT and SLT methods converge to the same optimal solution computed with the GT method, while reducing the storage cost by a factor of 1.5 and 4, respectively. For a larger number of time steps, the storage savings may be considerably higher. As has been pointed out in the foregoing section, the storage cost of the SLT algorithm is independent of the number of time steps and equal to 3 units, where one unit corresponds to memory that is required to store one flow solution vector at each grid point. Note that the SLT method requires the same storage as the steady state adjoint formulation. The storage cost of the GT method is $N + 3$ units and directly proportional to the total number of time intervals, N, while the storage cost of the LT method is $K + N/K + 2$ units, where K is the total number of time subintervals used.

We now evaluate the performance of the LT and SLT methods for minimization of the objective functional defined on a time interval that is smaller than $[0, T_{\text{final}}]$. For this test problem, it is assumed that the objective functional involves only the solution at the terminal time T_{final}, i.e.

$$F_{\text{obj}} = \sum_{j \in \Gamma_c} \left[C_j^N - \left(C_j^{\text{target}} \right)^N \right]^2 \Delta t. \tag{22}$$

The target pressure distribution in Eq. (22) is chosen in the same manner as in the previous test problem. Therefore, the target flow is feasible, and the optimization problem has at least one global minimum. Clearly, this problem is more challenging for the SLT method. Indeed, the SLT method takes into account only the contribution of the last time interval to the sensitivity derivative, while for the GT and LT methods, the adjoint variables at each time level are nonzero; thus, each time subinterval makes a nonzero contribution to the global sensitivity derivative. Fig. 4 shows convergence histories obtained with

Table 1
Sensitivity derivatives computed with the adjoint formulation and the forward mode differentiation based on the complex variable technique.

	$\frac{dL}{dD_1}$	$\frac{dL}{dD_2}$	$\frac{dL}{dD_3}$
Adjoint formulation	−10.5059070229186	−12.2910025055155	−12.8094954127715
Complex variables	−10.5059070229196	−12.2910025055174	−12.8094954127741

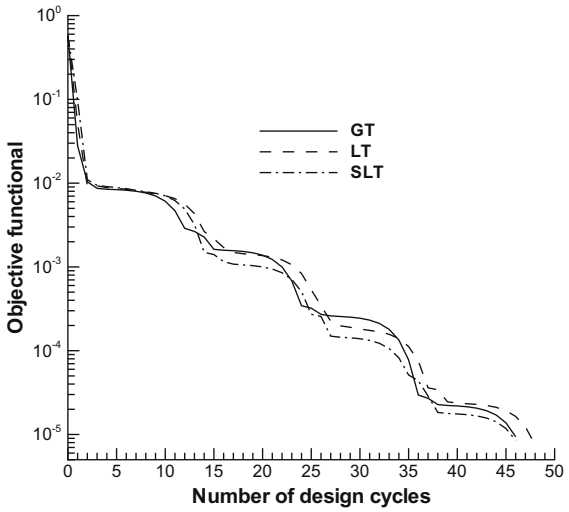

Fig. 2. Convergence of the objective functional obtained with the GT, LT, and SLT methods for the first design optimization problem.

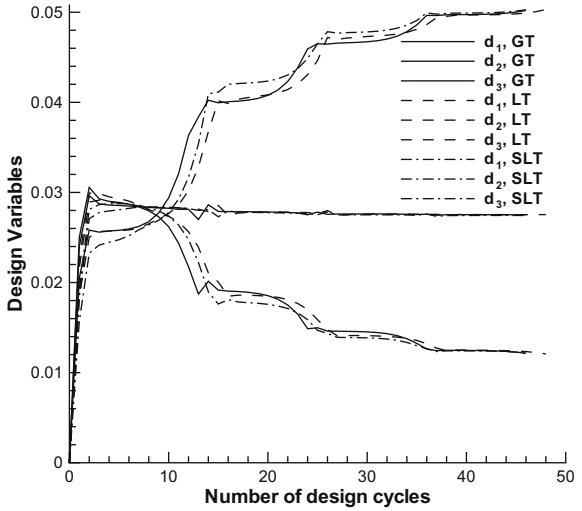

Fig. 3. Convergence of the design variables obtained with the GT, LT, and SLT methods for the first design optimization problem.

the global and both local algorithms for the minimization problem with the objective functional defined by Eq. (22). As follows from Fig. 5, all three methods converge to the global extremum of the minimization problem, demonstrating similar convergence rates. It takes 42 design cycles to reduce the objective functional by four orders of magnitude by using the LT method, while the SLT and GT algorithms require 36 and 37 iterations, respectively.

Despite that for the SLT method, contributions from all time levels except the last one are neglected, its solution and convergence rate are very close to those obtained with GT and LT algorithms. This is not surprising, because as has been shown at the end of Section 3 for this test problem, each component of the sensitivity derivative vector and the flow adjoint variables decay to zero in reverse time. Figs. 6 and 7 demonstrate this property of the sensitivity derivatives and adjoint variables computed with the GT algorithm for the objective functional given by Eq. (22). The result is that the contribution from the last time interval is dominant, which explains why the SLT method provides a good approximation of the total sensitivity derivative. Fig. 7 also shows that the adjoint variables computed with the GT and LT algorithms agree very well over the entire time interval considered, which corroborates our analysis presented in Section 4. Note that for the SLT method, the adjoint equations should be solved only at the final time level, thus reducing the computational cost as compared with the GT and LT algorithms.

For the third test problem, the target bump shape is set to $y = \sin^4(\pi(x-1))$, which is outside of the design space. As a result, the target flow is infeasible, and the value of the objective functional at the optimum is not equal to zero. Fig. 8 shows

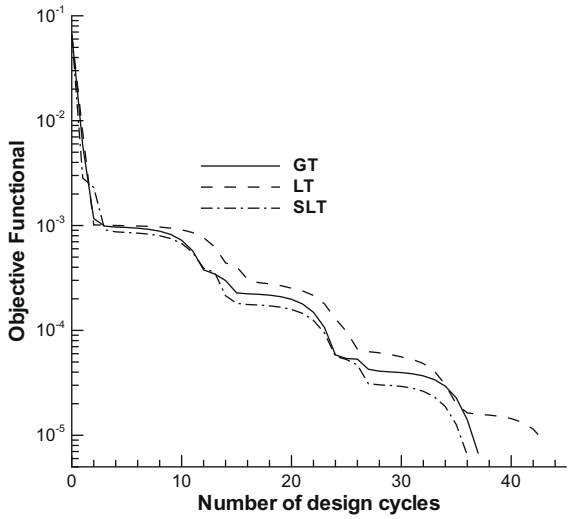

Fig. 4. Convergence histories of the objective functional computed with the GT, LT, and SLT adjoint-based methods for the second test problem.

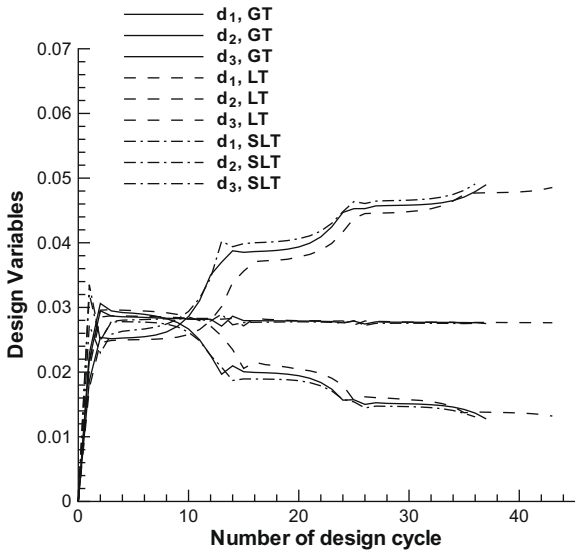

Fig. 5. Convergence of the design variables obtained with the GT, LT, and SLT methods for the second test problem.

convergence histories of the objective functional obtained with the GT, LT, and SLT algorithms. Overall, each optimization method reduces the value of the objective functional more than an order of magnitude.

During the first 15 design cycles, the LT method provides the fastest reduction in the objective functional among all three methods. By 25th design cycle, all the methods provide similar values of the objective functional and show practically the same convergence behavior thereafter. Convergence histories of all three design variables are depicted in Fig. 9. Despite the fact that each design variable changes dramatically during the design, both the SLT and LT methods demonstrate the convergence behavior that is very similar to that of the GT algorithm. As in the previous test cases, the GT, LT, and SLT algorithms converge to the same solution, which again indicates that this solution is optimal with respect to the original minimization problem. The comparison of the computed, target and initial lift coefficients are shown in Fig. 10. The relative difference between the initial lift coefficient and its target value is of the order of $O(1)$. In spite of the fact that the target flow is infeasible, the lift coefficients computed with all three optimization techniques agree reasonably well with the target lift coefficient over the entire time interval considered. Furthermore, the lift coefficients obtained with the GT, LT, and SLT algorithms are almost indistinguishable from each other, which indicates that all three methods converge to the same solution.

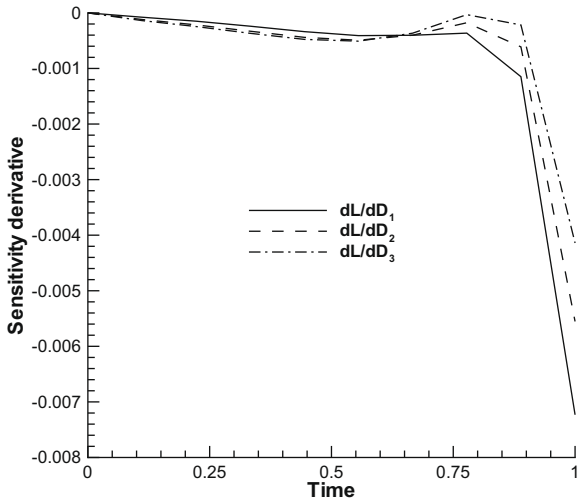

Fig. 6. Components of the sensitivity derivative vector obtained with the GT method.

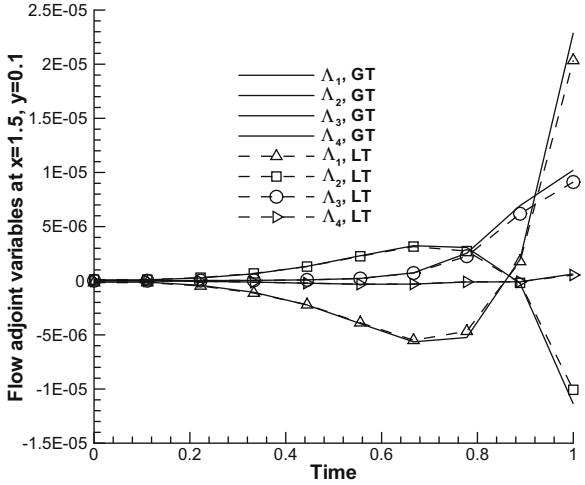

Fig. 7. The flow adjoint variables on the bump surface at $x = 1.5$, obtained with the GT, LT, SLT methods for the second test problem.

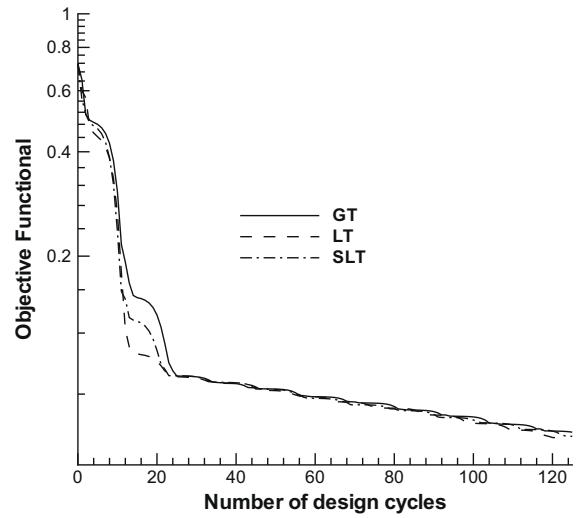

Fig. 8. Convergence histories of the objective functional computed with the GT, LT, and SLT methods for the third test problem (infeasible target flow).

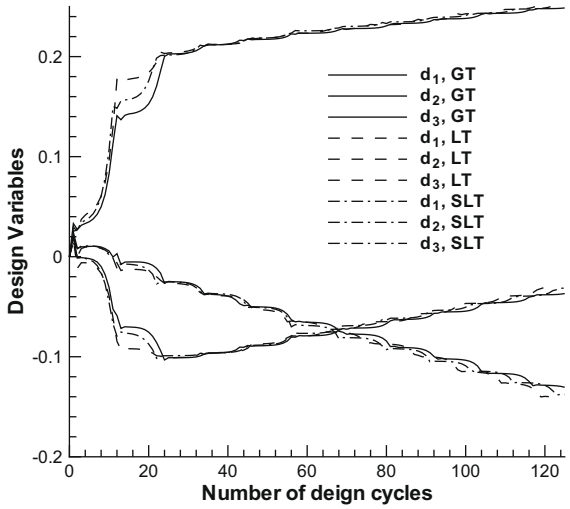

Fig. 9. Convergence of the design variables obtained with the GT, LT, and SLT methods for the third test problem (infeasible target flow).

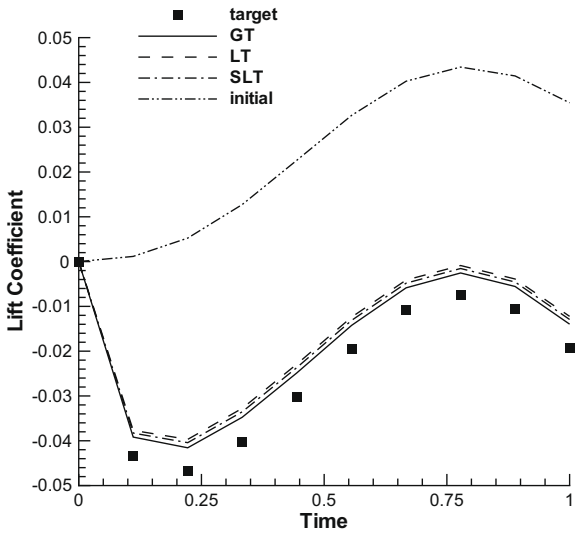

Fig. 10. Comparison of lift coefficients computed using the GT, LT, and SLT methods with their initial and target values for the third test problem (infeasible target flow).

6. Conclusions

The new local-in-time adjoint-based method for design optimization of unsteady flows has been developed. In contrast to the global-in-time (GT) algorithm that stores the flow solution for all time levels, the new algorithm sequentially solves the local adjoint equations on each time subinterval to form the global sensitivity derivative. Two different implementations of the local-in-time method have been considered. The first, simplified (SLT) implementation neglects the coupling between neighboring time subintervals. Since each set of local adjoint equations is integrated backward in time over only a small time subinterval, the storage cost of the SLT method is of the order of $O(N/K)$ flow variables, where N is the total number of time intervals and K is the number of time subintervals. In the limit, each time subinterval can consist of a single time step, thus, the storage cost can be reduced to the level of the steady state adjoint formulation. For the second, more general implementation of the local-in-time (LT) method, the term that couples the local sets of adjoint equations defined on neighboring time subintervals is retained and taken from the previous optimization iteration. The storage cost of the LT method is $O(N/K + K)$ versus $O(N)$ flow variables required for the GT method. For the LT method, the optimal number of time subintervals is $\sqrt{N}$, thus leading to the storage cost that is $\sqrt{N}/2$ times less than that of the conventional counterpart. The most distinctive

feature of the LT algorithm is that its solution is a local minimum of the original optimization problem, which is not necessarily the case for the SLT method. Furthermore, for the LT method, the number of operations per optimization cycle is equal to that of the GT algorithm, thus leading to the same CPU cost. For all test problems considered, the GT, LT, and SLT methods provide practically the same convergence rate and converge to the same local minimum of the original time-dependent optimization problem. These properties of the LT method open new avenues for solving a broad spectrum of realistic large-scale design optimization problems arising in various unsteady aerodynamic applications.

Acknowledgments

The work of the first and second authors has been supported by NASA under Contract NNL07AA23C. The first author would also like to acknowledge the partial support from the Army Research Laboratory through Grant W911NF-06-R-006.

References

[1] A. Jameson, N. Pierce, L. Martinelli, Optimum aerodynamic design using the Navier–Stokes equations, Theor. Comput. Fluid Dyn. 10 (1) (1998) 213–237.
[2] E.J. Nielsen, M.A. Park, Using an adjoint approach to eliminate mesh sensitivities in computational design, AIAA J. 44 (5) (2006) 948–953.
[3] E.J. Nielsen, W.K. Anderson, Recent improvements in aerodynamic optimization on unstructured meshes, AIAA J. 40 (6) (2002) 1155–1163.
[4] R. Soto, C, Yang, An adjoint-based methodology for CFD optimization problems, AIAA Paper 2003-0299, 2003.
[5] S.K. Nadarajah, A. Jameson, Optimal control of unsteady flows using a time accurate method, AIAA Paper 2002-5436, 2002.
[6] K. Mani, D.J. Mavriplis, An unsteady discrete adjoint formulation for two-dimensional flow problems with deforming meshes, AIAA Paper 2007-60, 2007.
[7] E.J. Nielsen, B. Diskin, N.K. Yamaleev, Discrete adjoint-based design optimization of unsteady turbulent flows on dynamic unstructured grids, AIAA Paper 2009-3802, 2009.
[8] M. Hinze, A. Walther An optimal memory-reduced procedure for calculating adjoints of the instationary Navier–Stokes equations, in MATH-NM-06-2002, 2002.
[9] M. Hinze, J. Strenberg, A-Revolve: an adaptive memory-reduced procedure for calculating adjoints; with an application to computing adjoints of the instationary Navier–Stokes system, Optim Method Softw 20 (6) (2005) 645–663.
[10] C. Min, H. Choi, Suboptimal feedback control of vortex shedding at low Reynolds numbers, J. Fluid Mech. 401 (1999) 123–156.
[11] T. Bewley, R. Temam, M. Ziane, A general framework for robust control in fluid mechanics, Physica D 138 (2000) 360–392.
[12] L.S. Hou, Y. Yan, Dynamics and approximations of a velocity tracking problem for the Navier–Stokes flows with piecewise distributed controls, SIAM J. Contr. Opt. 35 (6) (1997) 1847–1885.
[13] K.Y. Tang, W.R. Graham, J. Peraire, Optimal control of vortex shedding using low-order models: open loop model development. II: Model based control, Int. J. Numer. Methods Eng. 44 (1999) 945–990.
[14] M. Hinze, K. Kunisch, Three control methods for time-dependent fluid flows, Flow. Turbul. Combust. 65 (2000) 273–298.
[15] X. Cai, F. Ladeinde, A comparison of two POD methods for airfoil design optimization, AIAA Paper 2005-4912, 2005.
[16] A. Hay, J. Borggaard, D. Pelletier, Reduced-order models for parameter dependent geometries based on shape and sensitivity analysis of the POD, AIAA Paper 2008-5962, 2008.
[17] P.A. LeGresley, J.J. Alonso, Investigation of non-linear projection for POD based reduced order models for aerodynamics, AIAA Paper 2001-0926, 2001.
[18] S.K. Nadarajah, M. McMullen, A. Jameson Non-linear frequency domain based optimum shape design for unsteady three-dimensional flow, AIAA Paper 2006-1052, 2006.
[19] S. Choi, M. Potsdam, K Lee, G. Iaccario, J.J. Alonso, Helicopter rotor design using a time-spectral and adjoint-based method, AIAA Paper 2008-5810, 2008.
[20] W.K. Anderson, D.L. Bonhaus, An implicit upwind algorithm for computing turbulent flows on unstructured grids, Comput. Fluid 23 (1994) 1–21.
[21] P.L. Roe, Approximate Riemann solvers, parameter vectors, and differential schemes, J. Comput. Phys. 43 (2) (1981) 357–372.
[22] N.K. Yamaleev, B. Diskin, E.J. Nielsen, Adjoint-based methodology for time-dependent optimization, AIAA Paper 2008-5857, 2008.
[23] J.N. Lyness, C.B. Moler, Numerical differentiation of analytic functions, SIAM J. Numer. Anal. 4 (1967) 202–210. 124–134.