A framework for simultaneous aerodynamic design optimization in the presence of chaos

Stefanie Günther[a,*], Nicolas R. Gauger[a], Qiqi Wang[b]

[a] *TU Kaiserslautern, Chair for Scientific Computing, Paul-Ehrlich-Straße 34, 67663 Kaiserslautern, Germany*
[b] *Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, 77 Massachusetts Avenue, Cambridge, MA 02139, USA*

Abstract

Integrating existing solvers for unsteady partial differential equations into a simultaneous optimization method is challenging due to the forward-in-time information propagation of classical time-stepping methods. This paper applies the simultaneous single-step one-shot optimization method to a reformulated unsteady constraint that allows for both forward- and backward-in-time information propagation. Especially in the presence of chaotic and turbulent flow, solving the initial value problem simultaneously with the optimization problem often scales poorly with the time domain length. The new formulation relaxes the initial condition and instead solves a least squares problem for the discrete partial differential equations. This enables efficient one-shot optimization that is independent of the time domain length, even in the presence of chaos.

Keywords: simultaneous optimization, one-shot, unsteady flow, dual time-stepping, chaos, least squares shadowing

1. Introduction

This paper is motivated by aerodynamic design in the presence of massively separated flows. Important applications include launch vehicle design [1, 2], helicopter design [3, 4], designing highly maneuverable air vehicles

*Corresponding author
Email address: `stefanie.guenther@scicomp.uni-kl.de` (Stefanie Günther)

[5, 6], and designing turbomachinery components [7, 8]. In these applications, massively separation often generate large scale unsteady flows, which can cause large fluctuating pressure forces and may lead to undesirable structural vibration and acoustic noise [1, 9]. Avoiding and mitigating these engineering issues motivates efficient aerodynamic design optimization in unsteady simulations of massively separated flows.

The paper applies simultaneous optimization [10, 11, 12, 13, 14, 15, 16, 17, 18] to unsteady partial differential equations (PDEs) which mimic the behavior of massively separated flows, in particular flow past cylindrical bluff bodies [19, 20]. Such flows typically contain two most dynamically important regions, the near wake, and the far wake [19]. The near wake contains a recirculation region behind the bluff body, whose shear layers generate a repeating pattern of swirling vortices known as von Karman vortex street. These vortices shed into the far wake, where they advect downstream and slowly dissipate. In our 1D model, the near wake is mimicked by a nonlinear ordinary differential equation (ODE) exhibiting self-excited oscillations; the far wake is mimicked by an advection-diffusion equation, whose upstream boundary condition is determined by the ODE mimicking the near wake.

The nonlinear dynamics of the near wake dictates our choice of ODE mimicking it. In the near wake of a circular cylinder, regular, periodic vortex shedding occurs at Reynolds numbers between 40 and 150. When the Reynolds number exceeds a few hundred, the near wake become irregular and aperiodic[19]. These higher Reynolds number flows are of interest to most aerospace engineering applications. For high Reynolds number turbulent flows, direct simulation is currently impractical. Instead they are often simulated using unsteady Reynolds-averaged Navier-Stokes equations (RANS), large eddy simulation (LES), or their hybrid [21, 22, 23, 24]. RANS often models regular, periodic vortex shedding in the near wake, while LES often models irregular, aperiodic vortex shedding, which better resembles experiments [24, 25]. While regular, periodic vortex shedding can be explained by nonlinear limit cycles, irregular, aperiodic self-excited oscillation can be explained by strange attractors of chaotic nonlinear dynamics [26, 27, 28, 29] In this paper, we use well known ODE models of both nonlinear limit cycle and chaotic strange attractor, respectively the Van-der-Pol oscillator and the Lorenz attractor [30, 31, 32, 33, 34].

1.1. Simultaneous one-shot optimization with PDE constraints

We aim at solving an optimization problem that is constrained by an unsteady differential equation, as e.g. a Van-der-Pol oscillator, a Lorenz attractor, or a PDE that governs the dynamics of massively separated flows, as the following

$$\min_{y,u} J(y,u) \quad \text{s.t.} \quad c(y,u) = 0. \tag{1}$$

Here $y \in Y$ and $u \in U$ represent the state and the design variables in a differential equation $c(y,u) = 0$, respectively, with $c\colon Y \times U \to Y$. J denotes some objective function to be minimized with $J\colon Y \times U \to \mathbb{R}$. }Throughout the paper we assume that Y and U are finite dimensional Hilbert spaces so that their elements can be associated with coordinate vectors in $\mathbb{R}^m$ and $\mathbb{R}^n$, if $\dim Y = m$ and $\dim U = n$, and write duals as transposed vectors.

If solving the PDE is computationally expensive, so-called *one-shot* or *full-space* methods that solve the necessary optimality conditions for the problem (1) simultaneously in the full space for the state, the design and the corresponding adjoint variables have proven to be very efficient [14, 35, 18, 16]. However, many full-space methods require the computation of additional Jacobians of the PDE constraints which are not part of common PDE solvers. This makes it necessary to rewrite existing simulation codes for optimization.

In this paper, we rather pursue the so-called *single-step one-shot* optimization method [36, 37] that overcomes this difficulty and exploits existing simulation software by treating the simulation iterations in a more or less black box fashion. To this end, we assume that the user is provided with a simulation code of fixed-point type that computes a PDE solution y with the iteration

$$y_{k+1} = H(y_k, u) \tag{2}$$

for any given design u while the limit point $\lim_{k\to\infty} y_k = y$ satisfies

$$y = H(y,u) \Leftrightarrow c(y,u) = 0. \tag{3}$$

Following the *first-discretize-then-optimize* approach, we replace the constraints in (1) by the fixed-point formulation for the PDE solver and focus on the discrete optimization problem

$$\min_{y,u} J(y,u) \quad \text{s.t.} \quad y = H(y,u) \tag{4}$$

instead. With adjoint variables $\bar{y} \in Y^*$, the necessary optimality conditions are given by the Karush-Kuhn-Tucker conditions [38]

$$y = H(y, u) \qquad \textit{state equation} \tag{5}$$
$$\bar{y} = J_y(y, u)^T + H_y(y, u)^T \bar{y} \qquad \textit{adjoint equation} \tag{6}$$
$$0 = J_u(y, u)^T + H_u(y, u)^T \bar{y} \qquad \textit{design equation} \tag{7}$$

where subscripts denote partial derivatives. In the single-step one-shot method, this system of equations is solved simultaneously in a single coupled iteration:

$$\begin{bmatrix} y_{k+1} \\ \bar{y}_{k+1} \\ u_{k+1} \end{bmatrix} = \begin{bmatrix} H(y_k, u_k) \\ J_y(y_k, u_k)^T + H_y(y_k, u_k)^T \bar{y}_k \\ u_k - B_k^{-1} \left(J_u(y_k, u_k)^T + H_u(y_k, u_k)^T \bar{y}_k \right) \end{bmatrix}. \tag{8}$$

The first line corresponds to one iteration of the provided PDE solver. The second line performs one step of an adjoint PDE solver that attempts to solve the adjoint equation (6). It can be generated efficiently by applying Automatic Differentiation (AD) to H and J [39]. The last line of iteration (8) applies a preconditioned design update in the direction of the (inexact) reduced gradient $J_u^T + H_u^T \bar{y}$. The choice of the preconditioning matrix B_k ensures convergence of the single-step one-shot method according to [40]. It has been proven there, that B_k approximates the reduced Hessian of J near the optimal point. It can be approximated using secant updates based on the reduced gradient (e.g. BFGS updates [38]).

The single-step one-shot method simultaneously solves the PDE along with the optimization problem. It exploits existing simulation software by treating the mapping H in a black box fashion in the optimization process. In contrast to many other full-space optimization methods, no additional Jacobians of the PDE constraints are needed for optimization other than the simulation tool. Besides the single-step one-shot method, other so-called *multi-step* one-shot schemes have been developed which perform not only one step of the simulation iterations in each optimization cycle but several. A survey on these schemes and a discussion on optimal sequencing of the primal, the adjoint and the design steps can be found in [41].

The single-step one-shot method has been applied to various optimization problems with a major focus on steady state aerodynamics [36, 41, 12]. Steady state PDEs are often solved with a pseudo time-stepping method in which the flow evolves in (pseudo) time until a steady state is reached. In this case, the fixed-point PDE solver H corresponds to one (pseudo) time step of the method while the iterates y_k are the state at (pseudo) time t_k. Many applications have proven, that the cost for single-step one-shot optimization is only a small multiple of the cost of a pure PDE simulation, measured in iteration counts. The factor typically varies between 2 and 8 [41, 42].

However, since aerodynamic optimization with unsteady PDEs have become an active field of research [43, 13, 44], application of the single-step one-shot method to unsteady flow is desirable.

1.2. One-shot optimization with unsteady PDEs

For optimization with unsteady PDEs one is typically interested in minimizing some long-time averaged quantity of the flow

$$\lim_{T\to\infty} \frac{1}{T} \int_0^T J(y(t), u)\,\mathrm{d}t \tag{9}$$

while the state $y\colon [0,\infty) \to Y$ and the control $u \in U$ satisfy an unsteady PDE

$$\frac{\partial y(t)}{\partial t} + c(y(t), u) = 0 \quad \forall t > 0 \tag{10}$$

with appropriate initial data $y(0) = y^0$. Here, $c(y(t), u)$ represents the spacial derivative operators of the unsteady PDE.

Simulation tools for unsteady PDEs typically discretize the transient term by an implicit scheme and solve the resulting implicit equations sequetially in time as proposed in the famous dual time-stepping approach [45]. We assume throughout the paper, that the user is provided with a time-marching simulation code of that type. Moving forward in time, the simulation code applies a fixed-point solver G that iteratively solves the implicit equations at each time step:

$$\text{for } i = 1, \dots, N: \tag{11}$$

$$\text{iterate } y^i_{k+1} = G(y^i_k, y^{i-1}, \dots, u) \overset{k\to\infty}{\longrightarrow} y^i = G(y^i, y^{i-1}, \dots, u) \tag{12}$$

where $y^i \in Y$ approximate the PDE solution at the discrete time steps $0 = t_0 < \dots < t_N = T$. The inner fixed-point iterations at a certain time step (12) are based on the converged solutions at one or more previous states $y^{i-1}, y^{i-2}, \dots$, depending on the number of steps used in the multi-step time-marching scheme.

Since these schemes operate forward in time, computing time step after time step, integration in a simultaneous single-step one-shot optimization is not straightforward. In the single-step one-shot method, the design updates are integrated in the primal flow computation and are based on intermediate state variables which, in case of unsteady PDEs, are trajectories. This means, that the fixed-point iterator for solving the primal equation in a single-step one-shot framework should incorporate primal state updates on an entire intermediate time trajectory.

One possibility to achieve this while exploiting the unsteady time-marching scheme is, to reduce the number of inner fixed-point iterations in (12) and perform only one update at each time step. In an outer one-shot optimization iteration, this reduced time-marching scheme is then augmented with updates for the adjoint and design variables according to (8). This unsteady single-step one-shot method has been applied to an optimal active flow control problem in [10] where the flow is governed by the unsteady RANS equations. However, the method suffers from the fact that the number of iterations needed for convergence depends on the length of the time domain. In section 2 we numerically investigate the dependence by applying the method to an advection-diffusion flow problem driven by an oscillatory boundary term.

The numerical investigation in Section 2 will demonstrate, that the poor scalability with respect to the number of time steps is inherent in the unsteady PDE constraint that we account for. If the initial value of the unsteady PDE is fixed, small corrections at earlier time steps can result in large changes of the state at later times. This is especially the case for chaotic or turbulent flow and is well-known as the so-called butterfly effect: A small perturbation of the state at any time can result in large differences at later states. In other words: The initial value problem of turbulent dynamics, which we attempt to solve as constraints of the optimization problem, is ill-posed.

However, the objective function of the unsteady optimization problem does not depend on a certain initial value and a certain realization of the flow trajectory. In fact, the quantity of interest (e.g. average drag coefficient, mean flow etc.) is rather a statistical quantity of the flow which is independent of the initial condition. Changing the initial condition would lead to a different realization of the flow trajectory, but the statistical quantity of interest remains unchanged in the limit of infinite time.

In Section 3, we therefore reformulate the constraints of the unsteady optimization problem by relaxing the initial value and searching for trajectories that satisfies the differential equations only. From the manifold of trajectories that satisfy the PDEs, a least squares problem chooses one that is closest in some norm to a reference trajectory while the initial value can be adjusted accordingly. This so-called *Least Square Shadowing* (LSS) problem then serves as new constraints of the unsteady optimization problem. By solving it iteratively we can integrate it efficiently in a single-step one-shot optimization algorithm.

Numerical results for LSS-constrained single-step one-shot optimization in Section 4 show, that the dependency on the time domain length is no longer visible which makes the single-step one-shot method attractive for optimization with unsteady PDEs especially for chaotic and turbulent flow.

2. Single-step one-shot with reduced time-marching schemes

In this section, we recall the single-step one-shot method for unsteady time-marching schemes as presented in [10] and investigate its scalability barrier with respect to the time domain length numerically for an advection-diffusion flow problem that is driven by an oscillatory boundary term.

The single-step one-shot method for unsteady time-marching schemes attempts to solve the discrete optimization problem

$$\begin{aligned} &\min_{y^1,\dots,y^N\in Y,u\in U} \tilde{J}(y^0,\dots,y^N,u) \\ \text{s.t.} \quad & y^i = G(y^i,y^{i-1},u) \quad \forall i=1,\dots,N \end{aligned} \tag{13}$$

for a fixed initial value $y^0 \in Y$. The objective function $\tilde{J}\colon Y\times\ldots\times Y\times U \to \mathbb{R}$ is a discrete finite-time approximation to the objective funtion in (9). The constraints are the fixed points of the inner iterations of an implicit time-marching scheme as in (12). For simplicity, we drop the dependency of G on previous states other than y^{i-1}. Application of the method to other multi-step time-marching schemes is straightforward.

The first order optimality conditions for (13) are given by the KKT system

$$G(y^i,y^{i-1},u) - y^i = 0 \quad \forall i=1,\dots,N \tag{14}$$

$$\partial_{y^i}\tilde{J}^T + \left(\partial_{y^i}G^i - I\right)^T \bar{y}^i + \left(\partial_{y^i}G^{i+1}\right)^T \bar{y}^{i+1} = 0 \quad \forall i=1,\dots,N \tag{15}$$

$$\partial_u\tilde{J} + \sum_{i=1}^N \left(\partial_u G^i\right)^T \bar{y}^i = 0 \tag{16}$$

where $\bar{y}^i$ are the adjoint variables with $\bar{y}^N = 0$ and $G^i := G(y^i,y^{i-1},u)$.

In the single-step one-shot method, this set of equations is solved simultaneously in a single iteration for updating the state, the adjoint and the design variable. It has been shown in [10], that a suitable state update is obtained from the classical implicit time-marching scheme by reducing the number of inner fixed-point iterations in (12) to one. One state update then involves updates for the entire time trajectory:

$$\begin{aligned} &\text{for } i = 1, \ldots, N : \\ &\qquad \text{update } \; y^i_{k+1} = G(y^i_k, y^{i-1}_k, u). \end{aligned} \tag{17}$$

In contrast to (12), the update at a certain time step now depends on inexact states at previous time steps y^{i-1}_k instead of the converged fixed point y^{i-1}. Applying AD to the reduced time-marching scheme (17) and the discrete objective function $\tilde{J}$ generates an update for the adjoint variables to solve (15)

$$\begin{aligned} &\text{for } i = N, \ldots, 1 : \\ &\qquad \text{update } \; \bar{y}^i_{k+1} = \partial_{y^i} \tilde{J}^T_k + \left(\partial_{y^i} G^i_k\right)^T \bar{y}^i_k + \left(\partial_{y^i} G^{i+1}_k\right)^T \bar{y}^{i+1}_k \end{aligned} \tag{18}$$

and evaluates the (inexact) reduced gradient in (16). The reduced gradient is utilized to perform a preconditioned update for the design:

$$u_{k+1} = u_k - B_k^{-1} \left(\partial_u \tilde{J}_k + \sum_{i=1}^{N} \left(\partial_u {G^i}_k\right)^T \bar{y}^i_k \right). \tag{19}$$

with a preconditioning matrix B_k^{-1}. As in the steady case, the preconditioner should approximate the reduced Hession of the objective function which can be achieved by applying BFGS updates based on the reduced gradient.

In [10], the authors apply the single-step one-shot method to an optimal active flow control problem of unsteady flow around a 2D cylinder. 15 actuation slits are installed on the cylinder surface that apply sinusoidal blowing and suction in order to reduce vorticity downstream the cylinder. An industry standard flow solver of second order in space and time was available which solves the governing unsteady Reynolds-averaged Navier-Stokes equations (URANS) by applying pressure correction loops in each time step (SIMPLE algorithm). In order to transition from simulation to single-step one-shot optimization, the number of SIMPLE iterations at each time step is reduced to one. The reduced time-marching scheme has been differentiated with the AD tool Tapenade [46] in order to generate an adjoint solver and the reduced gradient. The resulting single-step one-shot optimization iterations solve the URANS equations while optimizing the amplitudes of the actuation at each slit simultaneously. An average drag reduction of about 30% has been obtained from the optimization. The cost for the single-step one-shot optimization was measured to be 3 times the cost for a pure simulation with the modified time-marching scheme, measured in iteration counts. However, since the modified time-marching scheme is computationally more expensive than the standard time-stepping scheme, the performance of the one-shot method using the modified scheme compared to the classical scheme remains to be investigated.

2.1. Pitfall for large time domains

It has also been observed in [10] that the number of iterations needed for convergence of the method is proportional to the time domain length. The dependency results from the fact, that the updates in the reduced time-marching scheme (17) are contaminated by inexact states at previous time steps. These errors are propagated through the entire time domain in one execution of (17).

To investigate this kind of error propagation numerically, we consider a one dimensional advection-driven flow actuated by an oscillatory boundary term

$$\begin{aligned}
\partial_t y(t,x) + a\partial_x y(t,x) - \mu\partial_{xx} y(t,x) &= 0 \qquad \text{in } \Omega, \forall t > 0 && (20)\\
y(t,0) - \mu\partial_x y(t,0) &= z(t) \quad \forall t > 0 && (21)\\
\partial_{xx} y(t,1) &= 0 \qquad \forall t > 0 && (22)\\
y(0,x) &= 0 \qquad \forall x \in [0,1] && (23)
\end{aligned}$$

in the domain $\Omega = (0,1)$ with advection velocity $a = 1$ and a small diffusion parameter $\mu = 10^{-5}$. The boundary term $z(t)$ satisfies a nonlinear ordinary differential equation (ODE), namely the Van-der-Pol oscillator

$$\begin{aligned}
\dot{x}(t) &= z(t) && (24)\\
\dot{z}(t) &= -x(t) + \gamma\left(1 - x(t)^2\right) z(t) && (25)
\end{aligned}$$

with parameter $\gamma = 2$ and initial values $x(0) = z(0) = 1$. We use the trapezoidal rule for discretization in time and linear upwinding and central finite differencing for the first and the second order spatial derivative terms, respectively. A Quasi-Newton fixed-point solver is applied at each time step to solve the implicit equations and resemble the scenario of a general implicit time-marching simulation tool. The reduced time-marching scheme according to (17) then performs one update of the Quasi-Newton solver at each time step.

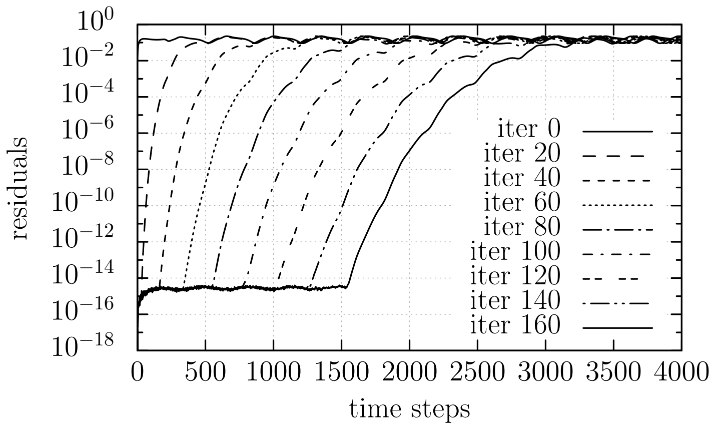
The performance of the reduced time-marching scheme is visualized in Figure 1. The upper figure plots the primal residuals $\|y_k^i - G(y_k^i, y_k^{i-1}, u)\|_Y$ for each time step i for different iterations k corresponding to state updates computed from (17). While the residuals decreases rapidly for t close to zero, the magnitudes of residuals at later time steps stay almost unchanged during the first iterations and get reduced only after previous time steps have already reached a good level of convergence. The lower plot of Figure 1 illustrates a linear dependence of the convergence on the number of physical time steps. The bigger the time domain, the more executions of (17) are necessary in order to reduce the residuals at later time steps.

Similar convergence behavior has been observed for solving unsteady PDEs with time domain decomposition methods [47, 48, 49]: The number of required iterations to reach a certain tolerance is proportional to the length of the time domain. This is especially true for flow dynamics in the presence of chaos: If the initial value is fixed, small corrections of the state at earlier times can result in large differences of the trajectory at later time steps. Later time chunks can therefore only converge after earlier states have already reached a good level of convergence and can then serve as good initial conditions for the later states.

In [47], it has therefore been proposed to relax the initial condition of the unsteady PDE and thereby transform the unsteady flow simulation into a well-conditioned problem. For relaxed initial condition, a perturbation of the state at some time could be accommodated by a change in the initial value with little effect on the state at later times.

3. Single-step one-shot with Least Squares Shadowing constraints

In this section, we reformulate the discrete optimization problem (13) and replace the constraints by the Least Squares Shadowing problem. The new optimization problem is then solved simultaneously with the single-step one-shot method.



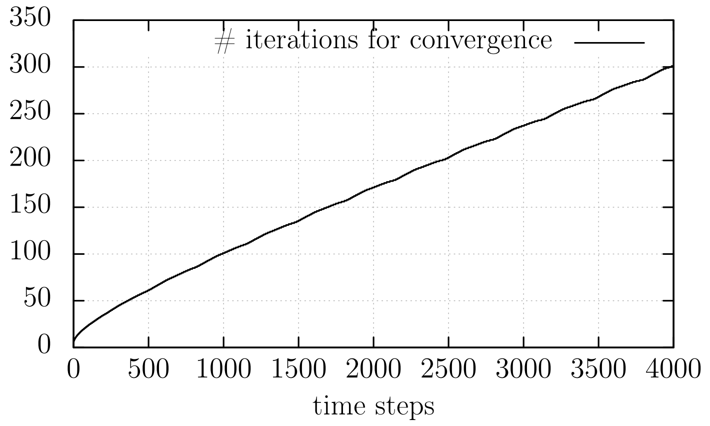


Figure 1: Solving the advection-diffusion equation with reduced time-marching scheme: Residuals over time for 9 different iterations (top); number of iterations needed for convergence over number of time steps (bottom).

3.1. Replacing the initial value problem

We follow the idea in [47, 50] where the Least Squares Shadowing (LSS) problem is introduced in the context of sensitivity analysis of chaotic dynamics. Instead of solving an initial value problem for unsteady PDEs where one tries to find a flow solution that satisfies the initial value and the PDEs, the authors suggest to focus on the governing equations only regardless of where the trajectory starts. Among all trajectories that satisfy the governing equations, a least squares minimization problem filters out one trajectory that is in some sense closest to a prescribed reference trajectory.

While the LSS problem is set up on the continuous level in [50], we prefer to formulate the problem based on the discrete fixed-point formulation of the time-marching scheme (12) in order to exploit existing unsteady simulation software. The discrete Least Squares Shadowing (LSS) problem with fixed-point constraints reads

$$\begin{aligned} \min_{\substack{y^i, i=0,\dots,N \\ \Delta t^i, i=1,\dots,N}} \quad & \frac{1}{N+1}\sum_{i=0}^{N}\frac{1}{2}\left\|y^i - y_r^i\right\|^2 + \frac{\alpha}{N}\sum_{i=1}^{N}\frac{1}{2}\left(1-\frac{\Delta t^i}{\Delta t_r^i}\right)^2 \\ \text{s.t.} \quad & y^i = G(y^i, y^{i-1}, \Delta t^i, u) \quad \text{for } i = 1,\dots,N. \end{aligned} \tag{26}$$

where $y_r^0, \dots, y_r^N \in Y$ denote the discrete states of a reference trajectory with given reference time steps $\Delta t_r^1, \dots, \Delta t_r^N \in \mathbb{R}$.

The solution of the LSS problem consists of a trajectory $y^0, \dots, y^N \in Y$ (including the initial value y^0) and corresponding time steps $\Delta t^1, \dots, \Delta t^N$ with $\Delta t^i := t_i - t_{i-1}$ that satisfy the discrete fixed-point formulation of the governing PDE and are closest to the reference trajectory in the metric given by the objective function. The second term of the objective function regularizes the problem by factoring out tangential movement along the trajectory. The regularization parameter $\alpha > 0$ is chosen such that both terms in the objective function have the same order of magnitude. The LSS problem has a unique and stable solution if the reference trajectory is close to the solution in the metric given by the objective function [50, 51]. The reference trajectory should therefore be chosen as a solution to the unsteady PDEs to a different parameter u. It can be computed in advance where the corresponding u could be the initial guess of the outer optimization problem (13).

3.2. Solving the discrete Least Squares Shadowing problem

Let $w^1, \dots, w^N \in Y^*$ denote the Lagrangian multipliers associated with the LSS problem (26). The necessary optimality conditions are given by the KKT system

$$R_w^i := \frac{1}{N+1}\left(y^i - y_r^i\right) + \left(\partial_{y^i} G^i - I\right)^T w^i + \left(\partial_{y^i} G^{i+1}\right)^T w^{i+1} = 0 \qquad (27)$$
$$\text{for } i = 0, \dots, N$$
$$R_{\Delta t}^i := -\frac{\alpha}{N \Delta t_r^i}\left(1 - \frac{\Delta t^i}{\Delta t_r^i}\right) + \left(\partial_{\Delta t^i} G^i\right)^T w^i = 0 \qquad (28)$$
$$\text{for } i = 1, \dots, N$$
$$R_y^i := G^i(y^i, y^{i-1}, \Delta t^i, u) - y^i = 0 \qquad (29)$$
$$\text{for } i = 1, \dots, N$$

where $w^0 = w^{N+1} = 0$.

We solve this system of equations iteratively by applying a Quasi-Newton method. Linearization with respect to $\boldsymbol{y} = \left(y^0, \dots, y^N\right)$, $\boldsymbol{\Delta t} = \left(\Delta t^1, \dots \Delta t^N\right)$ and $\boldsymbol{w} = \left(w^1, \dots, w^N\right)$ and neglecting second order derivatives of G^i lead to the approximate Newton step

$$\begin{pmatrix} \boldsymbol{y}_{k+1} \\ \boldsymbol{\Delta t}_{k+1} \\ \boldsymbol{w}_{k+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}_{k} \\ \boldsymbol{\Delta t}_{k} \\ \boldsymbol{w}_{k} \end{pmatrix} + \begin{pmatrix} \delta\boldsymbol{y} \\ \delta\boldsymbol{\Delta t} \\ \delta\boldsymbol{w} \end{pmatrix} =: H\left(\begin{pmatrix} \boldsymbol{y}_{k} \\ \boldsymbol{\Delta t}_{k} \\ \boldsymbol{w}_{k} \end{pmatrix}, u\right) \qquad (30)$$

where the update $(\delta\boldsymbol{y}, \delta\boldsymbol{\Delta t}, \delta\boldsymbol{w})^T$ is computed from the linear system

$$\begin{pmatrix} I & 0 & C^T \\ 0 & I & D^T \\ C & D & 0 \end{pmatrix} \begin{pmatrix} \delta\boldsymbol{y} \\ \delta\boldsymbol{\Delta t} \\ \delta\boldsymbol{w} \end{pmatrix} = -\begin{pmatrix} \boldsymbol{R_w} \\ \boldsymbol{R_{\Delta t}} \\ \boldsymbol{R_y} \end{pmatrix} \qquad (31)$$

with

$$C = \begin{pmatrix} \partial_{y^0} G^1 & \partial_{y^1} G^1 - I & & & \\ & \partial_{y^1} G^2 & \partial_{y^2} G^2 - I & & \\ & & \ddots & \ddots & \\ & & & \partial_{y^{N-1}} G^N & \partial_{y^N} G^N - I \end{pmatrix} \qquad (32)$$

and

$$D = \begin{pmatrix} \partial_{\Delta t^1} G^1 & & \\ & \ddots & \\ & & \partial_{\Delta t^N} G^N \end{pmatrix}. \tag{33}$$

Exploiting the block structure by building the Schur complement results in the equation

$$\left(CC^T + DD^T\right) \delta \boldsymbol{w} = \boldsymbol{R_y} + C\boldsymbol{R_w} + D\boldsymbol{R_{\Delta t}} \tag{34}$$

which needs to be solved for $\delta \boldsymbol{w}$ in each Quasi-Newton iteration. We can then compute the trajectory update from

$$\delta \boldsymbol{y} = -\boldsymbol{R_w} - C^T \delta \boldsymbol{w} \tag{35}$$
$$\delta \boldsymbol{\Delta t} = -\boldsymbol{R_{\Delta t}} - D^T \delta \boldsymbol{w}. \tag{36}$$

Although the linear system is of large scale, it is sparse and its block tridiagonal structure can be exploited. New High Performance Computing (HPC) techniques that solve the resulting space-time system (34) in parallel will help to reduce the computational costs. For now, we solve the equation with a direct solver. However, iterative methods will reduce the memory requirements drastically. Moreover, multigrid methods that are specially tailored for equation (34) have been investigated and are an active field of research [52].

3.3. Single-step one-shot with LSS constraints

Instead of solving the optimization problem (13), we now replace the ill-conditioned initial value problem in the constraints by the necessary optimality conditions of the discrete LSS problem written in terms of the fixed-point Quasi-Newton solver H defined in (30). The reformulated optimization problem is given by

$$\begin{aligned} &\min_{\boldsymbol{y}, \boldsymbol{\Delta t}, \boldsymbol{w}, u} \tilde{J}\left(\boldsymbol{y}, \boldsymbol{\Delta t}, u\right) \\ &\text{s.t.} \quad \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{\Delta t} \\ \boldsymbol{w} \end{pmatrix} = H\left(\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{\Delta t} \\ \boldsymbol{w} \end{pmatrix}, u\right). \end{aligned} \tag{37}$$

Although the fixed point that solves the LSS constraint depends on the prescribed reference trajectory $(\boldsymbol{y}_r, \boldsymbol{\Delta t_r})$, the objective $\tilde{J}$ is independent of the reference if the time domain is large.

The optimization problem with LSS constraints (37) can now be solved with the single-step one-shot method as explained in Section 1.1. Let $(\bar{\boldsymbol{y}}, \bar{\boldsymbol{\Delta t}}, \bar{\boldsymbol{w}})$ denote the Lagrange multipliers associated to the optimization problem (37), the single-step one-shot iteration is then given by

$$\begin{pmatrix} \boldsymbol{y}_{k+1} \\ \boldsymbol{\Delta t}_{k+1} \\ \boldsymbol{w}_{k+1} \end{pmatrix} = H\left(\begin{pmatrix} \boldsymbol{y}_k \\ \boldsymbol{\Delta t}_k \\ \boldsymbol{w}_k \end{pmatrix}, u_k\right) \tag{38}$$

$$\begin{pmatrix} \bar{\boldsymbol{y}}_{k+1} \\ \bar{\boldsymbol{\Delta t}}_{k+1} \\ \bar{\boldsymbol{w}}_{k+1} \end{pmatrix} = \nabla_{(\boldsymbol{y},\boldsymbol{\Delta t},\boldsymbol{w})}\tilde{J}_k + \left(\frac{\partial H_k}{\partial(\boldsymbol{y}, \boldsymbol{\Delta t}, \boldsymbol{w})}\right)^T \begin{pmatrix} \bar{\boldsymbol{y}}_k \\ \bar{\boldsymbol{\Delta t}}_k \\ \bar{\boldsymbol{w}}_k \end{pmatrix} \tag{39}$$

$$u_{k+1} = u_k - B_k^{-1}\left(\nabla_u \tilde{J}_k + \left(\frac{\partial H_k}{\partial u}\right)^T \begin{pmatrix} \bar{\boldsymbol{y}}_k \\ \bar{\boldsymbol{\Delta t}}_k \\ \bar{\boldsymbol{w}}_k \end{pmatrix}\right). \tag{40}$$

Since the primal updates (38) require the solution of a sparse linear system according to (34), so do the adjoint updates (38). However, since the Schur complement matrix $CC^T + DD^T$ is symmetric, only the right hand side of the adjoint linear equation differs, which can be exploited efficiently when computing the updates with AD.

4. Numerical Results

We consider numerical results for advection-dominated flow in one dimension similar to Section 2.1 while the upstream boundary condition is determined by an ODE:

$$\partial_t y(t,x) + a\partial_x y(t,x) - \mu\partial_{xx} y(t,x) = 0 \qquad \text{in } \Omega, t > 0 \tag{41}$$

$$y(t,0) - \mu\partial_x y(t,0) = z(t) \quad \forall t > 0 \tag{42}$$

$$\partial_{xx} y(t,1) = 0 \qquad \forall t > 0 \tag{43}$$

$$y(0,x) = 0 \qquad \forall x \in [0,1] \tag{44}$$

where $\Omega = (0,1)$, with advection velocity $a = 1.0$ and small diffusion parameter $\mu = 10^{-5}$. The upstream boundary condition given by $z(t)$ mimics the near wake of flow past cylindrical bluff bodies. We consider two scenarios: In the first test case, $z(t)$ satisfies the Van-der-Pol oscillator

$$\dot{x}(t) = z(t) \tag{45}$$
$$\dot{z}(t) = -x(t) + \gamma\left(1 - x(t)^2\right) z(t) \tag{46}$$

with $\gamma > 0$ which models periodic vortex shedding. The second test case corresponds to irregular and aperiodic oscillations that occur for higher Reynolds numbers. These are modeled by the Lorenz attractor:

$$\dot{x}(t) = \sigma(y(t) - r(t)) \tag{47}$$
$$\dot{r}(t) = x(t)(\rho - z(t)) - r(t) \tag{48}$$
$$\dot{z}(t) = x(t)r(t) - \beta z(t) \tag{49}$$

with $\sigma = 10$ and $\beta = \frac{8}{3}$ fixed and $\rho > 0$.

Again we assume, that we are provided with a classical implicit time-marching scheme solving the PDE equations (41) - (44) with Van-der-Pol or Lorenz boundary condition. The inner fixed-point iterator G consists of one update of the Quasi-Newton method that solves the implicit equations at each time step as in Section 2.1.

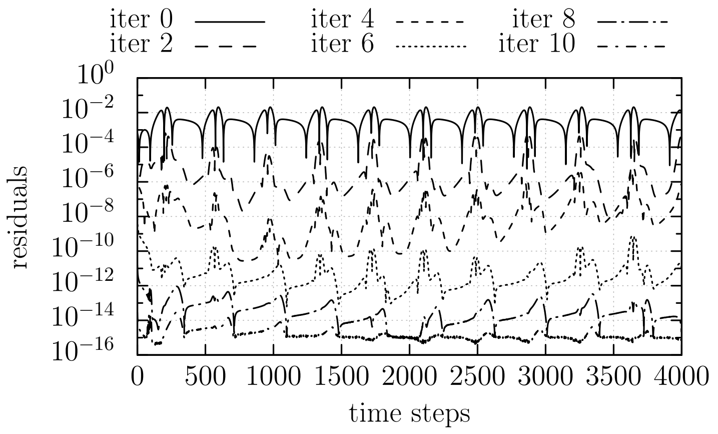
In Section 4.1, we solve the LSS problem for the model problem (41) - (44) with Van-der-Pol and with Lorenz boundary condition, respectively, where the parameters $\gamma > 0$ and $\rho > 0$ are fixed. These parameters then serve as design variables for one-shot optimization with LSS constraints presented in Section 4.2

4.1. Solving the LSS problem for advection-driven flow

For both test cases, a reference trajectory for the parameters $\gamma = 2$ and $\rho = 28$, respectively, is computed in advance with prescribed time step size $\Delta t_r^i = 0.005$. The necessary optimality conditions of the LSS problem are then solved iteratively for the perturbed parameters $\gamma = 3$ and $\rho = 29$, respectively. The update function H is set up according to Section 3.2 applying approximate Newton updates for the trajectory $\boldsymbol{y}$, the time steps $\boldsymbol{\Delta t}$ and the Lagrangian multipliers of the LSS problem $\boldsymbol{w}$.

Figure 2 visualizes the convergence of the LSS iterations for both the Van-der-Pol and the Lorenz boundary test cases. For both boundary types, the primal residuals $\|y_k^i - G(y_k^i, y_k^{i-1}, u)\|_Y$ of different intermediate trajectories are reduced uniformly over the entire time domain. The dependency on the time step number as observed for the reduced time-marching scheme in Section 2.1, Figure 1, has been removed. This is also visible in Figure 3 where LSS solutions with have been computed for various numbers of time steps and corresponding reference trajectories. The numbers of iterations needed for convergence are independent of the number of time steps for both, the regular periodic as well as the chaotic boundary test case.

A detailed view on the upstream chaotic boundary of the reference solution for $\rho = 28$ and the corresponding LSS solution computed with $\rho = 29$ is given in Figure 4. As expected, the LSS solution closely tracks the reference trajectory on a transformed time scale.



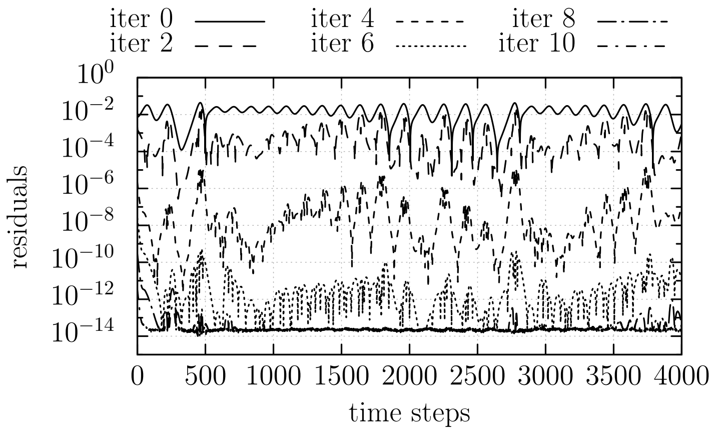


Figure 2: Residuals over time while solving the LSS problem for advection-diffusion equation with periodic (top) and chaotic boundary term (bottom).

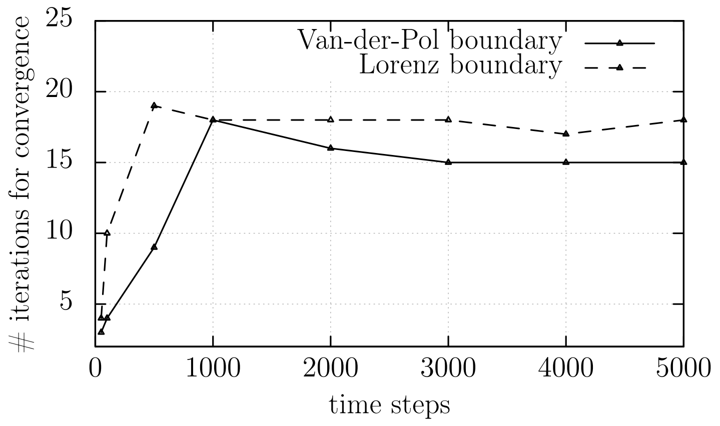


Figure 3: Number of iterations needed for solving the LSS problem over number of time steps for the periodic (Van-der-Pol boundary) and chaotic boundary test case (Lorenz boundary).

4.2. Single-step one-shot with LSS constraints for advection-driven flow

The iterative LSS solver from the previous section is now integrated in a single-step one-shot optimization loop according to Section 3.3. Again, the reference trajectories are computed with parameters $\gamma = 2$ and $\rho = 28$. These parameters then serve as design variables in order to minimize a tracking type objective function for matching the space-time averaged state to a prescribed value a_{target}

$$J_T = \frac{1}{2}\left(\frac{1}{T}\int_0^T \|y(t,\cdot)\|_2 \,\mathrm{d}t - a_{\text{target}}\right)^2. \tag{50}$$

The target values a_{target} have been computed in advance for the parameters $\gamma = 3$ and $\rho = 29$, respectively.

The adjoint updates as well as the reduced gradient are computed with the open source AD tool CoDiPack [53]. The preconditioner for the design update is approximated with σI where a fixed step size of $\sigma = 0.5$ is chosen.

In Figure 5, the single-step one-shot optimization history is plotted for the Van-der-Pol boundary (left) and the Lorenz boundary (right). For both cases, the primal and adjoint residuals computed from $\|(\boldsymbol{y}, \boldsymbol{\Delta t}, \boldsymbol{w})_k - (\boldsymbol{y}, \boldsymbol{\Delta t}, \boldsymbol{w})_{k-1}\|$ and $\|(\bar{\boldsymbol{y}}, \bar{\boldsymbol{\Delta t}}, \bar{\boldsymbol{w}})_k - (\bar{\boldsymbol{y}}, \bar{\boldsymbol{\Delta t}}, \bar{\boldsymbol{w}})_{k-1}\|$ are reduced simultaneously with the norm of the reduced gradient indicating that the one-shot method has been applied successfully. The objective function is reduced to machine precision. Comparing the number of iterations needed for convergence of the optimization

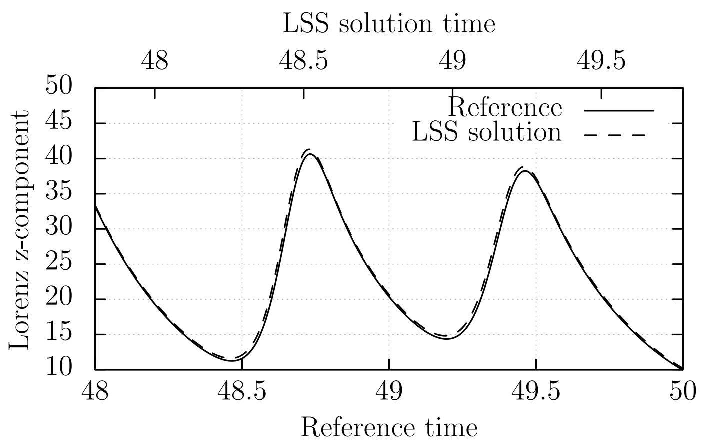


Figure 4: Image section of the chaotic boundary of the reference solution for $\rho = 28$ and the corresponding LSS solution computed with $\rho = 29$.

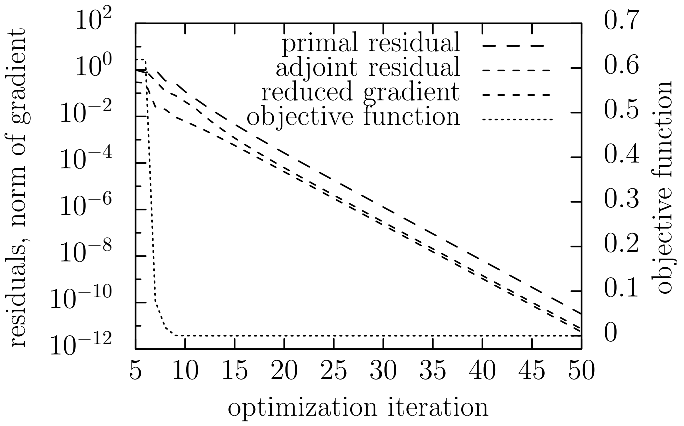
in Figure 5 with that of the pure LSS simulation in Figure 3 for number of time steps $N = 4000$ one can see that the cost for a single-step one-shot optimization for the Van-der-pol and the Lorenz boundary cases are about 3 and 5 times the cost for a pure LSS simulation, respectively, measured in iteration counts.

5. Conclusion

In this paper, we presented a reformulation of unsteady PDE-constrained optimization that enables the integration of existing unsteady simulation software into simultaneous single-step one-shot optimization.

The new formulation replaces the unsteady PDE constraint by a least squares problem that relaxes the initial condition and focuses only on the discrete implicit equations arising from classical implicit time-stepping methods. The necessary optimality conditions of the least squares problem are solved iteratively in space-time using a non-intrusive Quasi-Newton method. These iterations are integrated in a single-step one-shot optimization loop.

The new approach has been applied to a one dimensional advection-diffusion equation driven by an oscillating upstream boundary condition. Solving the corresponding least squares problem provides a uniform reduction of the residuals over the entire time domain - independent of the time domain length. Its integration into a simultaneous one-shot method has been demonstrated for an inverse design optimization problem.



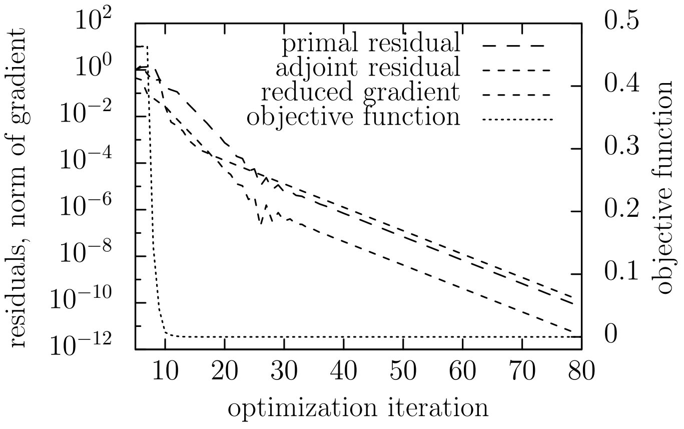


Figure 5: One-shot optimization with LSS constraints for advection-diffusion equation with Van-der-Pol boundary (top) and Lorenz boundary (bottom).

By relaxing the initial condition, changes to the trajectory at early time chunks and changes to the design parameters can be accommodated by a change in the initial condition of the trajectory. Transporting information is thereby no longer limited to forward-in-time propagation as it is the case for unsteady PDEs with fixed initial conditions. The formulation is therefore especially beneficial for simultaneous optimization of statistical quantities subject to chaotic and turbulent flow constraints.

Consequently, as a next step the presented framework will be applied to optimal control of massively separated flow. This necessitates the application of new HPC techniques that solve the resulting space-time system efficiently.

References

[1] G. J. Brauckmann, C. L. Streett, W. L. Kleb, S. J. Alter, K. J. Murphy, C. E. Glass, Computational and experimental unsteady pressures for alternate sls booster nose shapes, in: AIAA Science and Technology Forum and Exposition, Kissimmee, Florida, 2015.

[2] S. J. Alter, G. J. Brauckmann, B. Kleb, C. E. Glass, C. L. Streett, D. M. Schuster, Time-accurate unsteady pressure loads simulated for the space launch system at wind tunnel conditions, in: 33rd AIAA Applied Aerodynamics Conference, 2015, p. 3149.

[3] S. Choi, K. Lee, M. M. Potsdam, J. J. Alonso, Helicopter rotor design using a time-spectral and adjoint-based method, Journal of Aircraft 51 (2) (2014) 412–423.

[4] W. T. Jones, E. J. Nielsen, E. M. Lee-Rausch, C. Acree Jr, Multi-point adjoint-based design of tilt-rotors in a noninertial reference frame, AIAA Paper 290 (2014) 2014.

[5] S. A. Morton, R. M. Cummings, D. B. Kholodar, High resolution turbulence treatment of f/a-18 tail buffet, Journal of Aircraft 44 (6) (2007) 1769–1775.

[6] E. J. Nielsen, B. Diskin, N. K. Yamaleev, Discrete adjoint-based design optimization of unsteady turbulent flows on dynamic unstructured grids, AIAA Journal 48 (6) (2010) 1195–1206.

[7] L. He, D. Wang, Concurrent blade aerodynamic-aero-elastic design optimization using adjoint method, Journal of Turbomachinery 133 (1) (2011) 011021.

[8] H. Wu, F. Liu, H. Tsai, Aerodynamic design of turbine blades using an adjoint equation method, AIAA paper 1006 (2005) 10–13.

[9] A. Park, V. Vatsa, M. Khorram, D. Lockard, Aeroacoustic simulation of nose landing gear on adaptive unstructured grids with fun3d, in: Proc. of 19th AIAA/CEAS Aeroacoustics Conference, AIAA Paper, Vol. 2071, 2013.

[10] S. Günther, N. Gauger, Q. Wang, Simultaneous One-shot Optimization with unsteady PDEs, Journal of Computational and Applied Mathematics 294 (2016) 12–22. doi:10.1016/j.cam.2015.07.033.

[11] A. Griewank, A. Hamdi, Reduced quasi-Newton method for simultaneous design and optimization, Comput. Optim. Appl. 49 (2011) 521–548.

[12] S. Hazra, V. Schulz, Simultaneous pseudo-timestepping for PDE-model based optimization problems, Bit Numerical Mathematics 44 (2004) 457–472.

[13] D. Abbeloos, M. Diehl, M. Hinze, S. Vandewalle, Nested multigrid methods for time-periodic, parabolic optimal control problems, Computing and Visualization in Science 14 (1) (2011) 27–38.

[14] V. Akçelik, G. Biros, O. Ghattas, J. Hill, D. Keyes, B. van Bloemen Waanders, Parallel algorithms for PDE-constrained optimization, in: M. A. Heroux, P. Raghaven, H. D. Simon (Eds.), Parallel Processing for Scientific Computing, SIAM, 2006, pp. 291–322.

[15] L. T. Biegler, O. Ghattas, M. Heinkenschloss, B. van Bloemen Waanders, Large-scale PDE-constrained optimization, Vol. 30, Springer, 2003.

[16] G. Biros, O. Ghattas, Parallel Lagrange–Newton–Krylov–Schur Methods for PDE-Constrained Optimization. Part I: The Krylov–Schur Solver, SIAM Journal on Scientific Computing 27 (2) (2005) 687–713.

[17] G. Biros, O. Ghattas, Parallel lagrange–newton–krylov–schur methods for pde-constrained optimization. part ii: The lagrange–newton solver

and its application to optimal control of steady viscous flows, SIAM Journal on Scientific Computing 27 (2) (2005) 714–739.

[18] J. C. Ziems, S. Ulbrich, Adaptive multilevel inexact SQP methods for PDE-constrained optimization, SIAM Journal on Optimization 21 (1) (2011) 1–40.

[19] C. Williamson, Vortex dynamics in the cylinder wake, Annual Review of Fluid Mechanics 28 (1996) 477–539.

[20] Q. Wang, J. Gao, The drag-adjoint field of a circular cylinder wake at reynolds numbers 20, 100 and 500, Journal of Fluid Mechanics 730.

[21] S. Jordan, S. Ragab, A large-eddy simulation of the near wake of a circular cylinder, Journal of fluids engineering 120 (2) (1998) 243–252.

[22] M. Breuer, Large eddy simulation of the subcritical flow past a circular cylinder: numerical and modeling aspects, International Journal for Numerical Methods in Fluids 28 (9) (1998) 1281–1302.

[23] A. Travin, M. Shur, M. Strelets, P. Spalart, Detached-eddy simulations past a circular cylinder, Flow, Turbulence and Combustion 63 (1-4) (2000) 293–313.

[24] W. Rodi, Comparison of les and rans calculations of the flow around bluff bodies, Journal of Wind Engineering and Industrial Aerodynamics 69 (1997) 55–75.

[25] H. Lübcke, T. Rung, F. Thiele, et al., Comparison of les and rans in bluff-body flows, Journal of Wind Engineering and Industrial Aerodynamics 89 (14) (2001) 1471–1485.

[26] T. Leweke, M. Provansal, Model for the transition in bluff body wakes, Physical review letters 72 (20) (1994) 3174.

[27] C. H. Williamson, Three-dimensional vortex dynamics in bluff body wakes, Experimental Thermal and Fluid Science 12 (2) (1996) 150–168.

[28] J. Ravoux, A. Nadim, H. Haj-Hariri, An embedding method for bluff body flows: interactions of two side-by-side cylinder wakes, Theoretical and Computational Fluid Dynamics 16 (6) (2003) 433–466.

[29] G. E. Karniadakis, G. S. Triantafyllou, Three-dimensional dynamics and transition to turbulence in the wake of bluff objects, Journal of Fluid Mechanics 238 (1992) 1–30.

[30] T. Kanamaru, Van der pol oscillator, Scholarpedia 2 (1) (2007) 2202.

[31] E. Lorenz, Deterministic non-periodic flows, J. Atmos. Sci. 51 (1963) 1037–1056.

[32] W. Tucker, The lorenz attractor exists, Comptes Rendus de l'Académie des Sciences-Series I-Mathematics 328 (12) (1999) 1197–1202.

[33] J. Guckenheimer, R. F. Williams, Structural stability of lorenz attractors, Publications Mathématiques de l'IHÉS 50 (1979) 59–72.

[34] C. Sparrow, The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors, Springer-Verlag, New York, 1982.

[35] E. Özkaya, One-shot methods for aerodynamic shape optimization, Ph.D. thesis, RWTH Aachen University (2014).

[36] N. Gauger, E. Özkaya, Single-step one-shot aerodynamic shape optimization, Internat. Ser. of Numer. Math. 158 (2009) 191–204.

[37] T. Bosse, N. Gauger, A. Griewank, S. Günther, V. Schulz, One-shot approaches to design optimzation, Internat. Ser. of Numer. Math. 165 (2014) 43–66.

[38] J. Nocedal, S. Wright, Numerical Optimization, 2nd Edition, Springer Science+Business Media, 2006.

[39] A. Griewank, C. Faure, Reduced Functions, Gradients and Hessians from Fixed-Point Iterations for State Equations, Numer. Algorithms 30 (2002) 113–139.

[40] A. Griewank, A. Hamdi, Properties of an augmented Lagrangian for design optimization, Optim. Methods Softw. 25 (2010) 645–664.

[41] T. Bosse, N. Gauger, A. Griewank, S. Günther, L. Kaland, et al., Optimal design with bounded retardation for problems with non-separable adjoints, Internat. Ser. of Numer. Math. 165 (2014) 67–84.

[42] N. Gauger, A. Griewank, A. Hamdi, C. Kratzenstein, E. Özkaya, T. Slawig, Automated extension of fixed point PDE solvers for optimal design with bounded retardation, in: G. Leugering, S. Engell, A. Griewank, M. Hinze, R. Rannacher, V. Schulz, M. Ulbrich, S. Ulbrich (Eds.), Constrained Optimization and Optimal Control for Partial Differential Equations, Springer Basel, 2012, pp. 99–122.

[43] M. P. Rumpfkeil, D. W. Zingg, The optimal control of unsteady flows with a discrete adjoint method, Optim. Eng. 11 (2010) 5–22.

[44] A. Carnarius, F. Thiele, E. Özkaya, A. Nemili, N. R. Gauger, Optimal control of unsteady flows using a discrete and a continuous adjoint approach, in: System Modeling and Optimization, Springer, 2013, pp. 318–327.

[45] A. Jameson, Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings, in: Proc. 10th Comp. Fluid Dyn. Conf., Honolulu, USA, June 24-26, AIAA-Paper 91-1596, 1991.

[46] TROPICS (Research Team of INRIA Sophia-Antipolis), Tapenade 3.8, online automatic differentiation engine, `http://www-sop.inria.fr/tropics/tapenade.html`, `tropics@sophia.inria.fr` (2014).

[47] Q. Wang, S. A. Gomez, P. J. Blonigan, A. L. Gregory, E. Y. Qian, Towards scalable parallel-in-time turbulent flow simulations, Physics of Fluids 25 (11), 2013.

[48] M. Gander, E. Hairer, Nonlinear convergence analysis for the parareal algorithm, in: U. Langer, M. Discacciati, D. E. Keyes, O. B. Widlund, W. Zulehner (Eds.), Domain Decomposition Methods in Science and Engineering XVII, Vol. 60 of Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 2008, pp. 45–56.

[49] J. Reynolds-Barredo, D. Newman, R. Sanchez, D. Samaddar, L. Berry, W. Elwasif, Mechanisms for the convergence of time-parallelized, parareal turbulent plasma simulations, Journal of Computational Physics 231 (23) (2012) 7851 – 7867.

[50] Q. Wang, R. Hu, P. Blonigan, Least Squares Shadowing sensitivity analysis of chaotic limit cycle oscillations, Journal of Computational Physics 267 (2014) 210–224.

[51] Q. Wang, Convergence of the least squares shadowing method for computing derivative of ergodic averages, SIAM Journal on Numerical Analysis 52 (1) (2014) 156–170.

[52] P. Blonigan, Q. Wang, Multigrid-in-time for sensitivity analysis of chaotic dynamical systems, Numerical Linear Algebra with Applications 21, (2014). doi:10.1002/nla.1946.

[53] CoDiPack - Code Differentiation Package (Version 1.0), `http://www.scicomp.uni-kl.de/software/codi/`, `codi@scicomp.uni-kl.de` (2015).