Preprint version. Final version is Published in
Journal of Computational Physics **256**, 1-16 (2014)
http://dx.doi.org/10.1016/j.jcp.2013.08.044

Sequential Quadratic Programming (SQP) for optimal control in direct numerical simulation of turbulent flow

Hassan Badreddine[a,1], Stefan Vandewalle[b], Johan Meyers[a,*]

[a]*Department of Mechanical Engineering, KU Leuven, Celestijnenlaan 300A, B3001 Leuven, Belgium*
[b]*Department of Computer Science, KU Leuven, Celestijnenlaan 200A, B3001 Leuven, Belgium*

Abstract

The current work focuses on the development and application of an efficient algorithm for optimization of three-dimensional turbulent flows, simulated using Direct Numerical Simulation (DNS) or large-eddy simulations, and further characterized by large-dimensional optimization-parameter spaces. The optimization algorithm is based on Sequential Quadratic Programming (SQP) in combination with a damped formulation of the limited-memory BFGS method. The latter is suitable for solving large-scale constrained optimization problems whose Hessian matrices cannot be computed and stored at a reasonable cost. We combine the algorithm with a line search merit function based an an L_1-norm to enforce the convergence from any remote point. It is first shown that the proposed form of the damped L-BFGS algorithm is suitable for solving equality constrained Rosenbrock type functions. Then, we apply the algorithm to an optimal-control test problem that consists of finding the optimal initial perturbations to a turbulent temporal mixing layer such that mixing is improved at the end of a simulation time horizon T. The controls are further subject to a non-linear equality constraint on the total control energy. DNSs are used to resolve all turbulent scales of motion, and a continuous adjoint formulation is employed to calculate the gradient of the cost functionals. We compare the convergence speed of the SQP L-BFGS algorithm to a conventional non-linear conjugate gradient method (i.e. the current standard in DNS-based optimal control), and find that the SQP algorithm is more than an order of magnitude faster than the conjugate-gradient method.

Keywords: Sequential Quadratic Programming, damped limited-memory BFGS, Turbulent mixing layer, optimal control, Direct numerical Simulations, Adjoint equations

1. Introduction

The combination of Direct Numerical Simulations (DNS) or Large-Eddy Simulations (LES) and adjoint-based optimization approaches is becoming more common [1–5]. The numerical simulation of Navier–Stokes equations for turbulent flow, either using DNS or LES, yields three-dimensional time-varying turbulent flow fields, which represent in great detail the complex phenomena occurring in these type of flows. Since DNS (or LES) is computationally expensive, its combination with optimization, which requires a multitude of these simulations for function evaluations, is challenging. Nowadays, the optimization method of choice for DNS (or LES) and adjoint-based optimization is based on nonlinear conjugate-gradient algorithms [1–5]. These algorithms are robust and simple to handle in combination with the computational complexity introduced by solving Navier–Stokes and adjoint linearized Navier–Stokes equations in combination with large optimization parameter spaces. In the current work, we focus on the elaboration of a robust and fast Sequential Quadratic Programming (SQP) algorithm for optimization of turbulent flows with an equality constraint. We apply this to a turbulent test problem, i.e., optimal control of a turbulent mixing layer with DNS [3, 4].

[*]Corresponding author
Email address: johan.meyers@mech.kuleuven.be (Johan Meyers)
[1]Present address: Paul Scherrer Institut, 5232 Villigen PSI, Switzerland

In optimization of fluid mechanics applications, various optimization techniques are used. In particular when optimization parameter spaces are relatively small, a range of methods is used, including, e.g. artificial neural networks, evolutionary algorithms, etc. (cf., e.g., [6–8]). However, for (optimal) control cases in fluid mechanics for which optimization parameter spaces (also referred to as the control space) may easily span thousands of degrees of freedom, efficient algorithms such as one-shot methods, or gradient-based optimization approaches are required [9]. For very large problems (such as DNS-based optimization), for which the coupled optimality system is too large for a one-shot method, the only viable option is the use of a gradient-based approach. Moreover, as function evaluations using DNS are expensive, the gradients of the optimization cost functionals are obtained by solving a normal (forward) Navier–Stokes problem followed by an adjoint (backward) linearized Navier–Stokes problem [1, 10, 11], leading to a cost per cost-function gradient of roughly two regular DNSs (adjoint Navier–Stokes equations are, e.g., discussed in [10–15] among others).

Bewley *et al.* [1] were the first to perform optimal control in direct numerical simulation of turbulent flow. They employed a Polak–Ribière method, which is a non-linear Conjugate-Gradient (CG) variant [16–18], in combination with a Brent line-search method [19] to minimize the drag in optimal control of turbulent channel flow. In various later studies, the same algorithm was used . However, the number of function evaluations required for convergence can become very high. For instance, in [2] every CG step required approximately 20 function evaluations (forward or adjoint), totalling to 600 DNS equivalents for 35 CG steps performed in their optimization. In [3] and [4], every CG step required one adjoint and 8 forward simulations, and optimization up to 200 conjugate gradient iterations were performed. Hence, CG optimization is several orders of magnitude more expensive than a baseline DNS; a factor which is not easily recovered by further parallelization and partitioning of the computational DNS grid. Thus, even though conjugate-gradient methods may be very robust and easy to implement, the large number of function evaluations required based on expensive DNSs calls for more efficient and faster converging optimization algorithms.

Optimal control problems are often subject to equality and inequality constraints, e.g., the energy of the controls may be limited, etc. In previous DNS-based optimal control, this was usually circumvented by adding constraints on the control as a penalty in the cost functional [1, 2, 20, 21], though the result of the optimization then depends on the chosen penalty factor [2]. Delport *et al.* [3, 4] used a gradient-projection method to impose an equality constraint on the control energy. A theoretical drawback of the conjugate gradient projection method is that it is difficult to show that the projected direction is a descent direction, in contrast, e.g., to steepest descent methods where this is more straightforward [22].

In the current work, we investigate the use of SQP methods to solve equality constrained optimization problems with respect to turbulent flow simulations (with an equality constraint that imposes the total energy of the control variables, i.e., based on a L_2 norm). SQP methods apply Newton's method to the Karush–Kuhn–Tucker (KKT) conditions (i.e. the first-order optimality conditions for constrained optimization problems) to get the search directions when iteratively solving the equality constrained problem. This is equivalent to minimizing a quadratic programming sub-problem to get the new search direction.

In the past, SQP has been successfully applied to optimization in a number of 'non-turbulent' flow problems. For instance, Ghattas and Bark [23] applied SQP for steady flow problems around a cylinder and a sphere, while Heinkenschloss [24] applied SQP for optimal Dirichlet boundary control of steady Navier-Stokes problems. Hinze and Kunisch [25], Hintermüller and Hinze [26], and Ravindran [27] applied SQP for unsteady two-dimensional flows. However, in these studies, the main focus was on applying SQP on an optimization problem in both control and state space that is *constrained* by the Navier–Stokes equations, i.e., corresponding to a non-linear PDE-constraint on the state space. This is an alternative to an approach where the problem is reduced by implicitly expressing the state as a function of the controls (i.e. by solving the PDE), and using, e.g., a Newton-like method to solve the resulting unconstrained optimization problem (cf., e.g., [28]; a comparison of the merits of both approaches is, e.g., presented in Ref. [25]).

We choose to cast the optimization problem in its reduced form, using the Navier–Stokes equations to implicitly express the state as function of the controls in the cost functional, instead of explicitly keeping the PDE as a constraint in the optimization algorithm. To our knowledge, all earlier DNS-based optimal control studies (all using CG methods) follow this approach. The reason is that the state space in a DNS becomes very large. For instance, in the DNS test problem in the current work, the state space, i.e. the space–time turbulent flow solution, has order of 10^9 degrees of freedom (versus order of 10^4 for the control space); for many DNSs the dimensionality of the state space may be even a lot higher. Thus, the full solution space cannot be kept in memory, and has to be stored onto disk instead (cf.

Section 2.2 and 2.3 for details). By using the reduced cost function approach in the current mixing-layer optimal control problem, we mostly avoid vector operations on the full state space, as would be required when the PDE is kept explicitly as a constraint in the optimization algorithm. Thus, compared to earlier work on SQP for optimal control in fluid mechanics, the SQP algorithm in the current work is required for the additional non-linear energy constraint on the controls only. In absence of such a constraint, the remaining reduced cost-function approach would correspond to an unconstrained quasi-Newton method (i.e. L-BFGS, cf. below).

A remaining challenge for the SQP formulation of our optimal control problem is that the quadratic sub-problem contains the Hessian of the Lagrange function which is expensive to compute, and cannot be stored in memory when the number of control variables is high. The first issue (high computational cost) can be solved by employing the damped Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, which allows for a low cost approximation of the Hessian (cf. Ref. [18]). The damping is necessary for constrained optimization to provide a positive definite approximation of the Hessian at each iteration [40]. In order to circumvent the issue of storage, we formulate a limited-memory version of the damped BFGS method.

The resulting SQP damped L-BFGS optimization algorithm is first tested on a set of Rosenbrock type functions which are known to provide a challenging optimization benchmark. Subsequently, we test the method for the optimal control of a turbulent mixing layer studied earlier by Delport *et al.* [3, 4]. A further difficulty encountered is the robustness of the method in combination with direct numerical simulations. We find that initial step sizes obtained from initial guesses of the inverse Hessian in the damped L-BFGS method can lead to control variables that 'crash' the forward Navier–Stokes simulations. We solve this by introducing a region of feasibility for the controls based on the constraint surface, which is then used to properly scale the initial steps in the L-BFGS algorithm. Finally, a comparison of the SQP method with the prior CG algorithm in [3, 4] shows that the current SQP method is at least an order of magnitude faster than the CG method. An exact quantification is difficult as we lack the computational resources to converge our CG results up to the levels attained with the new SQP method.

The paper is organized as follows. First, in Section 2, the turbulent test problem related to optimal control of a turbulent mixing layer is briefly introduced. Subsequently, in Section 3, the SQP algorithm in combination with damped L-BFGS is discussed. In Section 4 we present results for the Rosenbrock test problem, and for the turbulent mixing layer. Conclusions are presented in Section 5.

2. Optimal control of turbulent flow: temporal mixing layer case

In the current section, we present the optimal control test problem that we use to evaluate the SQP algorithm. It consists of a temporal mixing layer, which evolves from a laminar shear layer with small spatially distributed perturbations into a fully turbulent mixing layer. The temporal mixing layer is a much studied theoretical test case that highlights a number of transition mechanisms of free shear layers (cf., e.g., [29–32]). In [3] this flow type was selected for its relative low cost when compared to direct numerical simulations of other turbulent flow types. In §2.1, the optimization problem formulation is briefly introduced. Adjoint-based determination of gradients is discussed in §2.2. Finally, discretization is discussed in §2.3.

2.1. Formulation of the optimal-control problem

The temporal mixing layer consists of a shear flow in a box with periodic boundary conditions in directions parallel to the shear (x_1 and x_2), and free-stream boundary conditions in the third direction x_3. The shear layer grows in time starting from an initial velocity profile

$$\boldsymbol{u}(\boldsymbol{x}, 0) = \tanh(x_3)\,\boldsymbol{e_1} + \boldsymbol{\phi}(\boldsymbol{x}) \tag{1}$$

with $\boldsymbol{u} = [u_1, u_2, u_3]$ the velocity field, $\boldsymbol{e_1}$ the unity vector in x_1-direction, and $\boldsymbol{\phi}$ small three-dimensional perturbations (with zero spatial mean) of the hyperbolic-tangent u_1 velocity field (sometimes, an "error function" is used as starting profile instead of a hyperbolic-tangent function [29–31]). Following common conventions for temporal mixing layers, all properties and equations in the current paper (including the initial profile in Eq. 1), are normalized using half of the velocity difference over the mixing layer $\Delta U/2$ as reference velocity, and half the initial mean vorticity thickness $\delta_\omega/2$ as reference length scale (with $\delta_\omega \equiv \Delta U/\max[\partial\langle u_1\rangle/\partial x_3]$).

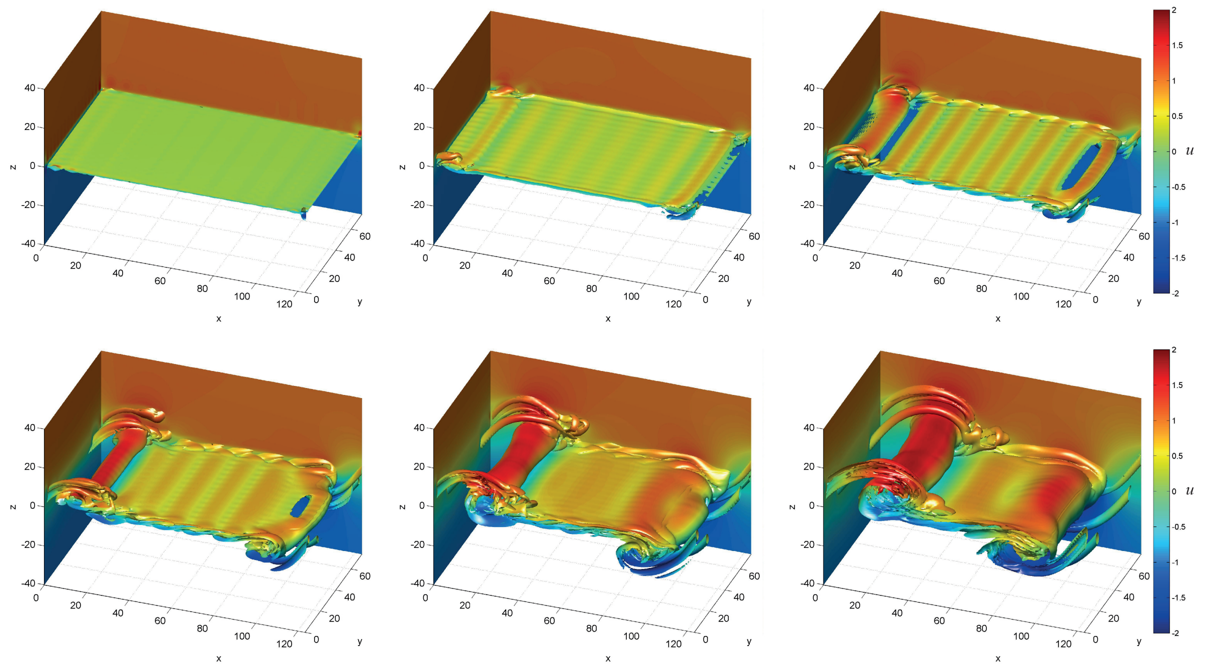
Figure 1: Typical evolution of a temporal mixing layer. Vertical planes $x = 0$, and $y = L_y$ colored by stream-wise velocity u_1, and iso-surface of vorticity (at $\omega = 0.9\max(\partial U/\partial y)$, with $\omega = \|\nabla\times\boldsymbol{u}\|$, and U the plane-average stream-wise velocity) colored by stream-wise velocity. From top left to bottom right: $t = 0, 20, 40, 60, 80, 100$. (In z-direction, solution is only displayed in the range $-40 \le z \le 40$.)

Temporal mixing layers are unconditionally linear unstable, and evolve fast into a three-dimensional turbulent flow. The dominant instability is the Kelvin–Helmholtz instability. In conventional temporal-mixing-layer simulations, perturbations $\boldsymbol{\phi}$ are often constructed by superimposing the most unstable (Kelvin–Helmholtz) eigenmode with a number of higher and lower harmonics, and some additional three-dimensional white noise [29, 30, 32]. The evolution of this initial condition into turbulence is then further governed by the Navier–Stokes equations, which for incompressible flows are given by

$$\nabla\cdot\boldsymbol{u} = 0, \tag{2}$$

$$\frac{\partial\boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} + \frac{1}{\rho}\nabla p - \frac{1}{Re}\nabla^2\boldsymbol{u} = 0, \tag{3}$$

with t the time, p the pressure, and Re the Reynolds number, and where the density ρ is constant. Following [3, 4], we employ $Re = 100$ in the current study. Further, the size of the computational domain Ω is chosen to be $L_1 = 8\lambda_1$ in the x_1 direction, with $\lambda_1 = 15.4$ the most unstable wavelength following from linear stability theory at $Re = 100$. The size of Ω in the x_2 direction corresponds to $L_2 = 8\lambda_2$, with $\lambda_2 = 0.6\lambda_1$, and where λ_2 corresponds to the wavelength of the most unstable span-wise disturbance on the rollers generated from the Kelvin–Helmholtz instability [33]. In the normal direction (x_3), we select a large domain size $L_3 = 60$ in order to prevent interactions between the boundary and the mixing region in the center of the domain around $x_3 = 0$. A typical evolution of a temporal mixing layer, using initial perturbations $\boldsymbol{\phi}$ based on the most unstable eigenmodes and some additional noise (cf. Ref. [4]), is shown in Figure 1. This figure illustrates the roll-up of and subsequent pairing of Kelvin-Helmholtz instabilities into large two-dimensional vortices. We refer to §2.3 for details on discretization, etc.

In [3, 4] an optimal-control problem was formulated in which various cost functionals of the form

$$\mathscr{J}[\boldsymbol{u}(\boldsymbol{x},T)] = \frac{1}{|\Omega|}\int_\Omega J[\boldsymbol{u}(\boldsymbol{x},T)]\,\mathrm{d}\boldsymbol{x}, \tag{4}$$

were optimized with respect to the perturbations $\boldsymbol{\phi}$. In this, J is an operator that is further specified below, and T is the optimization time horizon at which the DNS solution $\boldsymbol{u}$ on the computational domain Ω is evaluated, starting from an initial condition at time $t = 0$ with perturbations $\boldsymbol{\phi}$. In addition, $\boldsymbol{\phi}$ was subject to two constraints, namely it should be divergence-free, such that the initial condition $\boldsymbol{u}(\boldsymbol{x}, 0)$ satisfies continuity, and its energy level is fixed to a given value E_0. Hence, this leads to the optimal control problem [3, 4]

$$\begin{aligned} \min_{\boldsymbol{\phi}} \quad & \frac{1}{|\Omega|}\int_\Omega J[\boldsymbol{u}(\boldsymbol{x}, T, \boldsymbol{\phi})]\,\mathrm{d}\boldsymbol{x} \\ s.t. \quad & \nabla\cdot\boldsymbol{\phi} = 0 \\ & \frac{1}{2|\Omega|}\int_\Omega \boldsymbol{\phi}\cdot\boldsymbol{\phi}\,\mathrm{d}\boldsymbol{x} = E_0 \end{aligned} \tag{5}$$

Note that we explicitly add $\boldsymbol{\phi}$ as an argument to $\boldsymbol{u}(\boldsymbol{x}, T)$ to indicate its dependence on the initial perturbations. In fact, our optimization problem is here formulated without explicitly adding the Navier–Stokes equations as a constraint (cf. discussion in introduction). Instead, it is implicitly understood that $\boldsymbol{u}(\boldsymbol{\phi})$ is obtained as a solution of the Navier–Stokes equations (2,3). Further, in the current work, we employ $E_0 = 10^{-4}$, corresponding to the higher value used in Delport *et al.* [4].

In the current work, we consider two cost functionals from [3, 4] for the testing of the SQP method, with

$$J_E[\boldsymbol{u}] = \frac{1}{2}\boldsymbol{u}\cdot\boldsymbol{u}, \qquad \text{and} \tag{6}$$

$$J_M[\boldsymbol{u}] = \frac{1}{2}\langle\boldsymbol{u}\rangle\cdot\langle\boldsymbol{u}\rangle, \tag{7}$$

and where $\langle\boldsymbol{u}\rangle$ represents the volume average of $\boldsymbol{u}$ over the domain Ω. The corresponding cost functionals $\mathscr{J}_E$ and $\mathscr{J}_M$ minimize the total energy (subscript E) and the mean-flow kinetic energy (subscript M) respectively, at time horizon T. In [3] it was shown that optimization of these cost functionals leads to distinct mixing-layer behavior. In particular, $\mathscr{J}_E$ enhances mixing and fine-scale three-dimensional vortical structures at the time horizon, while $\mathscr{J}_M$ promotes large two-dimensional vortex structures.

2.2. Computation of the gradient

For the methods considered in this paper, gradient information is required. In order to describe how the gradients are computed, first the Gateau differential is introduced as (see, e.g., [22])

$$\delta\mathscr{J}[\boldsymbol{u}(\boldsymbol{x}, T, \boldsymbol{\phi}); \boldsymbol{\delta\phi}] \equiv \frac{\mathrm{d}}{\mathrm{d}\alpha}\mathscr{J}[\boldsymbol{u}(\boldsymbol{x}, T, \boldsymbol{\phi} + \alpha\boldsymbol{\delta\phi})] = \frac{1}{|\Omega|}\int_\Omega \frac{\partial J[\boldsymbol{u}(\boldsymbol{x}, T, \boldsymbol{\phi})]}{\partial\boldsymbol{u}}\cdot\boldsymbol{\delta u}\,\mathrm{d}\boldsymbol{x}, \tag{8}$$

with $\boldsymbol{\delta u}$ the sensitivity of the flow solution to changes $\boldsymbol{\delta\phi}$ on $\boldsymbol{\phi}$. The evaluation of the sensitivities $\delta\mathscr{J}$ by determining $\boldsymbol{\delta u}$ is very expensive when control with a large number of degrees of freedom is considered. Indeed, any possible change $\boldsymbol{\delta\phi}$ requires a DNS to determine $\boldsymbol{\delta u}$.

Instead, an adjoint formulation can be followed. Details of the underlying principles are, e.g., found in [10–13, 22, 34]. For the current optimization problem of a turbulent mixing layer the cost functional can be expressed as (cf. [3] for a detailed derivation)

$$\delta\mathscr{J}[\boldsymbol{u}(\boldsymbol{x}, T, \boldsymbol{\phi}); \boldsymbol{\delta\phi}] = \frac{1}{|\Omega|}\int_\Omega \boldsymbol{u}^*(\boldsymbol{x}, 0, \boldsymbol{\phi})\cdot\boldsymbol{\delta\phi}\,\mathrm{d}\boldsymbol{x}, \tag{9}$$

with $\boldsymbol{u}^*$ the solution of the adjoint linearized Navier–Stokes equations, linearized around the forward Navier–Stokes solution $\boldsymbol{u}(\boldsymbol{x}, t, \boldsymbol{\phi})$. The adjoint equations correspond to [3]

$$\nabla\cdot\boldsymbol{u}^* = 0, \tag{10}$$

$$-\frac{\partial\boldsymbol{u}^*}{\partial t} - \boldsymbol{u}\cdot\nabla\boldsymbol{u}^* - \boldsymbol{u}\cdot(\nabla\boldsymbol{u}^*)^T - \nabla p^* - \frac{1}{Re}\nabla^2\boldsymbol{u}^* = 0, \tag{11}$$

with p^* the adjoint pressure. These equations are solved for the current problem using periodic boundary conditions in x_1 and x_2, and symmetry boundary conditions in the x_3 direction. The equation is of parabolic nature in the backward time direction, with 'initial' condition at $t = T$ of [3]

$$\boldsymbol{u}^*(\boldsymbol{x}, T, \boldsymbol{\phi}) = \frac{\partial J[\boldsymbol{u}(\boldsymbol{x}, T, \boldsymbol{\phi})]}{\partial \boldsymbol{u}}. \tag{12}$$

Hence, based on one forward simulation to determine $\boldsymbol{u}(\boldsymbol{x}, T, \boldsymbol{\phi})$, and one backward simulation that determines $\boldsymbol{u}^*(\boldsymbol{x}, 0, \boldsymbol{\phi})$, the Gateau differential $\delta\mathscr{J}$ is determined through Eq. (9) with a cost of two simulations, independent of the number of degrees of freedom in the controls $\boldsymbol{\phi}$.

Finally, given the Gateau differential, the gradient of the cost functional may be determined. To this end, we first define the function space for the controls $\boldsymbol{\phi}$. If we choose $\boldsymbol{\phi} \in H^1(\Omega)$, i.e. the Sobolev space of order one in the domain Ω (which is, e.g., smoother than an L_2 space, and better ensures regularity of the optimal solution), it follows from the Riesz representation theorem for the derivative that the gradient $\boldsymbol{g} = \nabla\mathscr{J}$ is expressed as [22]

$$\delta\mathscr{J}[\boldsymbol{u}(\boldsymbol{x}, T, \boldsymbol{\phi}); \boldsymbol{\delta\phi}] = (\boldsymbol{g}, \boldsymbol{\delta\phi})_{H^1} \qquad \forall\, \boldsymbol{\delta\phi} \in H^1(\Omega). \tag{13}$$

Hence, using Eq. (9) in combination with the definition of the H^1 inner product, we find

$$\begin{aligned} \frac{1}{|\Omega|} \int_\Omega \boldsymbol{u}^*(\boldsymbol{x}, 0, \boldsymbol{\phi}) \cdot \boldsymbol{\delta\phi}\, \mathrm{d}\boldsymbol{x} &= \int_\Omega \boldsymbol{g} \cdot \boldsymbol{\delta\phi} + (\nabla\boldsymbol{g}) \cdot (\nabla\boldsymbol{\delta\phi})\, \mathrm{d}\boldsymbol{x} && (14)\\ &= \int_\Omega \boldsymbol{g} \cdot \boldsymbol{\delta\phi} - \nabla^2\boldsymbol{g} \cdot \boldsymbol{\delta\phi}\, \mathrm{d}\boldsymbol{x} \qquad \forall\, \boldsymbol{\delta\phi} \in H^1(\Omega), && (15) \end{aligned}$$

where the second equality follows from partial integration, and the boundary conditions for $\boldsymbol{\phi}$. Hence, we find the gradient of the cost functional to be a solution of the elliptic equation $\boldsymbol{g} - \nabla^2\boldsymbol{g} = \boldsymbol{u}^*/|\Omega|$. This corresponds to applying a second-order differential low-pass filter to $\boldsymbol{u}^*$. Instead of solving this elliptic partial differential equation, we follow a more pragmatic approach for the regularization of $\boldsymbol{u}^*$ that was also used in Refs. [3, 4] (this allows us to compare the SQP method elaborated in the current work with the CG method used in Refs. [3, 4] – cf. Section 4.2). First of all, in periodic directions, we restrict $\boldsymbol{\phi}$, and $\boldsymbol{\delta\phi}$ to wavelengths larger or equal to 1/16th of the box size. This effectively amounts to the use of a spectral cut-off filter in these directions. In the normal direction, we approximate the above second-order differential filter with a Gauss filter [3].

2.3. Discretization and computational set-up

A mixed pseudo-spectral finite volume code is used to discretize both the forward Navier–Stokes and the adjoint equations [3, 4]. In the two periodic directions, a pseudo-spectral Fourier discretization is used, and dealiasing is performed with the 2/3rd dealiasing rule [35]. The normal direction is discretized using a fourth-order energy-conserving finite-volume scheme [36], where the position of the normal velocities u_3 are shifted by half a cell in the normal direction, while u_1, u_2, and p are kept in the center of the cell. Continuity is imposed by solving a Poisson equation for the pressure using a direct solver. A fourth order explicit Runge–Kutta scheme is used to integrate the system of equations in time. The convective and diffusive Courant–Friedrichs–Lewy (CFL) numbers are set to 0.2. All simulations are performed on a N=128×128×256 mesh. The mesh is stretched in the normal direction with grid spacing $\Delta\, x_3$ proportional to $|x_3|^{1/4}$ [4].

For the formulation of the adjoint equations, we follow the continuous approach, in which the adjoint equations are first formulated before they are discretized (another approach is to first discretize the Navier–Stokes equations, and then linearize and formulate the discrete adjoint equations – cf., e.g. [12, 15]). We discretize the adjoint equations using the same method as the forward equations, and on the same grid. The equations are integrated backward in time, and require the solution $\boldsymbol{u}(\boldsymbol{x}, t, \boldsymbol{\phi})$ from the forward solution (cf. Eq. 11). To this end, the solution is stored on disk at every time step during the forward simulation.

A consequence of the discretization is that $\boldsymbol{\phi} \in H^1(\Omega)$ is mapped onto a discrete grid function $\boldsymbol{\phi}_d \in \mathbb{R}^n$, with $n = 3 \times N$. Following [4], N =32×32×256, i.e., in parallel directions $\boldsymbol{\phi}$ is restricted to wavelengths larger or equal to 1/16th of the box size (cf. also discussion at end of §2.2). The continuity constraint (Eq. 5) is a linear constraint that after discretization can be used to define a set of independent parameters $\boldsymbol{\psi}_d \in \mathbb{R}^m$, such that $\boldsymbol{\phi}_d = \boldsymbol{M} \cdot \boldsymbol{\psi}_d$ satisfies the discretized continuity equations, and with $m = 2 \times N$ (for details on $\boldsymbol{M}$, we refer to [3]). The resulting optimization problem after elimination of the continuity constraint then corresponds to

$$\begin{aligned} \min_{\boldsymbol{\psi}_d \in \mathbb{R}^m} \quad & \mathscr{J}_d[\boldsymbol{u}_d(\boldsymbol{M} \cdot \boldsymbol{\psi}_d)], \\ s.t. \quad & C(\boldsymbol{\psi}_d) = E_0 - \frac{1}{2|\Omega|} \boldsymbol{\psi}_d^T \boldsymbol{M}^T \boldsymbol{V} \boldsymbol{M} \boldsymbol{\psi}_d = 0, \end{aligned} \tag{16}$$

where $\mathscr{J}_d$ is the discrete approximation of $\mathscr{J}$ based on the discretized velocity field $\boldsymbol{u}_d$, and where $\boldsymbol{V}$ is a diagonal matrix with the cell volumes as diagonal elements. Further details are found in [3, 4].

3. Sequential Quadratic Programming with damped L-BFGS

We now focus on the optimization algorithm based on Sequential Quadratic Programming damped L-BFGS employed in the current work. First in §3.1, the formulation and solution of the quadratic programming problem is briefly reviewed. Subsequently, we discuss a damped limited-memory BFGS algorithm in §3.2. Next, in §3.3 we discuss some issues related to the initialization of the L-BFGS algorithm, and finally, in §3.4 the selection of step sizes using an L_1 merit function is further discussed.

In order to formulate the algorithm, we introduce the Lagrange functional associated with the constrained optimization problem (16), i.e.

$$\mathscr{L}(\boldsymbol{\psi}, \lambda) = \mathscr{J}[\boldsymbol{u}(\boldsymbol{M} \cdot \boldsymbol{\psi})] - \lambda C(\boldsymbol{\psi}), \tag{17}$$

with λ the Lagrange multiplier. The subscripts 'd' in $\mathscr{J}_d$, and $\boldsymbol{\psi}_d$ are dropped to keep notation sober. The Jacobian matrix of the constraint is defined as

$$\boldsymbol{A}(\boldsymbol{\psi})^T = \nabla C(\boldsymbol{\psi}). \tag{18}$$

Remark that in our case, with only one constraint, $\boldsymbol{A}\boldsymbol{A}^T$ is a scalar. Also note that the SQP algorithm is straightforward to elaborate with multiple constraints, and $\boldsymbol{A}(\boldsymbol{\psi})^T = [\nabla C_1(\boldsymbol{\psi}), \nabla C_2(\boldsymbol{\psi}), \cdots]$, but that is not necessary in the current case.

3.1. Step direction via Quadratic Programming

The SQP algorithm is briefly reviewed in the context of the current equality constrained optimization problem. More information on SQP algorithms may be found in [18, 37–39], amongst others. The SQP algorithm is an iterative procedure for solving nonlinear constrained optimization problems. Given an intermediate estimate of the optimum $\boldsymbol{\psi}_k$ during the kth step of the procedure, it determines the step direction $\boldsymbol{\chi}_k$ by solving a quadratic sub-problem. The latter is derived using a second-order Taylor approximation of $\mathscr{L}$, and a linearization of C around the point $(\boldsymbol{\psi}_k, \lambda_k)$. In the present study, we cast this quadratic problem into following standard form

$$\min_{\boldsymbol{\chi}} \quad \frac{1}{2}\boldsymbol{\chi}^T \nabla^2 \mathscr{L}(\boldsymbol{\psi}_k, \lambda_k)\boldsymbol{\chi} + (\nabla \mathscr{J}[\boldsymbol{u}(\boldsymbol{M} \cdot \boldsymbol{\psi}_k)])^T \boldsymbol{\chi} + \mathscr{J}[\boldsymbol{u}(\boldsymbol{M} \cdot \boldsymbol{\psi}_k)] \tag{19a}$$

$$s.t. \quad A(\boldsymbol{\psi}_k)\boldsymbol{\chi} + C(\boldsymbol{\psi}_k) = 0 \tag{19b}$$

The Lagrange functional of this quadratic sub-problem is expressed as

$$\mathscr{L}'(\boldsymbol{\chi}, \omega) = \frac{1}{2}\boldsymbol{\chi}^T \nabla^2 \mathscr{L}_k \boldsymbol{\chi} + (\nabla \mathscr{J}_k)^T \boldsymbol{\chi} + \mathscr{J}_k - \omega \left(\boldsymbol{A}_k \boldsymbol{\chi} + C_k\right), \tag{20}$$

with ω the Lagrange multiplier, and where we introduced the short-hand notations $\mathscr{L}_k = \mathscr{L}(\boldsymbol{\psi}_k, \lambda_k)$, $\mathscr{J}_k = \mathscr{J}[\boldsymbol{u}(\boldsymbol{M} \cdot \boldsymbol{\psi}_k)]$, $\boldsymbol{A}_k = \boldsymbol{A}(\boldsymbol{\psi}_k)$, and $C_k = C(\boldsymbol{\psi}_k)$. From this, the KKT conditions, written in matrix form, are derived as

$$\begin{bmatrix} \nabla^2 \mathscr{L}_k & -A_k^T \\ -A_k & 0 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\chi}_k \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} -\nabla \mathscr{J}_k \\ C_k \end{bmatrix}, \tag{21}$$

further using the result $\omega = \lambda_{k+1}$. The derivation of (21) is quite lengthy, but may, e.g., be found in Ref [18].

The matrix on the left hand side of Eq. (21) should be nonsingular in order to have a solution for the QP problem. This is satisfied if the following conditions hold in the point $(\boldsymbol{\psi}_k, \lambda_k)$:

1. The constraint Jacobian $\boldsymbol{A}_k$ has full row rank, and
2. $\boldsymbol{d}^T \nabla^2 \mathscr{L}_k \, \boldsymbol{d} > 0$ for $\forall \boldsymbol{d} \neq 0$, with $\boldsymbol{A}_k \boldsymbol{d} = 0$.

In the current study, the first condition is trivially satisfied as long as $\boldsymbol{\psi}_k \neq 0$, i.e. there is only one non-linear constraint, of which the gradient is easy to verify (cf. Eq. 16).

The second condition is an important constraint to consider when constructing approximations to the Hessian of the Lagrange functional. For large problems, as encountered in the current study, it is impossible to compute and even store the full Hessian of the Lagrange functional. Instead, using a damped L-BFGS method, further discussed below in §3.2, we will approximate

$$\left[\nabla^2 \mathscr{L}_k\right]^{-1} \approx \boldsymbol{H}_k, \tag{22}$$

and by construction, $\boldsymbol{d}^T \boldsymbol{H}_k^{-1} \, \boldsymbol{d} > 0$ $(\forall \boldsymbol{d} \neq 0)$ will be enforced.

Since both conditions discussed above are satisfied, Eq. (21) can be solved as

$$\lambda_{k+1} = (\boldsymbol{A}_k \boldsymbol{H}_k \boldsymbol{A}_k^T)^{-1} (\boldsymbol{A}_k \boldsymbol{H}_k \nabla \mathscr{J}_k - C_k), \tag{23}$$

$$\boldsymbol{\chi}_k = -\boldsymbol{H}_k \nabla \mathscr{J}_k + \boldsymbol{T} (\boldsymbol{A}_k \boldsymbol{H}_k \nabla \mathscr{J}_k - C_k), \tag{24}$$

where we introduced the projection matrix $\boldsymbol{T} = \boldsymbol{H}_k \boldsymbol{A}_k^T (\boldsymbol{A}_k \boldsymbol{H}_k \boldsymbol{A}_k^T)^{-1}$.

The calculation of $\boldsymbol{\chi}_k$ and λ_{k+1} requires the computation of

1. $\boldsymbol{H}_k \nabla \mathscr{J}_k$ and $\boldsymbol{H}_k \boldsymbol{A}_k^T$. In both cases, the product $\boldsymbol{H}_k$ with a vector is required. Instead of storing $\boldsymbol{H}_k$ itself and directly performing this product, a two-loop recursion algorithm is used in combination with the L-BFGS algorithm (cf. §3.2).
2. the inverse: $(\boldsymbol{A}_k \boldsymbol{H}_k \boldsymbol{A}_k^T)^{-1}$, appearing in the projection matrix $\boldsymbol{T}$. Since the the number of constraints in the current study equals one, this operation is trivial as the product $\boldsymbol{A}_k \boldsymbol{H}_k \boldsymbol{A}_k^T$ results in a scalar number.

Hence, the step direction $\boldsymbol{\chi}_k$ can be computed, provided we are able to construct matrix-vector products with $\boldsymbol{H}_k$. This is further discussed in next subsection. The selection of the step size is discussed in §3.4.

3.2. Updating the approximated inverse Hessian of the Lagrange functional

In the current section we discuss the construction of a limited-memory BFGS approximation of the inverse of the Hessian of the Lagrange functional (i.e. cf. Eq. 22). We inspire our algorithm on the damped BFGS algorithm proposed by Powell [40].

First, we review some aspects of the conventional L-BFGS approach, as used for unconstrained problems. In that case $\boldsymbol{H}_k$ represents an approximation to the inverse of the Hessian of the cost functional (since there are no constraints, a Lagrange functional does not need to be formulated). $\boldsymbol{H}_k$ is then constructed using the following recursion relation (cf. Ref. [18])

$$\boldsymbol{H}_{k+1} = \boldsymbol{V}_k^T \boldsymbol{H}_k \boldsymbol{V}_k + \rho_k \boldsymbol{s}_k \boldsymbol{s}_k^T, \tag{25}$$

with

$$\boldsymbol{V}_k = \boldsymbol{I} - \rho_k \boldsymbol{y}_k \boldsymbol{s}_k^T, \qquad \rho_k = \frac{1}{\boldsymbol{y}_k^T \boldsymbol{s}_k}, \tag{26}$$

and

$$\boldsymbol{s}_k = \boldsymbol{\chi}_{\boldsymbol{k+1}} - \boldsymbol{\chi}_k, \qquad \boldsymbol{y}_k = \nabla \mathscr{J}_{k+1} - \nabla \mathscr{J}_k. \tag{27}$$

In order to guarantee that $\boldsymbol{H}_{k+1}$ is positive definite (given that $\boldsymbol{H}_k$ is positive definite), which is required in order to get a descent direction for the objective functional in the unconstrained problem, the "curvature condition" should hold,

$$\boldsymbol{s}_k^T \boldsymbol{y}_k > 0. \tag{28}$$

It can be shown that this condition indeed holds provided that the strong Wolfe conditions are imposed for the determination of the step size. The resulting positive-definite construction of H_k has been shown to be very robust, cf. Ref [18] for further details. The startup of the algorithm further uses $\boldsymbol{H}_0 = \gamma_0 \boldsymbol{I}$, with $\boldsymbol{I}$ the identity matrix, and γ_0 a well-chosen strictly positive number (cf. section 3.3 for a discussion on the selection of γ_0).

For the constrained L-BFGS case, the quadratic programming sub-problem is based on the Lagrange functional, and an approximation H_k to the inverse of the Hessian of the Langrange functional is needed. In that case

$$\boldsymbol{y}_k = \nabla\mathscr{L}(\boldsymbol{\psi}_{k+1}, \lambda_{k+1}) - \nabla\mathscr{L}(\boldsymbol{\psi}_k, \lambda_{k+1}) \tag{29}$$

However, using this definition, and the conventional L-BFGS construction for $\boldsymbol{H}_k$, the curvature condition may not hold. For constrained BFGS, Powell [40] suggested a solution to this problem, called damped BFGS. His approach is based on the Hessian of the Lagrangian, instead of on its inverse. In the current work, we follow the same idea, but formulated in terms of $\boldsymbol{H}_k$. In a first step, define

$$\boldsymbol{r}_k = \theta_k\boldsymbol{s}_k + (1-\theta_k)\boldsymbol{H}_k\boldsymbol{y}_k, \tag{30}$$

with $\boldsymbol{s}_k = \boldsymbol{\chi_{k+1}} - \boldsymbol{\chi}_k$, $\boldsymbol{y}_k = \nabla\mathscr{L}_{k+1} - \nabla\mathscr{L}_k$, and where the scalar θ_k is defined as

$$\theta_k = \begin{cases} 1 & \text{for } \boldsymbol{s}_k^T\boldsymbol{y}_k \geq 0.2\boldsymbol{y}_k^T\boldsymbol{H}_k\boldsymbol{y}_k, \\ \dfrac{(0.8\boldsymbol{y}_k^T\boldsymbol{H}_k\boldsymbol{y}_k)}{(\boldsymbol{y}_k^T\boldsymbol{H}_k\boldsymbol{y}_k - \boldsymbol{s}_k^T\boldsymbol{y}_k)} & \text{for } \boldsymbol{s}_k^T\boldsymbol{y}_k < 0.2\boldsymbol{y}_k^T\boldsymbol{H}_k\boldsymbol{y}_k. \end{cases} \tag{31}$$

Using the definitions in Eqs. (30), and (31), a recursion relation for $\boldsymbol{H}_k$ is given by

$$\boldsymbol{H}_{k+1} = \boldsymbol{V}_k^T\boldsymbol{H}_k\boldsymbol{V}_k + \rho_k\boldsymbol{r}_k\boldsymbol{r}_k^T, \tag{32}$$

with

$$\rho_k = \frac{1}{\boldsymbol{r}_k^T\boldsymbol{y}_k}, \qquad \boldsymbol{V}_k = \boldsymbol{I} - \rho_k\boldsymbol{r}_k\boldsymbol{y}_k^T. \tag{33}$$

Hence, this mimics standard L-BFGS, but with $\boldsymbol{s}_k$ replaced by $\boldsymbol{r}_k$, and with $\boldsymbol{y}_k$ defined using the Lagrangian functional. For this version of damped L-BFGS it is possible to show that $\boldsymbol{H}_{k+1}$ is positive definite when $\boldsymbol{H}_k$ is positive definite. We refer to Appendix A for a short proof.

Finally, in the limited-memory version of the BFGS algorithm, $\boldsymbol{H}_k$ is not explicitly stored, since it is too large. Instead an approximation $\widetilde{\boldsymbol{H}}_k$ is stored implicitly using the q vector pairs $\boldsymbol{r}_{k-i}$, $\boldsymbol{y}_{k-i}$ $(i = 1\cdots q)$. To that end, the recursion relation (32) may be explicitly expressed for the last q steps, leading to

$$\boldsymbol{H}_k = \boldsymbol{W}_q^T\boldsymbol{H}_{k-q}\boldsymbol{W}_q + \sum_{i=2}^{q}\rho_{k-i}\boldsymbol{W}_{i-1}^T\boldsymbol{r}_{k-i}\boldsymbol{r}_{k-i}^T\boldsymbol{W}_{i-1} + \rho_{k-1}\boldsymbol{r}_{k-1}\boldsymbol{r}_{k-1}^T, \quad \text{with } \boldsymbol{W}_i = \prod_{j=1}^{i}\boldsymbol{V}_{k-j}. \tag{34}$$

For the limited-memory version, $\boldsymbol{H}_{k-q}$ is now simply replaced by $\boldsymbol{H}_k^0 = \gamma_k\boldsymbol{I}$, where (simillar to Ref. [18], but in terms of r_k instead of s_k)

$$\gamma_k = r_{k-1}^T y_{k-1}/(y_{k-1}^T y_{k-1}), \tag{35}$$

so that

$$\boldsymbol{H}_k \approx \widetilde{\boldsymbol{H}}_k = \gamma_k\boldsymbol{W}_q^T\boldsymbol{W}_q + \sum_{i=2}^{q}\rho_{k-i}\boldsymbol{W}_{i-1}^T\boldsymbol{r}_{k-i}\boldsymbol{r}_{k-i}^T\boldsymbol{W}_{i-1} + \rho_{k-1}\boldsymbol{r}_{k-1}\boldsymbol{r}_{k-1}^T. \tag{36}$$

The approximation $\widetilde{\boldsymbol{H}}_k$ nor the matrices $\boldsymbol{W}$ or $\boldsymbol{V}$ are ever explicitly calculated, i.e., they have the same size and storage requirements as $\boldsymbol{H}_k$. However, in the SQP algorithm, only products of $\widetilde{\boldsymbol{H}}_k$ with a vector are required, which allows to avoid the storage of these full matrices. In fact, it is appreciated from Eqs. (33,36) that all matrices $\boldsymbol{V}$ (and consequently $\boldsymbol{W}$) are constructed by outer products of the form $\boldsymbol{r}\boldsymbol{y}^T$, and $\boldsymbol{r}\boldsymbol{r}^T$. Hence, if a matrix-vector product is elaborated (with a vector $\boldsymbol{c}$), this would lead to expressions of the form $\boldsymbol{r}\boldsymbol{y}^T\boldsymbol{c}$, and $\boldsymbol{r}\boldsymbol{r}^T\boldsymbol{c}$. In these, the second (inner) product can be elaborated first, leading to a scalar, before the first (outer) product is calculated. Further elaboration leads to the so-called two-loop recursion algorithm (details are found in [18]). In Ref. [18], values for q between 3 and 20 are recommended. For further practical elaboration in the current study we make a somewhat ad hoc choice of $q = 5$ that is on the low side of this range, so that the total memory overhead remains limited.

3.3. Initialization of the L-BFGS algorithm

For the initialization of the L-BFGS algorithm at $k = 0$, γ_0 cannot be evaluated using Eq. (35), so instead $\gamma_0 = 1$ is often used. In fact, convergence of modern Broyden-like methods is usually insensitive to the initial matrix in the algorithm (cf. Ref. [41]). However, using $\gamma_0 = 1$ for the turbulent mixing layer optimal control problem, we obtained an initial step that led to a solver crash. We observed that the initial search step $\boldsymbol{\chi_0}$ calculated from the quadratic programming was giving a point $\boldsymbol{\psi_0} + \boldsymbol{\chi_0}$ that was by orders of magnitude too far off from the constraint surface to allow for a feasible Navier–Stokes solution. This is not so much related to convergence properties of the L-BFGS method, but rather to the computational robustness of the Navier–Stokes solver.

To remedy this, we determine γ_0 so that it gives a point $\boldsymbol{\psi_0} + \boldsymbol{\chi_0}$ that is sufficiently close to constraint surface. To that end, we require $C(\boldsymbol{\psi_0} + \boldsymbol{\chi_0}) \leq aE_0$, with a user-selected constant a. Further on, we shall simply take $a = 1$. Since C is a quadratic constraint (cf. Eq. 16), we find

$$C(\boldsymbol{\psi}_0 + \boldsymbol{\chi}_0) = C(\boldsymbol{\psi}_0) + \boldsymbol{A}_0\boldsymbol{\chi}_0 + \frac{1}{2}\boldsymbol{\chi}_0^T(\nabla^2 C)\boldsymbol{\chi}_0 \leq aE_0. \tag{37}$$

To further simplify this expression we use $C(\boldsymbol{\psi}_0) + \boldsymbol{A}_0\boldsymbol{\chi}_0 = 0$, since the search direction in SQP satisfies the linearized constraint (cf. Eq. 19a). Moreover, we apply the rough approximation $\nabla^2 C \approx \boldsymbol{I}$. This leads to

$$\boldsymbol{\chi_0}^T\boldsymbol{\chi_0} \leq 2aE_0. \tag{38}$$

Finally, $\boldsymbol{\chi_0}$ is given by Eq. (24) with $\boldsymbol{H}_0 = \gamma_0\boldsymbol{I}$. Inserting in Eq. (38) yields

$$\gamma_0^2\boldsymbol{g}^T\boldsymbol{g} \leq 2aE_0, \tag{39}$$

with $\boldsymbol{g} = (-\nabla\mathscr{J}_0 + \boldsymbol{A}_0^T\lambda_1)$, and $\lambda_1 = (\boldsymbol{A}_0\boldsymbol{A}_0^T)^{-1}\boldsymbol{A}_0\nabla\mathscr{J}_0$. Hence, we select the initial γ-value as

$$\gamma_0 = \frac{\sqrt{2aE_0}}{||\boldsymbol{g}||}. \tag{40}$$

3.4. Global convergence via the use of a merit function

It is possible to show that the SQP method corresponds to Newton's method applied to the KKT conditions [18]. Therefore, at least locally, it converges quadratically from a point which is close to the solution. To enforce convergence from any starting point, a line search method with a merit function is used. The merit function is a scalar-valued function that aims to minimize the objective functional while maintaining the constraint violations as low as possible. To that end, Han [42], and Powell [40] combine an SQP algorithm with an L_1 merit function

$$\varphi(\boldsymbol{\psi}, \mu) = \mathscr{J} + \mu\, ||C(\boldsymbol{\psi})||_1\,, \tag{41}$$

with μ a penalty parameter. Given an intermediate estimate of the optimum $\boldsymbol{\psi}_k$ from the kth step of the procedure, the line search method determines $\boldsymbol{\chi_{k+1}}$ such that

$$\boldsymbol{\psi}_{k+1} = \boldsymbol{\psi}_k + \alpha\boldsymbol{\chi}_k. \tag{42}$$

The step length α is determined based on a 'sufficient decrease' condition, i.e. the so-called Armijo condition [43]:

$$\varphi(\boldsymbol{\psi}_k + \alpha\boldsymbol{\chi}_k, \mu_k) \leq \varphi(\boldsymbol{\psi}_k, \mu_k) + \eta\alpha\, \delta\varphi[(\boldsymbol{\psi}_k, \mu_k); \boldsymbol{\chi}_k]\,, \qquad \text{and} \quad \eta \in (0, 1), \tag{43}$$

with the directional derivative in the point $\boldsymbol{\psi}_k$ and for direction $\boldsymbol{\chi}_k$ given by

$$\delta\varphi[(\boldsymbol{\psi}_k, \mu_k); \boldsymbol{\chi}_k] = \delta\mathscr{J}\,[\boldsymbol{\psi}_k; \boldsymbol{\chi}_k] - \mu_k\, ||C_k||_1\,. \tag{44}$$

We use a value of $\eta = 10^{-4}$ as suggested in [18]. In Eq. (44), the value of μ_k still needs to be specified. A sufficient condition for $\boldsymbol{\chi}_k$ to be a descent direction of the merit function is to select $\mu_k > |\lambda_{k+1}|$ [42], with λ_{k+1} coming from the solution of the SQP problem above. However, it has been observed that the performance of the merit function is sensitive to the choice of the weights μ_k [44]. Therefore we adopt Byrd *et al.*'s [44] proposal to set

$$\mu_k = \omega_k(|\lambda_{k+1}| + \delta), \qquad \text{with} \quad \omega_k = \max\left(1; \frac{2\,\delta\mathscr{J}\,[\boldsymbol{\psi}_k;\boldsymbol{\chi}_k]}{(|\lambda_{k+1}| + \delta)|C_k|}\right), \tag{45}$$

with δ set to 10^{-4}.

Finally, the step length α is then iteratively determined. It is initialized with $\alpha_0 = 1$, corresponding to the Newton value. In many cases this directly satisfies the Armijo condition (43). If not, α_i is iteratively updated, following the strategy described by Powell [40]. To that end, the minimum $\bar{\alpha}_i$ (normalized by α_i) of a quadratic interpolation using the points $\varphi_0 = \varphi(\boldsymbol{\psi}_k, \mu_k)$, $\varphi_\alpha = \varphi(\boldsymbol{\psi}_k + \alpha_i\boldsymbol{\chi}_k, \mu_k)$, and the directional derivative $\delta\phi[(\boldsymbol{\psi}_k, \mu_k);\boldsymbol{\chi}_k]$, is formulated. Hence,

$$\bar{\alpha}_i = \frac{1}{2}\frac{\alpha_i\delta\varphi}{\alpha_i\delta\varphi - \varphi_\alpha + \varphi_0}. \tag{46}$$

The next iterate α_{i+1}, is then formulated as $\alpha_{i+1} = \alpha_i \max[0.1, \min(0.9, \bar{\alpha}_i)]$, where the min-max construction is used to choose iterates that are not too close to the previous step α_i, or too close to zero.

4. Results and discussion

In a first step, in §4.1 we test the current damped L-BFGS SQP algorithm on a Rosenbrock function. Next, we apply the algorithm to the turbulent mixing layer case in §4.2, and we compare with a conventional non-linear conjugate gradient method.

4.1. Optimization of Rosenbrock function

The extended Rosenbrock problem is a well-known benchmark that serves as a good verification case for optimization algorithms [45, 46]. The problem is defined as

$$\min_{\boldsymbol{x}} \qquad f(\boldsymbol{x}) = \sum_{i=1}^{i=n/2}\left(x_{2i} - x_{2i-1}^2\right)^2 + (1 - x_{2i-1})^2, \qquad \boldsymbol{x} \in \mathbb{R}^n \tag{47}$$

$$\text{s.t} \qquad \sum_{i=1}^{i=n} x_i^2 - n = 0. \tag{48}$$

In the current work, we choose $n = 50000$ to mimic optimization in large parameter spaces. The global minimum of the problem corresponds with $\boldsymbol{x}^* = [1, 1, ..., 1]$.

To test convergence, we initialize the algorithm with a number different starting points, as listed in Table 1, and in Table 2. The construction of the starting points is rather ad hoc: in Table 1, we include a number of deterministic points, with different distances from the optimum, while in Table 2, a number of randomly generated starting points are included. For all the starting points, γ_0 is set to 1 in the L-BFGS method (cf. §3.3). The number of SQP iterations is also listed. We base convergence on the norms of the gradient of the Lagrangian and of the constraint $\leq 10^{-9}$. In all cases the global optimum $\boldsymbol{x}^*$ is found.

Table 1, and 2 also include the total number of function and gradient evaluations to reach the optimum. As these are particulary expensive in the context of DNS (either requiring a Navier–Stokes simulation, or an adjoint simulation), this total needs to remain as low as possible. For the starting points in Table 1, it is appreciated that the total number of evaluations ranges between 20 and 179. Clearly, the initial locations of the random-generated starting points in Table 1 are more challenging. Here we find that up to 168 evaluations are required for $0 \leq x_i^0 \leq 1$. This increases to up to 378 for $0 \leq x_i^0 \leq 1000$.

It is also well documented that Quasi-Newton SQP algorithms display superlinear convergence near the optimum [18]. Super-linear convergence is characterized by a Convergence Rate ratio

$$CR = \frac{||\boldsymbol{x}_{k+1} - \boldsymbol{x}^*||}{||\boldsymbol{x}_k - \boldsymbol{x}^*||} \tag{49}$$

Starting Point	SQP iterations	sum of function and gradient evaluations
$x_i^0 = 2$	9	20
$x_i^0 = 5$	11	25
$x_i^0 = 10$	15	37
$x_i^0 = 30$	38	95
$x_i^0 = 50$	20	51
$x_i^0 = 90$	30	76
$x_i^0 = 120$	20	52
$x_i^0 = 1111$	27	73
$x_i^0 = 2000$	56	154
$x_i^0 = 4786$	52	179
$x^0_{1\to n/2} = 7, x^0_{n/2+1\to n} = 7^2$	28	65
$x^0_{1\to n/2} = 10, x^0_{n/2+1\to n} = 10^2$	50	128
$x^0_{1\to n/2} = 35, x^0_{n/2+1\to n} = 35^2$	54	146
$x^0_{1\to n/2} = 68, x^0_{n/2+1\to n} = 68^2$	62	168
$x^0_{1\to n/2} = 89, x^0_{n/2+1\to n} = 89^2$	55	146

Table 1: Number of SQP iterations, and sum of function and gradient evaluations for the extended Rosenbrock problem, using different starting points $\boldsymbol{x}_0$ (with elements x_i^0, $i = 1 \cdots n$)

	Iterations	Evaluations	Iterations	Evaluations	Iterations	Evaluations
Starting Point	$0 \le x_i^0 \le 1$		$0 \le x_i^0 \le 100$		$0 \le x_i^0 \le 1000$	
seed = 120655	71	146	127	283	160	364
seed= 2802505	81	168	108	236	138	323
seed= 95012783	70	147	106	230	164	378

Table 2: Number of SQP iterations and function/gradient evaluations for randomly generated starting points, generated using Fortran's intrinsic random function, and three different seeds. Each of the random series is scaled such that $0 \le x_i^0 \le 1$; $0 \le x_i^0 \le 100$; or $0 \le x_i^0 \le 1000$; leading to 9 different starting points.

$\boldsymbol{x}_0$: $x_i^0 = 2$		$\boldsymbol{x}_0$: $x_i^0 = 50$		$\boldsymbol{x}_0$: $x_i^0 = 4786$	
$\|\boldsymbol{x}_k - \boldsymbol{x}^*\|$	CR	$\|\boldsymbol{x}_k - \boldsymbol{x}^*\|$	CR	$\|\boldsymbol{x}_k - \boldsymbol{x}^*\|$	CR
2.236E+02		1.014E+02		1.286E+01	
2.421E+02	1.083E+00	8.102E+01	3.761E-01	1.374E+01	1.069E+00
6.663E+01	2.752E-01	3.048E+01	1.009E+00	1.045E+01	7.601E-01
7.841E+00	1.177E-01	3.074E+01	3.733E-01	4.421E+00	4.232E-01
1.545E+00	1.970E-01	1.148E+01	3.662E-02	3.312E-01	7.492E-02
2.060E-01	1.333E-01	4.203E-01	5.978E-02	2.390E-01	7.217E-01
1.067E-03	5.178E-03	2.512E-02	6.865E-02	1.589E-03	6.646E-03
8.904E-07	8.348E-04	1.725E-03	1.248E-04	4.409E-04	2.775E-01
1.242E-10	1.395E-04	2.153E-07	6.472E-05	2.718E-09	6.166E-06

Table 3: Rate of Convergence CR of damped L-BFGS applied to Rosenbrock optimization problem.

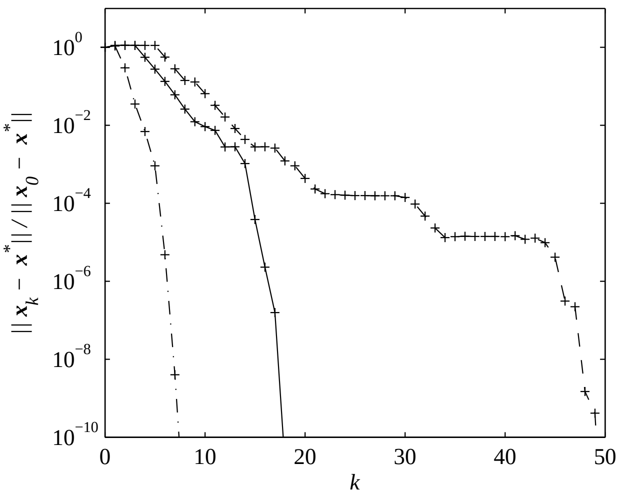

Figure 2: Convergence of Rosenbrock problem from different starting points $\boldsymbol{x}_0$ with elements (–·): $x_i^0 = 2$; (—): $x_i^0 = 50$; and (– –): $x_i^0 = 4786$.

that converges itself to zero. In Table 3 we report results from the last few iterations of the damped L-BFGS SQP algorithm for three selected starting points ($x_i^0 = 2,\ 50,\ 4786$), showing the values of $\|\boldsymbol{x}_k - \boldsymbol{x}^*\|$ and CR. It is observed that the value of CR converges to zero, indicating that convergence is indeed superlinear.

The convergence of the error $\|x_k - x^*\|$ normalized with the initial error $\|x_0 - x^*\|$ is shown in Figure 2 for three different starting points. We observe very fast convergence for starting points that are close to the solution. In that case, also almost no additional function evaluation are required in the line search algorithm, i.e., the SQP step with $\alpha = 1$ usually directly satisfies the Armijo condition. For instance, for $x_i^0 = 2$ twenty function or gradient evaluations are required for 9 SQP iterations (cf. Table 1). This corresponds to the minimum that is required, i.e., one function and one gradient evaluation per SQP step, and one of each at the end of the final step for the convergence evaluation. For starting points further away from the solution, the line-search algorithm described in §3.4 starts to play a role, e.g., for $x_i^0 = 4786$, seventy-three additional function evaluations are required in the line search on top of the 106 function or gradient evaluations that are minimally required for 52 SQP iterations (as is seen in Figure 2, the initial stages of convergence are in that case also slower). Note that the ratio of evaluations over SQP iterations in this case is approximately 3.4. This is the highest ratio for all starting points considered in Table 1 and 2.

Case	T	Method	Iterations	$(\phi_k - \phi_0)/E_m$	C_k/E_0	Wall time
MKE	20	SQP	200	−0.011	$\mathcal{O}(10^{-4})$	41 h
MKE	40	SQP	200	−0.039	$\mathcal{O}(10^{-4})$	70 h
TKE	20	SQP	700	−0.0026	$\mathcal{O}(10^{-4})$	147 h
TKE	40	SQP	300	−0.011	$\mathcal{O}(10^{-4})$	98 h
MKE	20	CG	200	−0.0074	—	160 h
MKE	40	CG	200	−0.031	—	265 h
TKE	20	CG	700	−0.0015	—	505 h
TKE	40	CG	300	−0.0060	—	413 h

Table 4: Mixing-layer optimization cases.

	SQP	CG	Ratio CG/SQP
MKE, T=20	2.17	8.35	3.85
MKE, T=40	2.06	8.72	4.23
TKE, T=20	2.11	9.15	4.34
TKE, T=40	2.05	8.95	4.37

Table 5: Average number of simulations (sum of function and gradient evaluations) per iteration for the SQP and CG methods.

4.2. Evaluation of the SQP damped L-BFGS algorithm for the optimal control of a temporal mixing layer

We now turn to an evaluation of the SQP L-BFGS algorithm for a turbulent mixing layer. As point of comparison, we also use a conventional non-linear Polak–Ribière Conjugate Gradient (CG) method, with projection of the gradient on the non-linear constraint surface, in combination with a Brent line-search method. We refer to [3, 4] for practical details of the implementation. Two different cost functionals are used (cf. §2.1, Eqs. 6,7), i.e. based on minimization of mean-flow kinetic energy (MKE), or of total kinetic energy (TKE). Next to that, two different time horizons T are used, i.e. $T = 20$, and $T = 40$. Details are provided in Table 4. To keep computational resources reasonable, optimization was stopped after 200 CG or SQP iterations for the MKE cases. For the TKE cases, we iterated somewhat longer, i.e. 300 iterations for $T = 40$, and 700 iterations for $T = 20$. All simulations were performed on a high-performance computer, using 64 processors, and optimization wall times are also listed in Table 4.

In Figure 3, the evolution of the merit function and cost functional are monitored for all cases. Note that in case of the CG projection method, the merit function and cost functional are identical, since the constraints are satisfied at every iteration (which is not the case in SQP damped L-BFGS). It is appreciated that the SQP method is converging much faster than the CG projection method for all cases. Moreover, the computational effort required per iteration is significantly lower for SQP than for CG. In Table 5 the average number of adjoint and/or forward simulations per iteration is listed for the different cases. For SQP, we find that the average number of simulations per iteration is close to 2 (which is the minimum required), indicating that the initial step length estimation of the SQP algorithm (i.e. $\alpha = 1$, cf. §3.4) in most cases directly satisfies the Armijo condition. For the CG algorithm, the number of function evaluations is much higher, i.e. approx. 9 per iteration. This is related to the Brent line-search algorithm required in this method.

A precise quantification of the speed differences between the SQP method and the CG method for the current mixing-layer case is difficult, as our optimization (in particular the CG cases) are not formally converged, but stopped after 200 (300, and 700) iterations for reasons of computational resources. However, it is appreciated from Figure 3 that the CG method may require at least twice the amount of iterations before the cost functionals reach the same levels as observed for SQP. Also taking into account that CG is 4 times more expensive per iteration than SQP (cf. Table 5), we believe that the SQP method is at least an order of magnitude faster than the CG method, while still retaining the computational robustness that is required for the use of direct and adjoint Navier–Stokes solvers.

Finally, in Figure 4 the evolution of the temporal mixing layer starting from the different optimized perturbations $\boldsymbol{\phi}$ is displayed at different time instances ($t = 0, 20, 40, 60$). For minimization of mean-flow kinetic energy (top

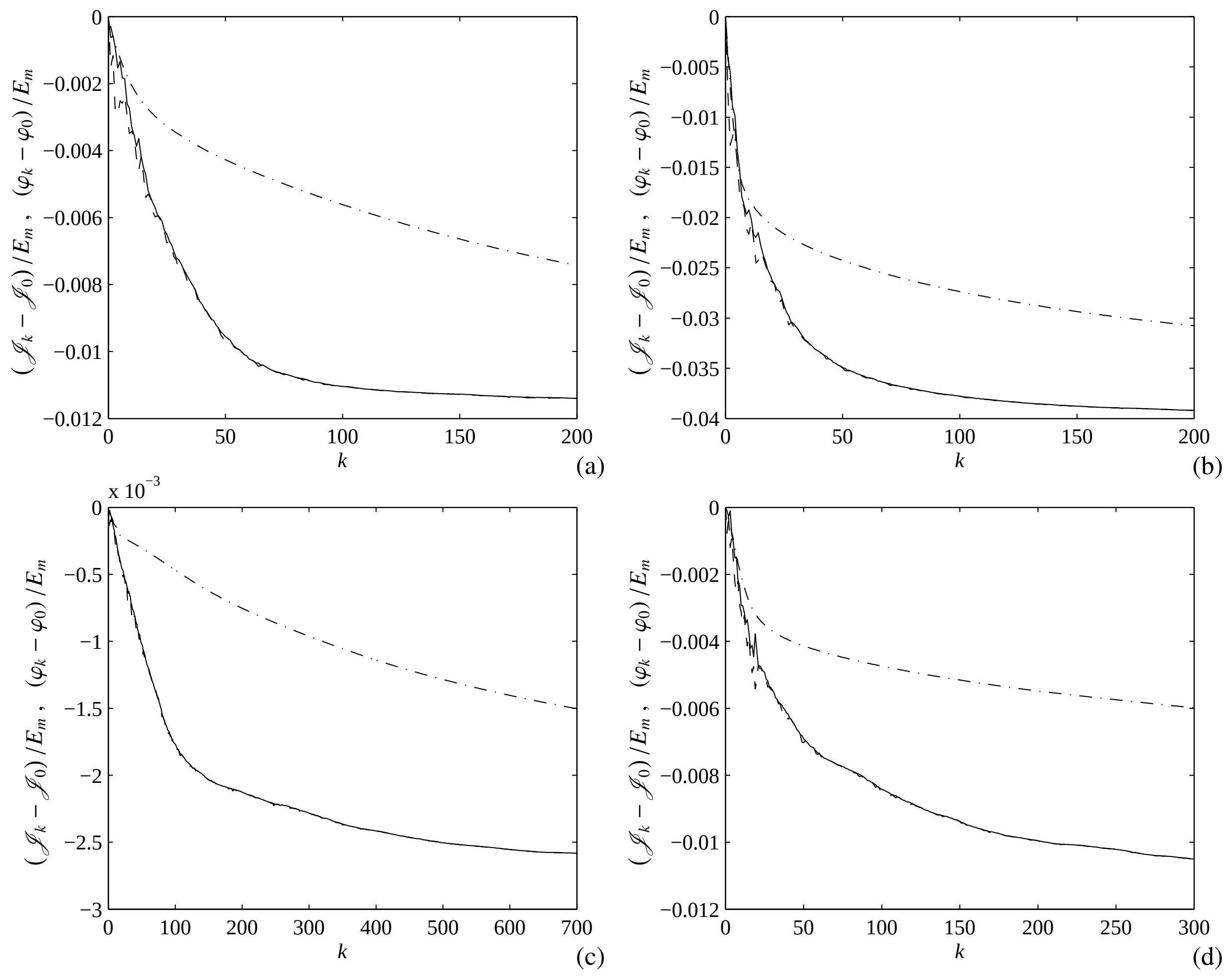

Figure 3: Convergence of SQP L-BFGS and conjugate gradient methods for the optimal control of a mixing layer using different cost functionals. (a) $T = 20$ and cost functional based on mean-flow kinetic energy (cf. Eq. 6); (b) $T = 40$ and cost functional based on mean-flow kinetic energy (cf. Eq. 6); (c) $T = 20$ and cost functional based on total kinetic energy (cf. Eq. 7); (d): $T = 40$ and cost functional based on total kinetic energy (cf. Eq. 7). Lines: (—) SQP cost functional $\mathscr{J}_k$; (−−) SQP merit function φ_k; and (−·) Conjugate gradient cost functional $\mathscr{J}_k$.

two rows), it is seen that the solution is dominated by large two-dimensional vortical structures. These structures are known to grow fast, rapidly increasing the unsteady motion in the mixing layer [3]. The turbulent kinetic energy contained in this unsteady vortical motion is extracted from the mean-flow kinetic energy, thus explaining their relation to the minimization of mean-flow kinetic energy. The minimization of total kinetic energy (bottom two rows), displays a totally different behavior. Here, the total dissipation of kinetic energy over the time window is maximized. This requires fine-scale vortical structures with large velocity gradients, such that the effect of viscosity on the flow increases. Finally, further details related to physical aspects of mixing-layer optimization and more in-depth discussion may be found in Ref. [4].

5. Conclusion

In the current work, we presented an efficient optimization algorithm for DNS-based optimization of turbulent flow with an equality constraint on the controls. Focus is on cases that are characterized by large parameter spaces. Thus, gradient-based methods are required, and gradient calculations are based on the solution of the adjoint equations.

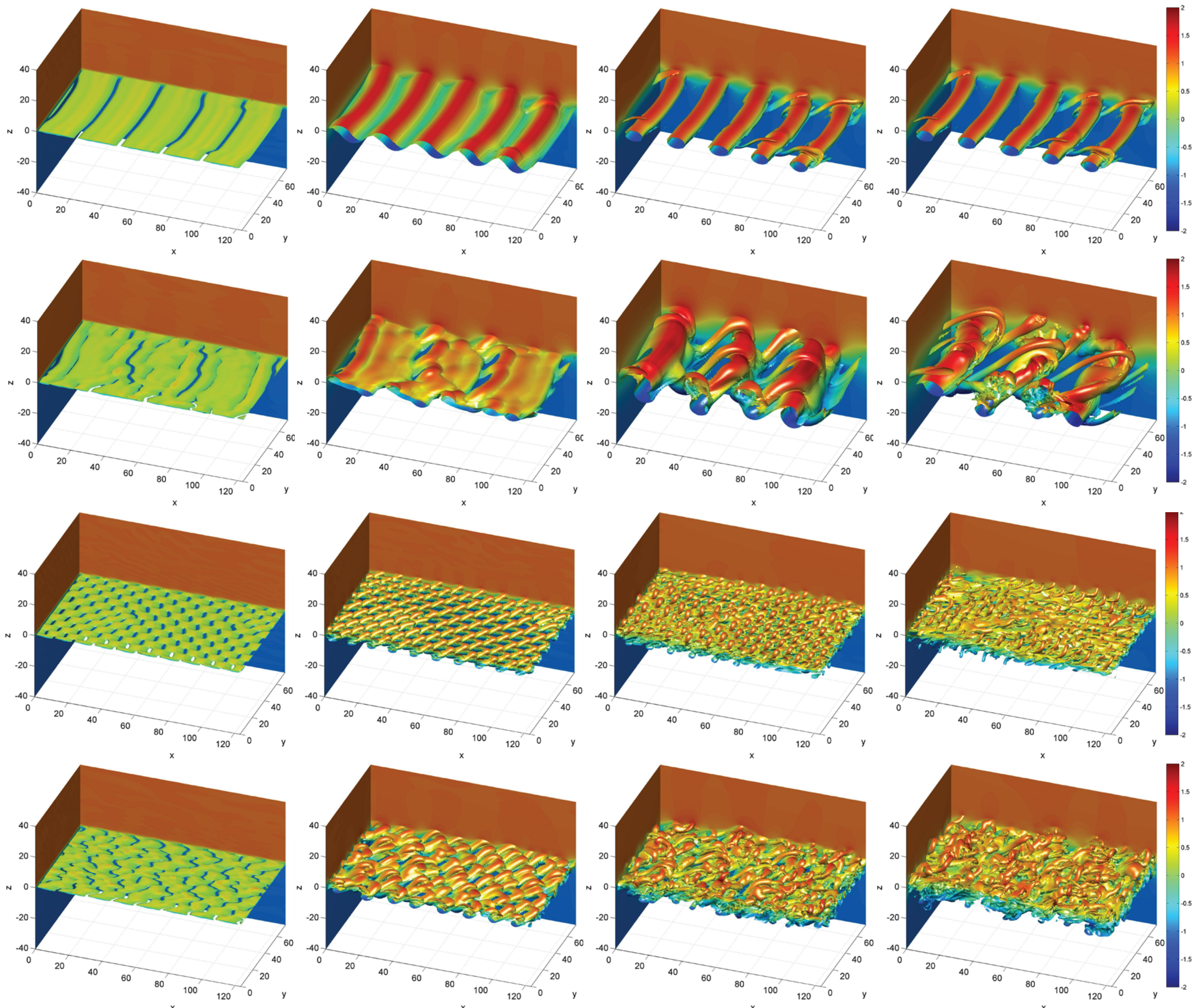

Figure 4: Evolution of optimized mixing layers (based on the SQP method). From top to bottom: MKE & $T = 20$, MKE & $T = 40$, TKE & $T = 20$, TKE & $T = 40$. From left to right: $t = 0, 20, 40, 60$. Vertical planes $x = 0$, and $y = L_y$ colored by stream-wise velocity u_1, and iso-surface of vorticity (at $\omega = 0.9 \max(\partial U/\partial y)$, with $\omega = \|\nabla \times \boldsymbol{u}\|$, and U the plane-average stream-wise velocity) colored by stream-wise velocity. (In z-direction, solution is only displayed in the range $-40 \leq z \leq 40$.)

Further, the large number of control dimensions does not allow to easily compute or store the Hessian of the problem. The algorithm is based on Sequential Quadratic Programming (SQP) in combination with a damped limited-memory BFGS method, and special care is taken to formulate a robust algorithm that works well in combination with an unsteady Navier–Stokes solver. In particular the initial guess of the L-BFGS algorithm needs to be selected carefully, so that first optimization steps are feasible for the solver. To that end, we proposed an appropriate scaling of the initial Hessian estimate that is based on a feasibility region around the constraint surface.

The method is first tested on an extended Rosenbrock problem which is a well-know benchmark for optimization methods, and we mimic optimization in large parameter spaces by selecting the number of dimensions in the Rosenbrock function equal to 50000. We found that the proposed method was fast converging, with a superlinear convergence rate in the neighborhood of the optimum. Also for starting points far from the optimal solution, the method was found to be robust. Moreover, the required number of function and gradients evaluations remained limited, and in most cases close to the minimum of one function and one gradient evaluation per SQP step.

To test the method in the context of DNS of turbulent flows, a temporal mixing-layer problem [3, 4] was selected, and four different cost functionals were tested. We compared optimization using the new SQP method with a conventional non-linear Polak–Ribière Conjugate Gradient (CG) method [3], which is the current standard in DNS-based turbulent-flow optimization. We found that the damped L-BFGS SQP method was at least an order of magnitude faster than conventional non-linear CG. This was both attributed to a faster decrease of cost functionals per iteration for the SQP method, and less function evaluations per iteration (4 times less than for CG).

Finally, we believe that the level of speed-up observed by using the SQP method, compared to conventional CG methods, is essential for computer intensive DNS-based turbulent flow optimization problems that are characterized by large parameter spaces, and require extensive parallel computing. Nowadays, many other efficient optimization algorithms are employed in the context of simpler PDE-constrained optimization (e.g., multi-grid methods, or single-shooting approaches [47]). Their combination with DNS-based optimization is however far from trivial, and subject of further research.

Acknowledgment

The authors acknowledge support from OPTEC (OPTimization in Engineering Center of excellence KU Leuven), which is funded by the KU Leuven Research Council under grant no PFV/10/002. Simulations were performed on the computing infrastructure of the VSC Flemish Supercomputer Center, funded by the Hercules Foundation and the Flemish Government.

Appendix A. Demonstration of the positive definiteness of $\boldsymbol{H}_k$

We briefly demonstrate that the recursive construction of $\boldsymbol{H}_{k+1}$ guarantees the matrix to be positive definite given $\boldsymbol{H}_k$ positive definite. The derivation largely follows the one by Powell [40] for damped BFGS, but is formulated in terms of $\boldsymbol{H}_k$, i.e., the inverse of the Hessian of the Lagrangian, instead of in terms of the Hessian of the Lagrangian itself.

First of all, $\boldsymbol{y}_k^T\boldsymbol{r}_k > 0$ is shown. Using Eq. (30), and Eq. (31) for the case that $\theta_k = 1$, we trivially find $\boldsymbol{y}_k^T\boldsymbol{r}_k = \boldsymbol{y}_k^T\boldsymbol{s}_k \geq 0.2\boldsymbol{y}_k^T\boldsymbol{H}_k\boldsymbol{y}_k > 0$, since $\boldsymbol{H}_k$ positive definite is given. For the other case in Eq. (31), we find

$$\begin{aligned}
\boldsymbol{y}_k^T\boldsymbol{r}_k &= \boldsymbol{y}_k^T[\theta_k\boldsymbol{s}_k + (1-\theta_k)\boldsymbol{H}_k\boldsymbol{y}_k] \\
&= \boldsymbol{y}_k^T\left[\frac{0.8\boldsymbol{y}_k^T\boldsymbol{H}_k\boldsymbol{y}_k}{\boldsymbol{y}_k^T\boldsymbol{H}_k\boldsymbol{y}_k - \boldsymbol{s}_k^T\boldsymbol{y}_k}\boldsymbol{s}_k + \frac{0.2\boldsymbol{y}_k^T\boldsymbol{H}_k\boldsymbol{y}_k - \boldsymbol{s}_k^T\boldsymbol{y}_k}{\boldsymbol{y}_k^T\boldsymbol{H}_k\boldsymbol{y}_k - \boldsymbol{s}_k^T\boldsymbol{y}_k}\boldsymbol{H}_k\boldsymbol{y}_k\right] \\
&= \frac{\boldsymbol{y}_k^T\boldsymbol{H}_k\boldsymbol{y}_k}{\boldsymbol{y}_k^T\boldsymbol{H}_k\boldsymbol{y}_k - \boldsymbol{s}_k^T\boldsymbol{y}_k}\left(0.8\boldsymbol{y}_k^T\boldsymbol{s}_k + 0.2\boldsymbol{y}_k^T\boldsymbol{H}_k\boldsymbol{y}_k - \boldsymbol{s}_k^T\boldsymbol{y}_k\right) \\
&= 0.2\boldsymbol{y}_k^T\boldsymbol{H}_k\boldsymbol{y}_k > 0.
\end{aligned} \tag{A.1}$$

Based on $y_k^T r_k > 0$, and $\boldsymbol{H}_k$ positive definite (starting with $\boldsymbol{H}_0$ positive definite), it is now possible to show that $\boldsymbol{H}_{k+1}$, constructed with Eq. (25), is also positive definite. Indeed, for any non-zero vector z and using (25), we find

$$z^T \boldsymbol{H}_{k+1} z = \boldsymbol{\omega}^T \boldsymbol{H}_k \boldsymbol{\omega} + \frac{(z^T \boldsymbol{r}_k)^2}{\boldsymbol{y}_k^T \boldsymbol{r}_k} \tag{A.2}$$

with $\boldsymbol{\omega} = z - \rho_k (\boldsymbol{r}_k^T z) \boldsymbol{y}_k$. We now have two cases, i.e., (1) $\boldsymbol{r}_k^T z = 0$, for which trivially follows that $z^T \boldsymbol{H}_{k+1} z = z^T \boldsymbol{H}_k z > 0$, and (2) $\boldsymbol{r}_k^T z \neq 0$. For the second case, we find $\boldsymbol{\omega}^T \boldsymbol{H}_k \boldsymbol{\omega} \geq 0$, and $(z^T \boldsymbol{r}_k)^2/(\boldsymbol{y}_k^T \boldsymbol{r}_k) > 0$, so that also for this case $z^T \boldsymbol{H}_{k+1} z > 0$.

[1] T. R. Bewley, P. Moin, R. Temam, DNS-based predictive control of turbulence: an optimal benchmark for feedback algorithms, Journal of Fluid Mechanics 447 (2001) 179–225.

[2] M. J. Wei, J. B. Freund, A noise-controlled free shear flow, Journal of Fluid Mechanics 546 (2006) 123–152.

[3] S. Delport, M. Baelmans, J. Meyers, Constrained optimization of turbulent mixing-layer evolution, Journal of Turbulence 10 (18) (2009) 26.

[4] S. Delport, M. Baelmans, J. Meyers, Maximizing dissipation in a turbulent shear flow by optimal control of its initial state, Physics of Fluids 23 (2011) 045105.

[5] J. B. Freund, Adjoint-based optimization for understanding and suppressing jet noise, Journal of Sound and Vibration 330 (2011) 4114–4122.

[6] A. Hilgers, B. J. Boersma, Optimization of turbulent jet mixing, Fluid Dynamics Research 29 (6) (2001) 345–368.

[7] S. Kern, P. Koumoutsakos, Simulations of optimized anguilliform swimming, The Journal of Experimental Biology 209 (2006) 4841–4857.

[8] T. Verstraete, Z. Alsalihi, R. A. Van den Braembussche, Multidisciplinary optimization of a radial compressor for microgas turbine applications, Journal of Turbomachinery – transactions of the ASME 132 (2010) 031004.

[9] M.D. Gunzburger, Perspectives in Flow Control and Optimization, SIAM, Philadelphia, 2003

[10] F. Abergel, R. Temam, On some Control Problems in Fluid Mechanics, Theoret. Comput. Fluid Dynamics 1 (1990) 303-325

[11] S.S. Sritharan (ed.), Optimal Control of Viscous Flow, SIAM, Philadelphia, 1998

[12] M. B. Giles, N. A. Pierce, Adjoint equations in CFD – duality, boundary conditions and solution behavior, in: 13th AIAA Computational Fluid Dynamics Conference, 1997, pp. AIAA–1997–1850.

[13] A. Jameson, L. Martinelli, N. A. Pierce, Optimum aerodynamic design using the navier-stokes equations, Theoretical and Computational Fluid Dynamics 10 (1-4) (1998) 213–237.

[14] H. Choi, M. Hinze, K. Kunisch, Instantaneous control of backward-facing step flows, Applied Numerical Mathematics 31 (1999) 133–158.

[15] M. B. Giles, N. A. Pierce, An introduction to the adjoint approach to design, Flow Turbulence and Combustion 65 (3-4) (2000) 393–415.

[16] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical recipes in FORTRAN 77: The art of scientific computing, 2nd Edition, Cambridge University press, 1996.

[17] D. G. Luenberger, Linear and nonlinear programming, 2nd Edition, Springer, New York, 2005.

[18] J. Nocedal, S. Wright, Numerical optimization, 2nd Edition, Springer, New York, 2006.

[19] R. Brent, Algorithms for minimization without derivatives, Series in automatic computation, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.

[20] F. Nicoud, J. S. Baggett, P. Moin, W. Cabot, Large eddy simulation wall-modeling based on suboptimal control theory and linear stochastic estimation, Physics of Fluids 13 (10) (2001) 2968–2984.

[21] J. A. Templeton, M. Wang, P. Moin, An efficient wall model for large-eddy simulation based on optimal control theory, Physics of Fluids 18 (2).

[22] M. Hinze, R. Pinnau, M. Ulbrich, S. Ulbrich, Optimization with PDE Constraints, Vol. 23 of Mathematical Modelling: Theory and Applications, Springer, 2009.

[23] O. Ghattas , J.-H. Bark, Optimal Control Of Two- And Three-Dimensional Incompressible Navier-Stokes Flows, Journal of Computational Physics 136 (1997), 231-244

[24] M. Heinkenschloss, Formulation and Analysis of a Sequential Quadratic Programming Method for the Optimal Dirichlet Boundary Control of Navier-Stokes Flow, In Optimal Control Theory, Algorithms, and Applications (eds. W.H. Hager, P.M. Pardalos), Applied Optimization 15, Springer, Dordrecht, 1998, pp. 178-203

[25] M. Hinze, K. Kunisch, Second order methods for optimal control of time-dependent fluid flow. Siam J. Control and Optim. 40 (2001), 925-946

[26] M. Hintermüller, M. Hinze, Globalization of SQP-Methods in Control of the Instationary Navier-Stokes Equations. ESAIM: Mathematical Modelling and Numerical Analysis 36 (2002), 725-746

[27] S.S. Ravindran, Numerical approximation of optimal control of unsteady flows using SQP and time decomposition, Int. J. Numer. Meth. Fluids 45 (2004), 21-42

[28] M. Ulbrich, Constrained Optimal Control of Navier-Stokes Flow by Semismooth Newton Methods, Pfeil, Systems & Control Letters 48 (2003), 297-311

[29] M. M. Rogers, R. D. Moser, The 3-dimensional evolution of a plane mixing layer - the Kelvin-Helmholtz rollup, Journal of Fluid Mechanics 243 (1992) 183–226.

[30] R. D. Moser, M. M. Rogers, The 3-dimensional evolution of a plane mixing layer - pairing and transition to turbulence, Journal of Fluid Mechanics 247 (1993) 275–320.

[31] M. M. Rogers, R. D. Moser, Direct simulation of a self-similar turbulent mixing layer, Physics of Fluids 6 (2) (1994) 903.

[32] B. Vreman, B. Geurts, H. Kuerten, Large-eddy simulation of the turbulent mixing layer, Journal of Fluid Mechanics 339 (1997) 357–390.

[33] R. T. Pierrehumbert, S. E. Widnall, The two-dimensional and 3-dimensional instabilities of a spatially periodic shear-layer, Journal of Fluid Mechanics 114 (Jan) (1982) 59–82.

[34] O. Pironneau, On optimum design in fluid mechanics, Journal of Fluid Mechanics 64 (1) (1974) 97–110.

[35] C. Canuto, M. Hussaini, A. Quarteroni, T. Zang, Spectral methods, fundamentals in single domains, Scientific computation, Springer, 2006.

[36] R. Verstappen, A. E. P. Veldman, Symmetry-preserving discretization of turbulent flow, Journal of Computational Physics 187 (1) (2003) 343–368.
[37] R. Fletcher, Practical methods of optimization, John Wiley, Chichester, 1987.
[38] P. E. Gill, W. Murray, M. Wright, Practical optimization, Academic press, London, 1982.
[39] P. T. Boggs, J. W. Tolle, Sequential quadratic programming, Acta Numerica 4 (1982) 1–51.
[40] M. J. Powell, A Fast Algorithm for Nonlinearly Constrained Optimization Calculations, Numerical Analysis, G.A.Watson ed., Lecture Notes in Mathematics, Springer Verlag 630 (1978).
[41] A. Griewank, A. The local convergence of Broyden-like methods in Lipschitzian problems in Hilbert spaces. SIAM J. Numer. Anal. 24 (1987), 684-705
[42] S. P. Han, A Globally Convergent Method for Nonlinear Programming, J. Optimization Theory and Applications 22 (1977) 297.
[43] L. Armijo, Minimization of functions having Lipschitz continuous first partial derivatives, Pacific Journal of Mathematics 16 (1966) 1–3.
[44] R. H. Byrd, R. A. Tapia, Y. Zhang, An SQP augmented Lagrangian BFGS algorithm for constrained optimization, SIAM J. Optimization 2 (1992) 210–241.
[45] Y.-W. Shang, Y.-H. Qiu, A note on the extended Rosenbrock function, Evolutionary Computation, 14 (2006), 119-126.
[46] H. H. Rosenbrock, An automatic method for finding the greatest or least value of a function, The Computer Journal 3 (1960) 175-184.
[47] A. Borzi, V. Schulz, Computational optimization of systems governed by partial differential equations, SIAM, Philadelphia, 2012.