Insensitive functionals, inconsistent gradients, spurious minima, and regularized functionals in flow optimization problems

John Burkardt[*]   Max Gunzburger[*]   Janet Peterson[*]

Abstract

We use the simple context of Navier-Stokes flow in a channel with a bump to examine problems caused by the insensitivity of functionals with respect to design parameters, the inconsistency of functional gradient approximations, and the appearance of spurious minima in discretized functionals. We discuss how regularization can help overcome these problems. Along the way, we compare the discretize-then-differentiate and differentiate-then-discretize approaches to optimization, especially as they relate to the issue of inconsistent functional gradients. We close with a discussion of the implications that our observations have on more practical flow control and optimization problems.

Key words: flow control, flow optimization, regularization, inconsistent gradients, spurious minima, insensitive functionals

1 Introduction

The optimization or control of flows through the use of high-fidelity CFD simulation codes coupled with sophisticated optimization strategies has, over the last dozen or so years, become a subject of great interest to the engineering and applied mathematics communities. Many great successes using this coupled strategy, both theoretical and practical, have been seen. However, the routine use of such a strategy is not yet realized for two reasons. First, for many problems, especially in three dimensions, the cost of approximate flow solves is still too high to make practical optimization (which usually requires multiple flow solves) a reality. Second, a number of pitfalls can arise when one applies the coupled CFD-optimization strategy which can render even the best optimizer useless or at least ineffective in producing desired optimal solutions. It is this second difficulty that we want to address in this paper. Using a very simple flow optimization problem, we systematically show how a number of pitfalls can arise.

The specific pitfalls we consider are:

- the possible insensitivity of functionals with respect to design parameters;
- the possible inconsistency of functional gradient approximations; and
- the possible appearance of spurious minima in discretized functionals.

We examine how regularization of the objective functional can help avoid or overcome the pitfalls. All this is done in the context of very simple optimization problems for low Reynolds number, viscous, incompressible flow in a two-dimensional channel having a bump along one of its walls. Along the way, we also compare the discretize-then-differentiate and differentiate-then-discretize approaches to optimization, especially in the context of the second pitfall. We end with a brief

[*]Department of Mathematics, Iowa State University, Ames IA 50011-2064. Supported by the Air Force Office of Scientific Research under grant number F49620-95-1-0407.

discussion of the implications that our study has to more complex, and therefore more useful, flow control and optimization problems.

We note that a nozzle design problem for Euler flows containing shock waves for which spurious minima also arise is discussed in [11]. The effects of inaccuracies in computed sensitivities for flows with discontinuities is discussed in [1].

1.1 The model problem

We consider steady, incompressible, viscous flow in the channel $0 < x < 10$ and $0 < y < 3$ having a bump on the lower wall extending from $x = 1$ to $x = 3$. The flow is described by solutions of the stationary Navier-Stokes system

$$-\frac{1}{Re}\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p = \mathbf{0} \qquad \text{and} \qquad \nabla\cdot\mathbf{u} = 0 \qquad \text{in the channel} \tag{1}$$

along with the boundary conditions

$$\mathbf{u} = \mathbf{0} \qquad \text{on the lower and upper walls,} \tag{2}$$

$$\mathbf{u} = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \alpha_0 y(3-y) \\ 0 \end{pmatrix} \qquad \text{at the inflow } x = 0\,, \tag{3}$$

along with standard outflow conditions at $x = 10$, the downstream end of the channel. In (1)–(3), $\mathbf{u}$ and p denote the velocity and pressure fields, respectively, Re is the Reynolds number, and α_0 is a parameter that determines the mass flow rate at the inflow. We denote by u and v the horizontal and vertical components, respectively, of the velocity vector $\mathbf{u}$.

The bump is determined as a sum of Bezier polynomials [4]; some of the coefficients in the sum are determined by the requirement that the bump continuously meets the straight channel walls on either side of it; there remain three coefficients $\{\alpha_k\}_{k=1}^{3}$ at our disposal to effect changes in the flow field. The coefficient α_0 appearing in the inflow boundary condition (3) is another parameter at our disposal. The exact details of the description of the bump and of other aspects of the model problem are not crucial to the discussion we are about to undertake. Here, it suffices to assume that the bump is described by

$$y_B(x;\alpha_1,\alpha_2,\alpha_3) = \Psi(x) + \sum_{k=1}^{3}\alpha_k\Phi_k(x) \qquad \text{for } 1 \le x \le 3\,, \tag{4}$$

where $\Psi(x)$ and $\Phi_k(x)$, for $k = 1, 2, 3$, are given functions.

We compute a *target flow* $(\widehat{u}^h, \widehat{v}^h, \widehat{p}^h)$ by solving the Navier-Stokes system (1)–(3) (by a finite element method) with the parameters chosen to be

$$\widehat{\alpha}_0 = 0.5, \qquad \widehat{\alpha}_1 = 0.375, \qquad \widehat{\alpha}_2 = 0.5, \qquad \text{and} \qquad \widehat{\alpha}_4 = 0.375. \tag{5}$$

Although the bump is described as a sum of higher degree polynomials, for the target values of the bump parameters, the target bump reduces to a parabola.

The *objective functional* is given by, for some x_m in the interval $[3, 10]$,

$$\mathcal{J}^h(\alpha_0,\alpha_1,\alpha_2,\alpha_3) = \int_0^3 \Big(u^h(x_m, y;\alpha_0,\alpha_1,\alpha_2,\alpha_3) - \widehat{u}^h(x_m, y)\Big)^2 dy \tag{6}$$

so that it measures the discrepancy between the horizontal velocity component u^h of the discretized flow and the horizontal velocity component $\widehat{u}^h$ of the target flow along a vertical line across the channel located at the position $x = x_m$ downstream of the bump. Note that (6) is a *discretized* functional since the integrand involves the discretized target flow $\widehat{u}^h$ and u^h, a candidate flow determined by solving the discretized flow equations for a particular guess for the parameters. The functional could be (and usually is) further discretized by approximating the integral in (6) by a numerical integration rule. In the finite element setting, the integrand is a polynomial and we can easily perform the integration in (6) exactly.

The *optimization problem* is then given as follows: for a given x_m, minimize the functional $\mathcal{J}^h$ with respect to the parameters $\{\alpha_k\}_{k=0}^3$, subject to (u^h, v^h, p^h) satisfying the *discretized* Navier-Stokes equations.

For all our calculations, we choose the value of the Reynolds number to be 10 and we apply a Taylor-Hood finite element discretization of the Navier-Stokes system [6]. Unless otherwise noted, we use a logically Cartesian grid of size $h = 0.25$, with the obvious squeezing in the region above the bump. (This level of grid refinement is sufficient to obtain accurate flow solutions for $Re = 10$.) Optimization is effected by a quasi-Newton BFGS/trust region method [3, 5] with a very tight convergence tolerance $= 10^{-9}$. For all cases, the initial values of the four parameters are chosen to be zero. Note that the target flow and parameters are feasible, i.e., an optimizer should be able to obtain exact values of the parameters and the value of the functional at the optimum is zero. Of course, gradient based optimization algorithms are required to find a point in parameter space at which the gradient of the functional vanishes; in general, they will not make use of the fact that, in our case, the functional itself vanishes at such a point. However, the vanishing of the functional at the optimum certainly is information we can use to monitor the performance of optimization methods.

The number of optimization steps given in the tables below are relatively high since we are converging solutions to an extremely tight tolerance; in practice, it makes no sense to iterate beyond something a little smaller than the discretization error for the approximate solution of the state equations; in that case, the number of iteration steps needed will be drastically reduced.

It is difficult to come up with a more straightforward or elementary flow control or optimization problem than the one we consider here. However, as we shall see, even in this simple setting, all sorts of difficulties can arise.

1.2 Determining the gradient of the functional

The optimization method we use employs the gradient of the functional to determine new guesses of the parameters from old guesses. We use two approaches for determining the gradient; the first is a discretize-then-differentiate approach and the second is a differentiate-then-discretize approach.

The first approach is to approximate the gradient of the functional by a finite difference quotient approximation. For example, for the first parameter, we have

$$\frac{\partial \mathcal{J}^h}{\partial \alpha_0} \approx \frac{\mathcal{J}^h(\alpha_0 + \delta\alpha_0, \alpha_1, \alpha_2, \alpha_3) - \mathcal{J}^h(\alpha_0, \alpha_1, \alpha_2, \alpha_3)}{\delta\alpha_0}\,, \tag{7}$$

where $\delta\alpha_0$ is a chosen (usually small) increment in the parameter α_0. Similar expressions hold for the other three parameters. Here, $\partial\mathcal{J}^h/\partial\alpha_k$, $k = 0, 1, 2, 3$, denote the components of the gradient of the functional with respect to the parameters $\{\alpha_k\}_{k=0}^3$. Note that to determine the approximate first component of the gradient by (7), we need to solve the flow system (1)–(3) for the parameter set $\{\alpha_0 + \delta\alpha_0, \alpha_1, \alpha_2, \alpha_3\}$. To determine approximations to all four components of the gradient of the functional requires four additional approximate flow solves.

An approximation to the gradient of the functional can be detemined more efficiently with the help of the *sensitivities*

$$\mathbf{u}_k = \begin{pmatrix} u_k \\ v_k \end{pmatrix} = \begin{pmatrix} \dfrac{\partial u}{\partial \alpha_k} \\ \dfrac{\partial v}{\partial \alpha_k} \end{pmatrix} \qquad \text{and} \qquad p_k = \frac{\partial p}{\partial \alpha_k} \qquad \text{for } k = 0,1,2,3\,. \tag{8}$$

In general, we cannot determine the sensitivities exactly. In a differentiate-then-discretize approach which we refer to as the *sensitivity equation method*, we first differentiate the flow system (1)–(3) with respect to each of the design parameters $\{\alpha_i\}_{k=0}^{3}$ to obtain the four continuous sensitivity systems: for $k = 0, 1, 2, 3$,

$$-\frac{1}{Re}\Delta\mathbf{u}_k + \mathbf{u}_k\cdot\nabla\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u}_k + \nabla p_k = \mathbf{0} \qquad \text{and} \qquad \nabla\cdot\mathbf{u}_k = 0 \qquad \text{in the channel,} \tag{9}$$

$$\mathbf{u}_k = \mathbf{0} \qquad \text{on the lower and upper walls except along the bump,} \tag{10}$$

and

$$\mathbf{u}_k = \begin{cases} \mathbf{0} & \text{if } k = 0 \\ -\left(\dfrac{\partial \mathbf{u}}{\partial y}\right)\Phi_k & \text{if } k = 1,2,3 \end{cases} \qquad \text{along the bump,} \tag{11}$$

$$u_k = \begin{cases} y(3-y) & \text{if } k = 0 \\ 0 & \text{if } k = 1,2,3 \end{cases} \qquad \text{and} \qquad v_k = 0 \qquad \text{at the inflow } x = 0\,, \tag{12}$$

and the same outflow conditions as used for the velocity $\mathbf{u}$ and pressure p. For details, see, e.g., [7, 8]. We then determine approximate sensitivities by discretizing (9)–(12), e.g., by using the same finite element method as was used to discretize the flow system (1)–(3). We denote the approximate sensitivities determined by this process, e.g., by first differentiating the flow equations and then discretizing the resulting continuous sensitivity equations, by

$$\left(\frac{\partial u}{\partial \alpha_k}\right)^h, \qquad \left(\frac{\partial v}{\partial \alpha_k}\right)^h, \qquad \text{and} \qquad \left(\frac{\partial p}{\partial \alpha_k}\right)^h, \qquad \text{for } k = 0,1,2,3\,. \tag{13}$$

One can instead use a discretize-then-differentiate approach for determining approximate sensitivities. In this approach, one first discretizes the flow equations, e.g., by a finite element method. Then one differentiates the discretized flow equations with respect to the design parameters to obtain four systems of discrete equations for the approximate flow sensitivities. We denote the approximate sensitivities determined by this process, e.g., by first discretizing the flow equations and then differentiating the resulting discretized flow equations, by

$$\frac{\partial u^h}{\partial \alpha_k}, \qquad \frac{\partial v^h}{\partial \alpha_k}, \qquad \text{and} \qquad \frac{\partial p^h}{\partial \alpha_k}, \qquad \text{for } k = 0,1,2,3\,. \tag{14}$$

These are the approximate sensitivities obtained when one uses an automatic differentiation methodology to obtain sensitivities; see, e.g., [2] or [10].

Although both (13) and (14) are approximations to the exact sensitivities (8), they are in general not the same, e.g.,

$$\left(\frac{\partial u}{\partial \alpha_k}\right)^h \neq \left(\frac{\partial u^h}{\partial \alpha_k}\right);$$

the differentiation and discretization steps do not commute.

The gradient of the discretized functional (6) can be determined from either

$$\frac{\partial \mathcal{J}^h}{\partial \alpha_k} \approx \int_0^3 \left((u^h - \widehat{u}^h) \left(\frac{\partial u}{\partial \alpha_k} \right)^h \right) \Bigg|_{(x_m,y)} dy \tag{15}$$

using (13) or from

$$\frac{\partial \mathcal{J}^h}{\partial \alpha_k} = \int_0^3 \left((u^h - \widehat{u}^h) \frac{\partial u^h}{\partial \alpha_k} \right) \Bigg|_{(x_m,y)} dy \tag{16}$$

using (14). Note that (15) only yields an *approximation* to the gradient of the discretized functional because we use (13) instead of (14). On the other hand, (16) yields the exact gradient of the discretized functional (6). Both (15) and (16) may be viewed as approximations of the gradient of the same continuous functional which involves exact solutions of the Navier-Stokes system (1)–(3); however, the fact that (16) is the exact gradient of the discretized functional while (15) is not the exact gradient of anything can and will have an important role to play in our discussions.

In our discussions, we will not consider both (7) and (16); both of these are discretize-then-differentiate approaches and have been found to have very much the same effects when used in gradient-based flow control and optimization methods, so long as the increments $\delta\alpha_k$ used in the finite difference approach are small; see, e.g., [1, 7]. Thus, we will use (7), i.e., finite difference quotient functional gradient approximations, as a representative discretize-then-differentiate approach and (15), i.e., the sensitivity equation method, as a representative differentiate-then-discretize approach.

We note that the gradient of the functional can also be determined or approximated through the use of solutions of *adjoint equations*; see, e.g., [8, 9]. Both discretize-then-differentiate and differentiate-then-discretize strategies may again be employed. With regard to the issues, difficulties, and remedies discussed here, adjoint equation approaches do not offer any advantages or disadvantages over the finite difference quotient or sensitivity-based approaches. Thus, we will not consider adjoint equation approaches in this paper.

2 Insensitive cost functionals

We choose the matching line to be located at $x_m = 9$; see Figure 1.

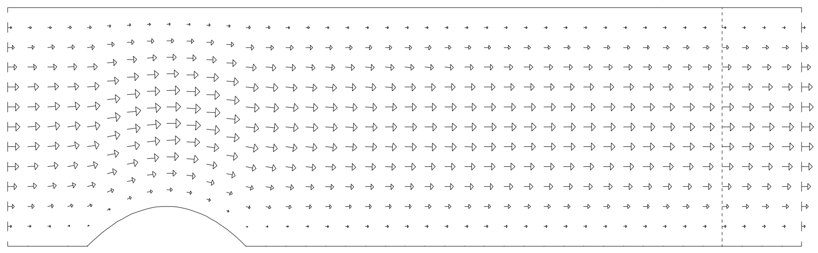

Figure 1: The target flow $(\widehat{u}^h, \widehat{v}^h)$ in the channel and the matching line at $x = 9$.

We first use (15) to evaluate the gradient of the functional, i.e., we use solutions of the discretized continuous sensitivity equations. With a tolerance of 10^{-9}, the optimizer declared that satisfactory convergence was achieved after 20 iterations and returned the values given in Table 1.

α_0	α_1	α_2	α_3	$\mathcal{J}^h$	$\max\lvert\nabla\mathcal{J}^h\rvert$
0.5000	0.0767	0.0656	0.2760	0.6×10^{-9}	0.3×10^{-5}

Table 1: Parameter, functional, and gradient values after 20 optimizer iterations using solutions of discretized continuous sensitivity equations to evaluate the gradient of the functional.

Note that the optimal value of α_0 is very close to its target value of 0.5 but that the three bump parameters are nowhere close to theirs, despite the fact that the functional and its gradient are very small. In fact, the value of the functional for the initial parameters is 0.4 so that the value of the functional has been reduced by 9 orders of magnitude!

If we instead determine the gradient of the functional using (7), i.e., a finite difference quotient approximation, we obtain an even more surprising set of parameter values after 10 iterations; see Table 2. Again, α_0 is very close to its target value and the functional and its gradient are very small, but the bump parameters are all zero, i.e., there is no bump! The bump shapes for the target parameters and for the parameters of Tables 1 and 2 are given in Figure 2.

α_0	α_1	α_2	α_3	$\mathcal{J}^h$	$\max\lvert\nabla\mathcal{J}^h\rvert$
0.5000	0	0	0	0.3×10^{-8}	0.2×10^{-6}

Table 2: Parameter, functional, and gradient values after 10 optimizer iterations using the finite difference quotient approach to evaluate the gradient of the functional.

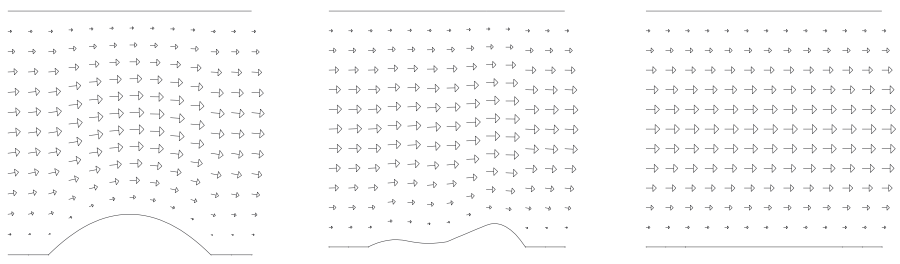

Figure 2: Bump and flow near the bump for the target flow (left) and the optimal flows determined using functional gradient approximations using the sensitivity equation method (middle) and finite difference quotient gradient approximations (right).

Why do we obtain such inaccurate bumps even though the functional and its gradient are very small? Why is the "optimal" inflow parameter so good but the bump parameters so bad?

First, let's see why α_0 is so good. The role of optimization with respect to α_0 is to get the mass flow at the inflow to be the same as that of the target flow at the matching line. Thus, it is no wonder that the optimizer very quickly finds the right value for α_0; this is typical for parameters that determine the values of gross, e.g. integral, features of the flow.

For our choices of Reynolds number and matching plane location, once the mass flow at the inflow is equal to the mass flow across the matching plane (as is true once α_0 has converged), not only is the optimal profile at the matching plane that for Poiseuille flow, but it is in fact *essentially identical to that of the target flow*, irrespective of the shape of the bump. This must be so because, for low values of the Reynolds number, at the matching line the target flow and the optimal flow both are very nearly *Poiseuille flows with the same mass flow.* Thus, for any values of the bump parameters, the functional

$$\mathcal{J}^h(\alpha_0, \alpha_1, \alpha_2, \alpha_3) = \int_0^3 \Big(u^h(9, y; \alpha_0, \alpha_1, \alpha_2, \alpha_3) - \widehat{u}^h(9, y) \Big)^2 dy$$

is small because $u^h(9, y; \alpha_0, \alpha_1, \alpha_2, \alpha_3)$ is very close to $\widehat{u}^h(9, y)$.

Next, let's see why the computed approximation to the gradient of the discrete functional is small, even though the bump parameters are way off their correct values. Part of the answer lies with the sensitivities; the sensitivities with respect to the bump parameters are big near the bump, but *are minute far downstream of the bump and, in particular, at the matching line* $x = 9$; see Figure 3. (In contrast, the sensitivity of the flow field with respect to α_0 at the matching line is not small.) This merely reflects the fact that at low values of the Reynolds number, no matter what shape the bump takes, the flow at a matching plane far downstream of the bump is going to be Pouisselle flow. The approximate gradient of the discrete functional with respect to the bump parameters given by (15) is small for any values of the bump parameters because, first, $u^h(9, y; \alpha_0, \alpha_1, \alpha_2, \alpha_3) = \widehat{u}^h(9, y)$, i.e., the flow and target flows are essentially the same, and second, because the velocity sensitivity at the matching line is small. The second cause, i.e., the smallness of the sensitivities, is the important one since it remains in effect even if we had chosen a target flow that was not feasible.

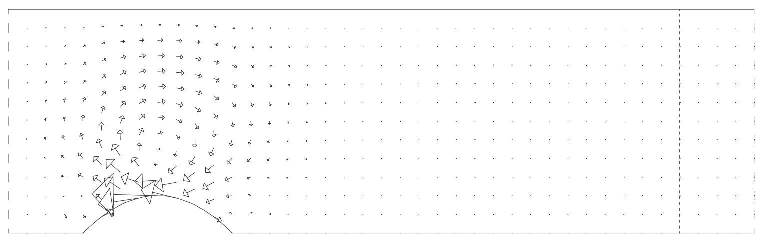

Figure 3: Sensitivity of the flow velocity with respect to the second bump parameter α_2 at the target values of the parameters.

Note that the problem is not with the optimizer; it did its job very well. The optimizer found values for the four parameters such that gradient of the functional is small; in addition, the functional itself was small at those values of the parameters, as we expect it to be for the functional we are using. The problem is that *the functional is very insensitive to changes in the bump parameters.* Thus, care must be exercised in setting up optimization problems to make sure that the functional is not insensitive to some or all of the design parameters. Alternately, if one has a functional that one wants to minimize and it is insensitive to changes in a design parameter, we might as well forget about that parameter and optimize with respect to only those parameters that can effect appreciable changes in the value of the functional.

3 Inconsistent functional gradients

In order to work with a more sensitive functional, from now on we set $x_m = 3$, i.e., we move the matching line to the back of the bump; see Figure 4. Otherwise, the problem specification remains the same. As we shall see, our troubles are just starting!

We determine the gradient of the functional using the approximate sensitivities determined by discretizing the continuous sensitivity equations. The optimizer gets confused after 33 iterations at which point it quits at the parameter values given in Table 3. Now, not only are the bump

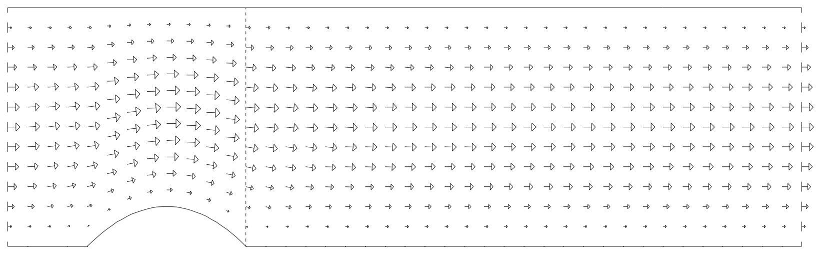

Figure 4: The target flow $(\widehat{u}^h, \widehat{v}^h)$ in the channel and the matching line at $x = 3$.

parameters not good and the value of the functional not small, but more importantly, the computed gradient of the functional is not small. The shape of the bump and the corresponding velocity field for the parameters of Table 3 are given in Figure 5. Note that the oscillatory nature of the bump is very different from the target bump given in Figure 4.

α_0	α_1	α_2	α_3	$\mathcal{J}^h$	$\max\lvert\nabla\mathcal{J}^h\rvert$
0.5058	-0.0185	0.4352	-0.0448	0.4×10^{-3}	0.2×10^{-1}

Table 3: Parameter, functional, and gradient values after 33 optimizer iterations using solutions of discretized continuous sensitivity equations to evaluate the gradient of the functional.

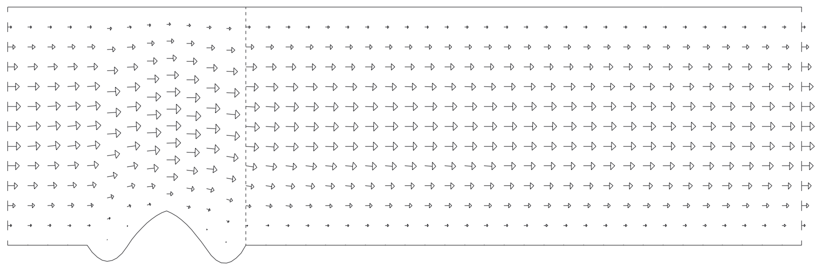

Figure 5: Bump geometry and flow field resulting after 33 optimizer iterations using solutions of discretized continuous sensitivity equations to evaluate the gradient of the functional.

The optimizer can handle functional values that are not small; after all, it doesn't know that for the exact minimizer, the value of the functional is zero. However, the optimizer can't live with nonzero gradients since at local or global minima, the gradient should be zero. Why did the optimizer quit even though the gradient isn't small?

To get an indication of what has gone wrong, let's draw a line (in the four-dimensional parameter space) through the global minimizer (the parameter values for the target flow given by (5)) and the parameter values returned by the optimizer after 33 iterations as given in Table 3. Let's evaluate the functional along that line as well as the derivative of the functional in the direction of the line. The results are given in Figure 6.

It is clear from Figure 6 that the optimizer did not stop at a local minimum of the functional; so why did it stop? From the plots of the functional and its directional derivative along the line,

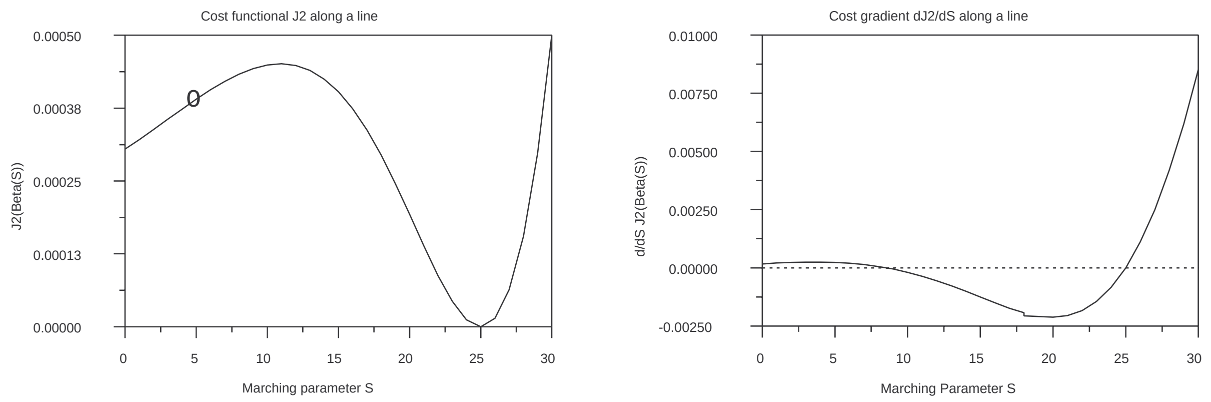

Figure 6: Functional (left) and directional derivative of the functional (right) along the line in parameter space joining the global minimum and the values of the parameter returned by the optimizer after 33 iterations; the circle identifies the point at which the optimizer stopped.

we glean the following information about the behavior of the functional

$0 \le s < 11$	the functional increases
$11 < s < 25$	the functional decreases
$25 < s \le 30$	the functional increases

and its directional derivative

$0 \le s < 8$	the gradient of the functional is positive
$8 < s < 25$	the gradient of the functional is negative
$25 < s \le 30$	the gradient of the functional is positive

as a function of arc length s along the line. There is a serious inconsistency in these figures: *in the interval $8 \le s \le 11$, the functional is increasing in s but the computed gradient of the functional determined by the differentiate-then-discretize sensitivity-based method is negative.*

Let's explore this a little more by examining a two-dimensional slice of parameter space that passes through the global minimum and the point in parameter space returned by the optimizer after 33 iterations. We compute the functional on that plane and the direction of the projection of the negative of the gradient onto that plane; these are displayed in Figure 7. Note that the negative of the gradient should be perpendicular to the level curves of the functional and should point in the direction of decreasing functional values.

From Figure 7 we see lots of places where the negative gradient is not perpendicular to the level curves. The optimizer doesn't care about this; all the optimizer cares about is that when it is given a functional and its gradient, that *the negative of the gradient points downhill.* Then it can guarantee that, for a sufficiently small step size, it can find new values of the parameters that reduce the value of the functional. Again, from Figure 7, we see that there certainly are locations in the two-dimensional slice at which *the negative of the gradient is pointing uphill.* Although examining a single two-dimensional slice out of a four-dimensional space is not definitive, what probably happened after 33 iterations is that the optimizer found a point in parameter space at which the functional it was given increases in the direction of the negative functional gradient it was also given. The optimizer cannot live with this inconsistency since *it cannot find a step along the direction of the negative gradient, no matter how small it chooses the step, which results in a decrease in the value of the functional.* At such a juncture, the optimizer just quits.

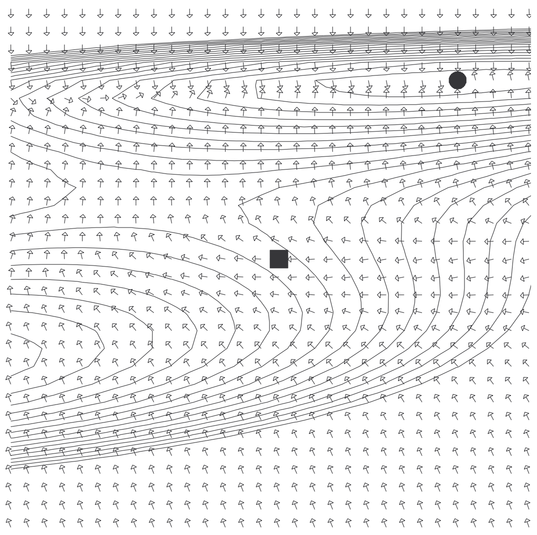

Figure 7: Level curves of the functional and projected negative approximate gradient of the functional on a two-dimensional slice of parameter space containing the global minimum (circle) and the values of the parameter returned by the optimizer after 33 iterations (square); the approximate gradient of the functional is determined by the differentiate-then-discretize approach.

Let's now compute an approximation to the gradient of the functional using a finite difference quotient approximation, i.e., a discretize-then-differentiate approach. In Figure 8, using the same two-dimensional slice as was used for Figure 7, we again provide level curves of the functional and direction vectors for the projected negative of the approximate gradient. Now there are no inconsistencies; the approximate gradient projected onto the two-dimensional slice is always nearly perpendicular to the level curves and the negative of the projected approximate gradient always points downhill.

In Figure 9, we zoom into the vicinity of the point in parameter space returned by the optimizer when it used the approximate gradients found by the differentiate-then-discretize approach. We see that there are lots of nearby locations at which the differentiate-then-discretize sensitivity based negative gradient points in the wrong direction (uphill); the finite difference quotient based negative gradient points in the right direction (downhill). Note the points for which the angle between the two approximate gradients is larger than $\pi/2$.

The discretize-then-differentiate sensitivity based approach, e.g., using automatic differentiation software, produces the exact gradient of the discrete functional which is being minimized, so of course, that gradient also points in the right direction; using finite difference quotient approximations with small increments $\delta\alpha_k$ also produces consistent gradients. For this reason, some believe that the discretize-then-differentiate approach is superior to the differentiate-then-discretize approach. The differentiate-then-discretize approach does not produce an exact gradient of any functional (continuous or discrete) and is more prone to producing inconsistent functional gradients. For the problem we are considering, if the grid size is sufficiently small, then all approaches must produce consistent gradients; however, the differentiate-then-discretize approach can produce an inconsistent gradient at a practical (from the point of simulation accuracy) grid size. Although this seems very bad, we shall see below that the solution of another difficulty also helps the differentiate-then-discretize approach produce consistent gradients. To this end, note that even *the differentiate-then-discretize approach produces a consistent gradient near the global minimizer.*

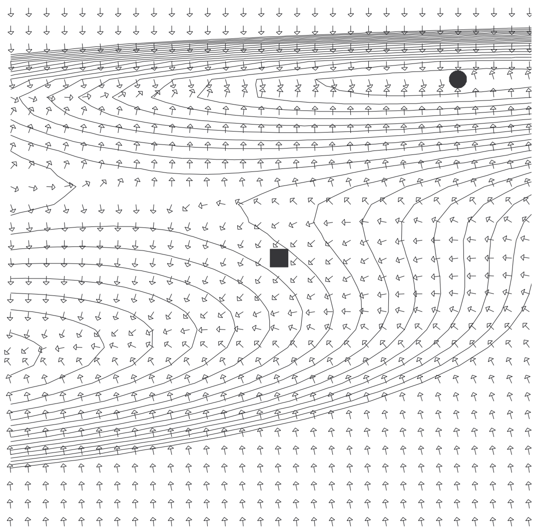

Figure 8: Level curves of the functional and projected negative approximate gradient of the functional on the same two-dimensional slice of parameter space used for Figure 7; the gradient of the functional is determined by the finite difference quotient approach. The square and circle have the same meaning as in Figure 7.

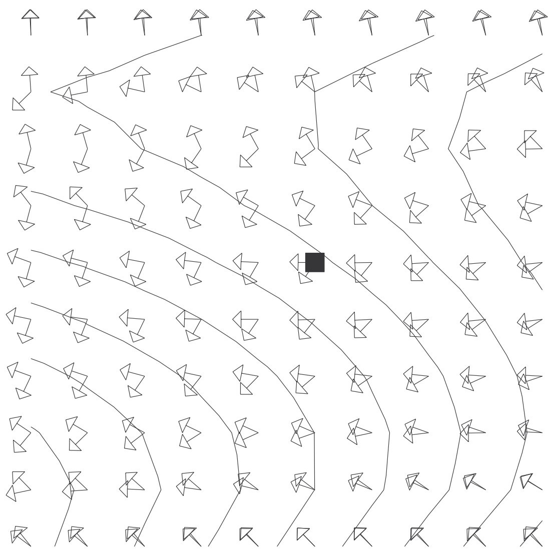

Figure 9: Level curves of the functional and projected negative approximate gradients of the functional on the same two-dimensional slice of parameter space used for Figures 7 and 8 and in the vicinity of the point (the filled square) returned by the optimizer after 33 iterations of the differentiate-then-discretize sensitivity equation approach; the direction of the approximate negative gradient of the functional determined by both the finite difference quotient approximation and by the sensitivity equation approach are displayed.

4 Spurious minima

Now that we know that using finite difference quotients to approximate the gradient of the functional yields consistent gradients, let's solve the optimization problem (with the matching line located at

$x = 3$) using those gradient approximations instead of the differentiate-then-discretize approach used for Table 3. The optimizer declares convergence after 44 iterations and returns the data in Table 4 for the optimal solution. Note the very small gradient of the functional which indicates that the solution provided by the optimizer is at least a local minimizer of the discretized functional. However, the bump parameters are again not close to those for the target flow. Note that although the bump parameters are far from those of the target flow, the value of the functional is quite small, i.e., the target flow is well matched at the line $x = 3$ even with the wrong bump parameters, i.e., the wrong bump looks a lot like the target bump as far as the flow at $x = 3$ is concerned; see Figure 10.

α_0	α_1	α_2	α_3	$\mathcal{J}^h$	$\max \lvert\nabla \mathcal{J}^h\rvert$
0.5078	0.1407	0.5394	0.0599	0.3×10^{-6}	0.2×10^{-9}

Table 4: Parameter, functional, and gradient values after 44 optimizer iterations using finite difference quotient approximations of the gradient of the functional.

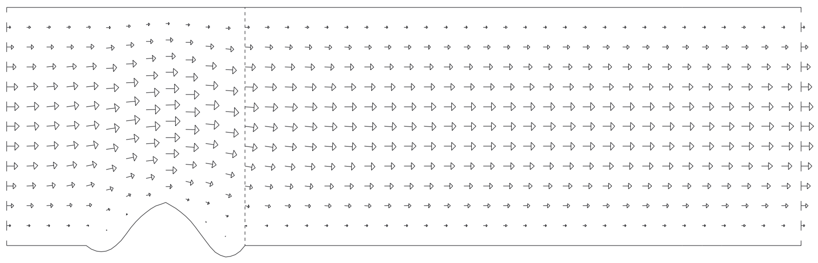

Figure 10: Bump geometry and flow field resulting after 44 optimizer iterations using finite difference quotient approximations of the gradient of the functional.

Let's again look at the functional along the line in four-dimensional parameter space joining the global minimizer and the computed minimizer of Table 4; see Figure 11. We see that it certainly looks like the computed solution is a local minimizer of the functional; in fact the value of the functional at this location is very small ($\approx 0.2 \times 10^{-9}$.)

Results along lines in four dimensions are not very conclusive, so let's again look at the functional on a two-dimensional slice determined by a plane going through the global minimizer and the optimal solution of Table 4. Figure 12 displays the curves of the functional and the direction of the negative approximate gradient. Again, both the position of the computed solution among the level curves of the functional and the gradient of the functional indicate that the optimizer has indeed found a local minimizer of the functional.

Is this local minimizer for real, or is it a spurious one introduced by the discretization process? A good (but not foolproof) way to check if a suspect solution (especially one with oscillatory behavior) is really a solution or whether it is a numerical artifact is to refine the grid. If upon refinement the solution seems to converge, then there is a good chance that it is an actual solution; if upon refinement the suspect solution gets worse, e.g., oscillates with bigger amplitude or higher frequency, then chances are good that it is a numerical artifact. In Table 5 and visually in Figure 13, one can see the non-convergence of the bump parameters as the grid size is reduced; the oscillations in the bump geometry have higher amplitude as the grid size is reduced.

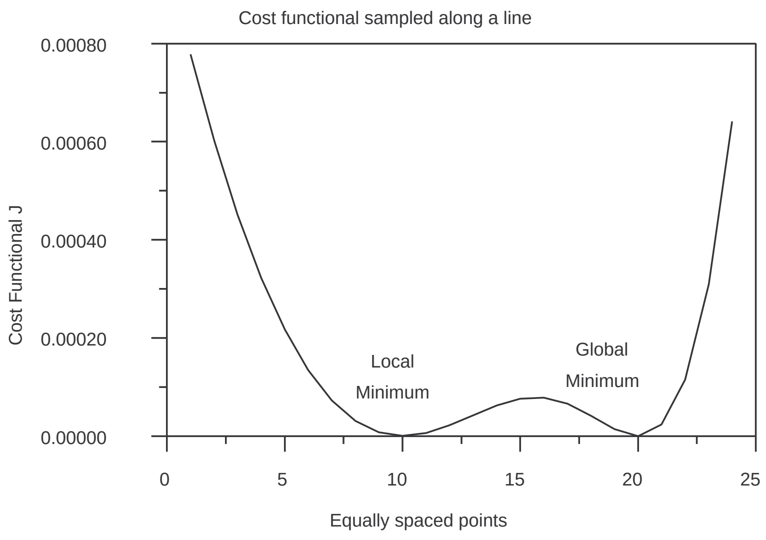

Figure 11: Functional along the line in parameter space joining the global minimizer and the computed minimizer of Table 4.

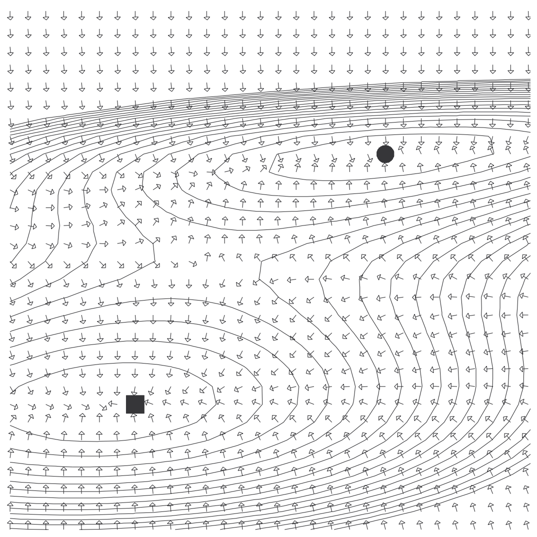

Figure 12: Level curves of the functional and projected negative gradient of the functional on a two-dimensional slice of parameter space containing the global minimum (circle) and computed minimum returned by the optimizer after 44 iterations (square); the gradient of the functional is determined by a finite difference quotient approach.

grid size	α_0	α_1	α_2	α_3	$\mathcal{J}^h$
0.250	0.5078	0.1408	0.5394	0.0600	0.3×10^{-6}
0.167	0.5087	-1.4610	0.3596	-0.0841	0.1×10^{-4}
0.125	0.5111	-1.9128	0.2984	-0.2906	0.1×10^{-4}

Table 5: Converged parameter and functional values for different grid sizes; the optimizer uses finite difference quotient approximations of the gradient of the functional.

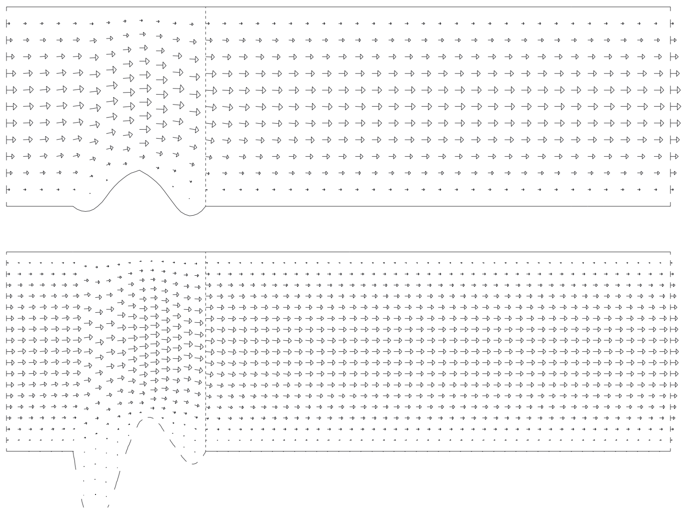

Figure 13: Bump geometry and flow field for a grid size $h = 0.25$ (top) and $h = 0.167$ (bottom).

Thus, we conclude that *the local minimizer found by the optimizer is a spurious, numerically induced one.*

5 Regularization of the functional

It is clear that, at least for some initial conditions, the optimizer leads us to the spurious local minimum. How can one get to the desired global minimum? To avoid the spurious minimum, in fact, to get rid of it altogether, one can used a penalized objective functional; since the solution corresponding to the spurious minimizer contained unwanted oscillations in the bump geometry, we add a penalty term which penalizes such oscillations. Thus, we choose *the penalized objective functional*

$$\begin{aligned}\mathcal{J}^h_\epsilon(\alpha_0,\alpha_1,\alpha_2,\alpha_3) &= \mathcal{J}^h(\alpha_0,\alpha_1,\alpha_2,\alpha_3) + \frac{\epsilon}{2}\int_1^3 \left(\frac{dy_B}{dx}\right)^2 dx \\ &= \frac{1}{2}\int_0^3 \Big(u^h(x_m,y;\alpha_0,\alpha_1,\alpha_2,\alpha_3) - \widehat{u}^h(x_m,y)\Big)^2 dy + \frac{\epsilon}{2}\int_1^3 \left(\frac{dy_B}{dx}\right)^2 dx\,,\end{aligned} \tag{17}$$

where $y_B(x;\alpha_1,\alpha_2,\alpha_3)$ is the function that determines the bump and ϵ is a penalty parameter. Of course, the case $\epsilon = 0$ corresponds to the unpenalized functional, i.e., $\mathcal{J}^h_0 = \mathcal{J}^h$.

First, in Table 6, let's see what the optimizer does as we change ϵ. In that table, for various values of the penalty parameter, we give the number of steps taken by the optimizer to converge, the converged values of the parameters and functional found by the optimizer, and the value of

$$\mathcal{G} = \frac{1}{2}\int_1^3 \left(\frac{dy_B}{dx}\right)^2 dx$$

which is a measure of the amount of oscillation in the bump geometry.

Recall that the parameters for the matching function, i.e., the parameters we are trying to find, are $\alpha_0 = 0.5$, $\alpha_1 = 0.375$, $\alpha_2 = 0.5$, and $\alpha_3 = 0.375$. For all values of ϵ, the inflow parameter α_0

ϵ	steps	α_0	α_1	α_2	α_3	$\mathcal{G}$	$\mathcal{J}^h$
1	22	0.5127	0.0026	0.0044	0.0032	0.4×10^{-4}	0.8×10^{-2}
10^{-1}	11	0.5122	0.0273	0.0467	0.0356	0.5×10^{-2}	0.8×10^{-2}
10^{-2}	19	0.5057	0.1740	0.3262	0.2706	0.2×10^{0}	0.4×10^{-2}
10^{-3}	31	0.5006	0.2515	0.4677	0.3665	0.5×10^{0}	0.5×10^{-3}
10^{-4}	40	0.5004	0.2849	0.5001	0.3636	0.5×10^{0}	0.6×10^{-4}
10^{-5}	47	0.5056	0.1305	0.5442	0.1527	0.1×10^{1}	0.1×10^{-4}
10^{-6}	47	0.5075	0.1387	0.5406	0.0712	0.6×10^{0}	0.1×10^{-5}
10^{-7}	43	0.5077	0.1405	0.5395	0.0611	0.6×10^{0}	0.4×10^{-6}
0	44	0.5077	0.1407	0.5394	0.0599	0.6×10^{0}	0.3×10^{-6}

Table 6: Converged parameter and functional values and optimizer steps for different values of the penalty parameter ϵ; $\mathcal{G}$ is a measure of the amount of oscillation in the bump geometry.

is pretty well matched. The best match for the bump parameters α_1, α_2, and α_3 occurs around $\epsilon = 10^{-4}$. The match of the bump parameters gets worse if we raise or lower the value of ϵ from that value, but for different reasons. For large values of ϵ, the match deteriorates because the global minimizer of the functional $\mathcal{J}_\epsilon^h$ moves further away from the global minimizer of the functional $\mathcal{J}^h$; the latter is what we are trying to find. In fact, as ϵ gets large, the penalty term in the functional $\mathcal{J}_\epsilon^h$ becomes dominant so that the minimization of that functional forces $dy_B/dx \to 0$, i.e., to a no bump situation; we already see evidence of this behavior even for $\epsilon = 1$. For small values of ϵ the match deteriorates because the penalty term ceases to provide enough regularization and a spurious local minimizer evidently captures the optimization iterates that start with the zero initial condition. Not surprisingly, the larger the value of ϵ, the better the penalty term does in reducing $\mathcal{G}$, the measure of the oscillations in the bump geometry.

Now, let's set $\epsilon = 2 \times 10^{-4}$, which is in the range of values for the penalty parameter which appeared to give solutions which reasonably match the target values. Using this value, let's examine the level curves of the functional on a two-dimensional slice of parameter space; these are displayed in Figure 14. It seems that the spurious local minimizer is now not present. Of course, the global minimizer of $\mathcal{J}_{0.0002}^h$ is not the same as the global minimizer of $\mathcal{J}^h$; this is the price one has to pay for regularization.

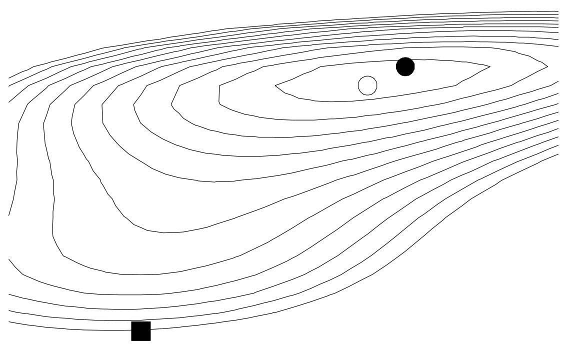

Figure 14: Level curves of the penalized cost functional (with $\epsilon = 0.0002$) and the positions of the global minimizer of that functional (open circle), the global minimizer of the unpenalized functional (filled circle), and the spurious local minimizer of the unpenalized functional (filled square).

5.1 A hybrid algorithm

We still have not accomplished our goal of finding the global minimizer of $\mathcal{J}^h$. Given that we now suspect that, for an appropriate value of ϵ, the spurious minimum is not present and thus we can get reasonably close to the global minimizer of $\mathcal{J}^h$, we can develop the following two-stage *hybrid algorithm*:

1. step through the optimization method for the penalized functional $\mathcal{J}^h_\epsilon$ until satisfactory convergence is achieved; we are now located at the global minimizer of the penalized functional;

2. using the results of stage 1 as an initial condition, step through the optimization for the unpenalized functional $\mathcal{J}^h$ until satisfactory convergence is achieved.

In this way, we trust that the iterates will converge to the global minimizer of the unpenalized functional.

The optimal values of the parameters determined by the hybrid algorithm are

$$\alpha_0 = 0.5000, \qquad \alpha_1 = 0.3735, \qquad \alpha_2 = 0.5001, \qquad \text{and} \qquad \alpha_3 = 0.3747.$$

At last we have obtained a very good match to the parameters of the target flow!

The convergence tolerances used in the two stages of the hybrid algorithm need not be the same. In fact, it is usually more efficient to use a looser tolerance for stage 1 since one is not interested in obtaining accurate minimizers of the penalized functional. The role of the iteration using the penalized functional is merely to produce a good enough initial condition for the optimization of the unpenalized functional, where "good enough" means we are at a point in parameter space which is in the attraction region of a true, i.e., not spurious, minimizer of the unpenalized functional. In our example, optimization of the penalized functional produced the parameter values $\alpha_0 = 0.5004$, $\alpha_1 = 0.2764$, $\alpha_2 = 0.4952$, and $\alpha_3 = 0.3649$ which are very good initial guesses for the optimization of the unpenalized functional.

The hybrid algorithm requires the choice of two parameters in addition to whatever parameters are set for the optimization of the unpenalized functional. One must choose a penalty parameter, e.g., ϵ in (17), and a convergence tolerance for the optimization of the penalized functional.

6 Concluding remarks

In the context of a *very simple,* low Reynolds number, steady, incompressible viscous flow in a channel, we have shown how three pitfalls can be encountered in the numerical approximation of flow control and optimization problems. We have also shown how these pitfalls can be avoided or circumvented, most notably through the regularization of the objective functional. We now discuss some of the implications that our observations have on more *complex* and more practical flow control and optimization problems.

6.1 Insensitive functionals

The first pitfall we encountered is insensitive functionals, i.e., the functional is insensitive to changes in one or more of the design parameters. For example, the functional chosen in section 2 was extremely insensitive to the parameters α_1, α_2, and α_3 which determine the shape of the bump.

There are actually two ways to view insensitive functionals, depending on whether the central goal of optimization is to extremize the objective functional as well as possible or is to find, in some sense, “good” values for the design parameters. In many, if not most, practical flow control and optimization problems, the central goal is the first of these. The choice for the mathematical objective functional is dictated by the physical objective one wants to achieve. If the goal of optimization is, e.g., to make a given objective functional as small as possible, what does the insensitivity of the cost functional with respect to a design parameter tell us? The answer is simple: we can eliminate that design parameter from the problem since it is useless for meeting our objective. For example, if in the simple problem of section 2 one really wanted to match the horizontal velocity component at the downstream position $x_m = 9$ to that of the target flow, one should not have used the bump parameters α_1, α_2, and α_3 as design parameters. Instead, one should have fixed these parameters to some convenient values, e.g., $\alpha_1 = \alpha_2 = \alpha_3 = 0$, and optimized with respect to the single inflow mass rate parameter α_0. Clearly, for any fixed values for α_1, α_2, and α_3, we can do a wonderful job of meeting the objective of making the functional (6) with $x_m = 9$ small using the single parameter α_0 for optimization. Of course, in practical situations, one may find that after eliminating any useless design parameters, the remaining parameters are not sufficient in number or effect to satisfactorily achieve the physical objective; in this case, one may be able to identify other effective design parameters which can be added to the optimization process; if not, then one must be content with what the remaining design parameters can do.

The second view of insensitive functionals for which the goal of optimization is to find a “good” set of design parameters, does arise in practice. If the central goal of optimization is to determine “good” parameter values and one finds that the cost functional is insensitive to one or more of those parameters, then one must change the cost functional to one that is more sensitive. For example, in section 2, we were not happy with the values of the bump parameters produced by the optimizer, even though the cost functional was clearly rendered as small as our tolerances allowed. We then went on, in section 3, to change the objective functional so that it is more sensitive to the bump parameters.

For both views of insensitive cost functional, one has to identify which design parameters have negligible or insufficient effect on the cost functional. Often, intuition can be used to eliminate such design parameters from the very start. For example, in section 2, it should have been obvious to us that the bump parameters were useless in the context of the functional (6) with $x_m = 9$. Thus, we either change the cost functional (the second view) or not use the bump parameters for optimization (the first view). Where intuition fails, sensitivity analyses can be very useful. For example, even if we had no intuition about the problem of section 2, a cursory look at Figure 3 would convince us that the second bump parameter α_2 is useless for affecting the value of the functional (6) with $x_m = 9$.

Thus, insensitivities of the cost functional can be used to

- *reduce the number of design parameters* by eliminating from the design process those parameters which do not appreciably affect the cost functional; or

- *induce changes in the choice of design parameters* by replacing the useless parameters by others that have a greater effect on the cost functional; or

- *induce changes in the cost functional* so that it becomes more sensitive to the design parameters.

All three of these will result in more efficient use of the optimizer, e.g., fewer iterations, and/or better results, e.g., lower functional values or better design parameter values.

6.2 Inconsistent gradients

The second pitfall we encountered was inconsistent approximate gradients of the objective functional. (Of course, we assume that the functional is differentiable.) This occurred when we used the differentiate-then-discretize approach. If one instead uses a discretize-then-differentiate approach, this is very unlikely to happen. In fact, if exact differentiations are used, as is the case when one employs automatic differentiation software, one determines the exact gradient of the discretized cost functional so that gradients cannot be inconsistent. If one uses a finite difference quotient approximation to the gradient of the functional, then one can always obtain consistent gradients by choosing small enough variations in the values of the parameters, e.g., by making $\delta\alpha_k$ small enough in formulas such as (7).

The approximate gradients found by the differentiate-then-discretize approach are not the exact gradients of any functional. However, they are approximations to the gradients of both the continuous and discretized functionals. If the approximate sensitivities are sufficiently accurate, e.g., if a sufficiently fine grid is used to determine sensitivities, then functional gradients found by formulas such as (15) should be consistent. After all, as the grid size tends to zero, approximate gradients found this way should converge to the exact gradient of the continuous functional. However, it may be the case, as it was in section 3, that a grid that is sufficiently fine to result in acceptable flow approximations may not result in sensitivities that are sufficiently accurate to avoid inconsistencies in the gradient of the functional. This is a situation one would like to avoid, i.e., having to use a finer grid for sensitivity calculations than is necessary for flow solutions.

At first glance, it would seem that the possibility of encountering inconsistent approximate gradients renders differentiate-then-discretize approaches much less desirable than discretize-then-differentiate approaches. There are two factors that should be taken into consideration before one jumps to this conclusion. The first is that the two most commonly used discretize-then-differentiate approaches are, for the same computational parameters, more costly than the sensitivity equation method. Finite difference quotient approximations to the gradient of the functional require additional expensive nonlinear flow solutions, in fact, at least one additional solution for each design parameter. Automatic differentiation software usually involve substantial overhead which makes them more costly to run than do hand-coded sensitivity equations software. However, there is a more important reason not to dismiss differentiate-then-discretize approaches, namely that regularization of the cost functional usually eliminates the occurrence of inconsistent approximate gradients. Furthermore, in the vicinity of true minimizers, computed gradients, even for differentiate-then-discretize approaches, are usually consistent. As a result, the hybrid algorithm of section 5.1 can be used not only to avoid spurious minima, but also to avoid inconsistent gradients.

6.3 Spurious minima

The third pitfall we encountered was artificial, numerically induced spurious minima. We also saw in Table 5 and Figure 13 an indication that as the grid size is reduced, the spurious minimizer does not disappear, but instead represents a point in parameter space that moves further away from the true minimizer. The implication of this observation is that for a sufficiently small grid size, the spurious minimizer will be sufficiently far way from the true minimum so that a reasonable starting guess, e.g., the origin in parameter space, would belong to the attraction region of the true minimizer. Thus, it seems that one can circumvent the spurious minima pitfall by using a sufficiently small grid size. This conclusion is of little practical utility. As we saw in our simple example, a good optimizer can yield iterates that converge to a spurious minimizer for grid sizes that are adequate to obtain good flow approximations. On the other hand, for the sake of efficiency, one does not want to use a grid size smaller than that needed to obtain acceptable flow approximations. Thus, reducing the grid size is not usually a viable way to circumvent spurious minima. Fortunately, in section 5, we saw that a more practical means of avoiding the spurious minima pitfall is through the regularization of the cost functional.

References

[1] J. Appel and M. Gunzburger; Difficulties in sensitivity calculations for flows with discontinuities, *AIAA J.* **35**, 1997, pp. 842–848.

[2] C. Bischof, P. Khademi, A. Mauer, and A. Carle, Adifor 2.0: automatic differentiation of Fortran 77 programs. *Comput. Sci. Engrg.* **3** 1996, pp. 18–32.

[3] J. Dennis and R. Schnabel; *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, Englewood Cliffs, 1983.

[4] G. Farin; *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, Academic, San Diego, 1988.

[5] D. Gay, Algorithm 611 subroutines for unconstrained minimization using a model/ trust-region approach, *ACM Trans. Math. Soft.* **9**, 1983, pp. 503-524.

[6] V. Girault and P. Raviart; *Finite Element Methods for Navier-Stokes Equations*, Springer, Berlin, 1975.

[7] M. Gunzburger; Sensitivities in computational methods for optimal flow control, in *Computational Methods for Optimal Design and Control*, Birkhäuser, Boston, 1998, pp. 197–236.

[8] M. Gunzburger; Sensitivities, adjoints, and flow optimization, *Inter. J. Num. Meth. Fluids.* **31**, 1999, pp. 53–78.

[9] M. Gunzburger; Adjoint equation-based methods for control problems in viscous, incompressible flows, to appear in *Flow, Turbul., Comb.*

[10] N. Rostaing, S. Dalmas, and A. Galligo; Automatic differentiation in Odysee, *Tellus* **45a** 1993, pp. 558–568.

[11] X. Wu, E. Cliff, and M. Gunzburger; An optimal design problem for a two dimensional flow in a duct, *Opt. Control Appl. Meth.* **17** 1996, pp. 329–339.