

# Chapter 10

## A POORLY SCALED OPTIMIZATION

### 10.1 Introduction

In this chapter, we consider a problem which can not be accurately optimized because the cost functional has been poorly chosen. We discuss the behavior of the optimization package, and the reasons for its failure. We examine the computed cost gradient and question its accuracy. We then look closely at the velocity sensitivity field, and see clear reasons for the failure of the optimization. We then propose a simple cure for the problem.

### 10.2 A Poor Optimization Using Discretized Sensitivities

We consider one of our standard problems, with four parameters: a single inflow parameter  $\lambda$ , and three parameters  $\alpha = (\alpha_1, \alpha_2, \alpha_3)$  which determine the shape of the bump. The Reynolds number  $Re$  is fixed at 1.

We set a profile line at  $x_s = 9$ , quite far down the channel from the bump. We then set the target parameter values  $\lambda^T = 0.500$  and  $\alpha^T = (0.375, 0.5, 0.375)$ , and compute the target

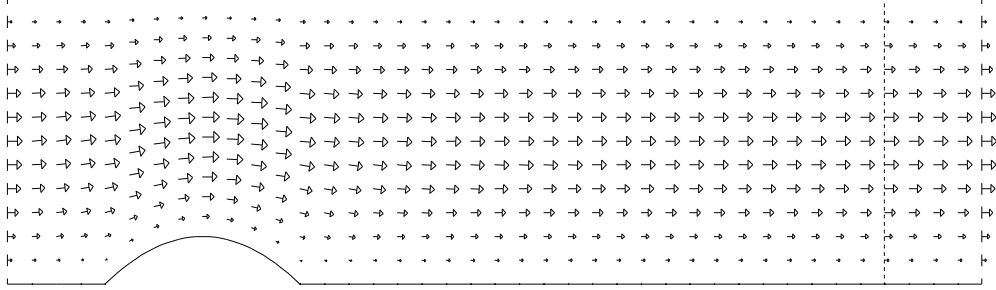


Figure 10.1: The flow region and velocity field for the target parameters. Note the profile line, at  $x_s = 9$ .

flow solution:

$$(u^T, v^T, p^T) = (u(\lambda^T, \alpha^T), v(\lambda^T, \alpha^T), p(\lambda^T, \alpha^T)). \quad (10.1)$$

which is displayed in Figure 10.1.

We may now define the cost functional  $J_1^h$ :

$$J_1^h(u^h, v^h, p^h, \lambda, \alpha) = \int_{x_s=9} (u^h(x_s, y) - u^T(x_s, y))^2 dy, \quad (10.2)$$

and in the usual way, define  $\mathcal{J}_1^h$  to be  $J_1^h$  restricted to flow functions that are consistent with the Navier Stokes equations as determined by the parameters:

$$\mathcal{J}_1^h(\lambda, \alpha) \equiv J_1^h(u^h(\lambda, \alpha), v^h(\lambda, \alpha), p^h(\lambda, \alpha), \lambda, \alpha) \quad (10.3)$$

$$= \int_{x_s=9} (u^h(x_s, y, \lambda, \alpha) - u^T(x_s, y))^2 dy. \quad (10.4)$$

We now face the problem of finding parameter values  $\lambda$  and  $\alpha$  which minimize  $\mathcal{J}_1^h$ . We use a discretization of the flow problem with mesh parameter  $h = 0.25$ . We begin the optimization attempt with  $\lambda$  and  $\alpha$  set to zero. Initially, we set the optimization error tolerance, called **TolOpt**, to  $1.0E - 09$ . When the optimization code requests a value of the gradient of  $\mathcal{J}_1^h(\lambda, \alpha)$ , we compute an approximation of the answer by computing the discretized sensi-

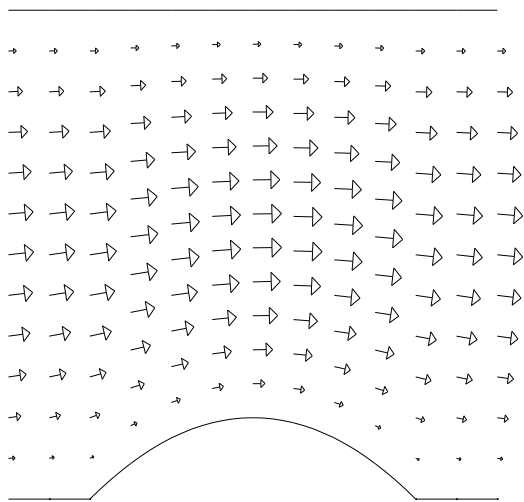


Figure 10.2: The bump and velocity field for the target parameters.  
This figure shows the portion of the flow region near the bump.

tivities  $((u_\lambda)^h, (v_\lambda)^h, (p_\lambda)^h)$  and  $((u_\alpha)^h, (v_\alpha)^h, (p_\alpha)^h)$  and applying Equation (9.9) to compute an approximation to the desired gradient, which we denote  $(\nabla \mathcal{J}_1(\lambda, \alpha))^h$ .

After 20 steps, the optimization code reports satisfactory convergence. However, the parameters that the optimization code returns are  $\lambda = 0.499998$ ,  $\alpha = (0.0767, 0.0656, 0.276)$ . These values seem unreasonably far from the target solution. A comparison of Figures 10.2 and 10.3 confirms that the difference between the shapes of the target and computed bumps is considerable. Despite this fact, the cost functional, which started out at  $\mathcal{J}_1^h(\lambda, \alpha) = 0.400$ , has dropped to  $0.572E - 09$ , very close to the minimum value of 0.

Seeking a set of parameters that are closer to the correct values, we tried lowering the tolerance **TolOpt**. The first reduction in the tolerance produces a “better” result in just 23 steps; the new bump has risen closer to its maximum height of 0.5, but the peak occurs at the *end* of the bump rather than at the middle. If we reduce the tolerance by a factor of 10 again, the optimization code fails to converge after 100 steps, and produces results almost

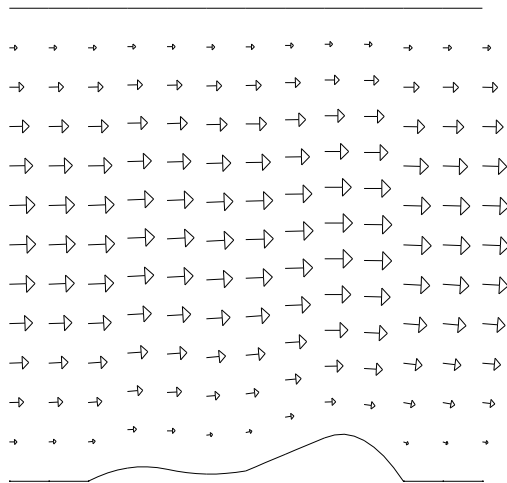


Figure 10.3: The computed bump and velocity field using discretized sensitivities.  
This is the solution computed for **TolOpt**=1.0E-9.

Table 10.1: Optimization results for  $\mathcal{J}_1^h$ , using discretized sensitivities.  
The third optimization did not converge after 100 steps.

<b>TolOpt</b>	Steps	$\lambda$	$\alpha_1$	$\alpha_2$	$\alpha_3$	$\mathcal{J}_1^h$	$\ (\nabla \mathcal{J}_1)^h\ _\infty$
1.0E-09	20	0.499998	0.0767	0.0656	0.276	0.572E-09	0.3E-05
1.0E-10	23	0.500000	0.1257	0.1075	0.453	0.375E-10	0.3E-05
1.0E-11	100	0.500001	0.1257	0.1075	0.453	0.358E-10	0.8E-09

identical to the previous set. The results are summarized in Table 10.1, where we list the optimization tolerance, the number of steps, the final point of the optimization, the cost functional evaluated at that point, and the maximum norm of the (approximate) gradient of the cost functional. It is clear that this problem is beyond the abilities of the optimization code to correct, and that we may need to look elsewhere for a remedy.

## 10.3 Approximating the Cost Gradient with Finite Differences

It is reasonable to assume that the optimization algorithm failed because of inconsistent data. In other words, we had asked the optimization algorithm to seek minimizers of the cost functional, and we provided it with an approximation to the gradient of the cost functional. However, especially near the end of an optimization, errors in the gradient can delay or defeat further progress.

We can investigate this possibility, since we have an alternative, and presumably more accurate, approximation of the cost gradient, via finite cost gradient differences. That is, once we had a flow solution  $(u, v, p)$  for a particular set of parameters  $\beta$ , and the corresponding cost  $\mathcal{J}_1^h(\beta)$ , we perturbed each component of  $\beta$  in turn, solved the corresponding flow problem, evaluated the cost of the solution, and used the formula:

$$\frac{\partial \mathcal{J}_1^h(\beta)}{\partial \beta_i} \approx \frac{\mathcal{J}_1^h(\beta + \Delta\beta_i) - \mathcal{J}_1^h(\beta)}{\Delta\beta_i}. \quad (10.5)$$

Using this modification, we again solved the flow optimization problem with **TolOpt** set to  $1.0E - 09$ . The optimization results were almost ludicrous: while the inflow parameter was well approximated, the bump parameters were all identically zero! We repeated the effort three times, with successively smaller optimization tolerances. None of these efforts produced any significant improvement in the bump parameters. The results are summarized in Table 10.2.

Unlike the previous optimization, this time the optimization code reported unsatisfactory convergence behavior, of the type “false convergence”. This message may be interpreted to mean that the optimization code believes there is a discrepancy between the cost functional and the gradient. This is quite a surprise, since we specifically sought to improve the gradient calculation. However, as will become plain later, this particular problem is especially

Table 10.2: Results for  $\mathcal{J}_1^h$ , using finite cost gradient differences.

<b>TolOpt</b>	Steps	$\lambda$	$\alpha_1$	$\alpha_2$	$\alpha_3$	$\mathcal{J}_1^h$	$\ \nabla \mathcal{J}_1^h\ _\infty$
1.0E-09	10	0.499996	0.0	0.0	0.0	0.286E-08	0.2E-06
1.0E-10	13	0.499996	0.0	0.0	0.0	0.286E-08	0.2E-06
1.0E-11	16	0.499996	0.9E-12	0.8E-12	0.3E-11	0.286E-08	0.2E-06
1.0E-12	20	0.499999	0.2E-11	0.1E-11	0.7E-11	0.286E-08	0.2E-06

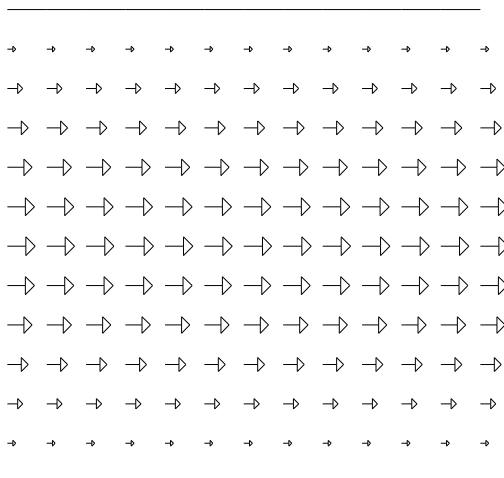


Figure 10.4: The computed bump and velocity field using a finite difference cost gradient.

unsuitable for such finite difference estimates.

## 10.4 Approximating the Sensitivities with Finite Differences

As a final approach to solving this problem, we tried going back to the chain rule calculation of the cost gradients, Equation (9.8). However, this time, we tried to approximate the sensitivities for the discrete problem by finite coefficient differences. That is, once we had computed each of the finite element coefficients  $u_k^h(\beta)$ , we computed  $u_k^h(\beta + \Delta\beta)$ , and made

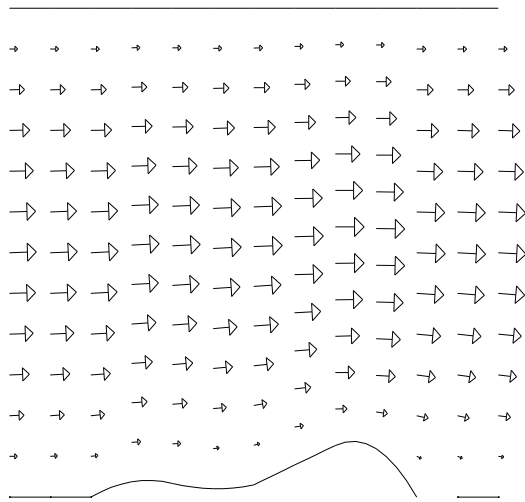


Figure 10.5: The bump and velocity using finite differences for sensitivities.

the approximation:

$$(u_k^h)_\beta(\beta) \approx \frac{(u_k^h)(\beta + \Delta\beta) - u_k^h(\beta)}{\Delta\beta}. \quad (10.6)$$

We already have discussed the fact that this finite difference approximation to  $\frac{\partial u^h}{\partial \beta_i}$  has a built-in inaccuracy whenever moving nodes are involved. Nonetheless, we wished to compare this approach with the use of discretized sensitivities.

A glance at Figure 10.5 will quickly confirm that the bump corresponding to the parameter values we have achieved looks nothing like the bump for the target data. And yet the cost functional is quite low, so that the optimization algorithm seems to have done part of its job; it has decreased the cost functional, though not enough.

But how can the cost functional be so low, while the bump parameters are so far from being correct? Perhaps it is the specification of the cost functional itself that is at fault. The fact that the computed optimizing bump is so dramatically far off from the correct one suggests to us that  $\mathcal{J}_1^h$  is not a very good measure of closeness.

Table 10.3: Results for  $\mathcal{J}_1^h$ , using finite coefficient differences.

<b>TolOpt</b>	Steps	$\lambda$	$\alpha_1$	$\alpha_2$	$\alpha_3$	$\mathcal{J}_1^h$	$\ \nabla \mathcal{J}_1^h\ _\infty$
1.0E-09	19	0.499998	0.084	0.074	0.316	0.321E-09	0.4E-05
1.0E-10	45	0.500004	0.114	0.100	0.426	0.155E-10	0.8E-05
1.0E-11	46	0.500001	0.112	0.098	0.419	0.334E-11	0.4E-08
1.0E-12	40	0.500001	0.112	0.099	0.422	0.318E-11	0.1E-09

## 10.5 Investigating the Poor Convergence

We have tended to judge the optimization by how well it “rediscovered” the parameter values  $\beta^T$  that were used to define the target data  $u^T(x, y)$ . This is a natural expectation, but it is not, strictly speaking, proper.

The optimization code’s task is simply to produce a set of parameters  $\beta$  that are “locally optimal” to within the specified tolerance **TolOpt**. This means, roughly, that the parameters  $\beta$  produce a value of  $\mathcal{J}_1^h$  that is believed to be very near to the minimum value in some small neighborhood. In other words, no “large” decrease in  $\mathcal{J}_1^h$  can be expected by taking a small step away from the estimated solution  $\beta$ .

Can we convince ourselves that the optimization code has actually performed its task well, despite our distaste for the results? To believe that this is so, we must show that no small step away from the estimated minimizer produces a large decrease in the cost. But this is quite easy to check. For instance, consider the very first solution we got, using discretized sensitivities. It lies at a Euclidean distance of about 0.53 units from the true solution, in parameter space. Over that distance, the cost functional decreases from 0.572E-09 to 0. If we suppose that in the neighborhood of the computed and correct minimizers, the cost functional does not vary significantly from these two values, then it is quite reasonable for the optimization code to assume that it has reached a minimizer (since the gradients must be extremely small) and that no “significantly” better minimizer lies close to the computed

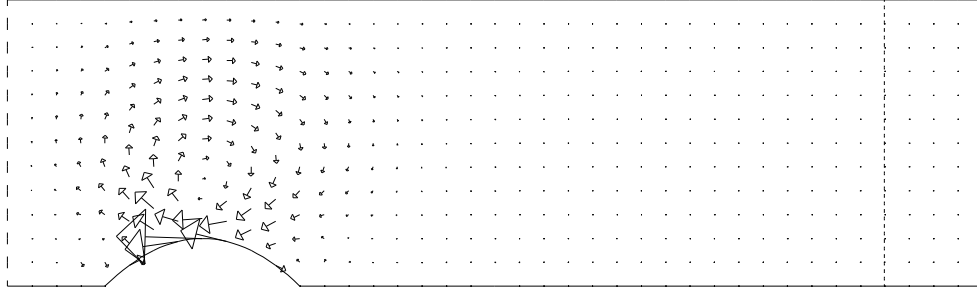


Figure 10.6: Discretized velocity  $\alpha_2$ -sensitivity field.  
The profile line is shown at  $x_s = 9$ .

one. In fact, given such small cost functional values, it would not be implausible to imagine that there is actually a slight local minimum at the computed point. These remarks apply to the points returned in the other computations as well, even to the displeasing results of using finite difference cost gradients.

This is not a satisfactory state of affairs. The values of the cost functional seem to be unusually small, even for parameters that are quite far from the target values. This makes for a very difficult optimization indeed. It suggests that the cost functional is relatively *insensitive* to changes in the parameters. It would be tempting to assume that therefore the flow solution  $(u^h, v^h, p^h)$  is also insensitive to parameter changes, but this is not correct, as is made plain when we plot the discretized sensitivities over the flow region, in Figure 10.6.

The figure makes the problem clear: the flow solution *as a whole* is not insensitive to parameter changes, but the portion of the solution along the profile line  $x_s = 9$  is indeed very insensitive. This is actually a feature of low Reynolds number flow through a channel; even when disrupted by the bump, the flow has a very strong tendency to return to Poiseuille flow once it is again passing through a rectangular region. Thus, by the time the flow has reached the profile line, almost all traces of the influence of the bump have been overwhelmed.

Once we understand the meaning of this sensitivity plot, a reasonable solution is clear. We don't want to try to change the physics of the flow, but we are free to decide where we make measurements (and for that matter, *what* we measure there). If we simply move the profile line from  $x_s = 9$  to some point much closer to the bump, then changes in the parameters will have a much larger effect on the portion of the flow solution that is sampled. This, in turn, will make the cost functional values larger, and much less flat. And this, finally, will make the optimization problem significantly easier to handle.

In fact, for all further computations, we will move the profile line right up to the back of the bump, setting  $x_s = 3$ . In the next chapter, we will begin using this new cost functional. Although the convergence behavior will be much improved, we will discover a new complication!