

Chapter 11

A DISCRETIZED SENSITIVITY FAILURE

11.1 Introduction

Having learned from the experiences with the profile line at $x_s = 9$, we now move the line much closer to the bump, and attempt to optimize the new problem.

We find that the optimization code returns a solution which is not the global minimizer, that is, the target parameters we used to set up the problem. Moreover, the returned solution does not even seem to be a local minimizer; the cost gradient entries are far too large. It is necessary that we understand what has happened before proceeding.

After a brief consideration of the numerical evidence, we turn to graphical analysis. We plot the cost functional along a line between the computed point and the global minimizer, and then consider the value of the approximated cost gradient in this direction.

We then carry out a similar analysis in a plane, which allows us to conclude that the cost gradients are not being adequately approximated for this problem.

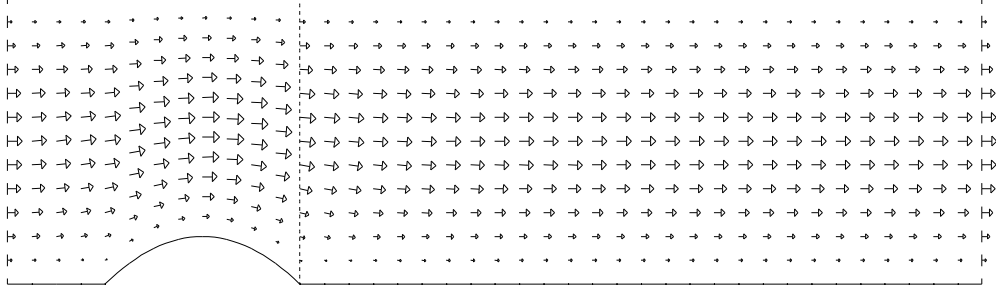


Figure 11.1: The target flow region, velocity field, and profile line.
The profile line has been moved to $x_s = 3$.

11.2 A Stalled Optimization

The experiments described in this chapter were all carried out on a problem with four free parameters: λ , representing the strength of the parabolic inflow, and α_1 , α_2 , and α_3 , used to control the shape of a cubic spline that represented a bump extending from $x = 1$ to $x = 3$. The Reynolds number parameter Re was held fixed at a value of 1.

The target profile data was generated by computing a flow solution corresponding to an inflow parameter $\lambda^T = 0.5$ and shape parameters $\alpha^T = (0.375, 0.5, 0.375)$, and measuring the state variables along the profile line, which we have moved quite close to the bump, at $x_s = 3$, as shown in Figure 11.1. The optimization code was given a zero starting estimate for values of the parameters.

We define the cost functional in terms of the state variables as:

$$J_2^h(u^h, v^h, p^h, \lambda, \alpha) = \int_{x_s=3} (u^h(x_s, y) - u^T(x_s, y))^2 dy, \quad (11.1)$$

and, in the usual way, define the constrained cost functional:

$$\mathcal{J}_2^h(\lambda, \alpha) \equiv J_2^h(u^h(\lambda, \alpha), v^h(\lambda, \alpha), p^h(\lambda, \alpha), \lambda, \alpha) \quad (11.2)$$

$$= \int_{x_s=3} (u^h(x_s, y, \lambda, \alpha) - u^T(x_s, y))^2 dy. \quad (11.3)$$

Table 11.1: Optimization results for \mathcal{J}_2^h .

TolOpt	Steps	λ	α_1	α_2	α_3	\mathcal{J}_2^h	$\ (\nabla \mathcal{J}_2^h)^h\ _\infty$
1.0E-09	33	0.505815	-0.0185	0.4352	-0.0448	0.390E-03	0.2E-01

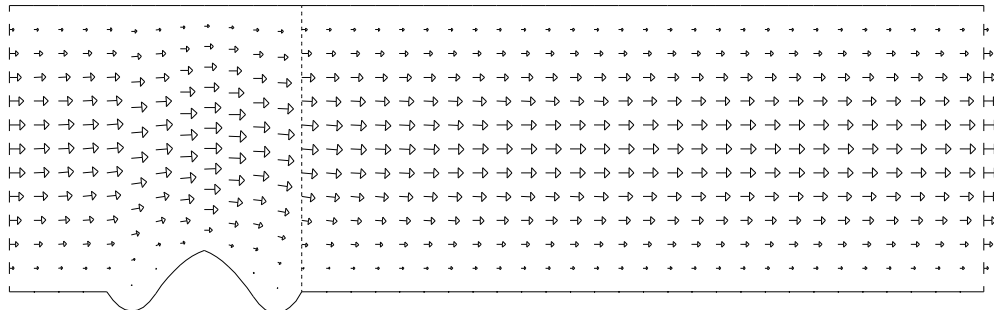


Figure 11.2: The flow produced with discretized sensitivities.

The cost functional at the starting point was $\mathcal{J}_2^h(0, 0, 0, 0) = 0.430$.

We approximate the cost gradient using discretized sensitivities. We present the result of the optimization in Table 11.1. The results presented are for an optimization tolerance of **TolOpt**=1.0E-9, but essentially identical results were achieved for tolerances of 1.0E-10, 1.0E-11 and 1.0E-12.

In this table, the value of the cost functional at the computed minimizer seems far too large. We happen to know that this value should be zero, although the optimization code has no way of knowing this. But more tellingly, the cost gradient is quite large. This is a serious problem, since the gradient must be zero at a minimizer, whether local or global, and the optimization code knows this. This means that it is very unlikely that the solution returned is even a local minimizer.

The picture of the computed flow in Figure 11.2 is disturbing, since we clearly have not achieved the secondary goal of reproducing the shape of the target bump.

The discrepancies between the known target and the solution returned by the optimization code are so serious that we must subject the solution to a series of tests to see if we are justified in discarding it, and to try to determine what went wrong.

11.3 Checking the Results Along a Line

We are now going to consider the optimization results that we got using the discretized sensitivities in the computation of the cost gradient. We have already made three simple checks. We compared the cost functional value at the computed minimizer to the known minimum value of 0. The value of 0.390E-03 does not seem suitably small, particularly since we are carrying out a double precision calculation. The second check we made was to look at the gradient of the cost functional. The fact that the maximum entry has a size of 0.2E-01 is very disconcerting; at a true local or global minimizer, this value would be 0. Finally, we compared the computed bump values to the target values, and were not satisfied, although we have learned from the previous chapter that this is not always a reliable measure.

With our suspicions raised, it is time to try to gain a feeling for what is going on. While hardly conclusive, some graphic evidence may prove useful here. Let us simply draw a straight line in parameter space between the computed point and the target point, pick evenly spaced parameter values along this line, and evaluate the cost functional and the cost gradient, as approximated with discretized sensitivities.

The actual table of values appears in Table 11.2, but perhaps the simple plot in Figure 11.3 best illustrates the problem. We must be careful to recall that the parameter space is actually four-dimensional, and that we are looking at a very narrow slice through it. Nonetheless, it is clear that something is quite wrong. It is surprising that there is a “hump” between the computed point and the global minimizer. If the computed point actually lies in a “valley”

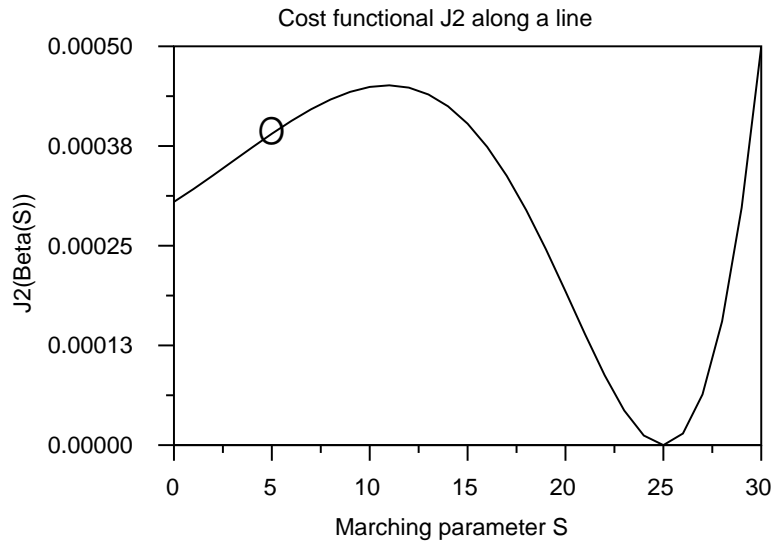


Figure 11.3: Values of $\mathcal{J}_2^h(\beta(S))$ along a line.
 The “optimal” point is marked, at $S=5$.
 The optimization used discretized sensitivities.
 The local maximum on this line occurs at $S=11$.
 The point $S=25$ is the global minimum.

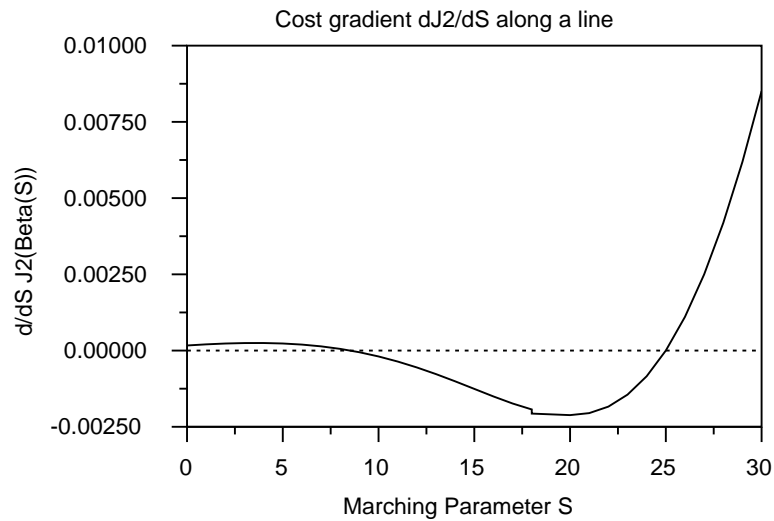


Figure 11.4: Values of $(\nabla \mathcal{J}_2)^h(\beta(S)) * dS$ along a line.
 The optimization used discretized sensitivities.
 Note that the computed slope changes sign between $S=8$ and $S=9$.

then the optimization, by its design, could never rise up out of the valley in order to reach the global minimizer. But this does not explain why the optimization did not then proceed downward to the left, where there are plenty of points with a lower functional value. The picture also shows us that the computed optimizer is not a local minimizer, nor does it seem to be an inflection point or local maximum. The behavior of the optimization code is still a mystery.

So let us now compute the gradient data that the optimization code had to work with. In Figure 11.4, we take the cost functional gradient, as approximated using the discretized sensitivities, and project it along the same line shown in Figure 11.3. Now some serious discrepancies are apparent. We see a local maximum near $S=11$, and a global minimum at $S=25$. Therefore, the correct slope must be positive up to $S=11$, and negative from there to $S=25$. But the computed slope is only positive up to $S=8$. Thus, there is an interval over which the cost functional is *increasing*, but the approximated directional derivative is *negative*. It is precisely this kind of discrepancy that will cause the optimization code to fail.

11.4 Checking the Results in a Plane

Our analysis of the optimization failure along a line has made it clear that something is wrong. In particular, it seems that the approximate gradients are inaccurate. But we'd like to get more evidence of the problem, to try to understand the context in which it occurs, how often it happens, and how severe the errors are.

To do so, we consider looking at the problem in a plane. That is, we start with the global minimizer and the computed minimizer, and specify some auxiliary point, and consider the plane that contains these three points. We then pick points on a regularly spaced grid over the plane, and evaluate the cost functional there. Finally, we make a contour map of the

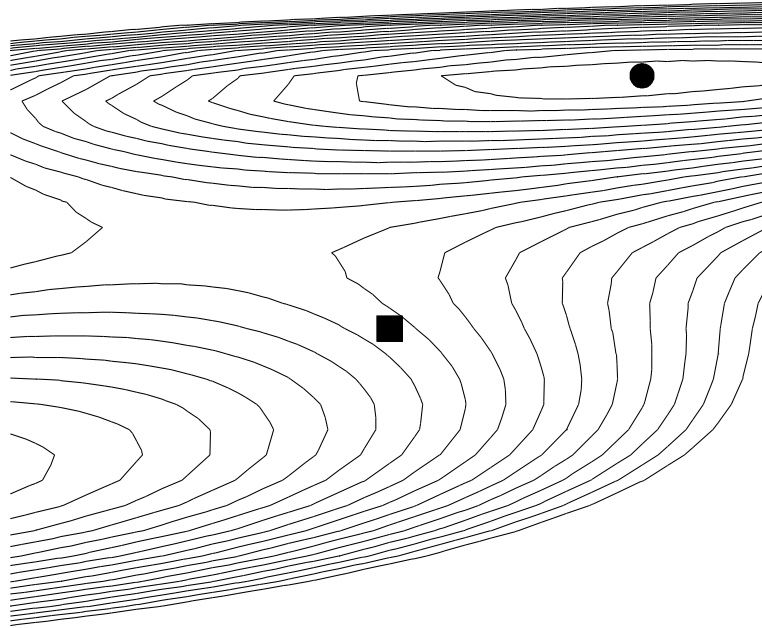


Figure 11.5: A contour map of the cost functional \mathcal{J}_2^h in a plane.
The global minimizer is at the black dot.
The computed minimizer is at the black square.

resulting data. The results are shown in Figure 11.5.

If we imagine a line drawn on this plot that goes from the computed minimizer to the global minimizer, then we can reproduce the one-dimensional plot in Figure 11.3. We can see that the “hump” between the two points is, in part at least, avoidable, simply by moving to the side slightly. From this graph, it seems possible that this could be done in such a way that we did not have to increase the cost functional value.

It is quite interesting to note what seems to be a depression associated with a local minimum, in the lower left corner of the graph. Since we will have occasion to examine a similar phenomenon in the next problem that we study, we will overlook it for now.

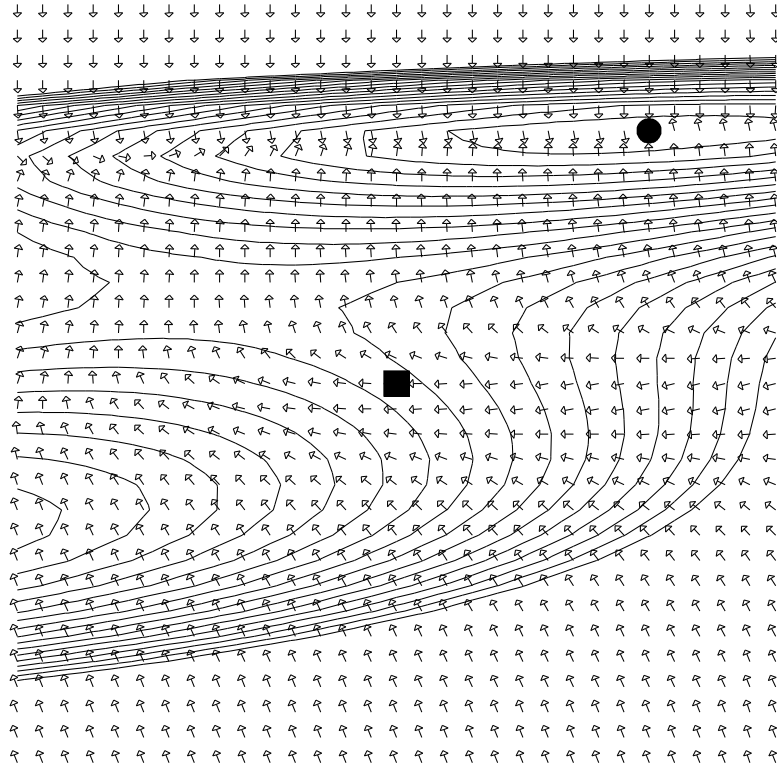


Figure 11.6: \mathcal{J}_2^h gradient directions by discretized sensitivities.
The global minimizer is at the black dot.
The computed minimizer is at the black square.

From the contour plot alone, we cannot see what has gone wrong. But let us overlay this plot with the normalized gradient direction vectors, in Figure 11.6. We have taken the liberty of reversing the direction of the gradient field, so that it points *down*, in the direction of cost functional decrease.

We may compare this plot with Figure 11.7, generated by approximating the cost gradients with the finite difference sensitivities as in Equation (9.11). Consider the requirement that a gradient vector must cross a contour line at a right angle. The discretized sensitivity gradients fail this test, while the finite difference sensitivity gradients behave correctly. Note, moreover, that the discretized sensitivity gradients seem to be completely “unaware” of the local minimizer that lies off the page, in the lower left hand direction.

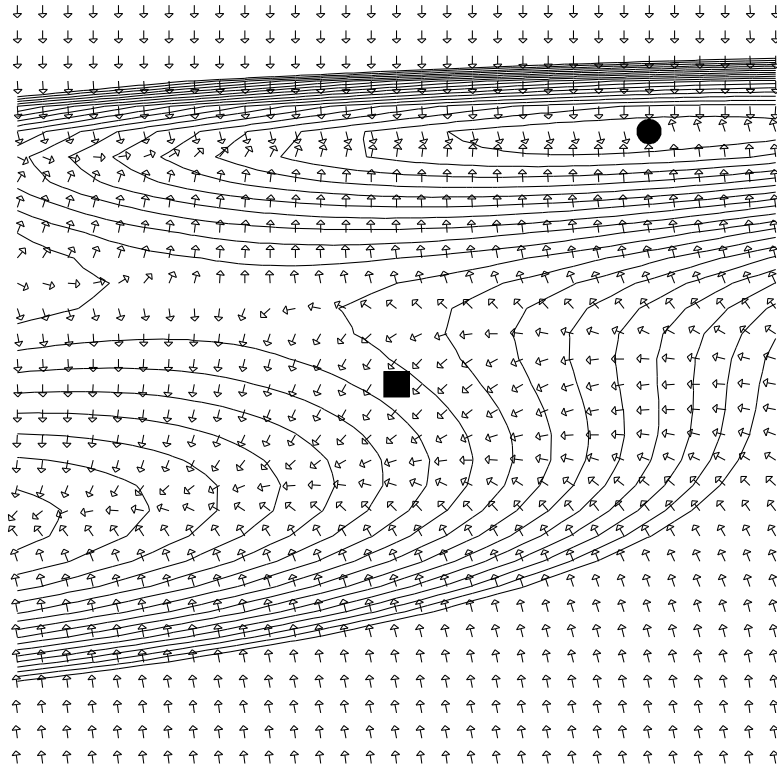


Figure 11.7: \mathcal{J}_2^h gradient directions by finite difference sensitivities.
The global minimizer is at the black dot.
The computed minimizer is at the black square.

Our one-dimensional plot reported the same fact: for discretized sensitivities, the computed cost gradients do not reliably estimate the cost gradient. However, the two-dimensional plot gives a much better feel for the kind of details that can be missed, and for the magnitude of the error.

Accurate computation of the magnitude of the gradient is not as important as getting the *direction* correct. In Figure 11.8, we have magnified the portion of the region around the computed minimizer, and shown the gradient direction fields derived from both the discretized sensitivities and the finite difference sensitivities. We take the finite difference sensitivity gradients to be correct, using the contour lines as guides. Then it is clear that in this small region, there is an abrupt growth of error in gradient direction as we move from top to bottom. Within a few sample points of the computed minimizer, we see points at which the gradient direction error is greater than 90 degrees.

Of course, the evidence from this plot is not conclusive. We are only showing a two-dimensional slice of the parameter space. If we compute the maximum angle between the two four-dimensional gradient estimates, the largest value we compute over the set of points we are displaying is roughly 17 degrees. To actually understand how the accuracy of the gradient direction affects performance, we would have to look at the details of the particular optimization algorithm we are using.

But since we have seen that, at least for this “slice” of the problem, the cost gradients computed via finite difference sensitivities do such a good job of tracking the changes in the cost functional, it is natural that we should use them to repeat the optimization of the same problem we have analyzed here. We make this study in the next chapter.

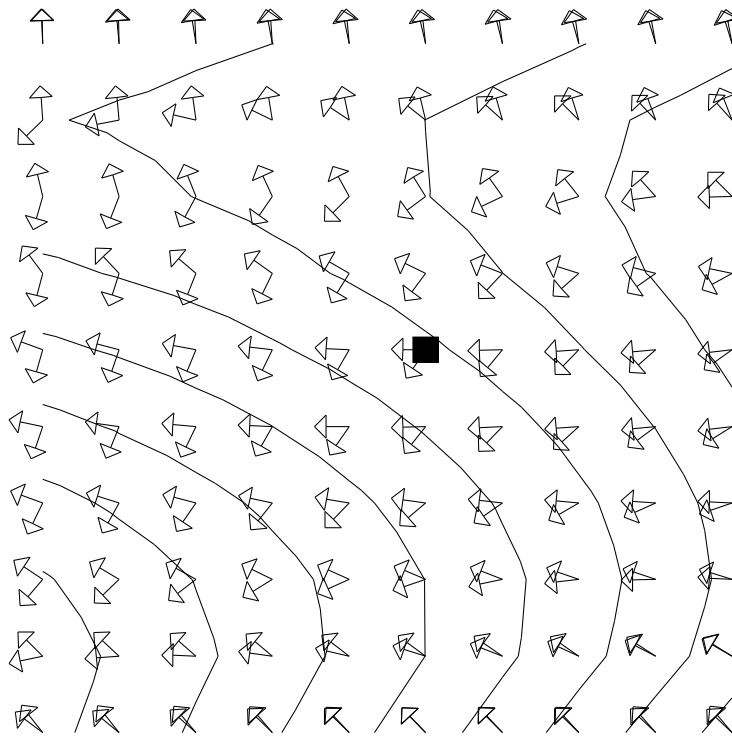


Figure 11.8: Comparison of \mathcal{J}_2^h gradient directions.
 Finite difference and discretized sensitivity approximations are shown.
 The computed minimizer is at the black square.

Table 11.2: Values of \mathcal{J}_2^h along a line in parameter space.
 Point 6 is the optimizer returned when using discretized sensitivities.
 Point 25 is the target parameters.

Note that a local maximum occurs between S=11 and S=12.

Point	λ	α_1	α_2	α_3	\mathcal{J}_2^h $\times 1,000$	$(\nabla \mathcal{J}_2)^h \cdot dS$ $\times 1,000$
0	0.507	-0.117	0.419	-0.149	3.05	0.16
1	0.507	-0.097	0.422	-0.128	3.21	0.20
2	0.506	-0.077	0.425	-0.107	3.38	0.23
3	0.506	-0.057	0.428	-0.086	3.55	0.24
4	0.506	-0.038	0.432	-0.065	3.73	0.24
5	0.505	-0.018	0.435	-0.044	3.90	0.23
6	0.505	0.001	0.438	-0.023	4.06	0.19
7	0.505	0.020	0.441	-0.002	4.21	0.13
8	0.504	0.040	0.444	0.018	4.33	0.05
9	0.504	0.060	0.448	0.039	4.42	-0.05
10	0.504	0.079	0.451	0.060	4.49	-0.19
11	0.504	0.099	0.454	0.081	4.51	-0.35
12	0.503	0.119	0.457	0.102	4.48	-0.55
13	0.503	0.138	0.461	0.123	4.39	-0.77
14	0.503	0.158	0.464	0.144	4.24	-1.00
15	0.502	0.178	0.467	0.165	4.03	-1.25
16	0.502	0.197	0.470	0.186	3.73	-1.50
17	0.502	0.217	0.474	0.207	3.37	-1.73
18	0.502	0.237	0.477	0.228	2.94	-1.93
19	0.501	0.256	0.480	0.249	2.45	-2.06
20	0.501	0.276	0.483	0.270	1.92	-2.11
21	0.501	0.296	0.487	0.291	1.39	-2.05
22	0.500	0.316	0.490	0.312	0.87	-1.83
23	0.500	0.335	0.493	0.333	0.43	-1.44
24	0.500	0.355	0.496	0.354	0.12	-0.84
25	0.500	0.375	0.500	0.375	0.00	0.00
26	0.499	0.394	0.503	0.396	0.14	1.10
27	0.499	0.414	0.506	0.417	0.63	2.49
28	0.499	0.434	0.509	0.438	1.55	4.18
29	0.498	0.453	0.513	0.459	2.97	6.19
30	0.498	0.473	0.516	0.480	4.99	8.51