

# Chapter 13

## OPTIMIZATION WITH A PENALTY FUNCTIONAL

### 13.1 Introduction

In this chapter, we try to avoid the local minimizer we found in Chapter 12 by adding a “penalty functional” to the cost functional. As described in Section 9.11, such a penalty functional can regularize a problem, that is, disqualify certain solutions with undesirable properties; in particular, we prefer solutions with a monotonic bump derivative, and we will use the penalty functional to try to select such solutions over those with “wiggles” or “gutters”.

It is hoped that this regularization will fill in the valley of attraction for the local minimizer seen in the previous chapter, and allow the optimization code to reach the global minimizer. We will see that regularization will allow us to come closer to the global minimizer, but a more elaborate procedure is needed to actually reach that point.

## 13.2 Regularizing the Cost Functional

We have already computed a number of solutions to the optimization problem, and found something objectionable with each. The biggest drawback, of course, was that none of these solutions were the global minimizer.

In a real problem, when the optimization code returns a purported minimizer, we won't know if there is a better, global minimizer. Nonetheless, there are some criteria that can be used to declare that a solution is unacceptable, or undesirable.

For the problems that we have been examining, a wind tunnel engineer might reject every solution with negative bump parameters because the concavities cause turbulence, are too hard to machine, or are simply implausible. Solutions in which the bump almost entirely occludes the channel might also be rejected for practical reasons.

We leave ourselves open to such undesirable results since we haven't placed any constraints on the feasible space of parameters to be examined by the optimization code. We could explicitly add new requirements, in the form of equalities or inequalities that the parameters would have to satisfy. However, this would change the problem to one of *constrained optimization*, adding a layer of complication we prefer to avoid.

It is enough, for our purposes, if we can *discourage* the optimization code from making certain choices, rather than forbidding them. This calls for the use of a penalty function, which allows the optimization code to investigate any set of parameters, but which attempts to make some choices unattractive by artificially increasing the cost functional there.

One possibility for controlling the oscillations of the bump is to compute a penalty functional  $\mathcal{J}_B^h(\alpha)$  based on the integral of the square of the derivative of the bump:

$$\mathcal{J}_B^h(\alpha) = \int_1^3 \left( \frac{\partial Bump(x, \alpha)}{\partial x} \right)^2 dx. \quad (13.1)$$

Table 13.1: Optimization results on  $\mathcal{J}_3^h$  for various values of  $\epsilon$ .

$\epsilon$	Steps	$\lambda$	$\alpha_1$	$\alpha_2$	$\alpha_3$	$\overline{\mathcal{J}}_B^h$	$\overline{\mathcal{J}}_3^h$
1.0	22	0.512681	0.0026	0.0044	0.0032	0.4E-04	0.8E-02
1.0E-01	11	0.512231	0.0273	0.0467	0.0356	0.5E-02	0.8E-02
1.0E-02	19	0.505713	0.1740	0.3262	0.2706	0.2E+00	0.4E-02
1.0E-03	31	0.500561	0.2515	0.4677	0.3665	0.5E+00	0.5E-03
1.0E-04	40	0.500382	0.2849	0.5001	0.3636	0.5E+00	0.6E-04
1.0E-05	47	0.505597	0.1305	0.5442	0.1527	0.1E+01	0.1E-04
1.0E-06	47	0.507467	0.1387	0.5406	0.0712	0.6E+00	0.1E-05
1.0E-07	43	0.507710	0.1405	0.5395	0.0611	0.6E+00	0.4E-06
0.0	44	0.507739	0.1407	0.5394	0.0599	0.6E+00	0.3E-06

Such a penalty functional would be zero for a straight line connecting the beginning and ending of the bump region, low for a small parabola or a piecewise linear function, and high for a curve with wiggles or severe curvature. Our optimization code would then work with a new cost functional  $\mathcal{J}_3^h$  defined by adding a “small” multiple  $\epsilon$  of the bump oscillation integral to the original cost:

$$\mathcal{J}_3^h(\lambda, \alpha) \equiv \mathcal{J}_2^h(\lambda, \alpha) + \epsilon \mathcal{J}_B^h(\lambda, \alpha) \quad (13.2)$$

$$= \int_{x_s=3} (u^h(x_s, y) - u^T(x_s, y))^2 dy + \epsilon \int_1^3 \left( \frac{\partial \text{Bump}(x, \alpha)}{\partial x} \right)^2 dx. \quad (13.3)$$

Here, we don’t include  $\epsilon$  as a parameter of  $\mathcal{J}_3^h$ , because we imagine it as being fixed to some suitable value during the computation.

The optimization code minimizes this new cost functional by attempting to decrease the flow discrepancy while not allowing the bump oscillation integral to get too large.

We ran a series of optimizations on the usual four parameter problem, with the target parameters at  $\lambda^T = 0.5$ ,  $\alpha^T = (0.375, 0.5, 0.375)$ . We considered a series of values of  $\epsilon$  to try to get an idea of the effect of the penalty term on the computed optimizing solution. The results are summarized in Table 13.1.

In this table, it is interesting to see how, as  $\epsilon$  decreases, the computed solution begins to

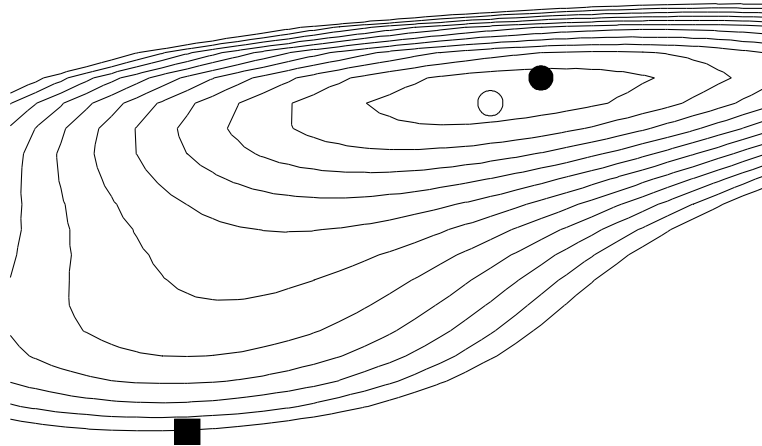


Figure 13.1: Contours for cost functional  $\mathcal{J}_3^h$  with  $\epsilon = 0.0002$ .  
 The open circle marks the minimizer of  $\mathcal{J}_3^h$ .  
 The filled circle marks the global minimizer of  $\mathcal{J}_2^h$ .  
 The square marks the local minimizer of  $\mathcal{J}_2^h$ .

move towards the global minimizer, down to the value  $\epsilon=0.0001$ . Then, the local minimizer becomes more attractive, and as  $\epsilon$  decreases further, the computed solution begins to approach that point instead. Of course, if we actually set  $\epsilon$  to zero, then that's precisely where we end up. From the table, we can conclude that the penalty functional can help, for *certain* values of the penalty parameter  $\epsilon$ ; the value mustn't be too large (or else we're not solving the flow problem we're interested in) but it mustn't be too small (or we'll end up approaching the undesirable local minimizer).

To get an idea of the smoothing effect of this change in the functional, consider Figure 13.1, which displays contour lines of the smoothed functional, for  $\epsilon = 0.0002$ , over the same

two dimensional plane used in Figure 12.3. The addition of the bump term seems to have completely smoothed away the valley containing the local minimizer.

However, the minimizer of the new functional  $\mathcal{J}_3^h$  is *not* the minimizer of the previous functional  $\mathcal{J}_2^h$ . This is to be expected; we could only guarantee that the minimizers were the same in the unlikely case that the global minimizer of  $\mathcal{J}_2^h$  also happened to minimize the penalty function.

### 13.3 Reaching the Global Minimizer

Since we didn't reach the global minimizer, it might seem that our penalty functional approach has not solved our problem. While our real goal was to reach the global minimizer of  $\mathcal{J}_2^h$ , we have reached a minimizer of a related functional, which is merely *near* our desired value.

But it was exactly the task of getting near the global minimizer that we could not accomplish earlier. Just as we would get near, the local minimizer would attract the optimization code, and once we entered the local minimizer's "valley" there was no way to escape.

The penalized functional  $\mathcal{J}_3^h$  gives us a way to restart such a computation from the local minimizer with the hope of exiting the valley. For values of  $\epsilon$  that aren't too small, the valley disappears. Assuming that the global minimizer has a lower value of  $\mathcal{J}_3^h$ , (though not necessarily the *lowest*), it is reasonable to assume that a few steps of optimizing  $\mathcal{J}_3^h$  will move us closer to the global minimizer. Then if we return to optimizing  $\mathcal{J}_2^h$ , we have a chance of reaching the desired solution.

As our first example of this approach, let us start an optimization of  $\mathcal{J}_2^h$  at the usual starting point. When that optimization is completed by reaching the local minimizer, let us use that point as the starting point of a new optimization of  $\mathcal{J}_3^h$ , with  $\epsilon = 0.0002$ . We don't need to

Table 13.2: Results at the end of each stage of the three-stage optimization.

Minimizing	Steps	$\lambda$	$\alpha_1$	$\alpha_2$	$\alpha_3$	$\mathcal{J}_2^h$	$\mathcal{J}_3^h$
$\mathcal{J}_2^h$	44	0.507739	0.1407	0.5394	0.0599	0.311E-06	0.297E-03
$\mathcal{J}_3^h$	15	0.502012	0.3411	0.5380	0.3086	0.144E-04	0.145E-03
$\mathcal{J}_2^h$	26	0.500007	0.3743	0.5000	0.3748	0.498E-09	NA

solve this problem well; we just want to get away from the local minimizer, so let's take 15 steps. Then we will use *that* set of parameters as the starting point for a third optimization of our unpenalized cost functional  $\mathcal{J}_2^h$ . Essentially, we're using the penalized function to “get over the mountain” between the local and global minimizers.

As Table 13.2 makes clear, our 15 steps of minimizing  $\mathcal{J}_3^h$  actually *raise* the value of  $\mathcal{J}_2^h$  as it moves away from the local minimizer. But this is exactly what must happen if we are to climb out of the local valley. Once we are on the other side, it is safe to return to minimization of  $\mathcal{J}_2^h$ , and we finally have the satisfactory result of reaching our global minimizer.

Even though we are optimizing  $\mathcal{J}_3^h$ , there is nothing that says we can't monitor the values of  $\mathcal{J}_2^h$ . If we do so, then we can take a sustained drop in the cost functional value as a sign that we have made it to the other side of the ridge. The actual sequence of cost functional values for the 15 iterates in our case is shown in Figure 13.2.

If we anticipate problems with a local minimizer, or we want to try to rule out certain behaviors with a penalty functional, then we could always begin by seeking a minimizer of the penalized functional  $\mathcal{J}_3^h$ . Then we restart the optimization from that point seeking the minimizer of  $\mathcal{J}_2^h$ . The results of carrying out this procedure on our current problem are shown in Table 13.3. Again, we used a value of  $\epsilon = 0.0002$  to define  $\mathcal{J}_3^h$ . As can be seen, the initial optimization stops quite close to the global minimizer, making the second stage easy.

Thus we have seen that, in the presence of an attractive local minimizer, a penalty functional may be used to escape from the local region of attraction, or, if used in advance, to avoid it

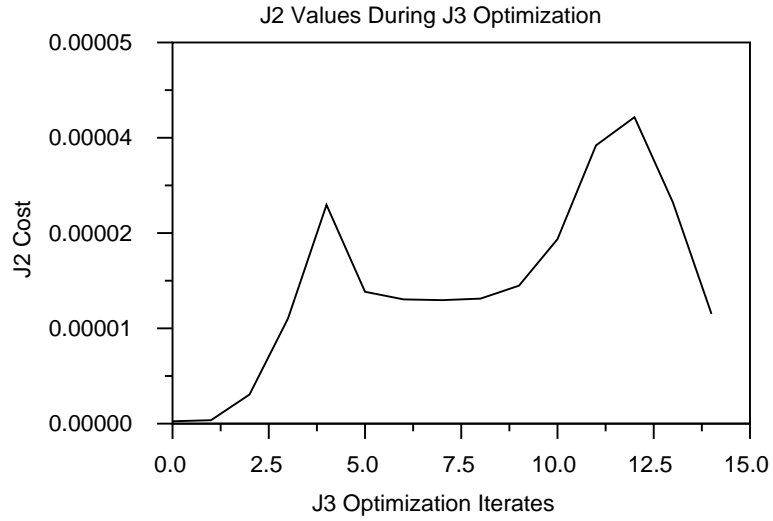


Figure 13.2: Values of  $\mathcal{J}_2^h$  during the optimization of  $\mathcal{J}_3^h$ .

Table 13.3: Results at the end of each stage of the two-stage optimization.

Minimizing	Steps	$\lambda$	$\alpha_1$	$\alpha_2$	$\alpha_3$	$\mathcal{J}_2^h$	$\mathcal{J}_3^h$
$\mathcal{J}_3^h$	45	0.500363	0.2764	0.4952	0.3649	0.194E-05	0.119E-03
$\mathcal{J}_2^h$	26	0.500015	0.3735	0.5001	0.3747	0.283E-09	NA

altogether. A number of *ad hoc* decisions must be made; the most important one being the choice of the form of the penalty functional. Typically, this choice is guided by seeking to eliminate objectionable or unphysical features of the local minimizer. Secondly, an appropriate weight  $\epsilon$  must be chosen when the penalty functional is added to the cost. Finally, the user must decide when to switch between optimization of the penalized and unpenalized cost functionals.