DISCRETIZATION OF COST AND SENSITIVITIES
http://people.sc.fsu.edu/~jburkardt/presentations/...
... bozeman_1994.pdf

# DISCRETIZATION OF COST AND SENSITIVITIES IN SHAPE OPTIMIZATION *

John Burkardt, Max Gunzburger, and Janet Peterson
*Department of Mathematics*
*Interdisciplinary Center for Applied Mathematics*
*Virginia Polytechnic Institute and State University*
*Blacksburg, Virginia, 24061*

## 1 The Physical Problem: A Forebody Simulator

We consider a problem in aircraft engine testing [1], [6]. Of special concern is the influence of the aircraft forebody on the flow that reaches the engine intake. Modern aircraft are too large to place in a wind tunnel; there may not even be room for just the forebody and engine. Resourceful engineers build small "forebody simulators" that roughly reproduce the flow disturbances known to be caused by the real forebody. A typical setup is shown in Figure 1. Designing an effective simulator this way is crude, tedious, and expensive, and computational guidance is desired. This paper investigates computational difficulties arising in a simplified version of this design problem.

## 2 Continuous Mathematical Flow Model

We model the wind tunnel problem by fluid flow in a two dimensional rectangular channel. The forebody is represented by a "bump" that partially obstructs the flow. The fluid obeys the Navier Stokes equations for steady, viscous, incompressible flow:

$$-\nu\Delta\vec{\mathbf{u}} + \vec{\mathbf{u}} \cdot \operatorname{grad}\vec{\mathbf{u}} + \operatorname{grad}\mathbf{p} = \vec{\mathbf{f}} \quad (1)$$

$$\operatorname{div}\vec{\mathbf{u}} = 0 \quad (2)$$

plus appropriate boundary conditions. Here $\vec{\mathbf{u}}$ is the velocity vector, with components $\mathbf{u}$ and $\mathbf{v}$; $\nu$ is the kinematic viscosity; $\vec{\mathbf{f}}$ is a given forcing function.

Our channel has opposite corners at (0,0) and (10,3). If we wish to include a bump, it will start at (1,0) and extend to (3,0), with its height defined by some given function $y = bump(x, \vec{\alpha})$, with $\vec{\alpha}$ a set of parameters.

*Supported by the Air Force Office of Scientific Research under grant AFOSR 93-1-0280, and the Office of Naval Research under grant N00014-91-J-1493.

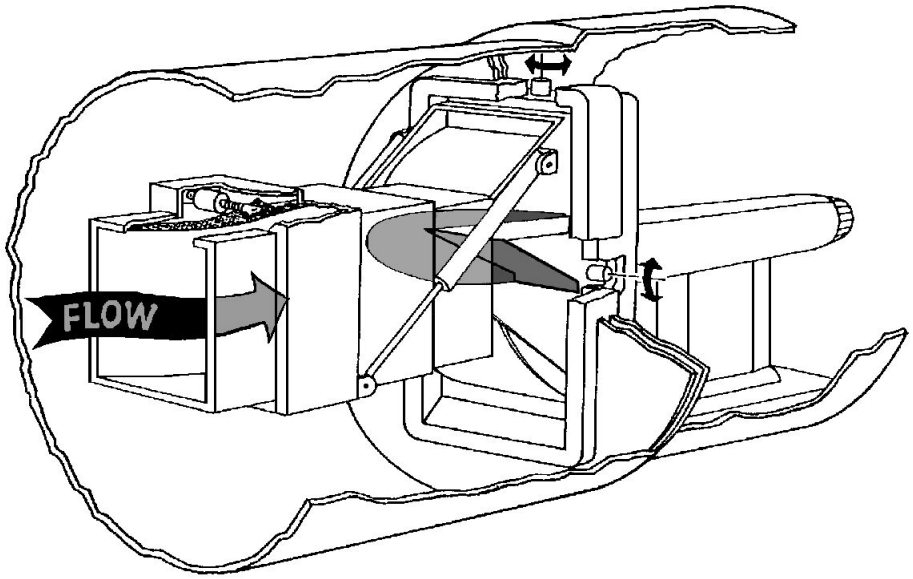


Figure 1: An aircraft engine and forebody simulator in a wind tunnel.

The fluid enters at the left, with velocity $\mathbf{u}(0, y) = inflow(y, \lambda)$, $\mathbf{v} = 0$, where $inflow$ is some given function, and $\lambda$ is a parameter. At the top and bottom of the channel we set the velocity to zero. On the right we set the usual outflow conditions $\mathbf{v} = 0$ and $\dfrac{\partial \mathbf{u}}{\partial x} = 0$.

Together, the flow equations, boundary conditions, and parameters produce a system of equations sufficient to determine the continuous quantities $(\mathbf{u}, \mathbf{v}, \mathbf{p})$ throughout the flow region, symbolized by:

$$\mathbf{G}(\mathbf{u}, \mathbf{v}, \mathbf{p}) = 0 \ . \tag{3}$$

Of course, $\mathbf{G}$ is a function of the parameters $\vec{\alpha}$ and $\lambda$, both explicitly and implicitly, through the dependence of $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{p}$ on the parameters.

Once the flow is determined, we measure the state variables on a fixed vertical plane called the "sampling line", attempting to match a set of given measurements made earlier. The exact evaluation of the discrepancy will be carried out by a cost functional, to be specified.

# 3 Discrete Mathematical Flow Model

Equation (3) is not immediately amenable to computational treatment. We must formulate a discrete set of equations for data which can approximate the

solution of that continuous problem. We use the weak formulation which follows [5]. We represent the region by a mesh of finite elements and approximate the continuously varying state variables **u** and **v** by coefficient vectors $u$ and $v$ multiplying a set of piecewise quadratic polynomials, and the pressure **p** by a set of coefficients $p$ multiplying a set of linear polynomials. Under mild assumptions on the data, it is known that, if $h$ is a measurement of the fineness of the finite element mesh, then the solution of the discrete equations approximates the solution of the continuous equations, as $h \rightarrow 0$.

The discretization results in a coupled system of nonlinear algebraic equations for the unknown coefficient vectors $(u, v, p)$, which we represent by

$$G(u, v, p) = 0 \ . \tag{4}$$

See [7] for the formulation and convergence results for the discretized Navier Stokes equations.

# 4 The Optimization Problem

Suppose our flow problem is fully specified, except for the values of a set of parameters, $\vec{\beta}$. Then the specification of the parameter values determines the flow field, and hence the flow values along the sampling line, and hence the discrepancy cost functional, which we denote by $J$.

$J$ will explicitly be a function of $(u, v, p)$, but we may instead regard it as a function of the parameters $\vec{\beta}$. Our fundamental task becomes an unconstrained optimization: given a functional $J(\vec{\beta})$, its partial derivatives $\frac{\partial J}{\partial \beta_i}$, and a starting point $\vec{\beta_0}$, we seek to minimize $J$.

As a sample cost function, we suppose $u_s(y)$ is a given set of flow measurements along the sampling line, and consider the integral:

$$J_1 = \int_{x=x_s} \left(u(x_s, y) - u_s(y)\right)^2 \ dy \ . \tag{5}$$

Since the dependence of $J$ on $\vec{\beta}$ is implicit, we cannot compute an explicit formula for the partial derivatives we will need. We might try finite differences:

$$\frac{\partial J}{\partial \beta_i} \approx \frac{\Delta J}{\Delta \beta_i} \ . \tag{6}$$

This method is straightforward but costly, each derivative requiring at least one additional Navier Stokes solution.

A cheaper alternative uses the *sensitivities*, derived from the implicit relationship between the state variables and parameters. We rewrite the continuous Navier Stokes equations to include the parameters:

$$\mathbf{G}(\mathbf{u}, \mathbf{v}, \mathbf{p}, \vec{\beta}) = 0 \ . \tag{7}$$

and if $\mathbf{G}$ is smooth, we may differentiate with respect to any $\beta_i$:

$$\frac{\partial \mathbf{G}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \beta_i} + \frac{\partial \mathbf{G}}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \beta_i} + \frac{\partial \mathbf{G}}{\partial \mathbf{p}} \frac{\partial \mathbf{p}}{\partial \beta_i} = -\frac{\partial \mathbf{G}}{\partial \beta_i} \ . \tag{8}$$

Because $\mathbf{G}$ generally involves derivatives of the continuous variables $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{p}$, we have implicitly assumed we may interchange differentiations. It is natural to consider the corresponding discrete version of Equation (8), which is called the *discrete sensitivity equations*:

$$\frac{\partial G}{\partial u} \widehat{\frac{\partial u}{\partial \beta_i}} + \frac{\partial G}{\partial v} \widehat{\frac{\partial v}{\partial \beta_i}} + \frac{\partial G}{\partial p} \widehat{\frac{\partial p}{\partial \beta_i}} = -\frac{\partial G}{\partial \beta_i} \tag{9}$$

where the quantities

$$(\widehat{\frac{\partial u}{\partial \beta_i}}, \widehat{\frac{\partial v}{\partial \beta_i}}, \widehat{\frac{\partial p}{\partial \beta_i}}) \tag{10}$$

are called the *(discrete) sensitivities* with respect to $\beta_i$.

It may be tempting to assume that a discrete sensitivity is equal to the derivative of the discrete state variable, but this is only true in the limit:

$$\widehat{\frac{\partial u}{\partial \beta_i}} \approx \frac{\partial \mathbf{u}}{\partial \beta_i} \approx \frac{\partial u}{\partial \beta_i} \tag{11}$$

where the left and right quantities approach the middle quantity (and hence, each other) as $h \to 0$.

If the mesh spacing $h$ is suitably fine, we may use the easily computable discrete sensitivities as approximations to the unknown state derivatives, producing an approximation to the desired cost function derivative:

$$\frac{\partial J}{\partial \beta_i} \approx \frac{\partial J}{\partial u} \widehat{\frac{\partial u}{\partial \beta_i}} + \frac{\partial J}{\partial v} \widehat{\frac{\partial v}{\partial \beta_i}} + \frac{\partial J}{\partial p} \widehat{\frac{\partial p}{\partial \beta_i}} \ . \tag{12}$$

If we have just used Newton's method to solve the discrete flow equations, the Newton system has the same form as the linear system in Equation (9); thus sensitivities can be computed at the trivial cost of a linear solve.

# 5 Simple Channel Flow

The first test of our program was unobstructed channel flow. The inflow was parabolic, with a strength determined by a single parameter $\lambda$. A target solution was generated with $\lambda = 0.5$, and the flow values were measured along the sampling line $x_s = 9$.

The optimization code was then given an initial value of $\lambda = 0.0$ and was requested to minimize the functional $J_1$ as given in Equation (5). It accomplished this minimization in one step; the exact flow solution, called *Poiseuille flow*, has a linear relationship between $\lambda$ and the horizontal velocity at any point $(x, y)$.

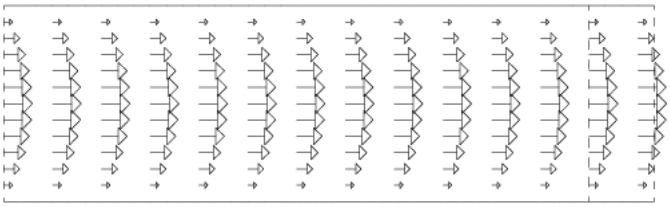

Figure 2: Derivative of velocity with respect to the inflow parameter.

This makes the functional $J_1$ a quadratic function of $\lambda$, which is why we can optimize it easily.

If we plot the velocity derivatives with respect to $\lambda$, as in Figure 2, we can easily see that the influence of the inflow parameter extends throughout the region, dying off only near the walls. Even in problems where the channel is obstructed, the inflow parameter will continue to have this very strong global effect upon the flow. This global influence of $\lambda$ could also be detected by monitoring the state variable derivatives.

# 6 Flow Past a Bump

We now turn to another problem, where the single parameter, $\alpha$, determines the height of a parabolic bump in the channel. The inflow parameter $\lambda$ will be fixed at a value of 0.5. We generate a target flow with $\alpha = 0.5$, and then begin the optimizer at $\alpha = 0$.

If we use approximate gradients based on the sensitivities, we find that the optimization does not reach the correct global minimum at $\alpha = 0.5$. Instead, the optimizer halts after 24 steps at $\alpha = 0.03$ giving the message "false convergence". Such a message generally means that the derivative data is inconsistent with the functional. We take that to mean the sensitivities aren't accurate enough, a difficulty that can be treated by refining the mesh. We will look at sensitivity failures more closely in Section 9.

However, there must be something more seriously wrong with this problem. We converted the program to use finite difference gradients, in which case the optimization came much closer to the correct answer, reaching $\alpha = 0.503$ in 17 steps. But this is hardly satisfactory for a one parameter optimization in double precision!

Why is this bump problem so different from the inflow problem? One hint comes if we look at the cost function values for the initial guess, $\alpha = 0$. The simple channel flow problem had a cost of $J_1(0) = 0.4$, but our bump problem has a cost of $J_1(0) = 10^{-8}$. Our problem is obviously very badly scaled.

If we graph the field of velocity derivatives with respect to the parameter,

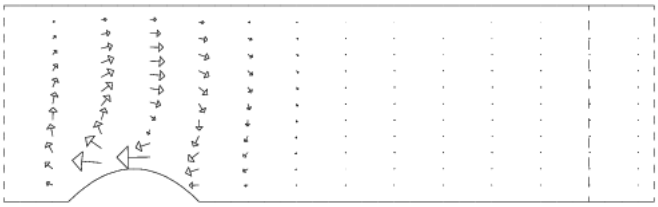

Figure 3: Derivatives of the velocity with respect to the bump parameter.

as in Figure 3, we see a great deal of influence near the bump, which dies away rapidly as we move down the channel towards the sampling line. Clearly, we should try moving the sampling line closer to the bump, to make our measurements as accurately and robustly as possible.

Simply moving our sampling line to $x_s = 3$, defining a new version of our cost functional, $J_2$, causes the cost of the $\alpha = 0$ solution to jump to a "healthy" value of $J_2(0) = 0.009$. Our optimization converges in just 10 steps to the more accurate value $\alpha = 0.500003$. Moreover, we can return to using sensitivities in our formulation. This suggests that sensitivities on a coarse grid aren't worthless. They just aren't accurate enough to solve problems that need a great deal of resolution.

Thus, the bump problem is harder to solve than the simple channel problem, because the influence of the bump parameter on the state variable $u$ is weak and local, a fact which we were able to deduce by looking carefully at the state derivatives.

# 7 Encountering a Local Minimum

We looked at problems where the bump was modeled by a cubic spline with equally spaced abscissas. We used a bump modeled by $\alpha_1$, $\alpha_2$, and $\alpha_3$, which represented the height of the bump at each of the interior abscissas. A single parameter $\lambda$ controlled the strength of the inflow. The target solution was generated with a parabolic bump described by $\alpha = (0.375, 0.5, 0.375)$ and inflow $\lambda = 0.5$.

The optimizer started from $\lambda = 0$, $\alpha = (0, 0, 0)$. Instead of reaching the target parameters, the optimizer settled down at $\lambda = 0.507$, $\alpha = (0.140, 0.539, 0.059)$, where it declared satisfactory convergence.

The cost of the zero solution was $J_2(\vec{0}) = 0.429$, so poor functional scaling was not to blame. Our next suspicion was that the cost functional might be so flat between our final iterate and the target value that further progress was not possible. But this belief was quickly dispelled when we computed the cost functional at a series of intermediate points, and found that it rose from $J_2 =$

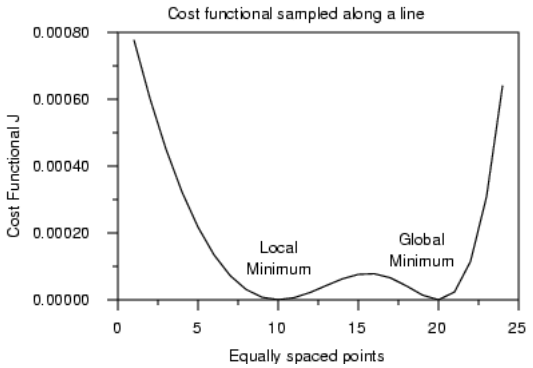


Figure 4: The functional between local and global minimizers.
The local minimizer has $J_2 = 0.3E-06$, the global minimizer has $J_2 = 0$.

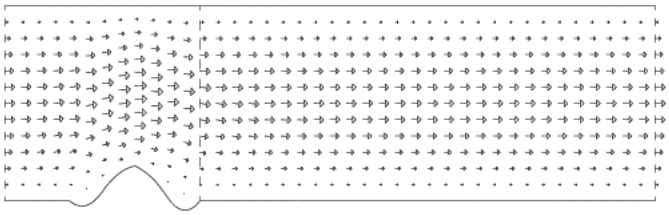

Figure 5: Locally minimizing flow produced by the optimizer.
The global minimizer has a parabolic bump, and no "gutters".

$0.3E-06$ to a value of $J_2 = 0.7E-04$ before falling to $J_2 = 0$ at the target, as shown in Figure 4, suggesting that we might have reached a local minimum.

We looked at the actual flow solution, as shown in Figure 5, to make sure it was acceptable and meaningful. The graph shows that the resulting bump had roughly the same height as the target bump, but with a "gutter" before and after it.

The question then arose as to whether this was actually a local minimum or a *spurious* numerical solution. There are numerical reasons for doubting the accuracy of this solution. The gutter regions are made up of elements that have become stretched and twisted, reducing the accuracy of the finite element discretization. This is an issue that is best addressed by a new calculation on a finer mesh.

If we used a mesh that is twice as fine, the gutters got almost twice as deep. This fact makes it unlikely that there is actually a physical solution that our data is trying to model. We therefore turned from investigating the meaning of this local minimum, and began to consider instead how we could avoid it.

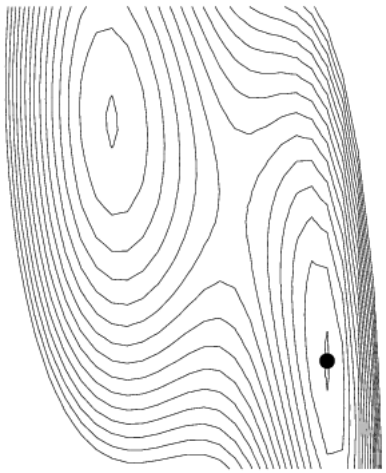

Figure 6: Contours for $J_2$ on plane including local and global minima. The global minimum is marked.

# 8 Smoothing the Cost Functional

It's possible that a solution with very deep gutters would be unacceptable to the wind tunnel engineer: such a forebody simulator might not fit the apparatus. We leave ourselves open to such results since we haven't placed any feasibility constraints on our parameter space.

We note that it is likely that a higher Reynolds number would simplify matters. The flow should be affected in a stronger way by the details of the shape of the bump, and these effects should be passed downstream to the sampling line.

However, let us suppose that we need to solve this problem, or problems similar to it, at the given, low, Reynolds number. What changes can we make so that we are likelier to avoid the local minimum? One possibility is to add a penalty $J_{bump}$ based on the integral of the square of the derivative of the bump. Such a penalty is zero for a flat line, low for a small parabola, and high for a curve with wiggles or severe curvature. Our formula would be:

$$J_{bump}(\vec{\alpha}) = \int_1^3 (bump_x(x, \vec{\alpha}))^2 \, dx \ . \tag{13}$$

Then we will work with a new cost function $J_3$ defined by adding a "small" multiple of the bump penalty to the original cost:

$$J_3 = J_2 + \epsilon \cdot J_{bump} \ . \tag{14}$$

To get an idea of the smoothing effect of this change in the functional, let's consider a two dimensional plane containing the local minimum and the global

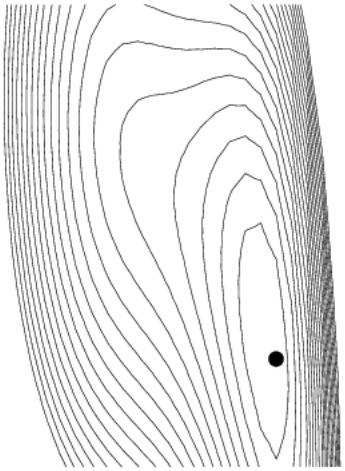

Figure 7: Contours of $J_3$ on the same plane, with $\epsilon = 0.0002$.
The local minimum evident in the previous figure has disappeared.

minimum. Contour lines of the functionals are displayed in Figure 7, and should be compared with those drawn in Figure 6.

The added bump term seems to have smoothed away the local minimum. And if we start from zero, the optimizer now finds the global minimizer. However, the global minimizer of $J_3$ is *not* the minimizer of $J_2$, the function we actually want to minimize. What is true is that, for small $\epsilon$, the minimizer of $J_3$ is close to the minimizer of $J_2$. That means that, in a case where we are bedeviled by a local minimum or other irregularities in the functional, we can try adding such a smoothing term. Starting from a zero initial guess, we can find the minimizer of the smoothed functional. Then we can restart the optimization from this point, but using the unsmoothed functional. If the minimizers of the two functions are close enough, we now have a much better chance of converging to the desired global minimum. For instance, an optimization of $J_3$ with $\epsilon = 0.0002$ converged to the point $\lambda = 0.500$, $\alpha = (0.276, 0.495, 0.364)$. If we now reset $\epsilon$ to 0, which restores our original optimization function $J_2$, and restarted the optimization, we reached the desired target point, bypassing the local minimizer.

Of course, this doesn't settle the question. We still have to detect that we have reached an undesired solution, and choose a smoothing function of a "suitable" type, and a smoothing parameter $\epsilon$ of a "suitable" size. For a more challenging case, we might have to smooth and restart several times, increasing the number of ad hoc choices made. In that case, a more suitable approach would be to add $\epsilon$ directly and explicitly as another parameter to the problem, and optimize once on the enlarged system. The partial derivative with respect to $\epsilon$ is trivial to compute. The only further complication is that there will be

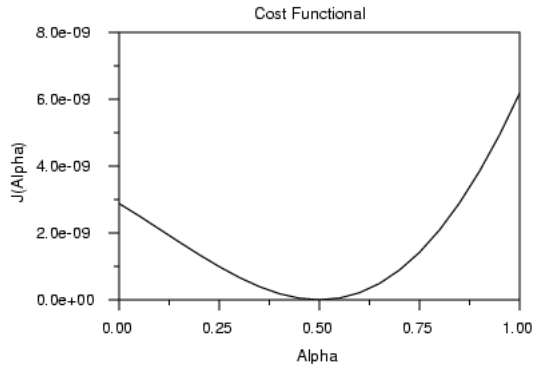


Figure 8: $J_1(\alpha)$, with sampling line at $x_s = 9$.

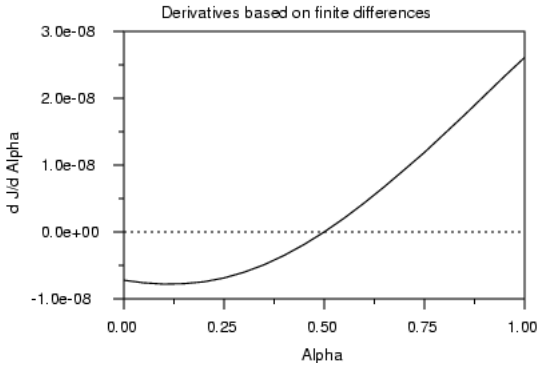


Figure 9: $\frac{dJ_1}{d\alpha}$ by finite differences.

new contributions to the derivative with respect to $\alpha$ coming from the term $\epsilon \cdot J_{bump}$.

# 9 Sensitivity Failure

Throughout our discussion and computations, we have used the sensitivities to arrive at cheap approximations for the derivatives of our state variables $(u, v, p)$ and cost functional $J$. Unfortunately, an optimization requires very accurate derivatives near the minimizer, precisely where the errors in the sensitivities become large, in the relative sense. We already encountered this problem in Section 6. When the optimizer returned the message "false convergence", it had reached a point where the approximate derivative of $J_2$ was too incorrect to use.

To get a feeling for what the optimizer was dealing with, let us look at the functional $J_1$ and its derivative $\dfrac{dJ_1}{d\alpha}$ as approximated by finite differences and

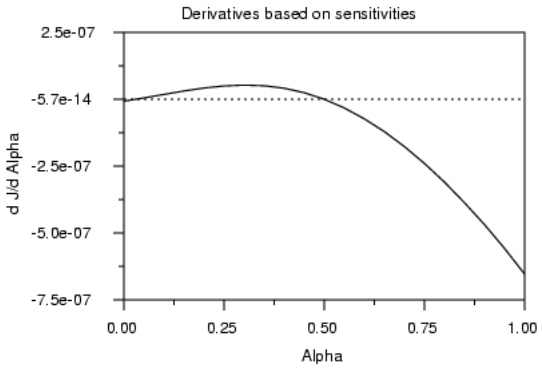


Figure 10: $\frac{dJ_1}{d\alpha}$ by sensitivities.

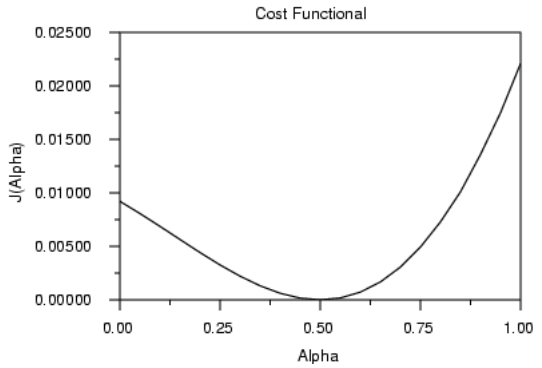


Figure 11: $J_2(\alpha)$ with sampling line at $x_s = 3$.

by sensitivities, in Figures 8 through 10.

The sensitivities provide an astonishingly bad "approximation". We might have been warned by the small magnitude of the quantities, though a better warning lies in the fact that the state derivatives we sample are much smaller than the same quantities elsewhere in the region, and hence are relatively poorly approximated. Figures 11 through 12 show how moving the sampling line to $x_s = 3$ corrects this problem.

We should keep in mind that the underlying data is *identical* for the two sets of plots we are comparing here. This includes the state variables and state derivatives. The difference is in the definition of the functional and its derivative, that is, in which state variables we sample.

Near the minimizer, errors in the sensitivities can become so serious that the partial derivatives are worthless. This occurred in the multiparameter bump problem, with one inflow parameter and three bump parameters, at a Reynolds number of 100. We were using a cost functional, $J_4$, which included the discrep-

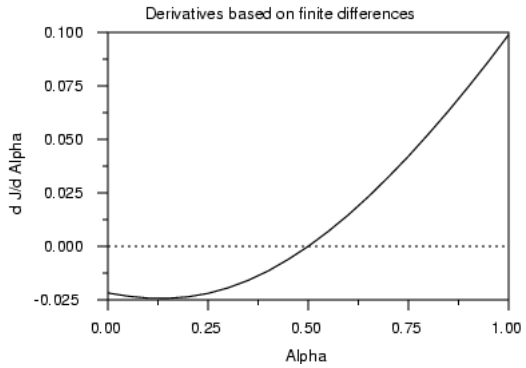


Figure 12: $\frac{dJ_2}{d\alpha}$ by finite differences and sensitivities.
The two calculations are now essentially identical.

ancies in both horizontal and vertical velocities along the line $x_s = 3$:

$$J_4 = \int_{x=x_s} \left(u(x_s, y) - u_s(y)\right)^2 + \left(v(x_s, y) - v_s(y)\right)^2 \, dy \; . \tag{15}$$

The target parameters were $\lambda = 0.500$, $\alpha = (0.375, 0.500, 0.375)$. The optimizer reached $\lambda = 0.493$, $\alpha = (-0.078, 0.476, 0.101)$ and reported "false convergence". Note that at this point, $J_4 = 0.0006$, hardly the sort of extremely small value we took as a warning earlier.

To see what was going on, consider a plane including the point where the optimizer stopped, and the global minimizer. We used sensitivities to compute the cost function gradients, projected onto the plane.

Figure 13 shows the functional contour lines, overlaid with the projected computed gradient field (normalized and multiplied by -1, so that it should point towards the minimizer). We can immediately see that, at least in this "slice" of parameter space, and near the minimizer, the errors of approximation are so serious that the direction field is utterly lost. It is no wonder that the optimizer is unable to find the minimizer. There can be little doubt that this problem is caused by discretization errors. If we halve the mesh size and recompute the same quantities, all the approximate gradients point inwards, towards the minimizer. Even then, small errors in the direction of the gradients plainly persist.

# 10 Conclusions

If a cost functional and the independent variables are only implicitly related, singularities, poor scalings, and local minima may easily occur. Diagnosis of such problems is usually possible through such means as making a plot of the state solution, a table of functional values along a line or plotting contours of the functional along two directions. Problems may be treated in a variety of ways.

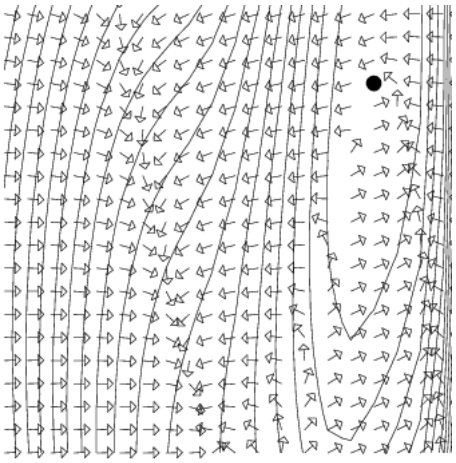

Figure 13: Gradients approximated by sensitivities.
The approximate gradients clearly do not match the contours.

The functional may be modified so that it depends on more state variables, or on state variables that are more sensitive to variations in the parameters. The functional may include the cost of control, or other terms that tend to "regularize" or smooth out the contour levels.

Sensitivities are a valuable method of approximating derivatives. However, their use entails an extra source of error, which must be monitored. In particular, a grid which is fine enough to get good approximations to the state variables may not be fine enough to provide approximations of the state variable derivatives.

Small errors caused by using sensitivities can also interfere with an optimization during the final steps, near a minimizer, when the gradient has dropped to a very low magnitude, essentially magnifying the significance of errors in the derivatives.

Simple checks can be applied after the failure of an optimization that uses sensitivities. These include trying a refined grid, comparing the results of approximating the derivatives with finite differences, plotting the state variable field, plotting the functional gradient field, and considering whether the relationship between the parameters and the functional is weakly or strongly mediated by the state variables.

# 11 Acknowledgements

The authors thank Ken Bowers and John Lund, the organizers of the fourth conference on Computation and Control at Montana State University.

# References


[1] J. BORGGAARD, J. BURNS, E. CLIFF, M. GUNZBURGER, *Sensitivity Calculations for a 2D, Inviscid Supersonic Forebody Problem*, in **Identification and Control of Systems Governed by Partial Differential Equations**, H T Banks, R Fabiano, K Ito, editors, SIAM Publications, 1993.

[2] J. BURKARDT and J. PETERSON, *Control of Steady Incompressible 2D Channel Flow*, **Flow Control**, The IMA Volumes in Mathematics and its Applications, Volume 68, pages 111-126, edited by Max Gunzburger, Springer Verlag, New York, 1995.

[3] C. DEBOOR, **A Practical Guide to Splines,** Springer Verlag, New York, 1978.

[4] D. GAY, *Algorithm 611, Subroutines for Unconstrained Minimization Using a Model/Trust Region Approach,* **ACM Transactions on Mathematical Software**, Volume 9, Number 4, December 1983, pages 503-524.

[5] M. GUNZBURGER, J. PETERSON, *On Conforming Finite Element Methods for the Inhomogeneous Stationary Navier-Stokes Equations*, **Numerische Mathematik**, Volume 42, pages 173-194.

[6] HUDDLESTON, *Development of a Free-Jet Forebody Simulator Design Optimization Method*, AEDC-TR-90-22, Arnold Engineering Development Center, Arnold AFB, TN, December 1990.

[7] O. KARAKASHIAN, *On a Galerkin-Lagrange Multiplier Method for the Stationary Navier-Stokes Equations*, **SIAM Journal of Numerical Analysis**, Volume 19, Number 5, October 1982, pages 909-923.