Control of Steady Incompressible 2D Channel Flow

John Burkardt *

Janet Peterson †

Department of Mathematics

Interdisciplinary Center for Applied Mathematics

Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 24061

Abstract

We consider steady incompressible flows in a 2D channel with flow quantities measured along some fixed, transverse sampling line. From a set of allowable flows it is desired to produce a flow that matches a given set of measurements as closely as possible. Allowable flows are completely specified by a set of control parameters which determine the shape of the inflow at the boundary and the shape of an internal bump which partially obstructs the flow. Difficulties concerning the transformation of this problem into a standard optimization problem are discussed, including the correct choice of functional and algorithm, and the existence of local minima.

1 Introduction

If a log falls into a stream, it disrupts the flow, creating a pattern of ripples and whirls. If the log lies hidden under a bridge, a wise observer standing on the bridge and staring downstream could nonetheless detect the change in the flow, and make a guess as to the size and position of the obstacle. But as the flow rushes on, it rapidly destroys this information, and just a few yards downstream there will be no discernible record of the intrusion into the flow.

In aeronautical design, a similar problem occurs. Instead of a stream, a wind tunnel is used, through which a steady flow of air is driven. It is not a log, but a mockup of an aircraft wing, or fuselage, or forebody, which is deliberately inserted into the flow. Instead of an observer on a bridge, a string of measuring devices are used to record the velocity and pressure of the flow at a fixed position downstream from the obstacle.

*Supported by the Air Force Office of Scientific Research under grant AFOSR 93-I-0061.

†Supported by the Office of Naval Research under grant N00014-91-J-1493.

For a given orientation and position of the object within the wind tunnel, and for a given pattern of inflowing air, the measured values of velocity and pressure can be regarded as the "signature" of the obstacle. Generally, if two objects differ in shape, their signatures will differ. However, it is possible for one shape to "forge" or approximate the signature of another. This fact can be very useful for some kinds of wind tunnel tests. Certain parts of a plane come "after" other parts; that is, they are further downstream in the airflow. Thus the flow field that strikes the downstream part has already been changed by its interactions with the upstream part and so although it's possible to test a propeller, say, by itself in a wind tunnel, to test a tail assembly requires a mockup of the entire forebody of the airplane as well.

Sometimes a full model cannot be tested because it is too large for the wind tunnel. But it may be possible to find a smaller shape which will fit in the wind tunnel ahead of the object to be tested and which will have the same "signature" as the true forebody. Such a forebody "simulator" should have the property that the velocity and pressure of the air flow, along some transverse downstream plane, will closely match the values associated with the true forebody. A complete description of the forebody simulator problem is available in Huddleston [**?**] and in Borggaard, Burns, Cliff and Gunzburger [**?**].

The problem described above is the motivation for the preliminary study given here. This is an ongoing project whose goal is the development of algorithms to select, from an allowable family, a set of flow parameters, and a shape which can be inserted into that flow, which will most closely match a given set of downstream measurements. In Section **??** we give the equations which model steady, viscous, incompressible flow in a channel. We choose to use finite elements to discretize these equations, so in Section **??** we discuss the choice of approximating spaces and give the set of nonlinear equations which must be solved. In Section **??** we discuss the optimization problem using flow sensitivities. The next three sections describe problems in which we allow one or more parameters to vary in order to obtain a flow which matches a given velocity profile. The first problem is simple channel flow with no obstacle in the flow field; here we allow the inflow to vary in order to match a given flow. For the second problem we allow an obstacle to be placed in the flow field but we require that it be modeled by a single parameter. In this case the inflow is fixed and the shape of the bump is allowed to vary. For the third problem we combine the first two and also allow the bump and inflow to be described by more than one parameter. We conclude the report by discussing future work in Section **??**.

2 Mathematical Model

The equations governing steady, viscous, incompressible flows are the Navier-Stokes equations which can be written in terms of the velocity $\mathbf{u} = (u, v)$ and the pressure p as

$$-\nu\Delta\mathbf{u} + \mathbf{u}\cdot\operatorname{grad}\mathbf{u} + \operatorname{grad} p \quad = \quad \mathbf{f} \quad \text{in } \Omega \tag{1}$$

$$\operatorname{div} \mathbf{u} \quad = \quad 0 \quad \text{in } \Omega \tag{2}$$

plus appropriate boundary conditions. Here ν is the kinematic viscosity, $\mathbf{f}$ the given forcing function and Ω the domain in $\mathbb{R}^2$ modeling the wind tunnel. We make the assumption that the problem can be restricted to two dimensions; that is, we assume that the behavior of the wind tunnel, the flow field, and the shape are all constant along the z-direction. For the case of simple channel flow with no obstacle, Ω is formed by two parallel horizontal walls. The boundary conditions chosen describe a flow entering the region from the left and passing out of the region at the right. At the inflow we set $u = q(y)$, $v = 0$; at the top and bottom of the channel we set both components of the velocity to zero; and at the outflow we set the usual conditions $v = 0$ and $\dfrac{\partial u}{\partial x} = 0$.

The weak formulation of Equations (**??**)–(**??**) which we consider follows [**?**]. We seek $\mathbf{u} \in \mathbf{H}^1(\Omega)$ and $p \in L_0^2(\Omega)$ such that

$$\nu \int_\Omega \operatorname{grad} \mathbf{u} : \operatorname{grad} \mathbf{w} \, d\Omega + \int_\Omega \mathbf{u} \cdot \operatorname{grad} \mathbf{u} \cdot \mathbf{w} \, d\Omega - \int_\Omega p \operatorname{div} \mathbf{w} \, d\Omega \quad = \quad \int_\Omega \mathbf{f}\mathbf{w} \, d\Omega \quad \forall \mathbf{w} \in \mathbf{H}^1(\Omega) \tag{3}$$

$$\int_\Omega \phi \operatorname{div} \mathbf{u} \, d\Omega \quad = \quad 0\,, \quad \forall \phi \in L_0^2(\Omega) \tag{4}$$

and such that $u = v = 0$ on the top and bottom walls, $u = q(y)$, $v = 0$ at the inflow, and $v = 0$ at the outflow. Here $\mathbf{H}^1(\Omega)$ represents the space of vector-valued functions each of whose components is in $H^1(\Omega)$, the standard Sobolev space of real-valued functions with square integrable derivatives of order up to one. $L_0^2(\Omega)$ is defined by all functions in $L^2(\Omega)$ with zero mean over Ω. Again see [**?**] for details.

3 Finite Element Approximations

In order to approximate the flow we must choose a particular discretization. Our choice here is to use the finite element method, although clearly other discretization methods could be employed. Using the standard techniques of finite elements we discretize our flow region into a finite number of subregions called *elements*, inside each of which we will assume that the flow has a simple structure. For our problems, we choose triangles to create this mesh. For our first problem, which has no internal obstacle, the flow region is rectangular and so it is a simple matter to divide the region up into rectangles, each of which can be split to form two triangles. However, for the problem with an internal obstacle, the flow region is thought of as being a mild distortion of a rectangle and so we are forced to use elements with curvilinear sides. This will require the use of isoparametric elements.

Having represented the region by a mesh of finite elements, we now approximate the continuously varying physical quantities $\mathbf{u}$ and p by functions which can be determined from a finite set of data associated with each finite element. Typically these functions will be represented over the entire region by continuous, piecewise polynomials. An examination of the error estimates for the

velocity and pressure indicate that one should usually choose one degree higher polynomial for the velocity than for the pressure. For our computations, the velocities are represented by quadratic polynomials and the pressure by linear polynomials. The finite data which represents the velocity, for instance, is then simply the value of the velocity at six particular *nodes* in the element, which are the vertices of the triangle and the midpoints of its sides. Similarly, the pressure is specified by its value at just the three vertices.

We now define a problem which will yield approximate solutions of the weak formulation given in Equations (**??**)–(**??**). Let $\mathbf{V}^h$ be the space of vector-valued functions whose components are continous piecewise quadratic polynomials over the triangles, let S^h be the space of piecewise linear polynomials over the triangles, and let S_0^h denote the functions in S^h which are constrained to have zero mean. Then we seek a $\mathbf{u}^h \in \mathbf{V}^h$ and $p^h \in S_0^h$ satisfying

$$\begin{aligned} \nu \int_\Omega \operatorname{grad} \mathbf{u}^h : \operatorname{grad} \mathbf{w}^h \, d\Omega &+ \int_\Omega \mathbf{u}^h \cdot \operatorname{grad} \mathbf{u}^h \cdot \mathbf{w}^h \, d\Omega \\ &- \int_\Omega p^h \operatorname{div} \mathbf{w}^h \, d\Omega = \int_\Omega f \mathbf{w}^h \, d\Omega \quad \forall \mathbf{w}^h \in \mathbf{V}^h \\ \int_\Omega \phi^h \operatorname{div} \mathbf{u}^h \, d\Omega &= 0 \quad \forall \phi^h \in S^h \, . \end{aligned}$$

The essential boundary conditions are enforced in the usual manner. In particular, at the inflow, $u^h = q^h(y)$, where $q^h(y)$ is the piecewise quadratic interpolant of $q(y)$.

Using standard techniques from the theory of finite elements, we can write these equations as a set of algebraic equations. Each pair of unknown velocities is uniquely associated with a velocity node at which we have two scalar equations. Similarly, each unknown pressure corresponds to a pressure node and a pressure equation. Thus, we should be able to solve the system and compute the values of the flow quantities at each node.

The finite element equations are *nonlinear*, and so they must be solved via an iterative method. The iterative method we employ is Newton's method. See [**?**] for the formulation and convergence results for the Navier-Stokes equations.

4 The Optimization Problem

Our goal in this study is to specify the values of some of the flow quantities along a line in the region Ω and then to deduce from that a flow over the whole region, whose values of $\mathbf{u}$ and p match (or come as close as possible to) the original given values along some sampling line. We will assume that we have some family of possible flows from which to select. In fact, we will assume that there are one or more *parameters* which characterize this family, so that specifying the value of the parameters completely specifies a flow. In such a case, we may regard the flow quantities as *functions* of the parameters. We will use the letters λ and α for typical parameters.

In order to solve the matching problem, we must first specify a mathematical measure of how well an arbitrary flow matches the given data. It would be desirable to produce a "score", that is, a single number which represents the closeness of the fit, and which is minimized for a perfect fit. One possible choice of a functional to minimize is the integral of the square of the differences between the data and the computed horizontal velocity variables; i.e.,

$$f_1(\mathbf{u}, p) = \int_S \left(u(x_s, y) - u_s(y)\right)^2 \, dy \,, \tag{5}$$

where S denotes the sampling line, x_s denotes the x-coordinate of the sampling line, and $u_s(y)$ represents the sampled velocity data we are attempting to match. Other choices of functionals to be minimized will be discussed in Section **??**.

Once we have chosen a particular functional, we can formulate a *minimization problem*, which is to find a flow ($\mathbf{u}$, p) which minimizes the given functional. If we have a single free parameter λ, we can phrase this problem as follows:

> Given a functional $f(\mathbf{u}, p)$, where $\mathbf{u}$, p are functions of a parameter λ, find the value of λ that minimizes f.

Clearly there are many different approaches to solving this one-dimensional minimization problem. Rather than seeking to minimize the functional $f(\mathbf{u}, p)$ itself, we choose to seek a zero of the derivative of the functional with respect to λ. It was not considered feasible to compute the derivative of the functional with respect to the parameter directly, since the effect of the parameter on the functional is expressed only indirectly, through the flow field. Instead, equations for the *flow sensitivities* are used to approximate the required derivative.

Suppose we can represent a flow field that satisifies a set of flow equations involving a single parameter λ as

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

Then the corresponding flow sensitivities

$$(\frac{du}{d\lambda}, \frac{dv}{d\lambda}, \frac{dp}{d\lambda})$$

are defined by the linear equations

$$\frac{\partial \mathbf{G}}{\partial u}\frac{du}{d\lambda} + \frac{\partial \mathbf{G}}{\partial v}\frac{dv}{d\lambda} + \frac{\partial \mathbf{G}}{\partial p}\frac{dp}{d\lambda} = -\frac{\partial \mathbf{G}}{\partial \lambda} \,.$$

If the original nonlinear flow equations have just been solved, the corresponding flow sensitivities are inexpensive to compute; this is because Newton's method, which is used to solve the nonlinear system, uses an iteration matrix which converges to the sensitivity matrix as the iterates converge to the correct solution. Thus, if the iteration has been deemed to converge, the current, factored iteration matrix may then be used to immediately solve for the sensitivities at very low computational cost.

Now we can reframe the problem of finding a minimum of the optimization functional $f(\mathbf{u}, p) = f(\lambda)$ in terms of finding a zero of the derivative of f with respect to λ given by

$$\frac{df}{d\lambda} = \frac{\partial f}{\partial u}\frac{du}{d\lambda} + \frac{\partial f}{\partial v}\frac{dv}{d\lambda} + \frac{\partial f}{\partial p}\frac{dp}{d\lambda}\,.$$

Once we have a method of computing $df/d\lambda$ we can pose the problem of finding a zero of this derivative, and hope that such a zero corresponds to a minimum of the original optimization function, that is, a best match to the flow data.

There are numerous choices for finding the zeros of a function. Since calculating the derivative of the function $df/d\lambda$ requires calculating the second derivative of f, the scalar secant method is an obvious choice when we have only one parameter since it uses only function evaluations. Of course, such a method is not guaranteed to find a minimum; a zero value of the derivative is just as likely to represent a maximum or inflection point. This problem can be forestalled by beginning the optimization with starting points close enough to the correct solution so that convergence to a minimum was very likely. Thus the usefulness of this approach is limited to testing one's code and some very simple problems. When we report on the solution of multiparameter problems in Section **??**, we discuss the choice of a suitable optimization package.

5 Simple Channel Flow

Our first example is the simple case of channel flow with no obstacles in the flow field. These computations were made to test the underlying flow solver and to begin to get some experience with the optimization techniques necessary to solve a general problem.

The channel is modeled by two parallel horizontal walls separated by 3 units and extending from $0 \le x \le 10$ units. The boundary conditions chosen for this problem describe a simple, parallel flow entering the region from the left and passing out of the region at the right. The inflow profile is required to be parabolic, but the actual strength of the inflow is allowed to vary, according to the value of a parameter λ. In particular, we set

$$u(0, y) = \lambda y(3 - y)\,.$$

As usual, the pressure must be required to satisfy an additional condition such as having zero mean or fixing its value at some point.

To simulate the experimental process of making measurements and then trying to produce a flow configuration that matched them, a "target" value of λ was chosen. The flow was determined for this value and the flow profile at the sampling line was recorded. It was this "experimental data" that we attempted to match. Because the target flow was actually generated by a particular value of λ, we knew that the minimum value of the functional was zero. This made it easy to determine when the search should halt or when the search was not converging.

A simple test case was set up where the correct solution was $\lambda = 1.0$ and the code was started with the two nearby estimates $\lambda_1 = 0.1$, $\lambda_2 = 0.5$. The secant method was used to find the zero of $df/d\lambda$ where f is given by Equation (**??**). For the family of solutions controlled by λ, the functional was actually a quadratic function of λ. Hence its derivative was linear, and the secant method converged to the solution *in one step*. The channel flow results were only useful in that they gave us confidence that the flow field was being solved correctly and that the sensitivities were correctly being used to evaluate the derivative function. In the next problem we allow an obstacle to lie in the flow field, but still require the obstacle to be characterized by just one parameter.

6 Flow Over a Bump Using One Parameter

For the second problem, we use a single parameter as a means of selecting a geometric shape which lies in the flow field as an obstacle. The inflow does not vary for this problem, but rather has a fixed strength and parabolic shape.

A "bump" is placed at a fixed location on the bottom of the channel which is again modeled by two parallel walls of length $0 \leq x \leq 10$ units separated by 3 units. The bump is required to be parabolic in shape and extend horizontally from $1 \leq x \leq 3$, but the height of the bump is allowed to vary, being characterized by a parameter α.

The boundary conditions are similar to those for the channel flow, with two exceptions. First, since the inflow does not vary, the inflow equations simplify to

$$u(0, y) = y(3 - y)\,.$$

Secondly, because the *height* of the lower boundary between $1 \leq x \leq 3$ now varies, the boundary conditions for that portion of the lower wall are rewritten as

$$\begin{aligned} u(x, y(x, \alpha)) &= 0 \\ v(x, y(x, \alpha)) &= 0 \\ y(x, \alpha) &= \alpha(x - 1)(3 - x)\,, \end{aligned}$$

for $1 \leq x \leq 3$.

Because of the curvature of the bump, the computational mesh of the region must also be curved, at least in the vicinity of the bump. Instead of triangular elements with straight sides, *isoparametric* elements were used so that the curved edges of the bump could be modeled. A typical triangulation of the channel with a bump is shown in Figure 1.

Another complication in this problem resulted from the fact that at each step of the optimization, a new value of α was produced for which the corresponding flow had to be computed. Because each α changed the shape of the region, all of the mesh calculations had to be redone at the beginning of every optimization. Thus, the geometry of the region changed at each step, with effects that were

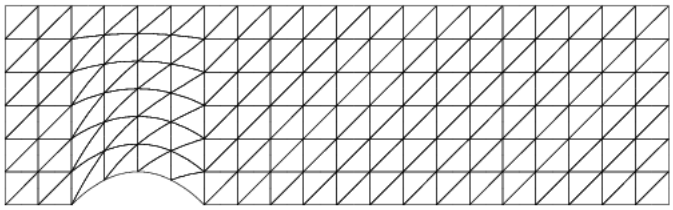

Figure 1: Figure 1: A typical region with a bump, showing elements.

harder to predict than those that were caused by simply varying the inflow as in the first example of simple channel flow.

For our computations we set the Reynolds number to one and chose the secant method to find the zero of the functional given by Equation (**??**). This is analogous to the first problem described in Section **??**.

The choice of the location of the profile sampling line considerably affected the results. If the profile sampling line was set near the outflow, say at $x_s = 9$, then we often encountered problems. For starting parameters that were quite close to the target value, we were able to get convergence, but often what seemed only slightly greater perturbations of the starting point would cause the program to take many more steps, or in some cases even to fail to converge. We concluded that the difficulty rested in the combined problem of the location of the sampling line relative to the bump and the low Reynolds number of the flow. We can use the sensitivities to see the problem. In Figure 2 the plot shows a bump of height 0.5 and the velocity sensitivity field. Each vector represents the effect that a unit increase in the height parameter would have on the local velocity. As is obvious from the graph, the influence is extremely strong above the bump, but drops off dramatically within a few units downstream. This illustrates the fact that low Reynolds numbers can be problematic for flow optimization; i.e., for such cases large changes in the shape control parameters produce only small changes in the flow.

For this reason, the profile line was moved to $x_s = 3$, immediately behind the bump. This vastly improved the responsiveness of the functional to changes in the bump.

7 Multiparameter Flow Past an Obstacle

This problem is a generalization, as well as a combination, of the first two problems. In this example we parameterize both the inflow and the shape of the obstacle and allow the number of parameters for each to be greater than one. Due to the more complicated nature of this problem, a major change in the optimization method was needed, since the secant method was not suitable for further use.

A search was made for a more suitable optimization package to use with the code. Robustness and flexibility were key considerations. We chose ACM TOMS algorithm 611 [**?**], which uses a model/trust region approach for choosing the step, and a BFGS procedure for updating an approximate Hessian matrix. Some of the advantages of this code included: access to the source code, good documentation, good portability with machine dependent quantities handled through calls to a machine dependent function, a *reverse communication* formulation which made it easy to integrate the package into the existing program, three versions of the code in case Hessian or gradients are not available, the fact that it will not accept an iterate if its functional value is higher than that of the current approximate solution, and the fact that it handles an arbitrary number of dimensions.

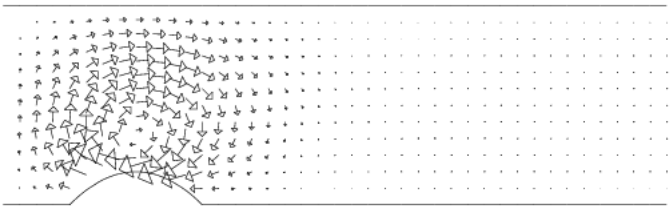

Figure 2: Figure 2: The velocity sensitivity field for a bump solution.

Another consideration was how to handle more complicated shapes. It was assumed that this would be done by adding more parameters, but it was not clear how those parameters should be used to determine the shapes. In the first two problems discussed above, a single parameter controlled the height of a simple, parabolic shape. In order to model more complicated shapes, we had to choose a reasonable set of shapes and a finite set of parameters to catalog them; we chose to use cubic splines [**?**] to represent the bumps. Such a representation requires the value of the shape at a specified sequence of nodes. Then a shape is produced which is a piecewise cubic polynomial between the nodes, and which is continuous, with continuous derivatives, at the nodes. In order to complete the system, typical spline representations also require that the slope of the shape be given at the end nodes.

For our shapes, whether they represent an inflow, or an obstacle, we set the value at the first and last nodes to zero. We did not specify the slopes at the end nodes, but rather, used the "not-a-knot" option. This permitted us to define a shape by specifying only the values. The penalty for this simplification was that the shape was required to have one greater degree of continuity at the first and last interior nodes.

Once any shape could be specified uniquely by giving its values at a sequence of nodes, it was natural to consider those values to be the parameters that would be varied in the optimization. One set of nodes would be placed along the inflow boundary. The value of the inflow at those nodes could be used to specify an inflow function defined along the entire line. A second set of nodes would be placed on the bottom of the channel, for $1 \leq x \leq 3$, where the internal obstacle would be placed. The height of the obstacle at each node would be enough to define the whole shape of the obstacle. The number of nodes placed at either location was arbitrary, and hence we could study different inflows or complicated obstacles or both.

The inflow parameters are given in a vector $\boldsymbol{\lambda}$ and the bump parameters in a vector $\boldsymbol{\alpha}$. The parameters enter the flow problem through the boundary conditions. In particular, the value of the inflow velocity at any inlet point $(0, y)$ is given by a function of y and the inflow parameter vector:

$$\begin{aligned} u(0,y) &= inflow(y, \boldsymbol{\lambda}) \\ v(0,y) &= 0 \end{aligned}$$

and the height of the lower channel for $1 \leq x \leq 3$ is determined as a function of x and the bump parameter vector:

$$\begin{aligned} u(x, y(x, \boldsymbol{\alpha})) &= 0 \\ v(x, y(x, \boldsymbol{\alpha})) &= 0 \\ y(x, \boldsymbol{\alpha}) &= height(x, \boldsymbol{\alpha})\,. \end{aligned}$$

Secondly, the vector of parameters requires that the optimization search be conducted in $\mathbb{R}^{\mathrm{M}}$ rather than $\mathbb{R}^1$. The sensitivities are now defined as *partial* derivatives of the flow quantities with respect to the several parameters.

Various problems became apparent as we attempted to solve these multiparameter problems. The first difficulty we encountered was when the optimization code seemed to "get stuck" on an incorrect minimizer. The optimization code at first produced a rapid decrease in the functional value and a correspondingly better approximation to the known solution. However, after a few steps the convergence ground to a halt. The optimization code took progressively smaller steps, and "converged" to a point that was still a significant distance from the target solution. A study of the data showed that the problem was rooted in a discrepancy between the functional and the approximate derivative data we were supplying. We had been computing the sensitivities of the functional with respect to the parameters. These quantities are easily computed from the same linear system used during the Newton-type iteration that produces the flow field itself. They are only approximations to the derivatives of the functional with respect to the parameters, and their accuracy depends on the fineness of the grid of the region. When the optimization code had gotten fairly close to the correct solution, greater accuracy in the derivatives was required than the sensitivities could deliver. In fact, at the false convergence point, the dot product of the sensitivities with the direction vector pointing towards the true solution was positive, suggesting incorrectly that the functional would increase in that direction, when in fact it was monotonically decreasing. This problem disappeared if the mesh was refined, which improved the sensitivities enough to permit convergence. On the other hand, if we wanted to do calculations on a coarse mesh, an alternative was to use the derivative-free version of the ACM code, in which derivatives with respect to the parameters are approximated internally by finite differences and the accuracy could be improved. This increased the cost of computation on a coarse mesh greatly, but also allowed the optimization method to reach the correct target solution.

A second problem that arose was in the choice of the cost functional given in Equation (**??**). To analyze the difficulty, consider the following problem which has one inflow and three bump parameters. The starting point was given as

$$\begin{aligned} \boldsymbol{\lambda} &= (0) \\ \boldsymbol{\alpha} &= (0,0,0) \end{aligned}$$

and the "target" profile was generated at

$$\begin{aligned} \boldsymbol{\lambda} &= \left(\frac{1}{2}\right) \\ \boldsymbol{\alpha} &= \left(\frac{3}{8},\frac{1}{2},\frac{3}{8}\right), \end{aligned}$$

which corresponds to an inflow with parabolic shape and strength $\frac{1}{2}$ and a bump which "happened" to be a parabola of height $\frac{1}{2}$ although it lies in a space of more complicated shapes. The computation proceeded satisfactorily at first, but after four or five steps, the solution ceased to approach the target solution. Instead, the second and fourth components of the parameter vector became negative!

In fact, after about twenty iterations, the optimization code returned with the message that the iteration had "converged", though the computed solution was not our intended target solution. The computed solution is shown in Figure 3.

A graph of the shape corresponding to the converged values shows that the resulting bump had roughly the same height as the target bump, but with a "gutter" before and after it. To the eye, at least, the resulting horizontal flow at the sampling line looks "close" to the target values.

The cost functional we used, f_1, seemed to have a local minimum which the optimizer had found. We tested this belief by marching along the line between the computed solution and the target solution. The corresponding functional values are shown in Figure 4 and clearly display a "double dip" curve.

The question then arose as to whether this was actually a local minimum or a *spurious* numerical solution. As other local minima solutions were found, they all tended to share the property of being oscillatory. That is, the shape would either be a crest sandwiched between two valleys, or a valley between two crests. When the mesh was refined for a particular case, the "valleys" doubled in depth, while the crest remained roughly where it had been. This suggested that the program was not approximating a real local minimum, which would have a fixed, finite shape. Rather, some instability in the program or in the problem formulation was generating numerical behavior that did not correspond to a physical solution.

One obvious modification to the problem formulation which might alleviate the problem was to increase the value of the Reynolds number. The fact that the functional seemed to be so insensitive to large changes in geometry was possibly due to the very low Reynolds of the problem. In such a setting, the viscous effects could be expected to dominate the flow, and quickly overwhelm disturbances that the functional would be trying to measure.

Some tests of the program seemed indirectly to bear out this statement. In one test we controlled the inflow with three parameters and the target profile was generated by specifying a parabolic inflow. The program produced as a solution an inflow with a "double hump", having a deep drop in the middle. Nonetheless, this contorted inflow assumed essentially a parabolic shape within two mesh units. In another problem we set up a simple channel flow with no bump. There were 11 equally spaced nodes along the left hand boundary at which an inflow velocity was specified with only the first node having a nonzero velocity. Nonetheless, as demonstrated in Figures 5 and 6, the flow very quickly took on a parabolic profile.

Therefore, it would be natural to expect that only at higher Reynolds number (faster inflow or lower viscosity), would the program be able to distinguish between the convex target bump, and the oscillatory solution that had been found. Calculations using higher Reynolds numbers can be done with our code by incorporating a continuation method. We plan to do this in future work.

Another modification to the problem formulation is a change in the cost function f_1 given by Equation (**??**). One possible change was to include the discrepancies in the vertical velocity and pressure along the sampling line as well. This gave the functional

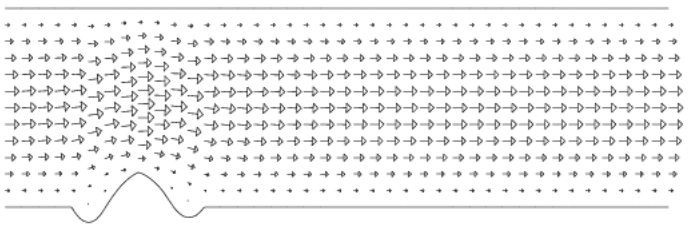

Figure 3: Figure 3: A “local minimum” solution that was not the “target”.

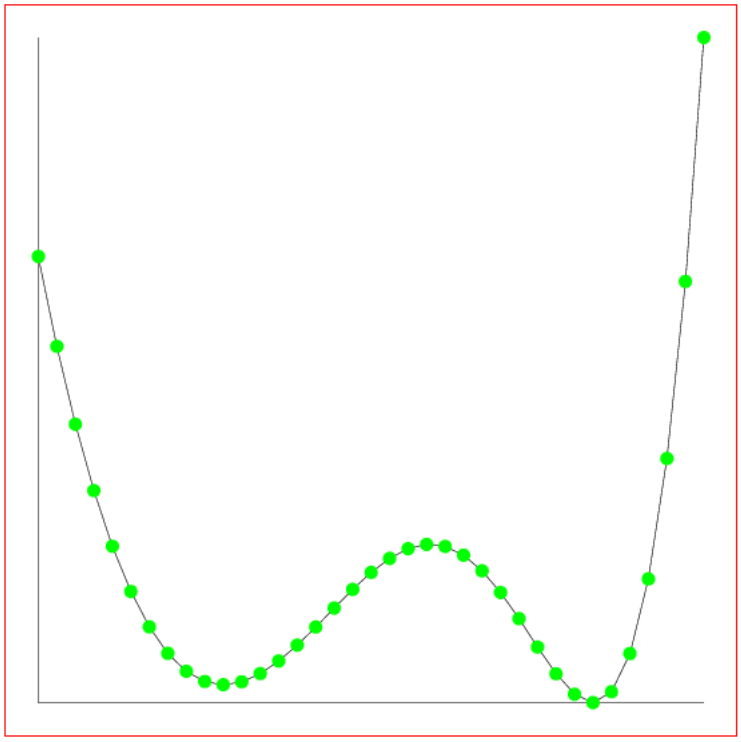

Figure 4: Figure 4: The functional values from local minimum to target.

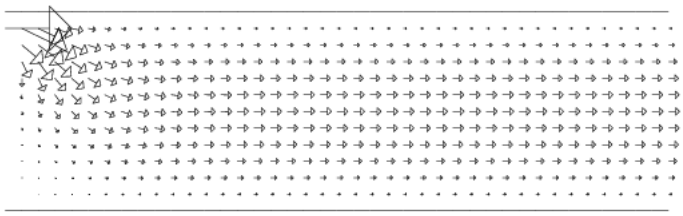

Figure 5: Figure 5: Velocity vectors for the single inflow node test.

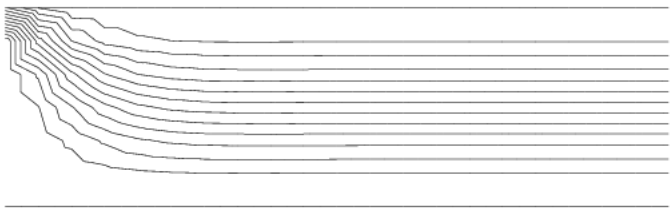

Figure 6: Figure 6: Streamlines for the single inflow node test.

$$f_2(u,v,p) = \left(\int_0^3 (u(x_s,y) - u_s(y))^2 + (v(x_s,y) - v_s(y))^2 + (p(x_s,y) - p_s(y))^2 dy \right)^{\frac{1}{2}} .$$

We used this cost functional f_2 for the same problem that the functional f_1 had displayed a local minimum. The new functional also produced a local minimum, differing from the previous one only in that the bump did not actually have "gutters", but rather "low shoulders".

Another choice of the cost functional was one that included the cost of the control. An obvious choice was to estimate the cost of the bump control by approximating the L^2 norm of the height of the bump about the channel bottom:

$$g_1(\boldsymbol{\alpha}) = \left(\int_1^3 (height(x,\boldsymbol{\alpha}))^2 dx \right)^{\frac{1}{2}} .$$

However, this still allowed oscillatory solutions. We only began to get smoother solutions when we took the L^2 norm of the *slope* of the height:

$$g_2(\boldsymbol{\alpha}) = \left(\int_1^3 (height_x(x,\boldsymbol{\alpha}))^2 dx \right)^{\frac{1}{2}} .$$

The smoothing was so effective that we immediately added the corresponding cost function for the inflow control:

$$h_2(\boldsymbol{\lambda}) = \left(\int_0^3 (inflow_y(y,\boldsymbol{\lambda}))^2 dy \right)^{\frac{1}{2}} .$$

Combining these, we have the cost function

$$Cost(\boldsymbol{\alpha},\boldsymbol{\lambda}) = f_2(\boldsymbol{\alpha},\boldsymbol{\lambda}) + g_2(\boldsymbol{\alpha}) + h_2(\boldsymbol{\lambda}) .$$

This produced smooth solutions, but not correct ones. The effect of the two added cost integrals depends partly on their scale relative to the first integral. If g_2 and h_2 are relatively small, then the optimization code will work for most of the time on minimizing f_2, and only towards the end of the optimization will the control costs perturb the solution slightly. But since a flow with a flat bump and a zero inflow will minimize the control integrals, it's clear that a mistake in scale would cause the optimizer to spend most of its effort smoothing a poor solution.

To avoid this problem, we modified the cost functional to include weights and then allowed these weights to be changed at any time during the run. If the weights were modified, however, the optimizer had to be restarted, since the functional was changed. The new cost function had the form

$$Cost(\boldsymbol{\alpha},\boldsymbol{\lambda}) = w_1 f_2(\boldsymbol{\alpha},\boldsymbol{\lambda}) + w_2 g_2(\boldsymbol{\alpha}) + w_3 h_2(\boldsymbol{\lambda}) .$$

With these modifications, a typical run of the code would involve several steps. A first run of the code would start from a zero solution, and find a minimizer of the cost function with weights (1, 0.001, 0.001). The solution would be used as the starting point for a second optimization of the cost function with weights (1, 0.00001, 0.00001). Finally, this solution would be used as a starting point for an optimization of the cost function with weights (1, 0, 0). Thus, the control costs kept the shapes from wiggling too much while a "crude" solution was being sought. After a few such procedures, with decreasing weight, the "crude" solution was close enough to the true solution that minimization of the original functional would produce the true solution. Using the control costs allowed the program to avoid the undesirable local minima that had trapped the original program.

8 Future Work

The original problem that motivated this investigation sought to find a flow obstacle from a given test set which had a downstream profile similar to one generated by a particular shape which was *not* a member of the test set. We have not handled such cases yet, although they require no change to the program. The optimizer does not require that the functional to be minimized achieve a value of zero.

The spurious solutions that were encountered with the multiparameter problem corresponded to a bump that had a very sharply varying profile. Because the simple gridding routine that was used determines the grid of the region by the shape of the boundary, sharp variations in the bump caused distortions of the grid. These distortions, if severe enough, could cause the calculations to become unreliable. Thus, a better gridding method is desirable, whose accuracy will be less dependent on the smoothness of the boundary.

We chose to represent the space of allowable shapes by cubic splines. We imposed no convexity or positivity requirements on the shapes generated by the splines. We found that the program often generated unacceptable shapes, sometimes only as trial solutions for an iteration, but on occasion as the final solutions of the overall optimization. It is possible that we could avoid this problem by choosing a more restrictive set of allowable shapes.

Finally, we note that we need to be able to handle higher Reynolds number flows. This is so for several reasons. The wind tunnel flow we are ultimately interested in has a very high Reynolds number. Also, our method of trying to match a downstream profile is hampered when the viscosity effects dampen out the perturbations caused by the obstacle.

9 Acknowledgements

The authors would like to thank the IMA for providing a forum for the discussion of flow control issues and they would like to thank Max Gunzburger for many

helpful suggestions.

References

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

[2] Borggaard, Burns, Cliff and Gunzburger *Sensitivity Calculations for a 2D, Inviscid Supersonic Forebody Problem*, in Identification and Control of Distributed Parameter Systems, to appear.

[3] M. D. Gunzburger and J. S. Peterson, *On Conforming Finite Element Methods for the Inhomogeneous Stationary Navier-Stokes Equations*, Numer. Math. Volume 42, pages 173-194.

[4] Ohannes A 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.

[5] David 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.

[6] Carl DeBoor, *A Practical Guide to Splines,* Springer Verlag, New York, 1978.