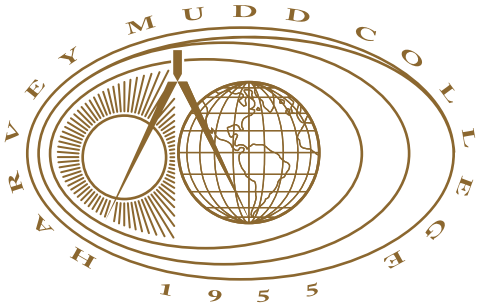
2003

Modelling Advection and Diffusion in Microchannels

Dan Beutel
Harvey Mudd College



Mixing In Microchannels

by
Dan Beutel
Andrew J. Bernoff, Advisor

Advisor: _____

Second Reader: _____

(Tom Donnelly)

June 2003

Department of Mathematics

HARVEY MUDD
COLLEGE

Abstract

Mixing In Microchannels

by Dan Beutel

June 2003

This project will investigate mixing in microchannels. Specifically, the advection and diffusion of a passive scalar, using a split-step Monte Carlo method. Numerically the implementation of this method is well understood. The current experimental geometry is a rectangular pipe with grooves on one wall. Mixing results with straight walls agree closely with experiment. The velocity field over grooves is also studied.

Table of Contents

List of Figures **iii**

Chapter 1: Introduction **1**

1.1 Organization 3

Chapter 2: Flow in a Straight Microchannel **4**

2.1 Geometry and Fluid Mechanics 4

2.2 Advection and Diffusion 7

2.3 Results 7

Chapter 3: Velocity Over Periodic Grooves **10**

3.1 Grooved Problem Setup 11

3.2 Solving for the Velocity 11

3.3 Analytical Solution 14

Chapter 4: Numerical Solution and Results for Periodic Grooves **17**

4.1 Numerical Solution 17

4.2 Numerically Motivated Revision of the Equations 18

4.3 Flow Over a Periodic Boundary - Results 21

4.4 Analysis of the Flow Over Periodic Ridges 27

4.5 Analyzing the Numerical Breakdown 27

Chapter 5: Numerical Methods **31**

5.1 Particle Method 31

5.2 Mixing Code 32
5.3 Velocity Interpolation 33
5.4 Solution of Flow over Grooves 35

Appendix A: Mixing Code **37**

Appendix B: Periodic Boundary Solution Code **40**
B.1 Calculate the Coefficients of the Stream Function 40
B.2 Calculate the Stream Function 46

Bibliography **48**

List of Figures

1.1 An active mixer. 2
1.2 Simple passive mixer using grooves showing two particle paths. . . . 3
1.3 Passive mixer using a herringbone pattern. 3

2.1 Geometry of the rectangular, straight walled pipe. 5
2.2 Diffusive broadening of two streams in a microfluidic system. 6
2.3 Numerical results of diffusive broadening, cross-section 1. 8
2.4 Cross-section 2. 8
2.5 Cross-section 3. 9
2.6 Experimental picture of diffusive broadening from Ismagilov et. al. [7]. 9

3.1 Coordinate system for the periodic boundary pipe. 10

4.1 Points to apply the boundary conditions at. 18
4.2 The points numerically integrated over. 21
4.3 Horizontal velocity u for $\varepsilon = .01$. 22
4.4 Vertical velocity w for $\varepsilon = .01$. 23
4.5 Perturbation stream function ψ for $\varepsilon = .01$. 24
4.6 Horizontal velocity u for $\varepsilon = .10$. 25
4.7 Vertical velocity w for $\varepsilon = .10$. 25
4.8 Vertical velocity w for $\varepsilon = .9$, showing the numerical instability along the edge. 26

4.9 Maximum vertical velocity $\max(w)$ as a function of ε. 28
4.10 z coordinate of $\max(w)$ as a function of ε. 28
4.11 The stream function modes β_0, β_1 and β_7 as a function of ε, the groove height. 29
4.12 Stream function modes α for $\varepsilon = .01, .3$ and 1.0. 30

5.1 Split-Step Method. 32
5.2 Graphical explanation of bilinear interpolation. 34
5.3 Comparison of interpolation with exact calculation. 35

Acknowledgments

I would especially like to thank Professor Bernoff for all of his help with my research.

I also want to thank Mike Gratton and others who worked on similar projects in the past for giving me a head start programming.

Chapter 1

Introduction

The study and application of fluid flows in micron scale channels is a rapidly growing area of research. Such channels, which are known as microchannels, have numerous potential applications in Chemisty and Biochemistry. In conjunction with various nanotechnology devices microchannels can be built into remarkably compact, sophisticated systems. Notably, mircofluidic systems can be used to mix very small, specific quantities of a solution, or even mix together solutions and allow them to react chemically. The potential applications of mixing in microchannels has inspired a great deal of research, including the work described here.

There are two ways to approach mixing, active and passive. Active mixing relies on pumping fluids through the system in some time-dependent manner. A great variety of mixing results can be obtained in this manner. However, active mixing is dependent on a complex system of pumps and control mechanisms, making it often impractical for applications. One example of a active mixing system is shown in Figure 1.1. This specific system is currently being used at UC Santa Barbara [18]. In this system fluid is pumped into multiple side channels in a time dependent manner. For instance, while fluid continuously flows from the black arrows, fluid is alternately pumped from all three red arrows, then all three blue arrows, then red again. Of course, this is only one simple pattern, more complex pumping arrangements are possible. While the depicted graphic shows the microchannel system where mixing takes place, this system depends upon a large

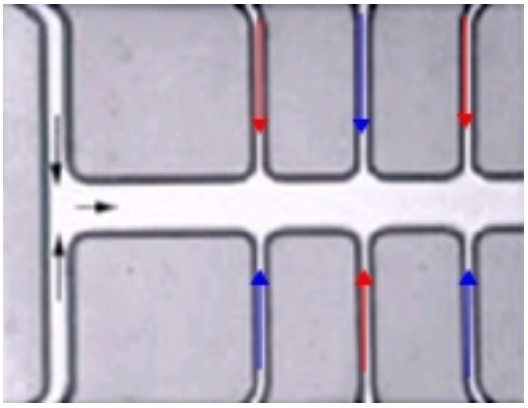


Figure 1.1: An active mixer.

external setup to control the various pumps. The complexity and bulk are a significant disadvantage of active mixing, as the size negates many of the benefits of having a micro-scale system.

Passive mixing is mixing that occurs due to the properties of the channel. Complex controllers and pumps are no longer needed, and it is more practical to build a passive mixer into a micro-scale system. The fluids are pumped or driven by a pressure gradient at a constant rate. Two or more streams come together and mix through diffusion, recirculation and other properties of the flow. By controlling the geometry, it is possible to optimize the mixing for a given pipe shape. One promising geometry, based on the work of Stroock et. al. [14], [15] is to have periodic grooves on one side of the channel at an angle to the flow, or a set of grooves in a herringbone pattern. See Figures 1.2 and 1.3, which are from the works just mentioned. In Figure 1.2 few particle paths which appear to swirl around the pipe are shown. The cross-section showing the recirculation below the pipe emphasizes this. If it is present, recirculation will greatly enhance mixing.

I have successfully simulated flow when two streams come together in a straight sided, rectangular microchannel. With these results in mind, I moved forward to

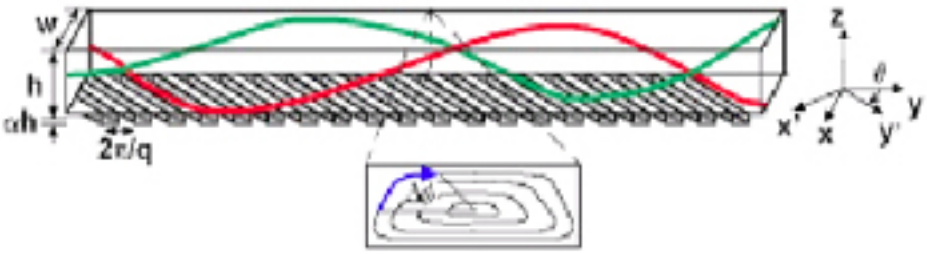


Figure 1.2: Simple passive mixer using grooves showing two particle paths.

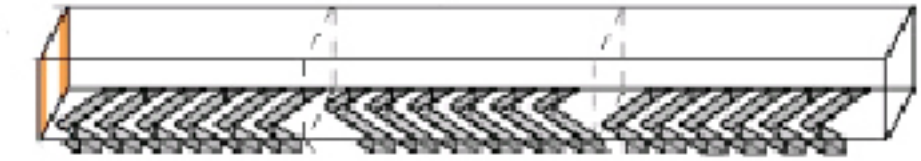


Figure 1.3: Passive mixer using a herringbone pattern.

develop the velocity field over periodic grooves, such as those in Figure 1.2.

1.1 Organization

This section explains how this report is organized. The first chapter provides some basic background and context for the problem. Chapter 2 provides a further introduction to the specific experimental geometries and the relevant equations of motion for the straight walled channel. Results for this geometry are also summarized in Chapter 2. The setup and analysis of the grooved pipe are in Chapter 3. Chapter 4 describes the numerical solution and results for the grooved pipe. Chapter 5 explains and gives some justification for the numerical methods used. The appendices contain a selection of the source code used.

Chapter 2

Flow in a Straight Microchannel

This research begins with a rectangular microchannel with straight walls and constant cross section. Specifically, I hope to reproduce the experimental results obtained by Ismagilov et. al. [7]. The experimental geometry is shown in Figure 2.1 and is described in more detail in Section 2.1.

This chapter also introduces several equations for the fluid motion and explains the results is Section 2.3

2.1 Geometry and Fluid Mechanics

One of the simplest channels to work with is straight channel with a constant rectangular cross section. The objective of starting with this system is to establish the validity of the code and the method comparison with experimental results obtained elsewhere. The velocity field in this setup is well known. Figure 2.1 shows the setup: y and z are the distance from the center axis of the pipe and x is the distance downstream. The velocity field $\vec{u} =< u(x,y,z), v(x,y,z), w(x,y,z) >$ can be found by understanding a bit about fluid mechanics.

2.1.1 Reynolds Number of a Microfluidic System

Since this problem is concerned with microfluidic systems, a number of simplifying assumptions can be made. First, the characteristic length scale is something like $10^{-3} - 10^{-4}m$ and the characteristic speeds are of roughly similar magnitudes,

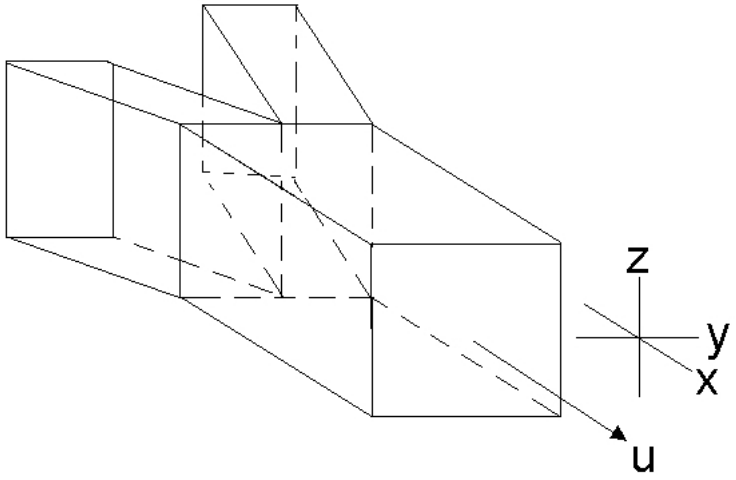


Figure 2.1: Geometry of the rectangular, straight walled pipe.

$10^{-3} - 10^{-4} m/s$. For any fluid the *Reynolds number* is given by

$$Re = \frac{Length \times Speed}{\nu} \tag{2.1}$$

where the $Length$ is a characteristic length scale of the problem and the $Speed$ is a characteristic speed of the flow. Thus, even for a fluid with $\nu = 10^{-6} m^2/s$ in a microfluidic system, $Re < 1$ and for more realistic (larger) values of ν, $Re \ll 1$. This guarantees a laminar flow.

2.1.2 Velocity Field

Given a laminar flow, the velocity in a rectangular pipe can be solved exactly. The velocity $\vec{u}$ has only a downstream component which is solely dependent on y and z, i.e. $\vec{u} =< u(y,z), 0, 0 >$. The downstream speed $u(y,z)$ is

$$u(y,z) = C\left(L^2 - y^2 - \frac{4}{L}\sum_{n=0}^{\infty} \frac{(-1)^n \, cosh(N_n x)}{N_n^3 cosh(N_n a)} cos(N_n y)\right) \tag{2.2}$$

where N_n is given by

$$N_n = \frac{(2n+1)\,\pi}{2L}.$$

The velocity is zero on the edges of the pipe and maximum in the center, with a symmetric distribution. This flow field is developed in Ward-Smith [20].

2.1.3 Experimental Results

Ismagilov et. al. [7] built a similar system and studied the mixing experimentally. They used two fluids which form a fluorescent complex when fluid A mixed into fluid B (but not vice-versa). Figure 2.2 shows how fluid A spreads into fluid B. The significant feature is the distinctive "C" shape of the mixed region when viewed in cross-section. This occurs because the fluid near the edges of the pipe is moving more slowly than the fluid in the center. Thus, the fluid on the edges has more time to diffuse before reaching a given cross-section.

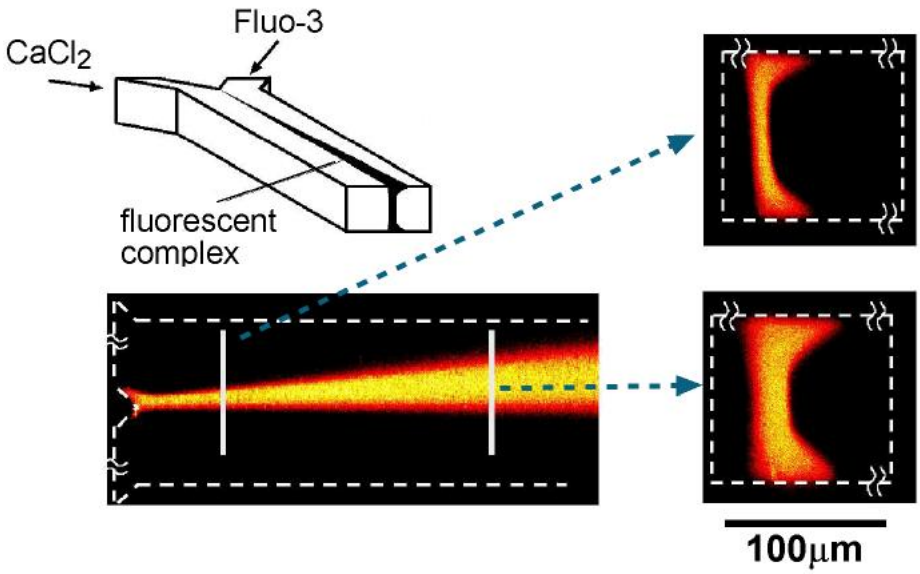


Figure 2.2: Diffusive broadening of two streams in a microfluidic system.

2.2 Advection and Diffusion

We can use this flow field, in combination with the advection diffusion equation:

$$\frac{\partial c}{\partial t} + \vec{u} \cdot \nabla c = D\nabla^2 c \tag{2.3}$$

to obtain a numerical approximation for the concentration $c(x, y, z, t)$ as a function of location and time. The concentration of one fluid mixing into another tells how "mixed" the fluids are. To understand (2.3) physically, separate advection and diffusion. Advection is movement with the velocity field of the fluid which is determined by the geometry of the pipe and nature of the fluid. Diffusion is a random spreading related to the random oscillation of the molecules. The method to simulate the advection and diffusion of a fluid is explained in more detail in Sections 5.1 and 5.2.

2.3 Results

Working with this channel, I have developed a program to simulate the flow by the method described in Chapter 5. The results, as expected, indicate that there is no mixing per-se, but just a diffusive broadening with a distinctive shape as seen in the figures below. The "C" shape mentioned above as an experimental result is seen again in my results. A series of Figures, 2.3, 2.4, 2.5 show my results at sequentially farther downstream points. A bit of explanation of the plots is in order: the red represents maximum concentration of fluid A, blue a minimum. As the fluid spreads out across the pipe, the difference between the maximum and minimum decreases and the colors blend together as in Figure 2.5 My results agree with—and provide more detail to—the experimental results of Ismagilov et. al. [7], who obtained the result shown in Figure 2.6.

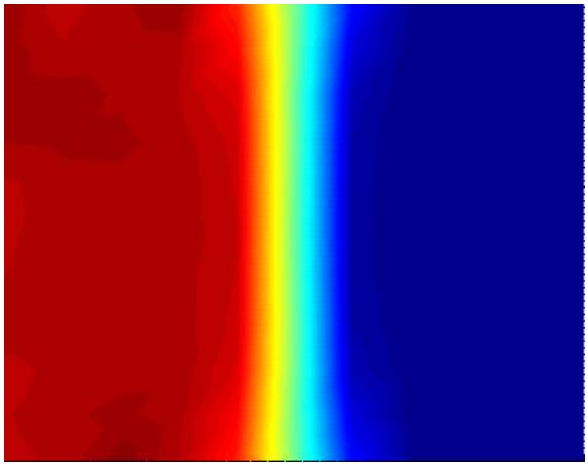


Figure 2.3: Numerical results of diffusive broadening, cross-section 1.

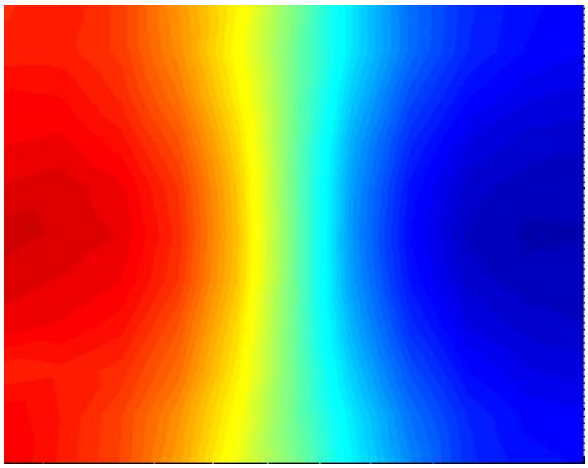


Figure 2.4: Cross-section 2.

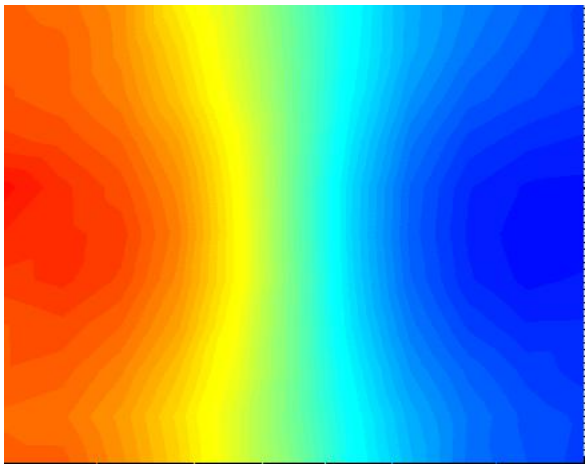


Figure 2.5: Cross-section 3.

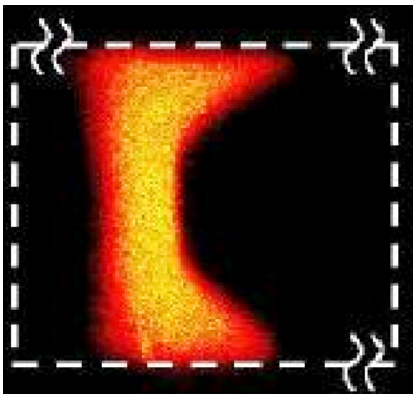


Figure 2.6: Experimental picture of diffusive broadening from Ismagilov et. al. [7].

Chapter 3

Velocity Over Periodic Grooves

To move beyond simple spreading and into passive mixing induced by periodic grooves, it is necessary to understand the manner in which the grooves alter the velocity field. Specifically, grooves along the bottom of the channel, perpendicular to the flow. From simple physical intuition, it seems that such grooves will cause the fluid to flow up, away from the grooved edge and will alter the horizontal speed, possibly even causing some bit of fluid to flow "backwards", against the main flow. This chapter will setup the problem and solve it as far as analytical techniques can be applied.

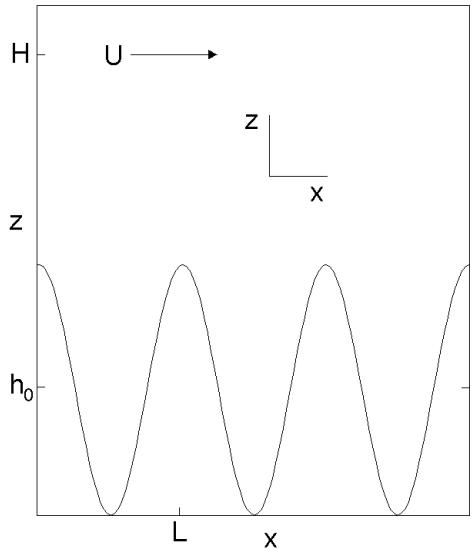


Figure 3.1: Coordinate system for the periodic boundary pipe.

3.1 Grooved Problem Setup

To simplify the problem, I worked in two dimensions in a deep pipe. The setup is shown in Figure 3.1. The pipe has a sinusoidal bottom boundary, defined by the function $z = h(x) = \varepsilon \cos(2\pi x/L) + h_0$, which has period L and average value of h_0. At the "top" of the pipe, the fluid flows with horizontal velocity U at $z = H$ and no vertical velocity.

3.2 Solving for the Velocity

Given these coordinates, I can begin to solve for the velocity. First, I start with the Navier-Stokes Equations,

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{u} + \vec{g} \tag{3.1}$$

$$\nabla \cdot \vec{u} = 0 \tag{3.2}$$

for a fluid with velocity $\vec{u}$, density ρ and viscosity ν under pressure p and gravity $\vec{g}$. In addition, *no-slip* condition—the "standard" boundary condition of fluid mechanics—is applied along the bottom boundary of the pipe. At H, there is horizontal velocity U and no vertical velocity, i.e. $\vec{u} =< U, 0, 0 >$. Since the system is two-dimensional, there is no y component to the velocity. Thus, if $\vec{u} =< u, v, w >$ the boundary conditions on the velocity can be succintly stated as

$$\begin{aligned} u(x, H) &= 0 \\ w(x, H) &= 0 \\ u(x, h(x)) &= 0 \\ w(x, h(x)) &= 0. \end{aligned}$$

For a suitably small boundary edge, there are two components of the solution: a linear shear over the equivalent flat bottom and a perturbation term associated

with the wavy bottom. Thus, it becomes convenient to write the velocity as $\vec{u} =< Uz/H, 0, 0 > + < \tilde{u}, 0, \tilde{w} >$. The boundary conditions are now

$$\tilde{u}(x, H) = 0 \tag{3.3}$$

$$\tilde{w}(x, H) = 0 \tag{3.4}$$

$$\tilde{u}(x, h(x)) = -U\frac{h(x)}{H} \tag{3.5}$$

$$\tilde{w}(x, h(x)) = 0. \tag{3.6}$$

With these boundary conditions and the Navier-Stokes Equations 3.2 it is possible to attempt to find a solution.

3.2.1 Simplifying the Navier-Stokes Equations

The Navier-Stokes equations, as written above, are highly nonlinear and very difficult to work with, even numerically. As discussed in Section 2.1.1, the Reynolds number is small because this is a microfluidic system. Since the system is in a small Reynolds number regime the viscosity term $\nu\nabla^2\vec{u}$ dominates the inertia term $(\vec{u}\cdot\nabla)\vec{u}$. Furthermore, assuming a time-independent system means that $\partial\vec{u}/\partial t = 0$ and if the system is horizontal, then gravity has no effect. Thus, the Navier-Stokes equations (3.2) reduce to

$$\nu\nabla^2\vec{u} = \nabla p \tag{3.7}$$

$$\nabla\cdot\vec{u} = 0 \tag{3.8}$$

for some pressure gradient p. With these physical assumptions, and the mathematical manipulations to follow, it is possible to solve for the velocity.

3.2.2 The Stream Function

Mathematical simplification begins by using Equation 3.8 to rewrite the perturbation velocity $\vec{\tilde{u}} =< \tilde{u}, \tilde{w} >$ (where I have omitted the y component since it must be

0):

$$\vec{\tilde{u}} =< \tilde{\psi}_z, -\tilde{\psi}_x > \tag{3.9}$$

where $\tilde{\psi}(x, z)$ is the *stream function* and the subscripts denote partial derivatives. Introducing the stream function turns the vector problem into a scalar and allows a great simplification of the equations of motion. Taking the divergence of $< \tilde{\psi}_z, -\tilde{\psi}_x >$ confirms that it is zero:

$$\nabla \cdot < \tilde{\psi}_z, -\tilde{\psi}_x >= \tilde{\psi}_{zx} - \tilde{\psi}_{xz} = 0$$

as required by Equation 3.8.

At this point I should note that the stream function $\tilde{\psi}$ given here is the *perturbation stream function* - that is, the portion of the stream function associated with the non-flat boundary. The total stream function of the system ψ is given by

$$\vec{u} =< \psi_z, -\psi_x >=< \tilde{\psi}_z + U\frac{z}{H}, -\tilde{\psi}_x > \tag{3.10}$$

and is related to the perturbation stream function by $\psi = \tilde{\psi} + Uz^2/(2H)$. From this point forward I do deal with (unless stated otherwise) the perturbation stream function, so I drop the ˜ and just refer to the perturbation stream function as ψ.

Substituting Equation 3.9 into Equation 3.7 and breaking out the vector components gives

$$\nu\partial_{xx}(\psi_z) = p_x \tag{3.11}$$

$$\nu\partial_{zz}(-\psi_x) = p_z \tag{3.12}$$

and taking $\partial/\partial z$ of Equation 3.11 and $\partial/\partial x$ of Equation 3.12 gives

$$\begin{aligned} \nu\partial_{xx}(\psi_{zz}) &= p_{xz} \\ \nu\partial_{zz}(\psi_{xx}) &= -p_{xz} \end{aligned}$$

by dividing out ν and equating the expressions for p_{xz}

$$\begin{aligned} \partial_{xx}(\psi_{zz}) = -\partial_{zz}(\psi_{xx}) &\implies \\ \nabla^2(\psi_{xx} + \psi_{zz}) = 0 &\implies \\ \nabla^4\psi &= 0. \end{aligned} \tag{3.13}$$

Thus, Equation 3.13 is the only equation of motion needed and it has analytical solutions (biharmonic functions).

3.2.3 Boundary Conditions

Now it is necessary to substitute $\vec{\tilde{u}} =< \psi_z, -\psi_x >$ into the boundary conditions, Equations 3.6 to find the boundary conditions to solve for ψ. This substitution gives

$$\tilde{\psi}_z(x, H) = 0 \tag{3.14}$$

$$\tilde{\psi}_x(x, H) = 0 \tag{3.15}$$

$$\tilde{\psi}_z(x, h(x)) = -U\frac{h(x)}{H} \tag{3.16}$$

$$\tilde{\psi}_x(x, h(x)) = 0 \tag{3.17}$$

as the boundary conditions.

3.3 Analytical Solution

Proceeding with an analytical solution for ψ, start by writing

$$\psi(x, z) = a_0(z) + \sum_{n=1}^{\infty}\left(a_n(z)\cos(k_n x) + b_n(z)\sin(k_n x)\right)$$

where $k_n = \frac{2\pi n}{L}$, the Fourier series in x. Thus the Laplacian is

$$\nabla^2\left(a_n(z)\cos(k_n x)+b_n(z)\sin(k_n x)\right) = \cos(k_n x)(\partial_{zz}-k_n^2)a_n(z)+\sin(k_n x)(\partial_{zz}-k_n^2)b_n(z)$$

so the BiLaplacian is

$$\nabla^4\Big(a_n(z)\cos(k_nx)+b_n(z)\sin(k_nx)\Big) = \cos(k_nx)(\partial_{zz}-k_n^2)^2a_n(z)+\sin(k_nx)(\partial_{zz}-k_n^2)^2b_n(z).$$

The equation of motion (Equation 3.13) is therefore

$$\begin{aligned}\nabla^4\left(a_0(z)+\sum_{n=1}^{\infty}\Big(a_n(z)\cos(k_nx)+b_n(z)\sin(k_nx)\Big)\right) &= \\ \nabla^4a_0(z)+\sum_{n=1}^{\infty}\Big(\cos(k_nx)(\partial_{zz}-k_n^2)^2a_n(z)+\sin(k_nx)(\partial_{zz}-k_n^2)^2b_n(z)\Big) &= 0\end{aligned}$$

but $\cos(k_nx)$ and $\sin(k_nx)$ are orthogonal, so each term can be broken out of the sum:

$$\begin{aligned}\nabla^4a_0(z) &= 0\\ \cos(k_nx)(\partial_{zz}-k_n^2)^2a_n(z) &= 0\\ \sin(k_nx)(\partial_{zz}-k_n^2)^2b_n(z) &= 0\end{aligned}$$

for all integer values of n. Thus, the equations of motion can be written as:

$$\nabla^4a_0(z) = 0 \tag{3.18}$$

$$(\partial_{xx}-k_n^2)^2a_n(z) = 0 \tag{3.19}$$

$$(\partial_{xx}-k_n^2)^2b_n(z) = 0 \tag{3.20}$$

again, for all integer values of n.

3.3.1 Finding $a_0(z)$

Starting with Equation 3.18, the solution is a polynomial in z:

$$a_0(z) = A_0+B_0z+C_0z^2+D_0^3. \tag{3.21}$$

Clearly the x derivative of Equation 3.21 is always zero so Equation 3.15 is guaranteed to be satisfied. By rewriting the polynomial as

$$a_0(z) = \left(1-\frac{z}{H}\right)^2\Big(\alpha_0+\beta_0z\Big) \tag{3.22}$$

Equation 3.14 will be satisfied as well.

3.3.2 *Finding $a_n(z)$ and $b_n(z)$*

Equation 3.19 has solutions

$$a_n(z) = (A_n + B_n z)e^{k_n z} + (C_n + D_n z)e^{-k_n z}.$$

However, the positive exponential terms go to infinity for large z which means that the coefficients of these terms must be zero to satisfy the boundary conditions at H. Therefore, the solution becomes

$$a_n(z) = (\alpha_n + \beta_n z)e^{-k_n z} \tag{3.23}$$

and, similarly

$$b_n(z) = (\gamma_n + \delta_n z)e^{-k_n z}. \tag{3.24}$$

3.3.3 *Solution for $\psi(x, z)$*

One solution for the perturbation stream function which satisfies the boundary conditions at H is thus given by

$$\psi(x,z) = \left(1-\frac{z}{H}\right)^2 \left(\alpha_0+\beta_0 z\right) + \sum_{n=1}^{\infty} e^{-k_n z}\Big((\alpha_n+\beta_n z)\cos(k_n x) + (\gamma_n+\delta_n z)\sin(k_n x) \Big). \tag{3.25}$$

This is the limit of analytical solutions. To proceed with finding the coefficients, and it is now necessary to obtain a numerical solution to Equations 3.16 and 3.17. This will be the topic of Chapter 4

Chapter 4

Numerical Solution and Results for Periodic Grooves

Going from the equations arrived at analytically to a numerical solution requires some revisions to the equations, which can then be solved numerically. The results can then be studied. This chapter will explain how I modify and solve the equations and the results obtained.

4.1 Numerical Solution

With the expression for ψ determined up to the constants α and β, and two boundary conditions still remaining unused, the obvious next step is to solve for α and β using these boundary conditions. The infinite sum must be truncated to N modes—more modes will increase accuracy—but since there is an e^{-n} term, higher order modes will contribute little to the sum. The truncated sum can then be substituted into Equations 3.16 and 3.17 which will be solved simultaneously at M points along $h(x)$. There are two equations to apply at each of the M points and there are $2N + 2$ unknowns, so $M = N + 1$ points will completely determine the system. Figure 4.1 shows the M points, equally spaced along the x-axis ranging from $x_0 = 0$ to $x_M = L - (M/L)$, as x_i, for $i = 0, 1, 2, \ldots, M - 1$ and located at $< x_i, h(x_i) >$. At each of these points, apply the boundary conditions on ψ_z and ψ_x. The result is a system of $2M$ equations in $2M$ unknowns, which can be solved by a packaged linear equation solver.

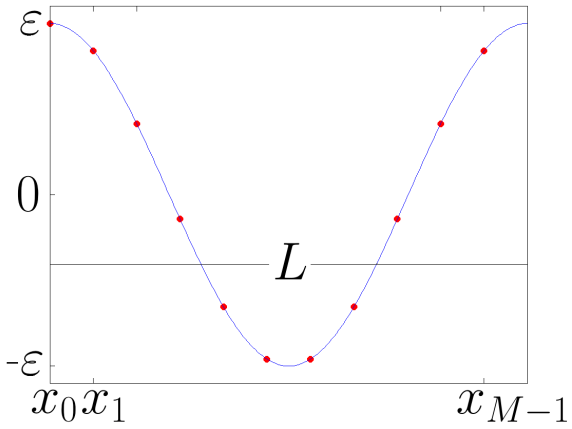


Figure 4.1: Points to apply the boundary conditions at.

4.1.1 Mass Flux Due to α_0 and β_0

There is one additional boundary condition that is imposed: the polynomial terms—the modes α_0 and β_0—must not carry any mass down the pipe. This condition can be satisfied by requiring that there is no mass flux due to these modes at $x = 0$. Mathematically speaking,

$$\int_{h_0}^{H} \left(1 - \frac{z}{H}\right)\Big(\alpha_0 + \beta_0 z\Big)\mathrm{d}z = 0.$$

Doing this integral gives

$$\alpha_0 + \frac{H + 3h_0}{4}\beta_0 = 0. \tag{4.1}$$

This equation will replace the ψ_x condition at $x = 0$ in the system of equations to be solved.

4.2 Numerically Motivated Revision of the Equations

Before beginning a discussion of the numerical solution I am going to rewrite the equations of motion in a form that proved more amenable to numerical solution.

First, I can drop the γ_n and δ_n terms since the remaining terms also solve the differential equation. I can also redefine the constants α_n and β_n such that the expression for ψ is

$$\psi(x,z) = \left(1 - \frac{z}{H}\right)^2 \left(\alpha_0 + \beta_0 z\right) + \sum_{n=1}^{\infty} e^{-k_n z} \cos(k_n x) \left(\alpha_n (1 + k_n z) + \beta_n\right) \tag{4.2}$$

the expression for ψ which I will proceed to solve numerically for α and β.

For this expression, the velocity components are

$$\begin{aligned}
\tilde{u}(x,z) &= \psi_z(x,z) \\
&= -\frac{2}{H}\left(1 - \frac{z}{H}\right)\alpha_0 + \left(1 - \frac{3z}{H}\right)\left(1 - \frac{z}{H}\right)\beta_0 \\
&+ \sum_{n=1}^{N} e^{-k_n z} \cos(k_n x) \left(- k_n^2 z \alpha_n + (1 - k_n z)\beta_n \right) \qquad (4.3) \\
\tilde{w}(x,z) &= -\psi_x(x,z) \\
&= \sum_{n=1}^{N} -k_n e^{-k_n z} \sin(k_n x) \left(\alpha_n (1 + k_n z) + \beta_n z\right). \qquad (4.4) \\
& \qquad (4.5)
\end{aligned}$$

These expressions will now be substituted into the boundary conditions to proceed towards numerical solution of α and β.

4.2.1 Revised Boundary Conditions

However, it also proved necessary to alter the boundary conditions by integrating them against $\cos(k_n x)$ so that the two boundary conditions which will be solved numerically at each of several points along the boundary are:

$$\begin{aligned}
\frac{2}{L}\int_{x=0}^{L} \tilde{u}(x, h(x)) \cos(k_n x) \mathrm{d}x &= \frac{2}{L}\int_{x=0}^{L} -\frac{U h(x)}{H} \mathrm{d}x \\
\frac{2}{L}\int_{x=0}^{L} \tilde{w}(x, h(x)) \cos(k_n x) \mathrm{d}x &= \frac{2}{L}\int_{x=0}^{L} 0 \mathrm{d}x
\end{aligned}$$

or, performing the integrals on the right hand side,

$$\frac{2}{L}\int_{x=0}^{L}\tilde{u}(x,h(x))\cos(k_n x)\mathrm{d}x = \begin{cases} -\frac{2Uh_0}{LH} & n=0 \\ -\frac{-U\varepsilon}{H} & n=1 \\ 0 & n\neq 0,1 \end{cases} \tag{4.6}$$

$$\frac{2}{L}\int_{x=0}^{L}\tilde{w}(x,h(x))\cos(k_n x)\mathrm{d}x = 0. \tag{4.7}$$

Substituting Equations 4.3 and 4.4 into Equations 4.6 and 4.7 shows the obvious benefit of this strategy: because $\cos$ terms are orthogonal, the integrals eliminate all but one term of the sums. The equations are now:

$$\begin{aligned} &-\frac{4}{L}\int_{x=0}^{L}\left(1-\frac{h(x)}{H}\right)\alpha_0\cos(k_n x)\mathrm{d}x+\frac{2}{L}\int_{x=0}^{L}\left(1-\frac{h(x)}{H}\right)\left(1-\frac{3h(x)}{H}\right)\beta_0\cos(k_n x)\mathrm{d}x \\ &+\frac{2}{L}\sum_{m=1}^{N}\int_{x=0}^{L}\cos(k_n x)\cos(k_m x)e^{-k_m h(x)}\Big(-k_n^2 h(x)\alpha_m+(1-k_m h(x))\beta_m\Big)\mathrm{d}x \\ &=\begin{cases} -\frac{2Uh_0}{LH} & n=0 \\ -\frac{-U\varepsilon}{H} & n=1 \\ 0 & n\neq 0,1 \end{cases} \end{aligned} \tag{4.8}$$

$$-\frac{2}{L}\sum_{m=1}^{N}\int_{x=0}^{L}\cos(k_n x)\sin(k_m x)e^{-k_m h(x)}\Big(\alpha_n(1+k_n h(x))+\beta_n h(x)\Big)\mathrm{d}x=0. \tag{4.9}$$

Each value of n gives two equations so $n = 0..N$ gives $2N + 2 = M$ equations to solve for the $2N + 2$ unknowns. Each of the integrals is performed numerically with p intervals between x_i and x_{i+1}—see Figure 4.2 where the x_i are as originally shown in Figure 4.1.

There still should not be mass flux due to the polynomial terms, so a modified version of Equation 4.1 still holds. This equation is now imposed in place of the $n = 0$ version of Equation 4.9 and is altered to read

$$\alpha_0 = 0. \tag{4.10}$$

Together, $N + 1$ versions of Equation 4.8, N versions of Equation 4.9 and Equation 4.10 form the complete system of equations to solve. I wrote a Matlab program

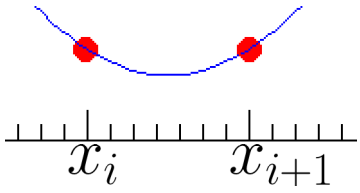


Figure 4.2: The points numerically integrated over.

to setup and solve these equations, and, using the solutions, calculate and plot the velocities and stream function. Section 5.4 describes the program and important portions of the code are listed in Appendix B.

4.3 Flow Over a Periodic Boundary - Results

To start with, it is necessary to select values for U, L, H and other constants. For a variety of reasons, $L = 2\pi$ was convenient. To keep the pipe dimensions at a uniform scale, I set $H = 2\pi$ as well. Finally, a slope of 1 for the velocity shear seemed reasonable, so I used $U = 2\pi$. These are all dimensionless parameters which can then be converted to a reasonable units system to compare with the results of an experimental setup. Furthermore, I used $h_0 = 0$. Keeping these fixed, I varied the size of the perturbation, with the following effects on the velocity fields.

4.3.1 Small Perturbations

Using $\varepsilon = .01 \Longrightarrow h = .01\cos(2\pi x/L)$—a fairly small perturbation—the horizontal velocity is almost entirely dominated by the linear shear which would be present in the absence of the grooves, as can be seen in Figure 4.3. The vertical velocity w clearly shows the perturbation in Figure 4.4, with fluid moving upwards in some areas and downwards in others. Before proceeding further, a bit of explanation of these surface plots is in order: the horizontal axes represent the x, z coordinates

within the pipe, the vertical axis and the shading show the value of the velocity (or stream function). Reds are the largest values on the surface, blues, the lowest values. For most plots, contours are also shown below the surface.

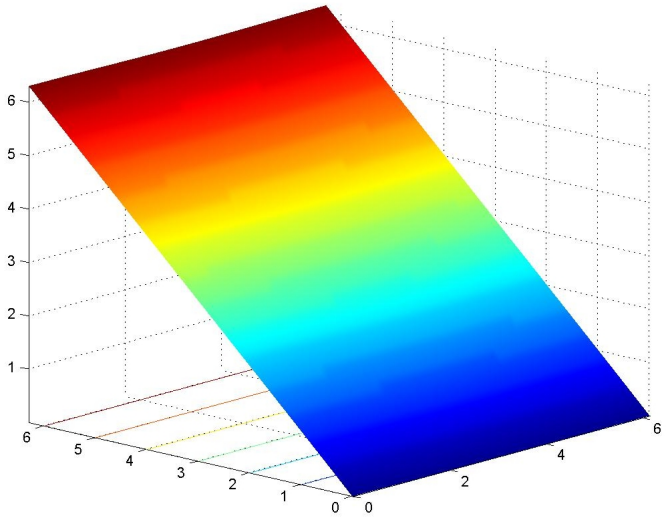


Figure 4.3: Horizontal velocity u for $\varepsilon = .01$.

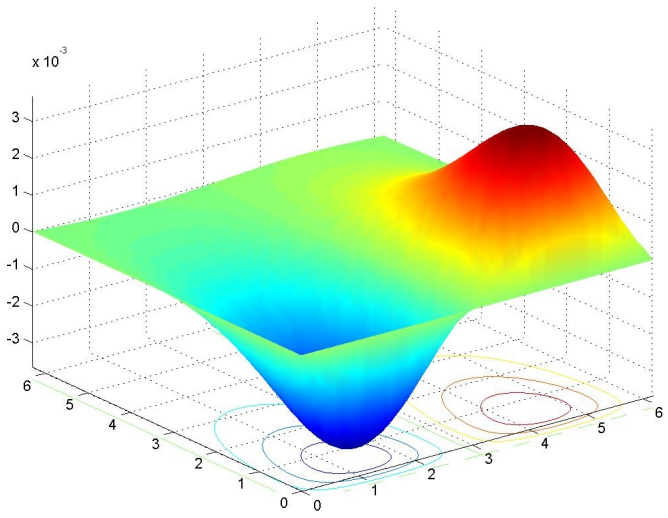


Figure 4.4: Vertical velocity w for $\varepsilon = .01$.

A more abstract view of the flow is the stream function. The perturbation stream function ψ is plotted in Figure 4.5.

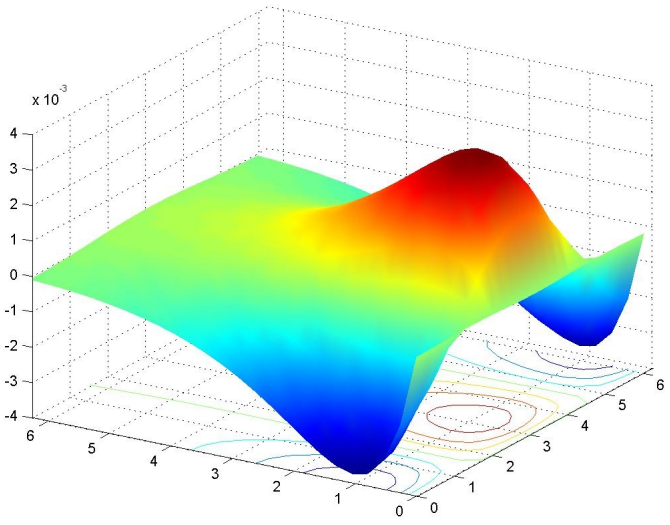


Figure 4.5: Perturbation stream function ψ for $\varepsilon = .01$.

4.3.2 Larger Perturbations

Moving to $\varepsilon = .1$ gives only a slightly more interesting result. The vertical velocity looks much the same but with larger magnitude (more on that later), and the horizontal velocity does seem a bit altered. Figures 4.6 and 4.7 illustrate. Looking at the perturbation stream function again, it also changes little qualitatively, but the magnitude increases. I didn't include another figure because the differences are slight.

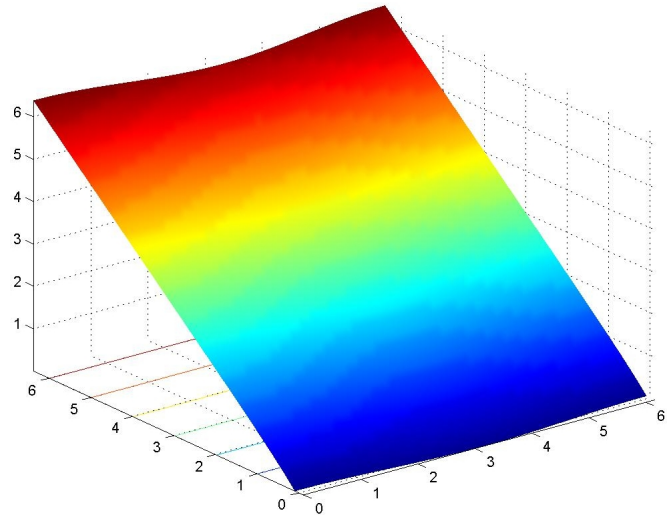


Figure 4.6: Horizontal velocity u for $\varepsilon = .10$.

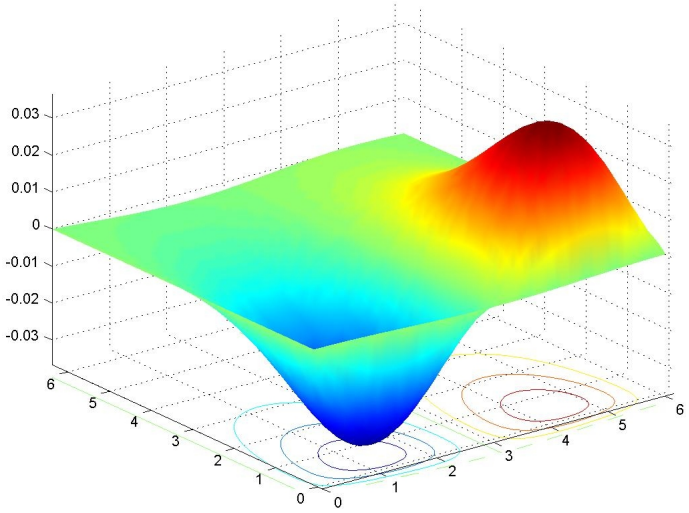


Figure 4.7: Vertical velocity w for $\varepsilon = .10$.

4.3.3 *Numerical Breakdown*

At this point, it seems that larger values of ε would give especially interesting flow with lots of recriculation. Unfortunately, the numerical solution of the system of equations begins to break down and becomes unstable, especially along the periodic boundary. Wild oscillations along this boundary quickly dominate the actual velocity values away from the boundary. For instance, with $\varepsilon = .9$, the z-velocity is shown in Figure 4.8. This clearly shows wild oscillations where the velocity should be constant at 0 along this edge.

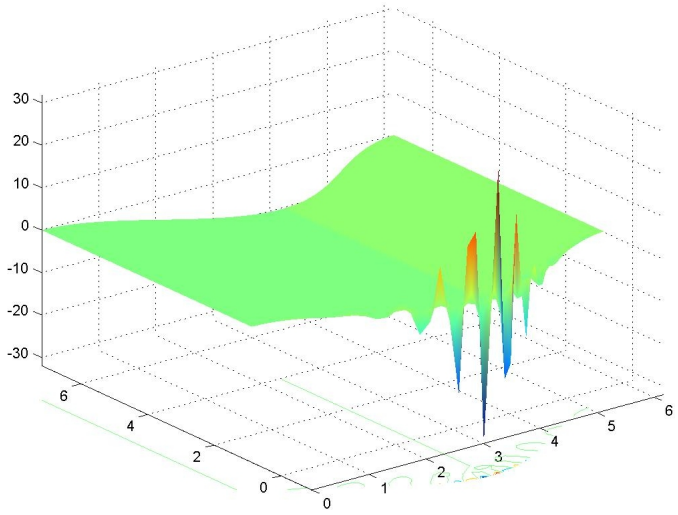


Figure 4.8: Vertical velocity w for $\varepsilon = .9$, showing the numerical instability along the edge.

These wild oscillations combined with the apparently correct results at low values of the perturbation magnitude suggest some form of numerical instability. Indeed a great deal of time and energy were invested in trying to remove the instability. Some progress was made (mostly as a result of the work detailed in Section 4.2), and successful solution of the problem for moderate perturbations was achieved. At the largest, $\varepsilon \approx .5$ gave stable results with $L = 2\pi$, so the magnitude of the perturbation was about $1/6$ the period. Section 4.5 further analyzes how the numerics

failed. These results have significant recirculation as shown above and analyzed below.

4.4 Analysis of the Flow Over Periodic Ridges

With the calculated values for the velocity, at least in the limit of small ridges, an understanding of the effect ridge size has on the flow is in order. This section will mainly look at the vertical velocity w and the perturbation stream function ψ which are the most clearly affected by the changes in ε since u tends to be dominated by the linear shear.

4.4.1 Maximum Vertical Velocity

As noted previously, the location of the maximum vertical velocity increases in magnitude and moves farther from the pipe bottom as ε increases. To study this, I looked at the maximum vertical velocity for $\varepsilon = .01, .02, \ldots, .20$. Figure 4.9 shows the dependence of the maximum vertical velocity as a function of ε, for ε =.01–.20. There is clearly a linear relationship, which turns out to be $\max(w) = .3605\varepsilon + 9.5 \times 10^{-5}$.

The location of the maximum vertical velocity also moves as ε increases. Specifically, the maximum (and minimum) move vertically away from the boundary remaining centered over the minima (maxima) of the ridge. The vertical movement is also linearly dependent upon ε according to $z(\max(w)) = 1.0259 + .0626\varepsilon$. The data are shown in Figure 4.10.

4.5 Analyzing the Numerical Breakdown

The results of the numerical solution are the coefficients α and β of the stream function ψ. Analyzing the individual coefficients (or modes) provides insight into how the solution fails as ε increases. A reasonable solution should have modes of

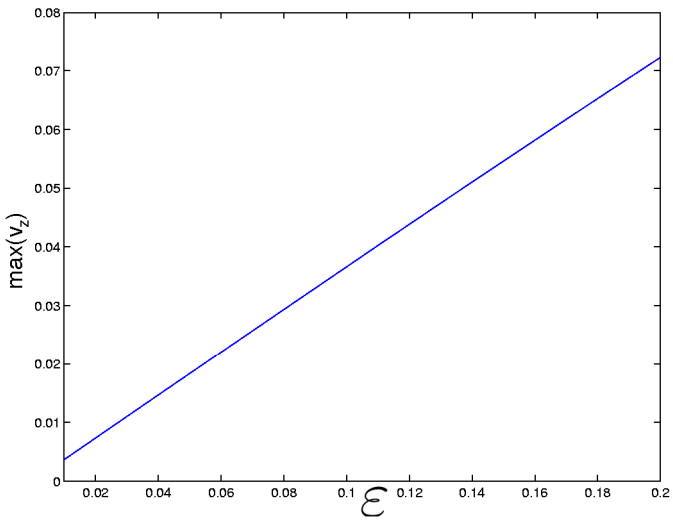


Figure 4.9: Maximum vertical velocity $\max(w)$ as a function of ε.

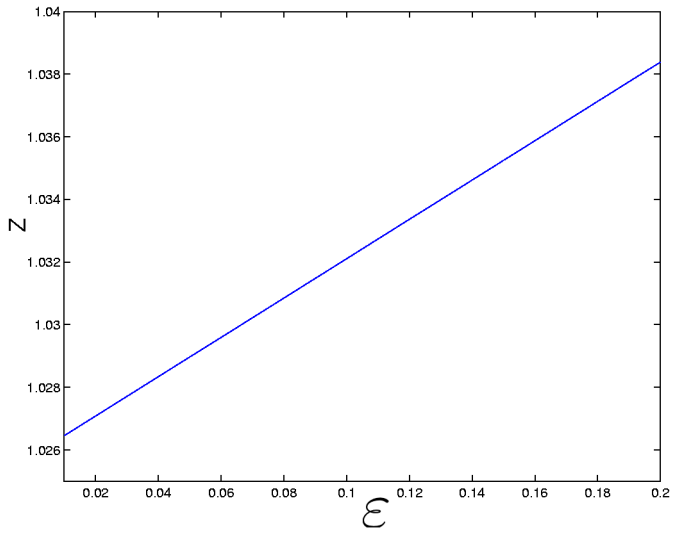


Figure 4.10: z coordinate of $\max(w)$ as a function of ε.

decreasing magnitude as n increases. The exception is that the $n = 0$ and $n = 1$ modes may be smaller than the $n = 2$ mode. For values of ε that give a correct solution, this is the behavior observed. When ε is too large, this is not the case

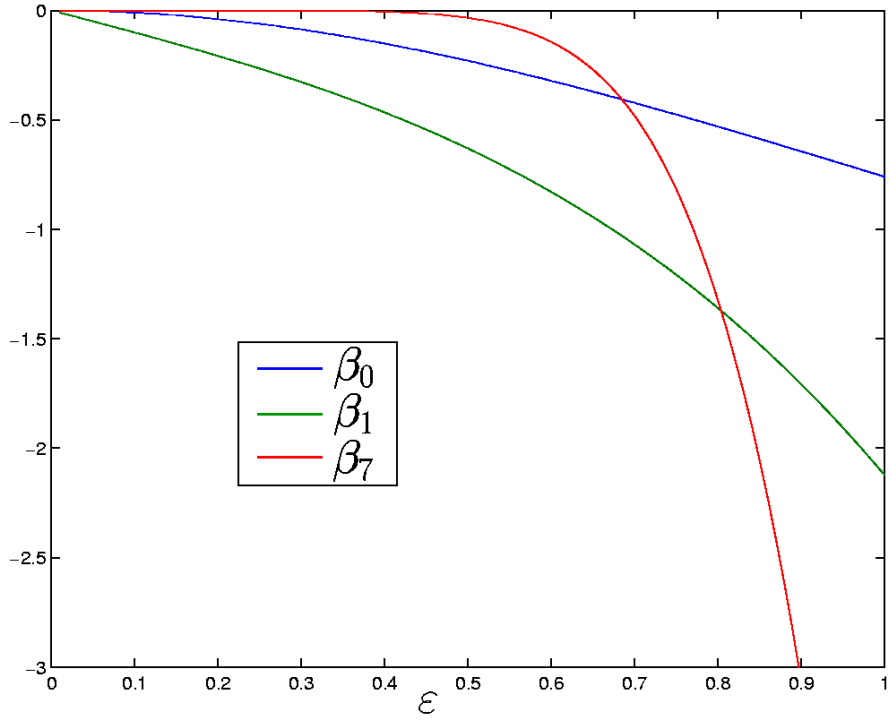


Figure 4.11: The stream function modes β_0, β_1 and β_7 as a function of ε, the groove height.

however. Figure 4.11 shows β_0, β_1 and β_7 as a function of ε. At low values of epsilon, β_0 and β_1 are indeed much larger (in magnitude) than β_7. Around $\varepsilon = .4$, β_7 begins to grow rapidly and quickly overwhelms β_0 and β_1

A different way to look at the result is to look at all the modes for a given value of ε. Figure 4.12 presents α's for $\varepsilon = .01, .3$ and 1.0. At $\varepsilon = .01$, the $n = 1$ mode dominates, as expected. With the larger perturbation, there should be some increase in the higher modes, as seen for $\varepsilon = .3$. Once $\varepsilon = 1.0$ however, there is a serious breakdown and the modes increase all the way out to $n = 5$.

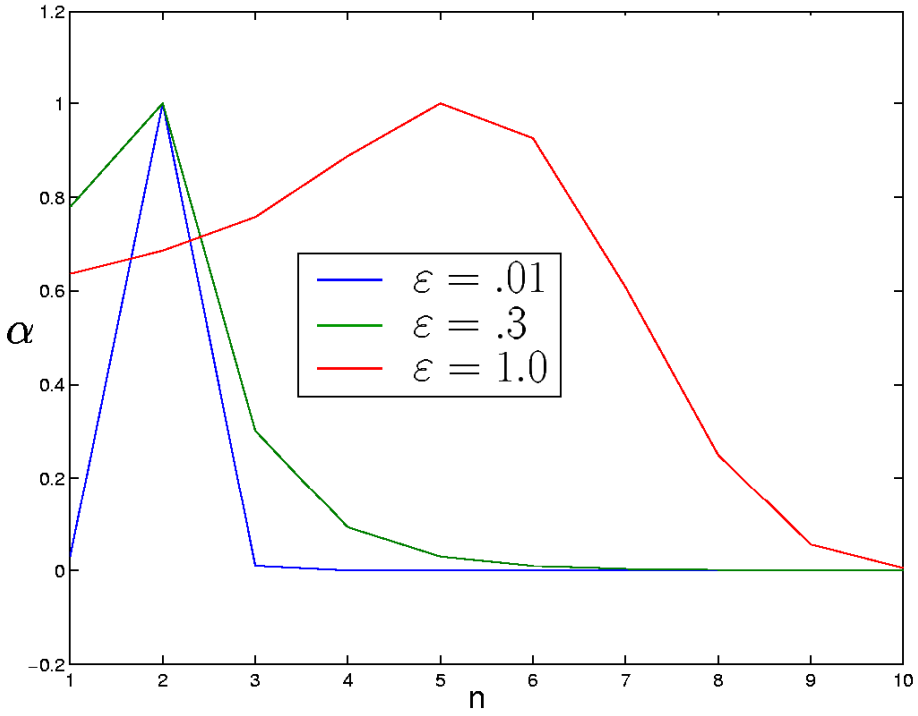


Figure 4.12: Stream function modes α for $\varepsilon = .01, .3$ and 1.0.

Chapter 5

Numerical Methods

This chapter is devoted to explaining the numerical methods and computer programs used in the research described in the previous chapters. The details of the split-step Monte Carlo method to simulate mixing, velocity field approximation by interpolation and the solution of the periodic boundary velocity are described.

5.1 Particle Method

To simulate mixing, I am using a particle method—a method that releases a lot of particles (about a million) in the velocity field and tracks their progress through the pipe being studied. The particles move according to the advection-diffusion equation as discussed in Section 2.2.

To simulate the simultaneous processes of advection and diffusion, separate advection and diffusion so that each time step for each particle consists of three substeps, which combine the advection and diffusion that would physically take place all at once. This division of labor presents an alternative more traditional grid-based simulations of fluid and the associated complexities in developing the simulation. The method has been used by a number of researchers, notably Ghoniem and Sherman [5]. This section describes how one particle moves for one time step, of length Δt, which is presumably a short time. Section 5.2 describes how it is applied to many particles for many time steps each. The method works as follows:

1. Take a random walk whose length is a Gaussian random variable in a random direction for one half time step to simulate diffusion.

2. Move with the velocity field from for one time step to simulate advection.

3. Take another random walk, again for one half time step.

Together, these steps simulate the advection diffusion equation, (2.3). A graphical illustration of the split step method is shown in Figure 5.1 which is modified from Marco Latini's thesis [8]. Lingevitch and Bernoff [9] show that the error of the method scales as the square of the time step ($error \propto \Delta t^2$).

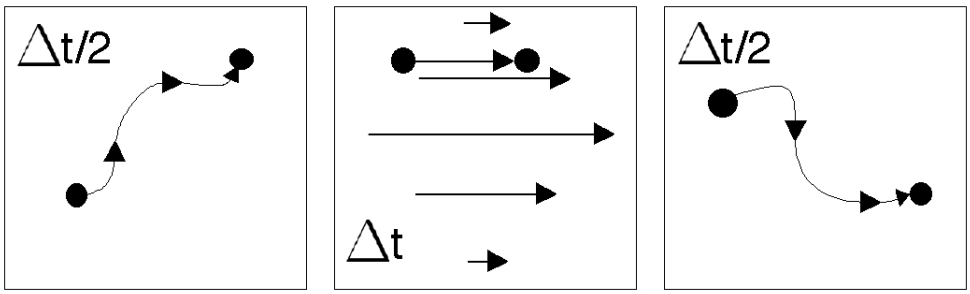


Figure 5.1: Graphical explanation of the split-step Monte Carlo method.

5.2 Mixing Code

To simulate mixing, I wrote a program which implements the particle method described in Section 5.1. The basic outline of the program is as follows:

1. Read an input file to determine the number of particles, velocity, diffusion constant and other parameters.

2. Start one particle at a random location in the $x = 0$ plane, in one half of the pipe.

3. Repeatedly apply the split-step Monte Carlo method described in Section 5.1 until the particle has moved sufficiently far downstream (as determined in Step 1). Keep track of the particle's trajectory through the pipe.

4. Figure out where that particle has crossed each of several cross-sections and store this data. Discard the trajectory data as it is no longer needed and requires an immense amount of disk space to save.

5. Repeat Steps 2–4 for many particles (how many is determined in Step 1).

The output of the program is data for a series of cross-sections. At each cross-section, the number of particles passing through a given point on a grid is recorded. Particles passing between points are weighted accordingly. The number of particles passing through a given point is proprotional to the concentration at that point. Thus, a contour plot showing the concentration at any desired cross-section can be created. This is how Figures 2.3, 2.4 and 2.5 were created. Additionally, it is easy to track how rapidly the concentration of fluid starting in the left half of the pipe moves to the right. Now this data can be analyzed as in Section 2.3.

5.3 Velocity Interpolation

Even in the "simple" square pipe the velocity field is very complex—an infinite expansion. This takes a long time to compute, even to compute just a few terms. Calculating the exact velocity every step for every particle takes up a significant amount of the program run time. One obvious potential speed-up is to tabulate velocity values in advance and interpolate to find the velocity at a given point. Specifically, calculate the velocity field at a grid of regularly spaced points before beginning the mixing simulation. I typically divided the pipe into a 50×50 grid, but any grid size that gives suitable accuracy could be used.

When the mixing program performs the advection step and requests the velocity at a given point $< x, y, z >$, which is presumably not on the grid, *bilinear interpolation* is used to find an approximate value for the velocity. Bilinear interpolation is the combination of linear interpolation in two variables. This explanation

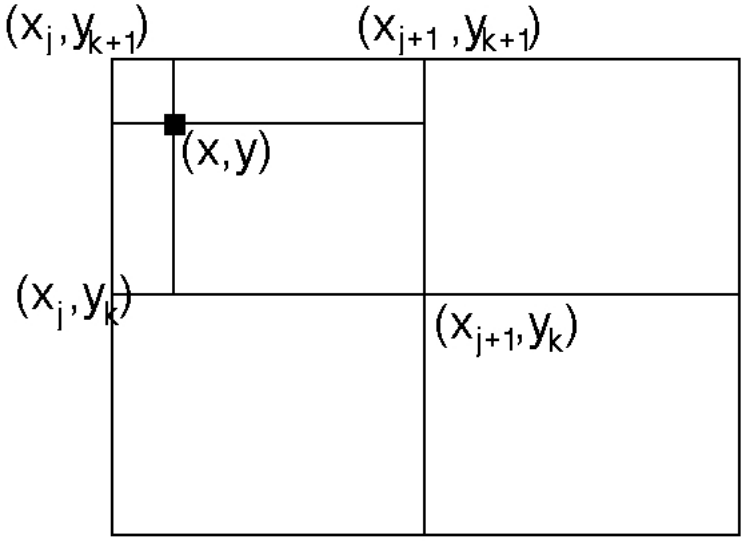


Figure 5.2: Graphical explanation of bilinear interpolation.

assumes that the velocity field is only two dimensional—the rectangular pipe for example—though it is easy to go from two to three dimensions. Bilinear interpolation is illustrated in Figure 5.2 and defined in (5.1).

$$f(x,y) = (1-\alpha)(1-\beta)f(x_j, y_k) + (1-\alpha)\beta f(x_j, y_{k+1}) + \alpha(1-\beta)f(x_{j+1}, y_k) + \alpha\beta f(x_{j+1}, y_{k+1}) \quad (5.1)$$

where

$$\alpha = \frac{x - x_j}{x_{j+1} - x_j} \qquad \text{and} \qquad \beta = \frac{y - y_k}{y_{k+1} - y_k}.$$

This has the desired effect of emphasizing the nearby values and de-emphasizing the points of the grid square which are farther away.

I compared the results of exact calculation of the velocity to the results of bilinear interpolation and found them to be consistent, allowing for the randomness inherent in this method. To quantify this, I plotted fraction of the fluid mixed versus distance downstream. The results for exact calculation and interpolation are shown in Figure 5.3. It is fairly obvious that the differences are random, and the

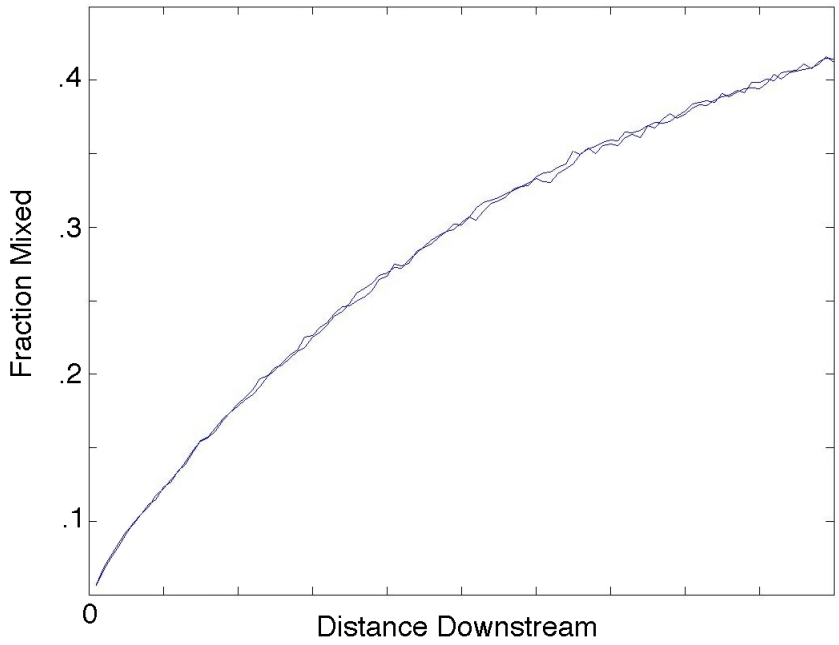


Figure 5.3: Fraction of fluid mixed due to diffusive broadening in a rectangular pipe as a function of distance downstream. Note the two lines, one each for interpolation and exact calculation, give essentially the same results.

magnitudes of the differences are small for the number of particles counted so the interpolation is valid.

5.4 Solution of Flow over Grooves

The development of the equations to be solved in Chapter 3 along with various numerically motivated revisions of Section 4.2 provide a completely determined linear system which can be solved numerically. To solve the system, I used Matlab's built-in linear equation solver. The code to solve this problem requires extensive setup to ensure that the large system of equations is correctly constructed. The system is then solved and the returned results are ready to be used to calculate the velocity components or the stream function. The Matlab function `coeff=calculate_coeffs()` which is included in Appendix B.1 sets up the equations and solves them.

The surface plots of the velocity components and stream function are generated by calculating the coefficients and using those coefficients to calculate the relevant velocity component at a regular grid of points in the $x-z$ plane. This calculation is performed by a series of functions such as `pert_str=calc_perturbation_str()` which is in Section B.2.

Appendix A

Mixing Code

There are about 2000 lines of code for the mixing program, so I have only included a small fraction of the program here. The code is written in C++ in a heavily object-oriented manner and compiled using g++ 3.1. Note that I took out most of the declarations, `#include`'s and other code that doesn't add to understanding, so don't be too surprised if something just shows up in the code. The main function for the basic mixing program follows.

```
// Function declarations:

// Reads the command line input to find the conditions
Pipe* readIn( int argc, char* argv[] );

// Save the contours to file.
void saveContours( Pipe* thePipe );

// Save the data and quit upon reciept of signal TERM.
void termHandler( int signalNumber );

// Global data:

// Pipe to work in - global to allow signal handler
// to recover data before exiting.
Pipe* thePipe;
```

```
// Main function for mixing program. Read the input,
// setup the pipe, send a bunch of particles through
// the pipe, and save the resulting contours. Most
// of the heavy lifting, including the advection and
// diffusion, is done in the Pipe class.
int main( int argc, char* argv[] ) {
    // Allow interupts without losing data.
    signal( SIGTERM, termHandler );

    // Read the input into a usable form and use it
    // to create the pipe.
    thePipe = readIn( argc, argv );

    // Move the numParts particles through the pipe,
    // saving data as we go.
    unsigned long i = 0;
    for( i = 0; i < numParts; ++i ) {

        // Make a new particle.
        Particle* part = thePipe->generateParticle();

        // Send that particle through the pipe
        thePipe->moveThroughPipe( *part );

        // Calculate the contour data for that
        // particle's trajectory.
        thePipe->updateContours( *part );
```

```
        // Save the contours to disk every so often.
        // Note that this time consuming but if not
        // done risks losing data to a mid-run crash.
        if( i % saveAfter == 0 && i != 0 ) {
            saveContours( thePipe );
        }

        // Get rid of the particle data that
        // is no longer needed.
        delete part;
    }

    // Save the contours one last time.
    saveContours( thePipe );

    delete thePipe;
    return 0;
}
```

Appendix B

Periodic Boundary Solution Code

This appendix contains select portions of the source code used to calculate the coefficients of the perturbation stream function over periodic grooves. The code was written in Matlab 6.5 and is also compatible with GNU Octave 2.1.40.

B.1 Calculate the Coefficients of the Stream Function

Function to calculate the coefficients of the perturbation stream function and velocity components.

```
function [coeff,error] = \
        calculate_coeffs( x, h, h0, eps, N, H, U, L );

% [ coeff, error ] = calculate_coeffs( x, h, N, H, U, L ) -
% solve for the coefficients of the perturbation stream
% function for a pipe with a periodic bottom boundary
% specified at h( x ) = h0 + eps * cos( 2 pi x / L )
% where h0, eps > 0. x is a vector of x-values to solve
% at, h is a vector of the values h( x ) at these x's
% over a period L. The total height of the pipe is H
% and the velocity at H is U. N is the number of terms
% in the expansion, and x is assumed to have length p * N.
% The return value is a 2 x N + 1 matrix of coefficients,
% with rows alpha and beta; columns 0 to N corresponding
```

```
% to the subscript.

  % Number of points to do integrals at
  pN = length( x );

  % Integral step
  dx = x( 2 ) - x( 1 );

  % x_m varies with column of xmat...col i => x_i, i = 1..pN
  xmat = ones( N, 1 ) * x;

  % k_n, which varies with row of k...row n => k_n, n=1..N
  k = ( ( 2 * pi / L ) * (1:N) )' * ones( 1, pN );

  % z_m, which varies with column of z...col i => z_i = h(x_i).
  z = ones( N, 1 ) * h;

  % precalculate some parts of the coefficients...note that k
  % varies by row, x and z=h(x) vary by column.
  kx = xmat .* k;
  coskx = cos( kx );
  sinkx = sin( kx );

  % Setup the matrix A, the coefficients of alpha and beta in
  % the linear system to solve.
  A = zeros( 2 * N + 2, 2 * N + 2 );

  % Setup the "solutions" vector so that we are solving
```

```
% A * coeff = b.
b = zeros( 2 * N + 2, 1 );

%
% Boundary condition that there is no mass flux due
% to the alpha_0 and beta_0 terms at x = 0.
%
A( 1, 1 ) = 1;                % alpha_0
A( 1, N + 2 ) = 0;            % beta_0
b( 1 ) = 0;

%
% (2/L) int( Psi_x * cos( k x ) ) = 0
%
for m = 1:N
  km = k( m, 1 );
  integrand = -km * sinkx .* sin( km * xmat ) \
              .* exp( -km * z );

  % alpha_m
  A( 2:(N+1), m + 1 ) = (2/L) * \
      integral( integrand .* ( 1 + km * z ), dx );
  % beta_m
  A( 2:(N+1), m + N + 2 ) = (2/L) * \
      integral( integrand .* z, dx );
end

b( 2:(N+1) ) = 0;
```

```
%
% (2/L) int( Psi_z cos( k x ), x=0..L ) =
% -(2/L) int( U h(x) /H cos( k x ), % x=0..L )
%

% alpha_0, n = 0
A( N + 2, 1 ) = -(4/(H*L)) * integral( 1 - h / H, dx );
% beta_0, n = 0
A( N + 2, N + 2 ) = (2/L) * \
      integral( 1 - ( 4 * h / H ) + ( 3 * h .^ 2 / H ), dx);

% alpha_0
A( (N+3):(2*N+2), 1 ) = -(4/(H*L)) * \
      integral( (-z/H) .* coskx, dx );

% beta_0
integrand = coskx .* \
      ( 1 - ( 4 * z / H ) + ( 3 * z .^2 / H^2 ) );
A( (N+3):(2*N+2), N + 2 ) = (2/L) * \
      integral( integrand, dx );

for m = 1:N
  km = k( m, 1 );
  integrand = coskx .* cos( km * xmat ) .* exp( -km * z );
  intn0 = cos( km * x ) .* exp( -km * h );

  % alpha_m, n = 0
```

```
  A( N + 2, m + 1 ) = -(2/L) * km ^ 2 * \
      integral( intn0 .* h, dx );

  % beta_m, n = 0
  A( N + 2, m + N + 2 ) = (2/L) * \
      integral( intn0 .* ( 1 - km * h ), dx );

  % alpha_m
  A( (N+3):(2*N+2), m + 1 ) = -(2/L) * km ^ 2 * \
      integral( integrand .* z, dx);

  % beta_m
  integrand_beta = integrand .* ( 1 - km * z );
  A( (N+3):(2*N+2), m + N + 2 ) = (2/L) * \
      integral( integrand_beta, dx );
end

% n = 0
b( N + 2 ) = -2 * U * h0 / ( L * H );
% n = 1
b( N + 3 ) = -U * eps / H;
% n > 1
b( (N+4):(2*N+2) ) = 0;

%
% Solve the system and translate into a
% more readable form.
%
```

```
coe = A \ b;
error = A * coe - b;
coeff = zeros( 2, N + 1 );
coeff( 1, : ) = coe( 1:(N+1) )';              % alphas
coeff( 2, : ) = coe( (N+2):(2*N+2) )';        % betas

%
% Independently reconstruct psi_x on boundary z = h(x)
%
alpha = coeff( 1, 2:(N+1) )';
beta = coeff( 2, 2:(N+1) )';
kn = k( :, 1 );
psi_x = zeros( 1, pN );
psi_xsinkx = zeros( 1, pN );
km = kn( 1 );

% Double check that the integral of psi_x = 0 along
% the boundary. Should be 0 to machine precision.
% Note that a zero value does _not_ guarantee a
% constant edge (it should be constant), but it does
% prove that the bc has been satisfied and give some
% idea of the rounding error associated with the
% integration.
for i = 1:pN
  xi = x( i );
  zi = h( i );
  psi_x( i ) = sum( -kn .* exp( -kn * zi ) .* sin( kn * xi ) .* \
      ( alpha .* ( 1 + kn * zi ) + beta .* zi ) );
```

```
    psi_xsinkx( i ) = psi_x( i ) * sin( km * xi );
  end

  int_psi_x = (2/L) * integral( psi_xsinkx , x(2) - x(1) )
```

B.2 Calculate the Stream Function

Calculate the perturbation stream function at a grid of points. This differs only in a few places from the functions to calculate the velocity components.

```
function pert_str = \
        calc_perturbation_str( x, z, A0, B0, A, B, k, U, H )

% pert_str =
% calculate_perturbation_str( x, z, A0, B0, A, B, k, U, H ) -
% Calculate the perturbation stream function at the points x,z
% given coefficients  A0, B0, A, B, wavenumbers k and total pipe
% height H. The max speed U isn't needed here and is ignored.

  % Number of points
  M = max( size( x ) );

  % Matrix to hold the solutions.
  pert_str = zeros( M, M );

  % Calculate the z polynomial terms first:
  pert_str = ( 1 - ( z / H ) ) .^ 2 .* ( A0 + B0 * z );

  % Calculate the sum at each x,z pair.
```

```
for i = 1:M
  for j = 1:M
    xij = x( i, j );
    zij = z( i, j );

    % a few precalculations:
    coskx = cos( k * xij );
    expkz = exp( -k * zij );
    abz = A .* ( 1 + k * zij ) + B * zij;

    % First, find a vector of each term of the sum
    stemp = expkz .* abz .* coskx;
    % Then add the sum to get the velocity
    pert_str( i, j ) = pert_str( i, j ) + sum( stemp );
  end
end
```

Bibliography

[1] A. Ajdari. Elctroosmosis on inhomogeneously charged surfaces. *Phys. Rev. Lett.*, 75:755–758, 1995.

[2] A. Ajdari. Generation of transverse fluid currents and forces by an electric field: Electro-osmosis on charge-modulated and undulated surfaces. *Phys. Rev. E*, 53:4996–5005, 1996.

[3] A. Ajdari. Transverse electrokinetic and microfluidic effects in micropatterned channels: Lubrication analysis for slab geometries. *Phys. Rev. E*, 6501:art. no.–016301, 2002.

[4] S. L. R. Barker, D. Ross, M. J. Tarlov, M. Gaitan, and L. E. Locascio. Control of flow direction in microfluidic devices with polyelectrolyte multilayers. *Anal. Chem.*, 72:5925–5929, 2000.

[5] Ahmed F. Ghoniem and Frederick S. Sherman. Grid-free simulation of diffusion using random walk methods. *Journal of Computational Physics*, 61:1–37, 1985.

[6] R. F. Ismagilov, D. Rosmarin, P. J. A. Kenis, D. T. Chiu, W. Zhang, H. A. Stone, and G. M. Whitesides. Pressure-driven laminar flow in tangential microchannels: an elastomeric microfluidic switch. *Anal. Chem.*, 73:4682–4687, 2001.

[7] R. F. Ismagilov, A. D. Stroock, P. J. A. Kenis, G. Whitesides, and H. A. Stone. Experimental and theoretical scaling laws for transverse diffusive broadening

in two-phase laminar flows in microchannels. *Appl. Phys. Lett.*, 76:2376–2378, 2000.

[8] Marco Latini. Mixing in curved pipes. Harvey Mudd College Mathematics, Senior Thesis, 2001.

[9] Joseph F. Lingevitch and Andrew J. Bernoff. Advection of a passive scalar by a vortex couple in the small-diffusion limit. *Journal of Fluid Mechanics*, 270:219–249, 1994.

[10] R. H. Liu, M. A. Stremler, K. V. Sharp, M. G. Olsen, J. G. Santiago, R. J. Adrian, H. Aref, and D. J. Beebe. Passive mixing in a three-dimensional serpentine microchannel. *J. Microelectromech. Syst.*, 9:190–197, 2000.

[11] M. J. Miksis and S. H. Davis. Slip over rough and coated surfaces. *J. Fluid Mech.*, 273:125–139, 1994.

[12] D. Ross and L. E. Locascio. Microfluidic temperature gradient focusing. *Anal. Chem.*, 74:2556–2564, 2002.

[13] H. A. Stone and S. Kim. Microfluidics: Basic issues, applications, and challenges. *Aiche J.*, 47:1250–1254, 2001.

[14] A. D. Stroock, S. K. Dertinger, G. M. Whitesides, and A. Ajdari. Patterning flows using grooved surfaces. *Anal. Chem.*, 74:5306–5312, 2002.

[15] A. D. Stroock, S. K. W. Dertinger, A. Ajdari, I. Mezic, H. A. Stone, and G. M. Whitesides. Chaotic mixer for microchannels. *Science*, 295:647–651, 2002.

[16] A. D. Stroock, M. Weck, D. T. Chiu, W. T. S. Huck, P. J. A. Kenis, R. F. Ismagilov, and G. M. Whitesides. Patterning electro-osmotic flow with patterned surface charge. *Phys. Rev. Lett.*, 84:3314–3317, 2000.

[17] S. Takayama, E. Ostuni, P. LeDuc, K. Naruse, D. E. Ingber, and G. M. Whitesides. Laminar flows - subcellular positioning of small molecules. *Nature*, 411:1016–1016, 2001.

[18] UC Santa Barabara Microfluidics Website. Available online at `http://www.engr.ucsb.edu/ microflu/index.html`, 2003.

[19] M. A. Unger, H. P. Chou, T. Thorsen, A. Scherer, and S. R. Quake. Monolithic microfabricated valves and pumps by multilayer soft lithography. *Science*, 288:113–116, 2000.

[20] A. J. Ward-Smith. *Internal Fluid Flow: The Fluid Dynamics of Flow in Pipes and Ducts*. Clarendon Press, Oxford, 1980.