Control of fluid mixing

P.G.M. Hoeijmakers
DCT 2009.44

Master's thesis

Coach: dr. F. Fontenele Araujo

Supervisors: prof. H. Nijmeijer, prof. G.J.F. van Heijst

Technische Universiteit Eindhoven
Department Mechanical Engineering
Dynamics and Control Group

Eindhoven, June, 2009
Abstract

In this thesis, we control mixing of isothermal fluids by prescribing the global entropy \( \langle s \rangle \) of the flow. In particular, based on the statistical coupling between the evolution of the global entropy and the global viscous dissipation \( \langle \epsilon \rangle \), we study stirring protocols such that \( \langle s \rangle \sim t^\alpha \Leftrightarrow \langle \epsilon \rangle \sim t^{\alpha-1} \), where \( 0 < \alpha \leq 1 \). For a linear array of vortices, we show that: (i) feedback control can be achieved via input-output linearization, (ii) mixing is monotonically enhanced for increasing \( \alpha \), and (iii) the mixing time \( \tau_m \) scales as \( \tau_m \sim \langle \epsilon \rangle^{-1/2} \).
## Contents

<table>
<thead>
<tr>
<th>Abstract</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>1</td>
</tr>
</tbody>
</table>

1 Introduction 4
   1.1 Vortices, mixing, and control 4
   1.2 Mixing in a linear array of vortices 4
   1.3 A guide through the chapters 5

2 A Linear array of counter rotating vortices 7
   2.1 Introduction 7
   2.2 Governing equations 7
      2.2.1 Streamfunction 7
      2.2.2 Driving force 9
      2.2.3 Method of normal modes 9
   2.3 The three mode model 10
      2.3.1 Spatial structure and interaction of the modes 10
      2.3.2 Equilibrium solutions 14
   2.4 Summary 15

3 Quantifying and imaging of fluid mixing 19
   3.1 Introduction 19
   3.2 Imaging of mixing 19
      3.2.1 Imaging methods 19
      3.2.2 Forward Advection 23
   3.3 Quantification of mixing 25
      3.3.1 Quantification methods 25
      3.3.2 Mixing number 25
   3.4 Summary 27

4 Entropy and viscous dissipation in the array of vortices 28
   4.1 Introduction 28
   4.2 Viscous dissipation and entropy 28
   4.3 Viscous dissipation and entropy in a linear array of vortices 29
   4.4 Viscous dissipation, entropy, and control 30
   4.5 Summary 31

5 Input output linearization of a linear array of vortices 32
   5.1 Introduction 32
   5.2 Input-output linearization 32
      5.2.1 Control Law 34
      5.2.2 Equilibria of the controlled system 34
      5.2.3 Internal dynamics 36
   5.3 Viscous dissipation, entropy, and control 39
| CONTENTS |
|----------|----------|
| Dankwoord | 83       |
| Bibliography | 85    |
1.1 Vortices, mixing, and control

Vortices are pervasive structures in nature and technology [19]. Typical examples include oceanic currents, wingtip vortices behind airplanes, and von Karman vortices. The latter is particularly appealing, as shown in figure 1.1.

A characteristic property of vortices is the rotation of matter around a common center [19]. Such movement is strongly related to velocity gradients in the flow. Velocity gradients also promote the stretching and folding of material elements, which is an important concept in the mixing of fluids [24]. A simple example is provided by stirring milk in a cup of coffee. On the other hand, industrial processes of fluid mixing often require some kind of automatic control of the stirring protocol. Examples include combination of paints, chemical components (e.g. polymers), or the ingredients of foods.

Control of fluid flows is a flourishing research field. The ability to control turbulent flows, for instance, is of considerable technological and economic interest. In particular, drag reduction by suppressing turbulent boundary layers could significantly reduce the operating costs of airplanes and ships [2, 18]. Regardless the motivation, flow control involves a remarkable interplay between fundamental sciences, applied mathematics, and engineering. In this spirit, we shall focus our attention on a simple but technologically relevant problem: mixing in an array of vortices.

1.2 Mixing in a linear array of vortices

In the present thesis, we study control of mixing in a vortex system. To this end, we consider a two-dimensional model such that counter rotating vortices are distributed in a linear array [13]. Figure 1.2 shows a typical snapshot of this flow. In particular, we address the following questions:

- What are the dynamical properties of the linear array of vortices shown in figure 1.2?
- How to quantify mixing of two fluids?
- How to control mixing using statistical properties of the flow?
1.3. A GUIDE THROUGH THE CHAPTERS

To answer such questions, we combine numerical simulations, dynamical systems theory, and nonlinear control design.

1.3 A guide through the chapters

The present thesis is organized as follows: In chapter 2, a model of a linear array of vortices is presented. Based on a normal mode analysis, the lowest three Fourier modes are used to construct a minimal nonlinear system. Then, the spatiotemporal dynamics of this model is discussed.
1.3. A GUIDE THROUGH THE CHAPTERS

In chapter 3, basic tools for the imaging and quantification of mixing are presented. In particular, two imaging methods are discussed: forward and backward advection. Since forward advection is our method of choice, it is considered in more detail. Likewise, a short overview of quantification methods is followed by a thorough discussion of the mixing number.

Chapter 4 is concerned with the derivation of a scalar measure of the velocity gradients. Such measure is provided by the global viscous dissipation. In essence, this chapter establishes a connection between mixing and control.

Chapter 5 is devoted to the input-output linearization of our linear array of vortices. By considering the viscous dissipation $\langle \epsilon \rangle$ as the output of the system, we prescribe the average velocity gradient. In this context, we study a family of powerlaws such that $\langle \epsilon \rangle \sim t^{-\beta}$. Here, a fundamental relation is found between the mixing timescale and the viscous dissipation in the flow.

Finally, Chapter 6 addresses the conclusions, shortcomings of our methods, and open questions.
2

A Linear array of counter rotating vortices

2.1 Introduction

Arrays of vortices are related to many physical phenomena. Prime examples include chaos [8], anomalous diffusion [7, 12], and transition to turbulence [23]. Here, we focus on a linear array of counter rotating vortices.

From the experimental standpoint, typical setups on counter rotating vortices use electrolytes [31, 4] or soap films [5] as working fluids. The former, in particular, is schematically shown in figure 2.1: a shallow layer of salt water driven by a Lorentz force. More precisely, such a driving just combines an electric current through the fluid with a space-periodic magnetic field (generated, for instance, by a row of magnets underneath the container).

From a theoretical standpoint, arrays of vortices are subject to elegant mathematical analysis. Classical examples include studies on flow stability [20, 11, 29, 22] and reduced models of coherent structures [11]. From such perspectives, we study a simplified model [13] that mimics the spatiotemporal dynamics observed in experiments. To this end, the present chapter is organized as follows. In section 2.2, the governing equations for two dimensional flows driven by a sinusoidal force are reviewed. Then, in section 2.3, we introduce the model proposed by Fukuta and Murakami [13]. In particular, the basic structure of the lowest Fourier modes is discussed in section 2.3.1. Finally, in section 2.3.2, the equilibrium solutions and the corresponding bifurcation diagram are studied.

2.2 Governing equations

2.2.1 Streamfunction

The velocity field in a two-dimensional flow is:

$$\mathbf{v} = (u, v),$$

where \((u, v)\) are the velocity components in the \(x\)- and \(y\)-direction, respectively. For incompressible flow, the continuity equation reads:

$$\nabla \cdot \mathbf{v} = 0,$$ (2.1)
2.2. GOVERNING EQUATIONS

Figure 2.1: Schematics of a counter rotating vortex array. a) Side view. b) Top view. The electrolyte (light red) is placed above an array of magnets with alternating polarity. The dashed blue lines represent the magnetic field. The electrodes are connected to a controlled voltage source.

where $\nabla$ is the gradient operator. In this context, the Navier-Stokes equation can be written as:

$$\frac{\partial \nu}{\partial t} + (\nu \cdot \nabla) \nu = -\nabla p + \frac{1}{\nu} \nabla^2 \nu + \frac{1}{R} f.$$  \hspace{1cm} (2.2)

Here, $p$ is the pressure, $\nu$ the kinematic viscosity, $f$ the driving force, and $R = \frac{U_0L_0}{\nu}$ denotes the Reynolds number, with $U_0$ a characteristic velocity and $L_0$ a typical length.

For two-dimensional and incompressible flow, one may introduce the streamfunction $\Psi$ as:

$$u = \frac{\partial \Psi}{\partial y}, \hspace{1cm} (2.3)$$

$$v = -\frac{\partial \Psi}{\partial x}, \hspace{1cm} (2.4)$$

which automatically satisfies equation (2.1).

By taking the curl of (2.2) and using (2.3) and (2.4) one finds\(^1\):

$$\frac{\partial \nabla^2 \Psi}{\partial t} + \frac{\partial (\nabla^2 \Psi, \Psi)}{\partial (x, y)} = \frac{1}{R} \nabla^4 \Psi + \frac{1}{R} \mathcal{F}, \hspace{1cm} (2.5)$$

where $\mathcal{F}$ is the $z$-component of $\nabla \times f$ and

$$\nabla^2 \Psi = \frac{\partial^2 \Psi}{\partial x^2} + \frac{\partial^2 \Psi}{\partial y^2}. \hspace{1cm} (2.6)$$

\(^1\)For details see Appendix A.
and

\[ \frac{\partial (a, b)}{\partial (x, y)} = \frac{\partial a}{\partial x} \frac{\partial b}{\partial y} - \frac{\partial a}{\partial y} \frac{\partial b}{\partial x}. \]  \hfill (2.7)

Equation (2.5) is the streamfunction representation of the vorticity equation.

### 2.2.2 Driving force

Now, consider the flow of an electrolyte in a shallow rectangular vessel as described in section 2.1. In this case, the forcing term \( \mathcal{F} \) is assumed to be spatially sinusoidal such that:

\[ \mathcal{F} = F(k^2 + 1)^2 \sin kx \sin y, \]  \hfill (2.8)

where \( 0 < k < \infty \) is the forcing wavenumber and \( F \) the forcing amplitude. By substitution of (2.8) in (2.5) one finds the streamfunction of the main flow:

\[ \bar{\Psi}(x, y) = \sin kx \sin y. \]  \hfill (2.9)

Note that \( \bar{\Psi} \) describes a linear array of counter rotating vortices.

### 2.2.3 Method of normal modes

To study the stability of (2.9) we write the streamfunction as:

\[ \Psi = \bar{\Psi} + \psi, \]  \hfill (2.10)

where \( \psi \) is a perturbation to \( \bar{\Psi} \). The linearized evolution of such a perturbation is obtained by substitution of (2.10) into (2.5);

\[ \frac{\partial \nabla^2 \psi}{\partial t} + \frac{\partial (\nabla^2 \bar{\Psi}, \psi)}{\partial (x, y)} + \frac{\partial (\nabla^2 \psi, \bar{\Psi})}{\partial (x, y)} = \frac{1}{R} \nabla^4 \psi. \]  \hfill (2.11)

Equation (2.11) must be complemented by a physically meaningful set of boundary conditions. In this specific case, Fukuta and Murakami [13], adopted stress free walls at \( y = [0, \pi] \). In terms of \( \psi \), those conditions read:

\[ \psi = \nabla^2 \psi = 0. \]  \hfill (2.12)

Thus, one can expand the streamfunction as:

\[ \psi(x, y, t) = e^{\sigma t + iax} \sum_{m=1}^{\infty} a_m \sin ny. \]  \hfill (2.13)

The lowest three modes of the linear stability analysis are the basis for the minimal non-linear system discussed in the next section.
2.3 The three mode model

The minimal truncated system consists of the main flow, the first term of the perturbation, and the nonlinear interaction of the former two. This leads to the truncated streamfunction:

\[ \psi(x, y, t) = \psi_0(t) \sin kx \sin y + \psi_1(t) \sin y + \psi_2(t) \cos kx \sin 2y, \quad (2.14) \]

where \( \psi_\alpha(t), 0 \leq \alpha \leq 2 \) are the amplitudes of the modes. Thus, the velocity components are given by:

\[ u = \frac{\partial \psi}{\partial y} = \psi_0(t) \sin kx \cos y + \psi_1(t) \cos y + 2\psi_2(t) \cos kx \cos 2y, \quad (2.15) \]

\[ v = -\frac{\partial \psi}{\partial x} = -k\psi_0(t) \cos kx \sin y + k\psi_2(t) \sin kx \sin 2y. \quad (2.16) \]

Substitution of (2.14) in (2.5) gives:

\[ \begin{align*}
\dot{\psi}_0 &= C_1 \psi_1 \psi_2 + C_2 \psi_0 + C_3 F \\
\dot{\psi}_1 &= C_4 \psi_2 \psi_0 + C_5 \psi_1 \\
\dot{\psi}_2 &= C_6 \psi_0 \psi_1 + C_7 \psi_2,
\end{align*} \quad (2.17) \]

where the coefficients \( C_\alpha, 1 \leq \alpha \leq 7 \), are given by:

\[ \begin{align*}
C_1 &= \frac{k(k^2 + 3)}{2(k^2 + 1)}, & C_2 &= -\frac{(k^2 + 1)}{R}, & C_3 &= -C_2, \\
C_4 &= -\frac{3k}{4}, & C_5 &= -\frac{1}{R}, \\
C_6 &= -\frac{k^3}{2(k^2 + 4)}, & C_7 &= -\frac{k^2 + 4}{R}.
\end{align*} \]

Thus, system (2.17) describes the dynamical behaviour of the three modes \( \psi_\alpha, 0 \leq \alpha \leq 2 \) in the streamfunction (2.14).

In what follows, we shall restrict our attention to the case in which \( k = 1 \). Physically, this corresponds to a main flow \( (\psi_0) \) of aspect ratio one vortices, and hence the width and height of the vortices are equal.

2.3.1 Spatial structure and interaction of the modes

This section aims to illustrate the spatial structure of the modes \( \psi_\alpha \). At this stage, we investigate the modes independently (i.e. without considering system (2.17)). In this way, a clear picture of the basic model is provided. Here, we depict the flow by plotting

\[ \text{For brevity, we drop the explicit dependence on } t \text{ and denote } \psi_\alpha(t) \text{ simply as } \psi_\alpha \]
the streamlines and velocity vectors of the model. Velocity fields consisting solely of one of the three modes are shown in figures 2.2a-2.2c. The first mode $\psi_0$ describes an counter rotating array of vortices. The second mode corresponds to a pure shear flow, and the third mode induces a vortex lattice.

Figure 2.2: Velocity fields associated with the lowest three modes. a) $\psi_0$ consists of a linear array of vortices. b) $\psi_0$ induces a shear flow. c) $\psi_2$ defines a vortex lattice. For all figures the forcing wavenumber $k = 1$. 
2.3. THE THREE MODE MODEL

By changing the relative weight between modes, a wealth of different flow fields can be created. Some basic examples containing two (equally weighted) modes are given in figures 2.3a-2.3c. Such cases can be readily explained by studying the velocity field due to the base modes. In figure 2.3a for example, a combination of the shear and the main mode, the shear mode cancels the middle vortex structure and amplifies the side vortices. Another example is the tilted vortex structure in figure 2.3b, which results from the diagonal orientation of equally signed vortices in the $\psi_2$ flow.

![Velocity fields](image)

Figure 2.3: Velocity fields associated with combinations of two equal modes. a) Main mode and the shear mode. b) Shear mode and the vortex lattice. c) Main mode and the vortex lattice. For all figures $k = 1$
More complicated situations arise for unequally weighted combinations of two modes, a selection is given in figures 2.4a-2.4c.

In what follows, we will designate a flow with $\psi = [1,0,0]$ as a $\psi_0$ flow, and a flow with $\psi = [1,1,1]$ as a $\psi_{0,1,2}$ flow.

Figure 2.4: Velocity fields for some specific cases containing two non-equal modes. a) $\psi_0 > \psi_1$ before elimination of middle vortex. b) $\psi_0 < \psi_1$, the middle vortex disappeared while the outer ones have merged. d) $\psi_0 > \psi_2$ the main flow is tilted due to the presence of the lattice mode. For all figures $k = 1$. 

2.3. THE THREE MODE MODEL
2.3. THE THREE MODE MODEL

2.3.2 Equilibrium solutions

In this section we investigate the equilibria of system (2.17) and illustrate the corresponding velocity fields.

Here, let us first choose an appropriate bifurcation parameter. Careful examination of system (2.17) reveals two possibilities: the forcing wavenumber $k$, and the forcing amplitude $F$. For a typical setup such as shown in figure 2.1, $F$ can be easily controlled by manipulating the current through the electrolyte. To change $k$ on the other hand, one has to modify the mutual distance between the magnets. Thus, in the following we study the steady flow behaviour as function of $F$.

Setting $\dot{\psi}_a$ in equation (2.17) to zero yields six equilibrium solutions, generated by:

\[
H_1 = F, \quad \text{(2.18)}
\]

\[
H_2 = \frac{2\sqrt{6}}{3} \frac{k^2 + 1}{k^2 + 3/2} R', \quad \text{(2.19)}
\]

\[
H_3 = \frac{2}{R} \frac{(k^2 + 1)(k^2 + 4)}{k^2\sqrt{k^2 + 3}} \sqrt{|F|} H_2 - 1, \quad \text{(2.20)}
\]

\[
H_4 = \frac{\sqrt{6}}{3} \frac{k}{k^2 + 4} H_3, \quad \text{(2.21)}
\]

where $H_2, H_3,$ and $H_4$ only exist if $|F| > H_2$. Table 2.1 summarizes the possible combinations between (2.18)-(2.21).

<table>
<thead>
<tr>
<th>Equilibrium</th>
<th>$\psi_0$</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A'_+^*$</td>
<td>$H_1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$A'_-^*$</td>
<td>$-H_1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$A'_+$</td>
<td>$H_2$</td>
<td>$H_3$</td>
<td>$-H_4$</td>
</tr>
<tr>
<td>$A'_-$</td>
<td>$-H_2$</td>
<td>$-H_3$</td>
<td>$-H_4$</td>
</tr>
<tr>
<td>$A_-^*$</td>
<td>$-H_2$</td>
<td>$H_3$</td>
<td>$H_4$</td>
</tr>
</tbody>
</table>

Table 2.1: Equilibria of system (2.17). The values of $H_1, ..., H_5$ are given by eqs. (2.18)-(2.21).

Now, we shall sketch the bifurcation diagram based on a linear stability analysis\(^3\). In particular, let us consider the case in which $k = 1$ and $R = 10$.

Figure 2.5 shows the stationary solutions $\psi_a$ as functions of $F$. Clearly, the system undergoes a supercritical pitchfork bifurcation. At $\pm F_{bif}$, where $F_{bif} = 0.8165$, $\psi_0$ and $\psi_1$ split into a lower (negative) and upper (positive) branch. In addition note that, $|\psi_0| = F_{bif}$ for $|F| \geq F_{bif}$. Physically, this means that for sufficiently large forcing, the $\psi_1$ and $\psi_2$ modes start to dominate the flow.

For $F > F_{bif}$, the diagram has two equilibrium solutions, denoted by $A_+$ and $A'_+$. Solution $A_+$ ($A'_+$) is defined by $\psi_0 = F_{bif}$, the upper (lower) branch of $\psi_1$, and the lower

\(^3\)For details see Appendix E.1

\(^4\)The choice for $R = 10$ is arbitrary.
Figure 2.5: Bifurcation diagram of the system for \( k = 1 \) and \( R = 30 \). Solid lines: stable branches. Dashed lines: unstable branches. Big arrows: stable equilibria for \( F > F_{bif} \).

(upper) branch of \( \psi_2 \). Likewise, two equilibria emerge for negative forcing: \( A_- (A'_-) \) consisting of \( \psi_0 = -F_{bif} \) and the lower (upper) branches of \( \psi_1 \) and \( \psi_2 \). Overall, the flow fields associated with \( A_+ \) and \( A_- \) are symmetric to each other. The same holds true for \( A'_+ \) and \( A'_- \).

To reveal such a symmetry, let us first consider typical snapshots from \( A_+ \). As shown in figure 2.6, the flow field becomes progressively tilted as \( F \) is increased. In particular, comparison between figure 2.6a and 2.6b shows that the middle vortex is absent at \( F = 3 \). This is in accordance with the findings in the last section, where figures 2.4a and 2.4b show a similar difference when the shear mode is stronger than the main mode.

Now, consider typical snapshots for \( A_- \). As shown in figure 2.7, the flow fields are symmetric to those in figure 2.6. More precisely, both the direction of the tilt and the sign of the dominant vortex are reversed.

Finally, let us comment on the similarities between \( A_+ \) and \( A'_- \). As depicted in figure 2.8, \( A'_- \) flows are simply shifted \( \frac{1}{2} \pi \) on the x-axis as compared to those of figure 2.6. The same holds for \( A'_+ \) and \( A_- \).

### 2.4 Summary

In this chapter we presented a model for counter rotating vortices (2.17), as proposed by Fukuta and Murakami [13]. Such model is based on a linear stability analysis of the main flow (e.g. the linear array of vortices). More precisely, the lowest three Fourier modes are used to construct the minimal nonlinear truncated system. Physically, the first mode corresponds to the main flow (counter rotating vortex array), the second to a shear flow,
and the third to a vortex lattice. The derived system of ordinary differential equations
describes the evolution of such modes as function of the forcing. It was shown that (i)
for increasing forcing the two higher modes start to dominate the flow (e.g. tilting) and
(ii) flow fields for positive and negative forcing are symmetric to each other.

Figure 2.6: Equilibrium solution $A_+$ for increasing forcing $F$. a) $F = 1$. b) $F = 3$. c) $F = 30$. 
Figure 2.7: Equilibrium solution $A_-$ for increasing forcing $F$. a) $F = 1$. b) $F = 3$. c) $F = 30$. 
Figure 2.8: Equilibrium solution $A'_\epsilon$ for increasing forcing $F$. a) $F = -1$. b) $F = -3$. c) $F = -30$. 
3.1 Introduction

Mixing of fluids is an important process in many industrial applications. Typical examples include the combination of paints, reaction of pharmaceutical liquids, and food processing as a whole.

To evaluate the quality of a mixing process, a common approach consists in taking snapshots of the flow. In an experimental setting, such images represent the concentration of a passive field like a fluorescent dye [32, 14]. On the other hand, in computer simulations it is more common to visualize mixing via the dynamics of passive tracers [15, 28, 26, 32, 27]. In both cases, these images are normally used for the quantification of mixing.

A variety of mixing measures is available. Examples include the intensity of segregation [10], the mix-norm [21], the Shannon entropy [6], and the mixing number [28]. The majority of these methods require the specification of an averaging volume. In particular, they are extremely sensitive to the choice of such volume’s size. The mixing number does not suffer from this drawback, and hence is better suited for our application.

In section 3.2 we discuss two imaging methods: forward and backward advection. Since forward advection is our method of choice, we then present the details of this method. Likewise, section 3.3 contains a short discussion of the different mixing measures along with a thorough description of the mixing number.

3.2 Imaging of mixing

3.2.1 Imaging methods

The kinematics of binary mixing (i.e. two fluids) can be described by a discrete black and white image. Two approaches towards such a description are forward and backward advection. As depicted in figure 3.1a, forward advection starts with a coloured grid at $t_0$, the initial configuration, and treats every coloured cell’s midpoint as a particle. Next, the particles are integrated forward in time to $t_1$. Finally, the grid is coloured according to the new particle distribution. An arbitrary grid cell inherits the colour of the particle
currently residing somewhere in the grid cell. This means that a black and white image can be obtained by forward advection of just one colour, since all the remaining cells are of the opposite colour by default.

![Figure 3.1: Forward versus backward advection. a) Forward advection of the coloured cells midpoints. b) Forward advection of points in the coloured cells. c) Backward advection of cells midpoints on the upper row.](image)

In order to improve contrast, one can use multiple particles per coloured cell, depicted in 3.1b.

Observe that the methods above evolve a particle along the velocity field forward in time. Backward advection\(^1\) on the other hand, shown in figure 3.1c, traces the midpoints of every cell backward in time. Observe that only backward advection of the upper row is shown. Therefore, the method starts at time \(t_1\) and evolves the cells midpoints along the inverse velocity field until \(t_0\). Since the initial configuration is known, a grid cell at time \(t_1\) can be coloured according to the colour of the cell it originated from, in the figure denoted by 'colour lookup'.

Thus, forward advection can be summarized by the question: where does a particle go? In contrast, the question for backward advection reads: From where does a particle come? The differences between the methods are also illustrated by figure 3.1. Forward advection pictures the stretching fluid by increasing the mutual distance between the particles. Improved forward advection however, colours the whole upper row black. Likewise, backward advection leads to a picture where the particles are 'stretched' from the lower to the upper row. Figure 3.3 shows how this behaviour translates to the case of a \(\psi_0\) flow. The initial configuration is depicted in figure 3.2. In terms of contrast, the images of the latter two methods are of higher quality. Although the example suggests differently, improved forward advection will ultimately suffer from the same effect as normal forward advection, hence backward advection can be considered superior.

The computational cost of backward advection, as compared to forward advection, is the main drawback of the method. In addition to the integration of all cell’s midpoints,

\(^1\)Also called backtrace method
instead of only the coloured ones, it has to repeat this process for every snapshot. Forward advection on the other hand, advects every particle only once through the entire time interval. The particle distributions at every time interval are known immediately. In order to demonstrate this, consider a situation where ten snapshots at a resolution of $200 \times 100$ are desired, while 50% of the grid is coloured black in the initial configuration. In this case, forward advection needs to integrate $1 \cdot 10^4$ midpoints, while backward advection needs to integrate $2 \cdot 10^5$ particles!

The mixing curves presented in the next chapters typically consist of hundreds of snapshots. Since the backtrace method becomes computationally expensive at such numbers, we shall use the (simple) forward advection method.

Note that in the case in which one would like to create 2D slices of a particle distribution in a 3D domain, backward advection is the only feasible method.

![Initial particle distribution](image.png)

Figure 3.2: Initial particle distribution. One third of the domain is occupied by black particles.
3.2. IMAGING OF MIXING

Figure 3.3: Particle distributions resulting from a $\psi_0$ flow after 20 dimensionless time units. a) Forward advection. b) Backward advection. c) Improved forward advection.
3.2. Imaging of Mixing

3.2.2 Forward Advection

Forward advection consists of evolving a particle forward in time. Primarily, such evolution determines a trajectory, which may be further sampled for imaging purposes. As shown in figure 3.4, this can be accomplished by taking successive snapshots of the positions of the particles in space and time.

Three steps are involved in the process: integration, discretization, and grid colouring. First, the integration is carried out by an explicit Runge-Kutta scheme. Second, the particle positions in physical space are translated to the cell space, and third, the appropriate cells are coloured black. This process is summarized in listing 3.1.

In the context above, let us denote the physical domain by $D = [D_x, D_y]$, and the cell domain as $(X_{reso}, Y_{reso})$, where $X_{reso}$ and $Y_{reso}$ are the number of cells in each direction. Consequently, a particle can have a certain position $(x, y)$ in the physical domain, which corresponds to a position $(Z_x, Z_y)$ in the cell domain. As depicted in figure 3.5, the combination of the size of the physical domain and the resolution of the cell domain leads to two scaling factors:

$$e_x = \frac{D_x}{X_{reso}},$$
$$e_y = \frac{D_y}{Y_{reso}}.\quad (3.1)$$

Thus, a coordinate transformation between the physical and cell spaces can be implemented as:

$$Z_x = \left\lceil \frac{x}{e_x} \right\rceil,\quad (3.2)$$
$$Z_y = \left\lceil \frac{y}{e_y} \right\rceil,$$

where $\lceil f \rceil$ represents rounding to the ceiling of $f$.

Figure 3.4: Typical snapshots at $t_f$ resulting from forward advection. The dashed line represents a particle trajectory.
3.2. IMAGING OF MIXING

%Initialize
N= Number of black particles
Zy,Zx= Cell indexes of black particles

%Main loop
% A parallel for loop to make use of both CPU cores.
parfor (i = (1:N))

% Do forward advection of every particle from
% t_0->t_end.
% ode45 is Matlab’s 4th order
% Runge-Kutta implementation.
ode45( . . )

% Convert physical–coordinates to cell–coordinates
% using a mex–file (C-code called via Matlab).
coordTransformation( . . )

% Colour the map accordingly, for all t at once.
% The map consists of a group of images stacked
% in a 3D array.
Map(Zy,Zx,t)=1;
end

Listing 3.1: The structure of the forward advection code.

Figure 3.5: Discretization of the physical domain. The scaling factors are $e_x$ and $e_y$. The cell coordinates are given by $Z_x$ and $Z_y$. 

24
3.3. QUANTIFICATION OF MIXING

3.3 Quantification of mixing

3.3.1 Quantification methods

Methods to quantify mixing always rely on images of the process. Such images are obtained from either experiments or computer simulations. Images originating from experiments often vary continuously in intensity.

In order to apply common quantification methods in computer simulations, the binary images have to be translated to an intensity profile. Usually, this is accomplished by dividing the image in small volumes, and calculating the ratio of black and white cells in each volume. Typical examples include Danckwerts intensity of segregation [10] and the Shannon entropy [6]. However, these measures are very sensitive to the size of the averaging volumes [28].

Such sensitiveness was carefully addressed by Stone and Stone [28], who proposed an improved averaging technique for binary images. Their quantification method does not involve an averaging volume, and hence can be applied directly to binary images.

3.3.2 Mixing number

The starting point of the quantification proposed by Stone and Stone [28] concerns how rapidly diffusion alone can homogenize a fluid distribution. Since the time needed for a particle to diffuse a distance $\delta$ is proportional to $\delta^2$, the squared distance between a white particle and a black particle provides an indication of how fast the two would homogenize. Although binary images are generated from pure kinematic considerations, the average distance between opposite colors provides an estimate of the degree of mixing.

Now, let us proceed with a more formal description of the mixing number. Let a domain of $N$ cells labeled by $n_i$, with $1 \leq i \leq N$. Given a cell $n_i$, consider the set $G_i$ of all cells whose color is opposite to that of $n_i$. Then compute the distance $\Delta(n_i, G_i)$ between $n_i$ and $G_i$ as:

$$\Delta(n_i, G_i) = \min_j \{d(n_i, n_j) : n_j \in G_i\}, \quad (3.3)$$

where $d(n_i, n_j)$ is the Cartesian distance between cells. In this way, the mixing number $m$ proposed by Stone and Stone [28] is defined as

$$m = \sum_{i=1}^{N} \frac{\Delta^2(n_i, G_i)}{N}. \quad (3.4)$$

Qualitatively, $m$ corresponds to the average distance between opposite colors.

Examples and implementation

Figure 3.6 shows an example of the computation of the mixing number. It should be noted that the mixing number has a theoretical minimum, which is exactly the scaling factor $e_{x,y}^2$, although only a perfect 'checkerboard' image would satisfy $m = e_{x,y}^2$. 
3.3. QUANTIFICATION OF MIXING

Figure 3.6: Typical computation of the mixing number (3.4).

Figure 3.7a depicts the time history of the mixing number for a $\psi_0$ flow and figure 3.7b for a $\psi_{0,1,2}$ flow. Both plots confirm that the mixing number is only slightly dependent on the grid size. However, the influence of increasing grid resolution on the absolute minimum value can be observed well.

Figure 3.7: Mixing time series for three grid resolutions. (a) $\psi_0$ flow. (b) $\psi_{0,1,2}$ flow

Calculation of the minimum distance is the most expensive step of the implementation. Since a considerable amount of points can be involved, e.g. $N = 2 \cdot 10^4$ on a $200 \times 100$ grid, an efficient algorithm is desirable. Therefore, we performed a comparison between three implementations: Matlab, a (CPU only) C mex-file, and a C mex-file involving the Graphical Processing Unit (GPU)². The C mex-file outperformed the Matlab code. However, as figure 3.8 shows, the GPU algorithm proved to be the most efficient. Clearly, the differences between the CPU and GPU implementation increase with grid resolution.

²For details see Appendix C
3.4 Summary

In this chapter we discussed the imaging and quantification of mixing. Three imaging methods were presented, forward advection, improved forward advection, and backward advection. Such techniques produce a binary image (e.g. black and white) of the mixing. Because backward advection is computationally expensive, forward advection is our method of choice. Next, we considered a mixing measure that can be directly applied to a binary image: the mixing number \cite{28}. It was shown that such measure is relatively independent of the grid resolution.

![Figure 3.8: Calculation time of mixing number for ten images. Blue: GPU. Red: CPU](image)

3.4. SUMMARY
4

Entropy and viscous dissipation in the array of vortices

4.1 Introduction

As discussed in chapter 3, the measure (3.4) reasonably quantifies mixing between two fluids. However, the relation between the mixing number $m$ and the dynamics of the flow is non-trivial. One can thus describe the problem from two perspectives, namely: (i) the hydrodynamics of stretching/folding in the fluid and (ii) the statistical properties of the flow.

From a fluid dynamics viewpoint, mixing is a process closely related to gradients in the velocity field. After all, such gradients are fundamental for stretching and folding of fluid. But how to characterize velocity gradients? Among the several possibilities (e.g. vorticity, strain rate, etc) the viscous dissipation rate $\epsilon$ provides a particularly simple description since it is a scalar quantity. Moreover, $\epsilon$ can be explicitly related to the entropy $s$ of the flow.

From a statistical perspective, mixing is a process that increases the entropy. This suggests that $\epsilon$ could provide a connection between the hydrodynamic and statistical descriptions of mixing. In particular, for system (2.17) we may write the entropy evolution as a direct function of the modes $\psi_a$. Such connection is also appealing from a control perspective. More precisely, we could consider $\epsilon$ as the output of the system.

To formalize the ideas mentioned above, this chapter is organized as follows. In section 4.2 we will review the equations for the viscous dissipation and entropy. Then, in section 4.3 an explicit expression is derived for the viscous dissipation in a linear array of vortices. Finally, in section 4.4 we will shortly discuss the results in a control context.

4.2 Viscous dissipation and entropy

The local viscous dissipation$^1$ $\epsilon$ in a two dimensional fluid flow is [17]:

$$\epsilon = 2 \left[ \left( \frac{\partial u}{\partial x} \right)^2 + \left( \frac{\partial v}{\partial y} \right)^2 + \frac{1}{2} \left( \frac{\partial u}{\partial y} \right)^2 + \frac{1}{2} \left( \frac{\partial v}{\partial x} \right)^2 + \frac{\partial u}{\partial y} \frac{\partial v}{\partial x} \right]. \quad (4.1)$$

$^1$In dimensionless units.
4.3. VISCOSOUS DISSIPATION AND ENTROPY IN A LINEAR ARRAY OF VORTICES

In order to describe the velocity gradients of the flow from a global perspective, we shall take the space average:

\[ \langle \ldots \rangle = \frac{1}{L_x L_y} \int_0^{L_x} \int_0^{L_y} (\ldots) \, dx \, dy, \quad (4.2) \]

where \( L_x \) and \( L_y \) are typical lengths along the \( x \)- and \( y \)-directions, respectively. Thus, the mean viscous dissipation is given by:

\[ \langle \epsilon \rangle = 2 \left[ \left( \langle \frac{\partial u}{\partial x} \rangle \right)^2 + \left( \langle \frac{\partial v}{\partial y} \rangle \right)^2 + \frac{1}{2} \left( \langle \frac{\partial u}{\partial y} \rangle \right)^2 + \frac{1}{2} \left( \langle \frac{\partial v}{\partial x} \rangle \right)^2 + \langle \frac{\partial u}{\partial y} \frac{\partial v}{\partial x} \rangle \right], \quad (4.3) \]

In general, equation (4.3) can be a function of time. Physically its evolution is related to the entropy generation. More precisely, for an incompressible and isothermal flow, \( \langle \epsilon \rangle \) is related to the mean entropy as [17, page 195, eq. (49.6)]:

\[ \frac{d}{dt} \langle s \rangle = \langle \epsilon \rangle. \quad (4.4) \]

In what follows, we shall apply (4.4) to the array of vortices (2.17).

4.3 Viscous dissipation and entropy in a linear array of vortices

Now, let us compute \( \langle \epsilon \rangle \) for the linear array of vortices (2.14). Since the streamfunction reads:

\[ \psi(x, y, t) = \psi_0(t) \sin k x \sin y + \psi_1(t) \sin y + \psi_2(t) \cos k x \sin 2 y, \quad (4.5) \]

the velocity components are given by:

\[ u = \frac{\partial \psi}{\partial y} = \psi_0(t) \sin k x \cos y + \psi_1(t) \cos y + 2 \psi_2(t) \cos k x \cos 2 y, \]
\[ v = - \frac{\partial \psi}{\partial x} = - k \psi_0(t) \cos k x \sin y + k \psi_2(t) \sin k x \sin 2 y. \]

Thus, the velocity gradients are:

\[ \frac{\partial u}{\partial x} = k \psi_0(t) \cos k x \cos y - 2 k \psi_2(t) \sin k x \cos 2 y, \]
\[ \frac{\partial u}{\partial y} = - \psi_0(t) \sin k x \sin y - \psi_1(t) \sin y - 4 \psi_2(t) \cos k x \sin 2 y, \]
\[ \frac{\partial v}{\partial x} = k^2 \psi_0(t) \sin k x \sin y + k^2 \psi_2(t) \cos k x \sin 2 y, \]
\[ \frac{\partial v}{\partial y} = - k \psi_0(t) \cos k x \cos y + 2 k \psi_2(t) \sin k x \cos 2 y. \]
Since the flow is periodic in space, we apply (4.2) over a period of the domain. Thus, the mean velocity gradients are given by:

\[
\langle \left( \frac{\partial u}{\partial x} \right)^2 \rangle = \frac{k^2}{4} \psi_0^2 + k^2 \psi_2^2, \\
\langle \left( \frac{\partial u}{\partial y} \right)^2 \rangle = \frac{1}{4} \psi_0^2 + \frac{1}{2} \psi_1^2 + 4 \psi_2^2, \\
\langle \left( \frac{\partial v}{\partial x} \right)^2 \rangle = \frac{k^4}{4} \psi_0^2 + k^4 \psi_2^2, \\
\langle \left( \frac{\partial v}{\partial y} \right)^2 \rangle = \frac{k^2}{4} \psi_0^2 + k^2 \psi_2^2, \\
\langle \frac{\partial u}{\partial y} \frac{\partial v}{\partial x} \rangle = -\frac{k^2}{4} \psi_0^2 - k^2 \psi_2^2.
\]

By substituting (4.12)-(4.16) into (4.3) one finds the space averaged viscous dissipation:

\[
\langle \epsilon \rangle = A_0 \psi_0^2 + A_1 \psi_1^2 + A_2 \psi_2^2,
\]

where

\[
A_0 = \frac{1}{4} \left( 1 + 2k^2 + k^4 \right), \\
A_1 = \frac{1}{2}, \\
A_2 = 4 + 2k^2 + \frac{k^4}{4}.
\]

Finally, by combination of (4.17) and (4.4), the space averaged entropy generation rate is:

\[
\frac{d\langle s \rangle}{dt} = A_0 \psi_0^2 + A_1 \psi_1^2 + A_2 \psi_2^2.
\]

### 4.4 Viscous dissipation, entropy, and control

From a control perspective, expression (4.21) satisfies our requirements: (i) it is closely related to mixing, and (ii) it is an explicit function of the modes \(\psi_0\). Moreover, \(\langle \epsilon \rangle = A_0 \psi_0^2 + A_1 \psi_1^2 + A_2 \psi_2^2\) can be seen as the output of the system. In this spirit, the dynamical equations (2.17) supplemented by (4.17) can be summarized as:

\[
\begin{align*}
\dot{\psi}_0 &= C_1 \psi_1 \psi_2 + C_2 \psi_0 + C_3 F, \\
\dot{\psi}_1 &= C_4 \psi_2 \psi_0 + C_5 \psi_1, \\
\dot{\psi}_2 &= C_6 \psi_0 \psi_1 + C_7 \psi_2, \\
\xi &= \langle \epsilon \rangle = A_0 \psi_0^2 + A_1 \psi_1^2 + A_2 \psi_2^2,
\end{align*}
\]

where \(F\) represents the input and \(\xi\) denotes the output. Systems of the form (4.22) are common in control.

\footnote{For details see appendix B.}
4.5 Summary

In this chapter we proposed that the global entropy/viscous dissipation may serve as a bridge between the hydrodynamic and statistical descriptions of mixing. After reviewing the space-averaged fluid mechanical equations, we derived an explicit expression of the entropy generation in a linear array of vortices. Such expression is a direct function of the modes $\psi_\alpha$. Finally, we discussed that the latter property allows us to view the linear array of vortices as a regular input-output system.
5

Input output linearization of a linear array of vortices

5.1 Introduction

In engineering, it is often desirable to have a form of automatic control. A familiar example is provided by a thermostat, which controls heating and cooling until the desired temperature is reached.

Active control also plays an important role in the transport and processing of fluids. Among the many applications are boundary feedback in pipes [3], neural networks in turbulent channel flows [18], and optimal control of Stokes flows [21]. In the context of mixing, a typical example is the stirring of two component adhesives, like epoxy, which should evolve neither too fast nor too slow.

As discussed in chapter 4, mixing is a process closely related to velocity gradients. Since the global viscous dissipation rate \( \langle \epsilon \rangle \) is a (scalar) measure of such gradients, one would expect that mixing could be controlled via \( \langle \epsilon \rangle \). But what is the relation to the entropy of the flow? And how does the mixing time scale depend on \( \langle \epsilon \rangle \)? To answer such questions, this chapter is organized as follows. In section 6.2 we apply input-output linearization to the linear array of vortices (2.17). In particular, section 6.2.2 addresses the equilibria and section 6.2.3 the internal dynamics of the controlled system. Then, we prescribe the entropy \( \langle s \rangle \) of the flow. Section 6.3.1 is devoted to dynamical power laws such that \( \langle s \rangle \sim t^\alpha \). Finally, in section 6.3.2, we define a mixing time scale and study its dependence on \( \langle \epsilon \rangle \).

5.2 Input-output linearization

In the previous chapter we discussed that the dynamical model of the linear array of vortices (2.17) can be seen as an input-output system, with \( F \) as input and the viscous dissipation (4.17) as output. Here, we rewrite system (4.22) in the following form:

\[
\frac{d\psi}{dt} = a(\psi) + b(\psi)F, \\
\xi = h(\psi),
\]  

(5.1)

where
5.2. INPUT-OUTPUT LINEARIZATION

\[
\psi = \begin{bmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \end{bmatrix}, \quad a(\psi) = \begin{bmatrix} C_1 \psi_1 \psi_2 + C_2 \psi_0 \\ C_4 \psi_2 \psi_0 + C_5 \psi_1 \\ C_6 \psi_0 \psi_1 + C_7 \psi_2 \end{bmatrix}, \quad b = \begin{bmatrix} C_3 \\ 0 \\ 0 \end{bmatrix},
\]

and

\[
\xi = h(\psi) = \langle \epsilon \rangle = A_0 \psi_0^2 + A_1 \psi_1^2 + A_2 \psi_2^2.
\] (5.2)

Let us now design a controller via input-output linearization. A short theoretical overview of this method can be found in appendix D.

To begin, we compute the Lie derivatives of (5.2):

\[
L_a h = 2A_0 \psi_0 (C_1 \psi_1 \psi_2 + C_2 \psi_0) + 2A_1 \psi_1 (C_4 \psi_2 \psi_0 + C_5 \psi_1) + 2A_2 \psi_2 (C_6 \psi_0 \psi_1 + C_7 \psi_2),
\] (5.3)

\[
L_b h = 2A_0 C_3 \psi_0.
\] (5.4)

Thus, the output satisfies [see eq. (D.2)]:

\[
\dot{\xi} = 2A_0 \psi_0 (C_1 \psi_1 \psi_2 + C_2 \psi_0) + 2A_1 \psi_1 (C_4 \psi_2 \psi_0 + C_5 \psi_1) + 2A_2 \psi_2 (C_6 \psi_0 \psi_1 + C_7 \psi_2) + 2A_0 C_3 \psi_0 F.
\] (5.5)

Since the first derivative of the output is explicitly dependent on \(F\), the system has relative degree \(\gamma = 1\). To linearize the input-output map, we choose \(F\) as [see eq. (D.6)]:

\[
F = \frac{1}{L_b h(\psi)} [-L_a h(\psi) + \xi] = \frac{1}{2A_0 C_3 \psi_0} [-2A_0 \psi_0 (C_1 \psi_1 \psi_2 + C_2 \psi_0) - 2A_1 \psi_1 (C_4 \psi_2 \psi_0 + C_5 \psi_1) - 2A_2 \psi_2 (C_6 \psi_0 \psi_1 + C_7 \psi_2) + \xi],
\] (5.6)

where \(\xi\) is the new control law.

Expression (5.6) requires \(L_g h(\psi) \neq 0\), which only holds on the domain \(\mathbb{L} = \{\psi \in \mathbb{R}^3 | \psi_0 \neq 0, A_0 \neq 0, C_3 \neq 0\}\). Therefore, the input output linearization is not globally valid. To stay in \(\mathbb{L}\), the trajectories of the controlled system should thus satisfy \(\psi_0 \neq 0\) at all times. As we shall see, this constraint can, at least locally, be satisfied.

The linearized input-output map can be written as a single integrator by introducing \(q\) as [see eq.(D.10)]:

\[
q_0 = \xi^{(0)},
\] (5.7)

which yields

\[
q_0 = \xi, \quad \xi = q_0,
\] (5.8)

where \(\xi\) is the input. The control law for \(\xi\) remains to be chosen, as addressed next.
5.2. CONTROL LAW

5.2.1 Control Law

Since $E_0 = q_0 - r$, with $r(t)$ a desired reference function\(^1\), the error dynamics is governed by:

$$
\dot{E}_0 = \zeta - r^{(1)}.
$$

(5.9)

Choosing $\zeta = -k_0 E_0 + r^{(1)}$ leads to:

$$
\dot{E}_0 = -k_0 E_0.
$$

(5.10)

Equation (5.10) is stable for all $k_0 > 0$. Thus, we choose $F$ as:

$$
F = \frac{1}{2A_0 C_3 \psi_0} [-2A_0 \psi_0 (C_1 \psi_1 \psi_2 + C_2 \psi_0) \\
- 2A_1 \psi_1 (C_4 \psi_2 \psi_0 + C_5 \psi_1) \\
- 2A_2 \psi_2 (C_6 \psi_0 \psi_1 + C_7 \psi_2) - k_0 (A_0 \psi_0^2 + A_1 \psi_1^2 + A_2 \psi_2^2 - r) + r^{(1)}],
$$

(5.11)

with $k_0 > 0$.

Since system (5.1) has relative degree one, the internal dynamics is two dimensional. From a geometric perspective, the uncontrolled system (5.1) evolves in three dimensional (state) space $\mathbb{D} = \mathbb{R}^3$. On the other hand, for constant $r$, system (5.1) under (5.11) is tracked to the surface $T = \{ \psi \in \mathbb{D} | \xi = r \}$. The dynamics on $I = T \cap \mathbb{L}$ is the internal dynamics. Before we investigate $I$ in more detail, let us study the equilibria of the controlled system.

5.2.2 Equilibria of the controlled system

By substitution of control (5.11) into equation (5.1) one finds:

$$
\begin{align*}
\dot{\psi}_0 &= \frac{1}{2A_0 \psi_0} [-2A_1 \psi_1 (C_4 \psi_2 \psi_0 + C_5 \psi_1) - 2A_2 \psi_2 (C_6 \psi_0 \psi_1 + C_7 \psi_2) \\
&\quad + \dot{r} + K (A_0 \psi_0^2 + A_1 \psi_1^2 + A_2 \psi_2^2 - r)] \\
\dot{\psi}_1 &= C_4 \psi_2 \psi_0 + C_5 \psi_1 \\
\dot{\psi}_2 &= C_6 \psi_0 \psi_1 + C_7 \psi_2.
\end{align*}
$$

(5.12)

System (5.12) describes the full dynamics of the controlled system. By construction, the equilibria of (5.12) should satisfy $\xi = r$. Thus, we can expect the equilibria to be dependent on $r$.

To reveal such dependence on $r$, let us consider a constant reference such that $\dot{r} = 0$. Thus, system (5.12) has six equilibria generated by:

$$
H_1 = \sqrt{\frac{r}{A_0}},
$$

(5.13)

$$
H_2 = \sqrt{\frac{C_5 C_7}{C_4 C_6}},
$$

(5.14)

\(^1\)For brevity we drop the explicit time dependence and denote $r(t)$ as $r$.  

---

34
5.2. INPUT-OUTPUT LINEARIZATION

\[ H_3 = \sqrt{\frac{C_7(C_4C_6r - A_0C_5C_7)}{C_6(A_1C_4C_7 + C_5A_2C_6)}}, \quad (5.15) \]

\[ H_4 = \sqrt{\frac{C_5(C_4C_6r - A_0C_5C_7)}{C_4(A_1C_4C_7 + C_5A_2C_6)}}, \quad (5.16) \]

Table 5.1 summarizes the six possible combinations between (5.13)-(5.16). As anticipated, the equilibria are dependent on \( r \). But how about stability? To answer this question, we perform a linear stability analysis\(^2\) of (5.12). Note that the stability is dependent on the gain \( k_0 \). Nevertheless, the results discussed below are valid for all \( k_0 < 0 \).

<table>
<thead>
<tr>
<th>Equilibrium</th>
<th>( \psi_0 )</th>
<th>( \psi_1 )</th>
<th>( \psi_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A^*_+ )</td>
<td>( H_1 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( A^*_- )</td>
<td>( -H_1 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( A_+ )</td>
<td>( H_2 )</td>
<td>( H_3 )</td>
<td>( -H_4 )</td>
</tr>
<tr>
<td>( A^- )</td>
<td>( H_2 )</td>
<td>( -H_3 )</td>
<td>( H_4 )</td>
</tr>
<tr>
<td>( A'_+ )</td>
<td>( -H_2 )</td>
<td>( -H_3 )</td>
<td>( -H_4 )</td>
</tr>
<tr>
<td>( A'_- )</td>
<td>( -H_2 )</td>
<td>( H_3 )</td>
<td>( H_4 )</td>
</tr>
</tbody>
</table>

Table 5.1: Equilibria of the controlled system. The values of \( H_1, ..., H_4 \) are given by eqs. (5.13)-(5.16)

As shown in figure 5.1, the equilibria of the controlled system undergo a bifurcation as function of the reference. For \( r < r_{\text{bif}} = 0.66 \) two stable solutions exist, namely: \( A^*_+ \) and \( A^*_- \). These solutions become unstable for \( r > r_{\text{bif}} \). On the other hand, for \( r > r_{\text{bif}} \) four stable equilibria emerge: \( A_+, A_-, A'_+, \) and \( A'_- \). Thus, the controlled system undergoes a supercritical pitchfork bifurcation.

\(^2\)For details see Appendix E.2
5.2.3 Internal dynamics

The output of system (5.1) defines the surface towards which the system is tracked. Physically, this corresponds to the viscous dissipation (4.21):

\[ \xi = \langle \epsilon \rangle = A_0 \psi_0^2 + A_1 \psi_1^2 + A_2 \psi_2^2. \]  \hspace{1cm} (5.17)

Geometrically, for a constant \( \xi \) equation (5.17) represents an ellipsoid in state space. The size of such ellipsoid is dependent on \( \langle \epsilon \rangle \), and its shape on \( A_0, A_1, \) and \( A_2 \). In the case of the controlled system, \( \xi \rightarrow r \) for \( t \rightarrow \infty \). Thus, if the reference \( r \) is constant, the system will asymptotically converge to an ellipsoid of size \( r \). Here, all equilibria shown in table 5.1 lie on the ellipsoid.

Nevertheless, the input-output linearization only holds in the domain \( \mathbb{L} \). Since \( \mathbb{L} \) excludes the plane \( \psi_0 = 0 \), the ellipsoid is split in two halves. Thus, for a successful control approach, the internal dynamics should not evolve towards the plane \( \psi_0 = 0 \). As figure 5.1 shows, on both sides of the plane \( \psi_0 = 0 \) stable solutions exist. This indicates that, at least locally, the controlled system will asymptotically converge to a stable point on the ellipsoid, and is not driven to the singularity.

To further clarify the statements above, we consider next the corresponding geometric structures in state space.

**Geometric interpretation**

Figures 5.2 and 5.3 show the ellipsoid (5.17) from different viewpoints. The subspace of possible equilibria as functions of \( r \) is depicted by red lines. More specifically, the
5.2. INPUT-OUTPUT LINEARIZATION

thick red lines are the stable equilibria for $r > r_{bif}$ and the thin red lines are the stable equilibria for $r < r_{bif}$. The stable equilibria on the ellipsoid are now simply given by the intersections of the red lines and the ellipsoid's surface.

Figure 5.2: Geometric structures in state space. Ellipsoid: surface for which $\langle \epsilon \rangle = 0.4$. Red lines: stable equilibria for $0 \leq \langle \epsilon \rangle \leq 1$.

Figure 5.3: Geometric structures in state space. Ellipsoid: surface for which $\langle \epsilon \rangle < 0.4$. Red lines: stable equilibria for $0 \leq \langle \epsilon \rangle \leq 1$. (a) Top view. (b) Side view.

Figure 5.4 shows the (normalized) vector field on the ellipsoid. In addition, the purple line represents a typical trajectory of the system. Figure 5.4 confirms that, at least locally, the controller will track the system to a stable point on the ellipsoid. Also, note that the vector field is always pointing away from the plane $\psi_0 = 0$. As a result, trajectories on the ellipsoid will tend to evolve away from the $\psi_0 = 0$ plane and will thus stay in $I$.

Finally, in figure 5.5 the norm of the vector field ($|\vec{\psi}|_2$) is represented by a color map. Due to the singularity in equation (5.11), the vector norm is infinite for $\psi_0 = 0$. Clearly,
5.2. INPUT-OUTPUT LINEARIZATION

Figure 5.4: Geometric structures in state space. Ellipsoid: surface such that $\langle \epsilon \rangle = 2$. Red lines: stable equilibria for $2 \leq \langle \epsilon \rangle \leq 2.5$. Vectors: internal dynamics. Purple line: typical trajectory of the system.

The vector field is not only pointing away from the plane $\psi_0 = 0$, but is also relatively strong in the neighborhood of $\psi_0 = 0$. This observation provides another argument for the statement that trajectories on the ellipsoid will tend to evolve away from the $\psi_0 = 0$ plane.

Figure 5.5: Colormap of the vector norm $|\dot{\psi}|_2$ on the ellipsoid $\langle \epsilon \rangle = 2$. The magnitude increases from blue (0) to red (0.8). The blue band around the singularity $\psi_0 = 0$ is a numerical artifact.
5.3. VISCOUS DISSIPATION, ENTROPY, AND CONTROL

**Speed of the internal dynamics**

As discussed above, the controlled system is readily tracked to the ellipsoid. But once on the ellipsoid, how fast does the system approach equilibrium? To answer this question, we measure the distance $d_q$ and the time $t_q$ to reach equilibrium, both as functions of $\langle \epsilon \rangle$. Figure 5.6a shows the distance $d_q$ to $A'_\epsilon$ as function of time for different $\langle \epsilon \rangle$ values. Figure 5.6b shows the time to reach equilibrium $t_q$ as function of ellipsoid size. Clearly, for increasing $\langle \epsilon \rangle$ the equilibrium is reached faster.

![Figure 5.6: Approach to equilibrium. (a) Distance and (b) time to reach equilibrium as functions of $\langle \epsilon \rangle$. For all simulations the initial condition is $(\psi_0, \psi_1, \psi_2) = (1, 0.1, 0.1)$](image)

5.3 **Viscous dissipation, entropy, and control**

In the previous section, we saw that the prescription of a time independent output (the viscous dissipation rate) corresponds to a time independent structure in state space (an ellipsoid). Physically, this corresponds to a steady stirring of fluid. Now, let us take a step further and study time dependent stirring protocols.

Non-steady stirring is pervasive in everyday life. Classical examples range from stirring in biological systems [25] to industrial processing of alloys. Time decaying stirring, in particular, plays an important role in the study of turbulent flows. A central issue in such scenario is the decay rate, which typically follows power laws [9].

In this section, we focus on stirring protocols such that $\langle \epsilon \rangle \sim t^{-1+\alpha}$, where $0 < \alpha \leq 1$. In particular, based on the statistical relation (4.4), we systematically quantify the mixing dynamics.

5.3.1 **Power law dynamics**

As shown in chapter 4, the viscous dissipation $\langle \epsilon \rangle$ and the entropy $\langle s \rangle$ are statistically coupled via [see eq. (4.4)]:

$$\frac{d\langle s \rangle}{dt} = \langle \epsilon \rangle.$$
Thus, if $\langle \epsilon \rangle$ evolves as a power law, so does the entropy $\langle s \rangle$ of the flow. From this perspective, we shall study stirring protocols such that:

$$\langle s \rangle = \frac{D}{\alpha} (t + t_0)^\alpha,$$

(5.18)

where $D$ is the coefficient, $\alpha$ the scaling exponent ($0 < \alpha \leq 1$), and $t_0$ an off-set ($t_0 = 1$). Prescription (5.18) is completely equivalent to:

$$\langle \epsilon \rangle = D(t + t_0)^{\alpha - 1}.$$

(5.19)

Here, $D$ may be interpreted as the stirring amplitude and $\alpha$ as a decay parameter.

**Role of the coefficient $D$**

To reveal the dependence of the mixing dynamics on $\langle \epsilon \rangle$, we fix $\alpha = 1$ and compute the mixing number (3.4) for increasing values of $D$.

As shown in figure 5.7, increasing $D$ leads to faster mixing. This result seems to support the notion that entropy and mixing are closely related concepts. Furthermore, note that the curves tend to collapse on a minimum mixing number $m \approx 6 \cdot 10^{-3}$. In this ultimate regime, the time series become indistinguishable from each other due to the spatial resolution of the mixing number.

![Figure 5.7: Mixing number for $\langle s \rangle = D(t + t_0) \Leftrightarrow \langle \epsilon \rangle = D$. For increasing $\langle \epsilon \rangle$, mixing is clearly enhanced.](image)

**Role of the exponent $\alpha$**

To reveal the dependence of the dynamics on the scaling exponent $\alpha$, we fix $D = 1$ and compute the mixing number for increasing $\alpha$. Figure 5.8a shows the resulting mixing time series and figure 5.8b the corresponding entropy curves. As expected, the larger the entropy generation in the flow, the better the mixing.

But what is the relation between entropy and mixing? To answer this question, we plot in figure 5.9a the mixing number as function of the entropy. Clearly, for fixed entropy $\langle s \rangle$, mixing is enhanced for decreasing $\alpha$. For instance, consider the curves for
5.3. VISCOUS DISSIPATION, ENTROPY, AND CONTROL

Figure 5.8: Tracking of \( \langle s \rangle \sim t^\alpha \Rightarrow \langle c \rangle \sim t^{\alpha-1} \) for \( \alpha = 0.2, \ldots, 0.8 \). (a) Mixing enhancement (\( \alpha \) increasing from top to bottom). (b) Entropy evolution (\( \alpha \) increasing from bottom to top).

Figure 5.9: Tracking of \( \langle s \rangle \sim t^\alpha \Rightarrow \langle c \rangle \sim t^{\alpha-1} \) for \( \alpha = 0.2, \ldots, 0.8 \). Mixing as function of (a) entropy and (b) control effort.

\( \alpha = 0.2 \) and \( \alpha = 1.0 \). Although the former corresponds to less entropy generation, it leads to lower mixing numbers as compared to the case \( \alpha = 1 \).

A different viewpoint is provided by figure 5.9b, which shows the cumulative control effort \( \Theta \) as function of the mixing number. In terms of energy input, figure 5.9b suggests that most efficient mixing protocol should involve subdiffusive (i.e. \( \alpha < \frac{1}{2} \)) entropy dynamics.

From a physical perspective, if one wants to mix as fast as possible, regardless of energy expenditure, entropy should be generated as fast as possible. On the other hand, if one desires to achieve a certain degree of mixing with the least amount of energy input, regardless of time, subdiffusive (i.e. \( \alpha < 0.5 \)) entropy dynamics would suffice.
5.3.2 Viscous dissipation and the mixing time scale

As shown above, mixing tends to evolve faster for increasing viscous dissipation rates (figure 5.7). But what is the connection between the viscous dissipation rate and the mixing rate? To answer this question, let us examine the explicit relation between $\langle \epsilon \rangle$ and a mixing time scale $t_m$. To quantify such relation, we define $t_m$ as the time at which the average distance between two species has halved. Thus, in terms of the mixing number this corresponds to $\frac{m(t_m)}{m(0)} = 0.25$.

Figure 5.10 shows the relation between $t_m$ and $\langle \epsilon \rangle$. Clearly, the mixing time scale decays as the inverse square root of $\langle \epsilon \rangle$. But what is the physics behind this result? On dimensional grounds we may write:

$$\tau_m = a \sqrt{\frac{\nu}{\langle \epsilon \rangle}}, \quad (5.20)$$

where $a$ is a dimensionless coefficient. This simple analysis supports our numerical result. Moreover, using the definition of the viscous dissipation rate (5.17), equation (5.20) can be rewritten as:

$$\tau_m = \frac{a}{\sqrt{\sum \left( \frac{\partial U_\alpha}{\partial x_\mu} \right)^2}}. \quad (5.21)$$

This result also supports our notion that the mixing dynamics is essentially determined by velocity gradients.

Figure 5.10: Mixing time scale $t_m$ as a function of the global energy dissipation $\langle \epsilon \rangle$. Circles: numerical simulations. Solid line: linear fit $t_m = 3.5\langle \epsilon \rangle^{-0.51}$
5.4 Summary

In this chapter, we controlled mixing of isothermal fluids by prescribing the global entropy $\langle s \rangle$ of the flow. In particular, based on the statistical coupling between the evolution of $\langle s \rangle$ and the global viscous dissipation $\langle \epsilon \rangle$, we studied stirring protocols such that $\langle s \rangle \sim t^\alpha \Leftrightarrow \langle \epsilon \rangle \sim t^{\alpha-1}$, where $0 < \alpha \leq 1$. For the linear array of vortices (2.17), we have shown that: (i) feedback control can be achieved via input-output linearization, (ii) mixing is monotonically enhanced for increasing $\alpha$, and (iii) the mixing time $\tau_m$ scales as $\tau_m \sim \langle \epsilon \rangle^{-1/2}$. 
Conclusions and recommendations

6.1 Conclusions

This thesis focussed on the control of mixing in a linear array of vortices. In particular, we addressed the following questions:

- What are the dynamical properties of the linear array of vortices shown in figure 1.2?
- How to quantify mixing of two fluids?
- How to control mixing using statistical properties of the flow?

It was shown that a linear array of vortices can be described by a system of ordinary differential equations consisting of three modes. Physically, the first mode corresponds to the main flow, the second to a shear flow, and the third to a vortex lattice. We discussed that (i) for increasing forcing the two higher modes start to dominate the flow (e.g. tilting) and (ii) flow fields for positive and negative forcing are symmetric to each other.

We showed that mixing measures usually rely on snapshots of the flow. Such snapshots can be generated by several methods, e.g. forward advection or backward advection. Based on the tradeoff between the quality of the image (i.e. contrast) and computational cost we opted for forward advection. Then, we applied a statistical measure \( m \) (3.4) to quantify the mixing. Such measure is based on the average distance between two fluids. However, the explicit relation between \( m \) and the flow variables is not clear. To overcome that, we proposed a bridge between mixing and velocity gradients (\( \langle \epsilon \rangle \)) via the global entropy \( \langle s \rangle \) of the flow.

To control mixing, we prescribed \( \langle s \rangle \) in a linear array of vortices. Since the vortex model which we considered is two-dimensional and of low-order, \( \langle s \rangle \) can be prescribed via input-output linearization. A constant output merely constrains the system to a two-dimensional surface, and hence internal dynamics are present. However, the results of a linearized stability analysis showed that the internal dynamics is (at least locally) stable. We found that (i) mixing is enhanced for increasing entropy and (ii) the mixing timescale is related to the viscous dissipation as \( t_m \sim \langle \epsilon \rangle^{-1/2} \). Both results are physically reasonable, supporting our idea that entropy may indeed serve as a bridge between the hydrodynamic and statistical descriptions of mixing.
Nevertheless, several questions remain. For instance: is the scaling \( t_m \sim \langle \epsilon \rangle^{-0.5} \) universal? Or does it depend on the geometry/dimensionality of the flow? What about different mixing measures? Can one uncover their explicit relation to the flow variables? And for which class of experimentally realizable flows can the present control approach be applied?

### 6.2 Recommendations

#### Mixing time scale

The most significant result requiring further investigation is the scaling of \( t_m \) as \( t_m \sim \langle \epsilon \rangle^{-0.5} \). Such scaling must be validated for different initial conditions and configurations. Moreover, one could study the effect of changing the definition of the mixing time scale. A different mixing measure can be used, for instance, such as the intensity of segregation. It will also be valuable to investigate the effect of the forcing wavenumber \( k \) (and \( R \)). The results from such simulations will give important indications to the generality of the scaling law.

#### Control

The proposed controller is not globally valid, and hence it is worthwhile to study the region of attraction. Preliminary simulations suggest that such region is a function of the gain \( k_0 \). Since accurate information of all states is necessary for the controller to function it seems difficult to implement in an experimental setup. To resolve such issues, a technique such as particle image velocimetry (PIV) may provide a good starting point. In addition, one could investigate the feasibility and application of alternative nonlinear control strategies.

#### Imaging and quantification

With respect to the imaging one could improve on both image quality (e.g. backward advection) and speed of calculation. A good starting point for the latter issue would be to rewrite the advection code in a c-mex file. Ultimately, it is even possible to port the code to the GPU. With respect to the quantification it is worthwhile to investigate in alternative mixing measures, such as the Shannon Entropy.

#### Flow model

As discussed, the vortex model is only two-dimensional and of low dynamic order. Model complexity can be increased by incorporating more modes, leading to further nonlinear interactions. Naturally, it is not certain that input-output linearization is still a valid control approach for a higher order model. Likewise, different two-dimensional flows (e.g. pipe flow) might not allow the same control approach. For three-dimensional flows the issues are even greater and beyond the scope of this thesis.
To conclude, closer investigation of the scaling law $t_m \sim \langle \epsilon \rangle^{-0.5}$ should receive the highest priority.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon$</td>
<td>Local dimensionless viscous dissipation</td>
</tr>
<tr>
<td>$\langle \epsilon \rangle$</td>
<td>Global dimensionless viscous dissipation</td>
</tr>
<tr>
<td>$\Psi$</td>
<td>Streamfunction</td>
</tr>
<tr>
<td>$\nu$</td>
<td>Kinematic viscosity $[\frac{m^2}{s}]$</td>
</tr>
<tr>
<td>$\psi_\alpha$</td>
<td>Fourier modes in model (2.17)</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Density $[\frac{kg}{m^3}]$</td>
</tr>
<tr>
<td>$E$</td>
<td>Error vector</td>
</tr>
<tr>
<td>$\vec{v}$</td>
<td>Velocity vector</td>
</tr>
<tr>
<td>$\xi$</td>
<td>Output</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>Auxiliary control variable</td>
</tr>
<tr>
<td>$C_\alpha$</td>
<td>Coefficients in model (2.17)</td>
</tr>
<tr>
<td>$e_{x,y}$</td>
<td>Scaling factors in $x$- and $y$-directions</td>
</tr>
<tr>
<td>$F$</td>
<td>Forcing amplitude</td>
</tr>
<tr>
<td>$F_{bif}$</td>
<td>Bifurcation point</td>
</tr>
<tr>
<td>$k$</td>
<td>Forcing wavenumber</td>
</tr>
<tr>
<td>$k_0$</td>
<td>Control gain</td>
</tr>
<tr>
<td>$m$</td>
<td>Mixing number</td>
</tr>
<tr>
<td>$p$</td>
<td>Pressure $[Pa]$</td>
</tr>
<tr>
<td>$R$</td>
<td>Reynolds number</td>
</tr>
<tr>
<td>$r(t)$</td>
<td>Reference trajectory</td>
</tr>
<tr>
<td>$s$</td>
<td>Dimensionless entropy</td>
</tr>
<tr>
<td>$t$</td>
<td>Dimensionless time</td>
</tr>
<tr>
<td>$u$</td>
<td>X-velocity component</td>
</tr>
<tr>
<td>$v$</td>
<td>Y-velocity component</td>
</tr>
<tr>
<td>$Z_{x,y}$</td>
<td>Cell coordinates in $x$- and $y$-direction</td>
</tr>
</tbody>
</table>
Appendices
A.1 Navier-stokes equation

The two-dimensional incompressible Navier-Stokes equation reads:

\[
\begin{align*}
\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla) \mathbf{v} &= -\nabla p + \frac{1}{R} \nabla^2 \mathbf{v} + \frac{1}{R} \mathbf{f}, \\
\nabla \cdot \mathbf{v} &= 0.
\end{align*}
\]

(A.1)  \hspace{1cm} (A.2)

Here, \( R = \frac{L_0 U_0}{\nu} \) denotes the Reynolds number, where \( L_0 \) is a characteristic length, \( U_0 \) a characteristic velocity, and \( \nu \) the kinematic viscosity of the fluid. Note that equation (A.1) is in dimensionless form.

A.2 Vorticity equation

The vorticity is defined as \( \Omega = \nabla \times \mathbf{v} \). For a two dimensional flow, \( \Omega = (0, 0, \omega) \). In particular, the equation for \( \omega \) is obtained by taking the curl of (A.1):

\[
\frac{\partial \omega}{\partial t} + (\mathbf{v} \cdot \nabla) \omega = \frac{1}{R} \nabla^2 \omega + \frac{1}{R} \mathcal{F},
\]

where \( \mathcal{F} \) is the \( z \) component of \( \nabla \times \mathbf{f} \).

(A.3)

A.3 Streamfunction representation

The streamfunction \( \Psi \) is defined as:

\[
\begin{align*}
u &= \frac{\partial \Psi}{\partial y}, \\
v &= -\frac{\partial \Psi}{\partial x}.
\end{align*}
\]

(A.4)  \hspace{1cm} (A.5)

Thus:

\[
\nabla^2 \Psi = -\omega.
\]

(A.6)
A.3. STREAMFUNCTION REPRESENTATION

Substitution of (A.6) in (A.3) gives:

$$\frac{\partial \nabla^{2} \Psi}{\partial t} + (\nu \cdot \nabla)(\nabla^{2} \Psi) = \frac{1}{R} \nabla^{4} \Psi + \frac{1}{R} \mathcal{F}, \quad (A.7)$$

where $\nabla^{4} \Psi = \nabla^{2}(\nabla^{2} \Psi)$. Since

$$ (\nu \cdot \nabla) = u \frac{\partial}{\partial x} + v \frac{\partial}{\partial y} = \frac{\partial \Psi}{\partial y} \frac{\partial}{\partial x} - \frac{\partial \Psi}{\partial x} \frac{\partial}{\partial y} , \quad (A.8)$$

substitution of (A.8) in (A.7) yields:

$$\frac{\partial \nabla^{2} \Psi}{\partial t} + \frac{\partial (\nabla^{2} \Psi, \Psi)}{\partial (x, y)} = \frac{1}{R} \nabla^{4} \psi + \frac{1}{R} \mathcal{F}, \quad (A.9)$$

where

$$\nabla^{2} \Psi = \frac{\partial^{2} \Psi}{\partial x^{2}} + \frac{\partial^{2} \Psi}{\partial y^{2}} , \quad (A.10)$$

and

$$\frac{\partial (a, b)}{\partial (x, y)} = \frac{\partial a}{\partial x} \frac{\partial b}{\partial y} - \frac{\partial a}{\partial y} \frac{\partial b}{\partial x} . \quad (A.11)$$

In particular, $\mathcal{F} = -(k^{2} + 1)^{2} F \sin kx \sin y$ leads to the vortex system proposed by Fukuta and Murakami [13].
Space averaged viscous dissipation in a linear array of vortices

B.1 Space-averaged viscous dissipation

The mean viscous dissipation in a two-dimensional flow is given by:

$$
\langle \varepsilon \rangle = 2 \left[ \langle (\frac{\partial u}{\partial x})^2 \rangle + \langle (\frac{\partial v}{\partial y})^2 \rangle + \frac{1}{2} \langle (\frac{\partial u}{\partial y})^2 \rangle + \frac{1}{2} \langle (\frac{\partial v}{\partial x})^2 \rangle + \langle \frac{\partial u}{\partial y} \frac{\partial v}{\partial x} \rangle \right]. \quad (B.1)
$$

In what follows, we shall compute the space-averaged viscous dissipation in a linear array of vortices.

B.2 Velocity gradients

The linear array of vortices proposed by Fukuta and Murakami [13] is described by the streamfunction:

$$
\psi(x, y, t) = \psi_0(t) \sin kx \sin y + \psi_1(t) \sin y + \psi_2(t) \cos kx \sin 2y \quad (B.2)
$$

Thus, the velocity components are:

$$
u = - \frac{\partial \psi}{\partial x} = - k\psi_0(t) \cos kx \sin y + k\psi_2(t) \sin kx \sin 2y, \quad (B.3)
$$

$$
u = - \frac{\partial \psi}{\partial x} = - k\psi_0(t) \cos kx \sin y + k\psi_2(t) \sin kx \sin 2y, \quad (B.4)
$$

The velocity gradients are given by:

$$
\frac{\partial u}{\partial x} = k\psi_0(t) \cos kx \cos y - 2k\psi_2(t) \sin kx \cos 2y, \quad (B.5)
$$

$$
\frac{\partial u}{\partial y} = - \psi_0(t) \sin kx \sin y - \psi_1(t) \sin y - 4\psi_2(t) \cos kx \sin 2y, \quad (B.6)
$$

$$
\frac{\partial v}{\partial x} = k^2 \psi_0(t) \sin kx \sin y + k^2 \psi_2(t) \cos kx \sin 2y, \quad (B.7)
$$

$$
\frac{\partial v}{\partial y} = - k\psi_0(t) \cos kx \cos y + 2k\psi_2(t) \sin kx \cos 2y. \quad (B.8)
$$
Space average of (B.5)

Let us start with the first term in equation (B.1). Here, \( \frac{\partial u}{\partial x} \) is given by:

\[
\left( \frac{\partial u}{\partial x} \right)^2 = (k \psi_0 \cos kx \cos y)^2 + (2k \psi_2 \sin kx \cos 2y)^2 - 4k^2 \psi_2 \psi_0 \cos kx \cos y \sin kx \cos 2y.
\]  

(B.9)

Now, the space average \( \langle \rangle \) is computed as:

\[
\langle \ldots \rangle = \frac{1}{L_x L_y} \int_0^{L_x} \int_0^{L_y} (\ldots) \, dx \, dy,
\]  

(B.10)

where \( L_x \) and \( L_y \) are given by:

\[
L_x = \frac{2\pi}{k}, \quad L_y = \pi,
\]  

(B.11, B.12)

with \( k \) the forcing wavenumber.

The surface integral evaluated over the first term in (B.9):

\[
\frac{1}{L_x L_y} \int_0^{L_x} \int_0^{L_y} (k \psi_0 \cos kx \cos y)^2 \, dx \, dy = \frac{k^2 \psi_0^2}{L_x L_y} \int_0^{L_x} \cos^2 kx \, dx \int_0^{L_y} \cos^2 y \, dy,
\]  

\[
= \frac{k^2 \psi_0^2}{L_x L_y} \left[ \frac{x}{2} + \frac{1}{4k} \sin 2kx \right]_0^{L_x} \left[ \frac{y}{2} + \frac{1}{4} \sin 2y \right]_0^{L_y},
\]  

\[
= \frac{k^2 \psi_0^2}{L_x L_y} \left[ \frac{L_x}{2} \right] = \frac{k^2 \psi_0^2}{4},
\]

where we have used:

\[
\sin 2kL_x = 0, \\
\sin 2L_y = 0.
\]

In the same spirit, the space average of the second term in (B.9) is:

\[
\frac{1}{L_x L_y} \int_0^{L_x} \int_0^{L_y} (2k \psi_2 \sin kx \cos 2y)^2 \, dx \, dy = \frac{4k^2 \psi_2^2}{L_x L_y} \int_0^{L_x} \sin^2 kx \, dx \int_0^{L_y} \cos^2 2y \, dy,
\]  

\[
= \frac{4k^2 \psi_2^2}{L_x L_y} \left[ \frac{x}{2} - \frac{1}{4k} \sin 2kx \right]_0^{L_x} \left[ \frac{y}{2} + \frac{1}{8} \sin 4y \right]_0^{L_y},
\]  

\[
= \frac{4k^2 \psi_2^2}{L_x L_y} \left[ \frac{L_x}{2} \right] = k^2 \psi_2^2.
\]
Finally, the space average of third term of (B.9) is:

\[
\frac{1}{L_x L_y} \int_0^{L_x} \int_0^{L_y} -4k^2 \psi_2 \psi_0 \cos kx \cos y \sin kx \cos 2y \, dx \, dy
\]

\[
= -4k^2 \psi_2 \psi_0 \int_0^{L_x} \cos kx \sin kx \, dx \int_0^{L_y} \cos y \cos 2y \, dy, \\
= -4k^2 \psi_2 \psi_0 \left[ -\frac{1}{2k} \cos^2 kx \right]_0^{L_x} \left[ \frac{1}{2} \sin y + \frac{1}{6} \sin 3y \right]_0^{L_y}, \\
= -4k^2 \psi_2 \psi_0 \left[ ... \right]_0^{L_x} [0] = 0.
\]

Thus, summation of all three contributions yields:

\[
\langle \left( \frac{\partial u}{\partial x} \right)^2 \rangle = k^2 \psi_0^2 + k^2 \psi_2^2.
\]  

(B.13)

**Space average of (B.6)**

Equation (B.6) squared yields:

\[
\left( \frac{\partial u}{\partial y} \right)^2 = \psi_0^2 \sin^2 kx \sin^2 y + \\
\psi_1^2 \sin^2 2y + \\
16 \psi_2^2 \cos^2 kx \sin^2 2y + \\
2 \psi_0 \psi_1 \sin kx \sin^2 y + \\
8 \psi_0 \psi_2 \sin kx \cos kx \sin y \sin 2y + \\
8 \psi_1 \psi_2 \sin y \sin 2y \cos kx,
\]

which consists of six terms.

Space average of the first term:

\[
\frac{1}{L_x L_y} \int_0^{L_x} \int_0^{L_y} \psi_0^2 \sin^2 kx \sin^2 y \, dx \, dy \\
= \frac{\psi_0^2}{L_x L_y} \int_0^{L_x} \sin^2 kx \, dx \int_0^{L_y} \sin^2 y \, dy, \\
= \frac{\psi_0^2}{L_x L_y} \left[ \frac{x}{2} - \frac{1}{4k} \sin 2kx \right]_0^{L_x} \left[ \frac{y}{2} - \frac{1}{4} \sin 2y \right]_0^{L_y}, \\
= \frac{\psi_0^2}{L_x L_y} \left[ \frac{L_x}{2} \frac{L_y}{2} \right] = \frac{\psi_0^2}{4}.
\]
B.2. VELOCITY GRADIENTS

Space average of the second term:

\[
\frac{1}{L_x L_y} \int_{0}^{L_x} \int_{0}^{L_y} \psi_1^2 \sin^2 y \, dx \, dy = \frac{\psi_1^2}{L_x L_y} \int_{0}^{L_x} dx \int_{0}^{L_y} \sin^2 y \, dy, \\
= \frac{\psi_1^2}{L_x L_y} \left[ \frac{L_y}{2} \right] dx, \\
= \frac{\psi_1^2}{L_x L_y} \left[ \frac{L_x L_y}{2} \right] = \frac{\psi_1^2}{2}.
\]

Space average of the third term:

\[
\frac{1}{L_x L_y} \int_{0}^{L_x} \int_{0}^{L_y} 16 \psi_2^2 \cos^2 kx \sin^2 2y \, dx \, dy \\
= \frac{16 \psi_2^2}{L_x L_y} \int_{0}^{L_x} \cos^2 kx \, dx \int_{0}^{L_y} \sin^2 2y \, dy, \\
= \frac{16 \psi_2^2}{L_x L_y} \left[ \frac{x}{2} + \frac{1}{4k} \sin 2kx \right]_0^{L_x} \left[ \frac{y}{2} - \frac{1}{8} \sin 4y \right]_0^{L_y}, \\
= \frac{16 \psi_2^2}{L_x L_y} \left[ \frac{L_x L_y}{2} \right] = 4 \psi_2^2.
\]

The fourth, fifth, and sixth term all evaluate to zero. Thus:

\[
\left\langle \left( \frac{\partial u}{\partial y} \right)^2 \right\rangle = \frac{\psi_0^2}{4} + \frac{\psi_1^2}{2} + 4 \psi_2^2. 
\] (B.14)

**Space average of (B.7)**

Equation (B.7) squared yields:

\[
\left( \frac{\partial u}{\partial x} \right)^2 = k^4 \psi_0^2 \sin^2 kx \sin^2 y \\
+ k^4 \psi_2^2 \cos^2 kx \sin^2 2y \\
+ 2k^4 \psi_0 \psi_2 \sin kx \cos kx \sin y \sin 2y,
\]

which consists of three terms.

Space average of the first term:

\[
\frac{k^4 \psi_0^2}{L_x L_y} \int_{0}^{L_x} \sin^2 kx, dx \int_{0}^{L_y} \sin^2 y \, dy \\
= \frac{k^4 \psi_0^2}{L_x L_y} \left[ \frac{x}{2} - \frac{1}{4k} \sin 2kx \right]_0^{L_x} \left[ \frac{y}{2} - \frac{1}{4} \sin 2y \right]_0^{L_y}, \\
= \frac{k^4 \psi_0^2}{L_x L_y} \left[ \frac{L_x L_y}{2} \right] = \frac{k^4 \psi_0^2}{4}.
\]
Space average of the second term:

\[
\frac{k^4 \psi_2}{L_x L_y} \int_0^{L_x} \cos^2 kx \, dx \int_0^{L_y} \sin^2 2y \, dy = \frac{k^4 \psi_2}{L_x L_y} \left[ \frac{x}{2} + \frac{1}{4k} \sin 2kx \right]_0^{L_x} \left[ \frac{y}{2} - \frac{1}{8} \sin 4y \right]_0^{L_y},
\]

\[
= \frac{k^4 \psi_2}{L_x L_y} \left[ \frac{L_x}{2} \frac{L_y}{2} \right] = \frac{k^4 \psi_2^2}{4}.
\]

The space average of the third term is zero. Thus:

\[
\left\langle \left( \frac{\partial v}{\partial x} \right)^2 \right\rangle = \frac{k^4}{4} (\psi_0^2 + \psi_2^2). \tag{B.15}
\]

**Space average of (B.8)**

Equation (B.8) squared yields:

\[
\left( \frac{\partial v}{\partial y} \right)^2 = k^2 \psi_0^2 \cos^2 kx \cos^2 y
\]

\[
+ 4k^2 \psi_2^2 \sin^2 kx \cos^2 2y
\]

\[
+ 4k^2 \psi_0 \psi_2 \cos kx \sin kx \cos y \cos 2y,
\]

which consists of three terms.

Space average of the first term:

\[
\frac{k^2 \psi_0^2}{L_x L_y} \int_0^{L_x} \cos^2 kx \, dx \int_0^{L_y} \cos^2 y \, dy
\]

\[
= \frac{k^2 \psi_0^2}{L_x L_y} \left[ \frac{x}{2} + \frac{1}{4k} \sin 2kx \right]_0^{L_x} \left[ \frac{y}{2} + \frac{1}{4} \sin 2y \right]_0^{L_y},
\]

\[
= \frac{k^2 \psi_0^2}{L_x L_y} \left[ \frac{L_x}{2} \frac{L_y}{2} \right] = \frac{k^2 \psi_0^2}{4}.
\]

Space average of the second term:

\[
\frac{4k^2 \psi_2^2}{L_x L_y} \int_0^{L_x} \sin^2 kx \, dx \int_0^{L_y} \cos^2 2y \, dy
\]

\[
= \frac{4k^2 \psi_2^2}{L_x L_y} \left[ \frac{x}{2} - \frac{1}{4k} \sin 2kx \right]_0^{L_x} \left[ \frac{y}{2} - \frac{1}{8} \sin 4y \right]_0^{L_y},
\]

\[
= \frac{4k^2 \psi_2^2}{L_x L_y} \left[ \frac{L_x}{2} \frac{L_y}{2} \right] = k^2 \psi_2^2.
\]

Again, the third term evaluates to zero. Thus:

\[
\left\langle \left( \frac{\partial v}{\partial y} \right)^2 \right\rangle = k^2 \psi_0^2 + k^2 \psi_2^2. \tag{B.16}
\]
Space average of product (B.6) and (B.7)

The product of (B.6) and (B.7) is:

\[
\left(\frac{\partial u}{\partial y} \frac{\partial v}{\partial x}\right) = -\psi_0 k^2 \sin^2 kx \sin^2 y \\
- k^2 \psi_0 \psi_2 \sin kx \sin y \cos kx \sin 2y \\
- \psi_1 \psi_0 k^2 \sin^2 y \sin kx \\
- k^2 \psi_1 \psi_2 \cos kx \sin 2y \sin y \\
- 4k^2 \psi_2 \psi_0 \cos kx \sin 2y \sin kx \sin y \\
- 4\psi_2^2 k^2 \cos^2 kx \sin^2 2y,
\]

which consists of six terms. The space average of the second, third, fourth, and fifth term of evaluate to zero, leaving:

\[
\frac{-k^2 \psi_0^2}{L_x L_y} \int_0^{L_x} \int_0^{L_y} \sin^2 kx \, dx \, \sin^2 y \, dy \\
= \frac{-k^2 \psi_0^2}{L_x L_y} \left[ \frac{x}{2} - \frac{1}{4k} \sin 2kx \right]_0^{L_x} \left[ \frac{y}{2} - \frac{1}{4} \sin 2y \right]_0^{L_y} ,
\]

\[
= \frac{-k^2 \psi_0^2}{L_x L_y} \left[ \frac{L_x}{2} \frac{L_y}{2} \right] = -\frac{k^2 \psi_0^2}{4} ,
\]

and

\[
\frac{-4k^2 \psi_2^2}{L_x L_y} \int_0^{L_x} \int_0^{L_y} \cos^2 kx \, dx \, \sin^2 2y \, dy \\
= \frac{-4k^2 \psi_2^2}{L_x L_y} \left[ \frac{x}{2} + \frac{1}{4k} \sin 2kx \right]_0^{L_x} \left[ \frac{y}{2} - \frac{1}{8} \sin 4y \right]_0^{L_y} ,
\]

\[
= \frac{-4k^2 \psi_2^2}{L_x L_y} \left[ \frac{L_x}{2} \frac{L_y}{2} \right] = -k^2 \psi_2^2 .
\]

Thus:

\[
\left\langle \left(\frac{\partial u}{\partial y} \frac{\partial v}{\partial x}\right) \right\rangle = k^2 \left( -\frac{\psi_0^2}{4} - \psi_2^2 \right) . \tag{B.17}
\]

B.3 Summary

Finally, substitution of (B.13), (B.14), (B.15), (B.16), and (B.17) in (B.1) gives:

\[
\langle \epsilon \rangle = A_0 \psi_0^2 + A_1 \psi_1^2 + A_2 \psi_2^2 , \tag{B.18}
\]
where

\[ A_0 = \frac{1}{4} \left( 1 + 2k^2 + k^4 \right), \quad (B.19) \]

\[ A_1 = \frac{1}{2}, \quad (B.20) \]

\[ A_2 = 4 + 2k^2 + \frac{k^4}{4}, \quad (B.21) \]
Computation of the mixing number on graphics card.

In recent years, the graphical processing unit (GPU) has developed into a highly parallel device. Since such parallel power is also convenient for applications other than graphical rendering, graphics card manufacturer NVIDIA has created a software development kit named CUDA (Compute Unified Device Architecture). Such kit enables the development of algorithms for a general purpose (GPGPU) via a C-library.

Because the computation of the mixing number is time consuming, we developed an algorithm to compute $m$ on the GPU. Before we describe the details of our implementation, let us shortly review the GPU architecture and its close connection to the programming model.

C.1 GPU architecture and CUDA

C.1.1 Architecture

As depicted by figure C.1a, CUDA exposes the hardware as a set of SIMD\(^1\) multiprocessors. Each multiprocessor consists of a number of processors, able to access memory of the following types [1];

- One set of 32 bit registers per processor to store local variables.

- A parallel data cache, or shared memory, that is shared between all processor cores of one multiprocessor. This shared memory is currently fixed at 16 KB.

- A read only constant cache that is shared by all multi-processors. This cache speeds up reads from constant memory space, a read-only region of device memory.

- A read-only texture cache that is shared by all multi-processors. This cache speeds up reads from the texture memory space, also a read-only region of device memory.

\(^1\)Nvidia officially defines it as single instruction multiple thread (SIMT) [1], since the behavior of a single thread can be controlled. Therefore SIMT can be seen as a more advanced level of SIMD.
In addition, there is the local and global memory spaces, which are read/write spaces of the device memory and are not cached. Thus they are not really on-chip memory.

All data transfers to and from the shared memory are managed by the user. This is unlike a CPU where data from the main memory is automatically cached.

Because the access to global memory is a factor hundred slower than to shared memory, 4 cycles compared to 400-600 cycles, the key to good performance is efficient use of the shared memory [1]. For the application at hand here, the device, shared and constant memory are used.

![Diagram of GPU architecture](image)

**Figure C.1:** The GPU hardware model and a typical execution configuration [1]

### C.1.2 CUDA programming model

A CUDA program is made up of a (i) host function and (ii) one or more kernel functions. The host function is the ‘base’ code which typically runs on the CPU. Such host function initializes the GPU and ‘sends’ the kernel functions to the multi-processors. Then, the kernel functions define the operations to be carried out at each input data element.

As clarified in figure C.1b the kernel(s) are executed as follows. A single kernel function is executed on the device (GPU) N times in parallel by N different ‘threads’. The threads (typically > 1000) are grouped in blocks on a grid. A block can be one- two- or three dimensional (e.g. a thread in a block can have a x,y, and z coordinate) while a grid can be either one- or two-dimensional. Each thread is coupled to an identifier (or coordinate) which denotes to which location in the block and grid it belongs. Ultimately, the host function determines the exact configuration of grid and blocks (the ‘execution configuration’). This model allows a user to conveniently map the data structure to the thread structure. However, there are some limitations to the execution configuration.
The most important constraint is that the number of threads per block cannot exceed 512.

One thread block is guaranteed to execute on one single multiprocessor. Thus all the threads in a block have access to the shared memory of the block (e.g. multiprocessor). In the case all threads in a block need to be synchronized, a synchronization function is available. For instance, such function can be used if one has to share data in the block through shared memory. In this case, one has to assure that all threads are finished with their tasks on the shared data.

Because the number of threads in a block are always greater than the number of processors per multiprocessor, the hardware contains a scheduling scheme. In particular, this scheduler allows the hardware to hide memory latencies. This is accomplished by letting some threads calculate while others wait for data. In this way, the utilization of the processors is optimized.

C.2 The algorithm

C.2.1 General goal

From a programming perspective, the definition of the mixing number (3.4) is not very convenient. Thus, let us rewrite in the following way.

Consider a binary grid of \( N = N_b + N_w \) cells, where \( N_b \) and \( N_w \) are the number of white and black cells. Now let the set of black cells be called \( S_b \), indexed by \( n_i \) with \( i = 1..N_b \), and the set of white cells be called \( S_w \), indexed by \( n_j \) with \( j = 1..N_w \). Next, the Cartesian distance between an arbitrary white and black cell will be denoted by \( d(n_i, n_j) \). We compute the minimum distance \( \Delta(n_{i,j}, S_{w,b}) \) as \( \Delta(n_i, S_w) \equiv \min_i \{d(n_i, s_j)\} \) and \( \Delta(n_j, S_b) \equiv \min_j \{d(n_j, s_i)\} \), where \( s_j \in S_w \) and \( s_i \in S_b \). Thus, \( \Delta(n_{i,j}, S_{w,b}) \) denotes the distance from a black cell \( n_i \) to the nearest white cell, or visa versa. The mixing number can now be defined as:

\[
m = \frac{\sum_{i=1}^{N_b} \Delta^2(n_i, S_w)}{N_b} + \frac{\sum_{j=1}^{N_w} \Delta^2(n_j, S_b)}{N_w}.
\] (C.1)

Now, the goal of the algorithm is to calculate \( m \). In principle this consists of the following steps:

1. Find the vector of coordinates of all \( N_b \) black points which will be called the \textit{blackSet} \((= S_b)\).
2. Find the vector of coordinates of all \( N_w \) white points which will be called the \textit{whiteSet} \((= S_w)\).
3. Calculate the \( N_b \times N_w \) distances \(^2\) for every point in the \textit{blackSet} to every point in the \textit{whiteSet}.
4. Find the \( N_b + N_w = N \) minimum distances \(^2\) of every single point to the point of the opposite color by sorting out all distances.
5. Average the distances to get $m(t)$.

Matlab is fast enough to handle the first, the second and the last step but not the intermediate steps. Thus, the algorithm is designed to calculate and sort out the distances. As an added advantage, this approach enables us to calculate higher order statistics in Matlab.

The input of the algorithm consists of two vectors, $blackSet$ of size $(N_b, 2)$ and $whiteSet$ of size $(N_w, 2)$, in which the $x$- and $y$-coordinates of all pixels are stored.

The output is consists of a vector of minimum distances with length $N$.

### C.2.2 GPU implementation

Various possibilities exist to map this problem to the CUDA programming model. Each method will have its own advantages/disadvantages. Since the shared memory plays such a crucial role in performance, this implementation focusses on the efficient use of such shared memory.

Roughly speaking, any data elements which are to be used more than once, benefit from residing in shared memory. Therefore, the challenge is to arrange the data in a way such that shared memory usage is optimized. In addition, the shared memory can be read/written in parallel by all threads in a block. Thus, using shared memory is also a very efficient way to limit the required memory bandwidth.

Figure C.2: Schematic representation of the first part of the algorithm showing only one of four thread blocks.

The algorithm consists of two parts: calculation of distances (and intermediate minima), and sorting out the absolute minima.

#### Part 1: calculation of distances and intermediate minima

The first part is shown schematically in figure C.2.

In figure C.2, the labeled gray boxes denote the individual memory spaces: device memory and shared memory. The black and white points in the $blackSet$ and $whiteSet$
C.2. THE ALGORITHM

are designated by \( P_{i}^{b} \) and \( P_{i}^{w} \). Furthermore, the yellow, green and orange colors represent the structural blocks of the algorithm.

The \( blackSet \) is divided in a number of tiles (the green boxes) of \( blackTileSize \) points length, whereas the \( whiteSet \) is divided in a number of tiles of \( whiteTileSize \) length. Let us assume that \( blackTileSize = 2 \) and \( whiteTileSize = 3 \). Also, assume that the length of the \( white- \) and \( blackSet \) are always an integer multiple of the respective tile sizes. Now, let each thread block be viewed as a pair of a white and a black tile. The block size is defined to be equal to the \( blackTileSize \).

The execution configuration is made of a two dimensional grid of thread blocks, each of which has a one dimensional grid of threads. The x-location of the threadblock determines to which black tile it belongs, while the y-location determines to which white tile it belongs. The grid dimensions are now simply determined as \((x_{\text{length}}, y_{\text{length}}) = (N_{b}/blackTileSize, N_{w}/whiteTileSize)\).

The calculation can now be carried out as follows: First, all threads in the block will copy a tile of white points from device to shared memory. Here, all threads will copy the data in parallel, since each thread copies one element of the \( whiteTile \) to the shared memory. Next, each thread is coupled to a unique black point of the relevant \( blackTile \). Then, every single thread in a block will calculate the distances from the respective black point to all points in the white tile. Thus, all white points in the \( whiteTile \) (now residing in shared memory) are accessed \( blackTileSize \) times. This process yields \( whiteTileSize \) distances per thread in the block. Now, each thread writes the minimum of these \( whiteTileSize \) distances the device memory. The result from each block is thus a vector of minima (of length \( blackTileSize \)). In summary, this procedure leads to the creation of a total of \( N_{b} \cdot N_{w}/\text{whiteTileSize} \) minima.

Because not only the minimum distance from a black point to all white points are needed, but also the minimum distance from all white point to all black points, the first part is executed twice. For the second execution the 'inputs' \( blackSet \) and \( whiteSet \) are exchanged. Naturally, every distance is calculated twice, but this is probably faster than other options (like synchronizing all threads in between or store all intermediate distances).

Since the \( i-th \) thread reads the \( i-th \) memory address, as shown in figure C.4, the access is likely to be coalesced. This means that all memory calls are united to a single call, thus saving bandwidth. Another advantage of this approach is that a very efficient broadcasting mechanism is used when all threads access the same element in shared memory at the same time. Such a call is also translated to only one memory call. Broadcasting will (probably) happen when all threads in a block start to 'walk' over all the \( whiteTileSize \) points.

Part 2: calculation of absolute minima

After the first execution of part 1, every black point has \( N_{w}/\text{whiteTileSize} \) minima. After the second execution of the first part one is left with, \((N_{b}\cdot N_{w}/\text{whiteTileSize} + N_{w}\cdot N_{b}/\text{whiteTileSize}) \) minima, all stored in device memory. To get the absolute minimum for every point, these minima still have to be sorted out. This is the task of the second part of the algorithm.
C.2. THE ALGORITHM

The second part of the algorithm is schematically depicted in figure C.3. Here, every thread calculates the absolute minimum from all intermediate minima, coupled to a single point, and writes the result to device memory. In this case, the shared memory is not used, since every data element is only accessed once.

The execution configuration now consists of a \((x_{\text{length}}, y_{\text{length}}) = (N_w, b/\text{blackTileSize}, 2)\) grid, where \(N_w\) or \(N_b\) is taken depending on which number is greater. Contrary to figure C.2, in figure C.3 all thread blocks are shown. Observe that although the \(1,2\) block is created, it is inactive. A block can be inactive by including an if statement in the code which, depending on the \(x\)– or \(y\)– location in the grid, is evaluated to 0 and skips all code in the kernel.

The algorithm is completed by copying all data back to the host memory, where it is accessible for Matlab.

C.2.3 General remarks

Until now, it was assumed that \(N_b\) and \(N_w\) are integer multiples of the \(\text{whiteTileSize}\) and \(\text{blackTileSize}\). In reality this is a rare event. Therefore, the code itself is somewhat more complicated. If \(N_w\) is not an integer multiple of the \(\text{whiteTileSize}\), the code handles the last white tile as a special case.

If \(N_b\) is not an integer multiple of \(\text{blackTileSize}\), an extra execution configuration is initiated. This configuration uses an adjusted kernel, and a grid size of \(1, N_w/\text{whiteTileSize}\). The \(\text{blackTileSize}\) of these blocks is adjusted to fit the length of the last black tile.

Furthermore, the copy to shared memory of a white tile cannot be entirely parallel since the \(\text{whiteTileSize}\) is usually greater than the \(\text{blackTileSize}\). Thus the number of threads available for a parallel copy is smaller than the needed amount of white points. A simple loop solves this problem to a small number of sequential parallel copies.

In addition, a number of improvements can still be made, and thus this implementation may not necessary be the fastest possible. Furthermore, we stress that all calculations are done in single precision.
Figure C.4: Left: coalesced float memory access, resulting in a single memory transaction. Right: coalesced float memory access (divergent warp), resulting in a single memory transaction. [1]
C.2. THE ALGORITHM

C.2.4 Code listings

A selection of the most important parts of the code are given below. The host function which initiates all kernels is omitted.

First part of algorithm

```c
__global__ void mixNumberTest(float *xB, float *yB, float *xW, float *yW,
float *minDistance, int BLOCK_SIZE, int Nw, int NwhiteTiles, float exS, float eyS)
{
    const int whiteTileSize=VALUE; // 1536;
    // Floats for the x, y-distances
    float xD, yD, d;
    float min;
    min=200000;
    // Block and thread id 's
    int bx=blockIdx.x;
    int by=blockIdx.y;
    int tx(threadIdx.x);

    //Allocate shared memory
    __shared__ float whitePoints[whiteTileSize][2];

    //Test if it is the last tile in the set, which has less the whiteTileSizePoints,
    //and thus is handled differently
    if(by<(NwhiteTiles-1)) {

        // Parallel assign the values to shared memory,
        // each block does only up till whiteTileSize points
        // of the whiteset, by is the block counter in the whiteset.
        // Loop over all data to be put in shared memory, since the block_size
        // is smaller then the amount of shared data between the
        // threads (whiteTileSize).
        // Both whiteTileSize and BLOCK_SIZE need to be powers of 2.
        for(int i=0;i<(whiteTileSize/BLOCK_SIZE);i++)
            {
                whitePoints[tx+i*BLOCK_SIZE][0]=xW[by*whiteTileSize+tx+i*BLOCK_SIZE];
                whitePoints[tx+i*BLOCK_SIZE][1]=yW[by*whiteTileSize+tx+i*BLOCK_SIZE];
            }

        //Synchronize all threads to make sure all data is copied.
        __syncthreads();

        // Now every thread acces 1 point of the blackset, from global memory and
        // every point of the whiteset, divided in partitions of
        // whiteTileSize in the shared memory. The min returned is only the min
        // of the whiteTileSize part of the whiteset,
        // so in the end it will return whiteset/whiteTileSize minima,
        // of which the absolut min is chosen.
        for(int i=0;i<whiteTileSize;i++)
```
C.2. THE ALGORITHM

```c
{ 
    xD = xB[ bx*BLOCK_SIZE+tx ] - whitePoints[ i ][0];
    yD= yB[ bx*BLOCK_SIZE+tx ] - whitePoints[ i ][1];
    d=xD*xD*exS+yD*yD*eyS;
    if (d<min) { min=d; }
}

// Create minima vector, every black point creates Nw/whiteTileSize minima
// Minima are ordered like [(Bx0 By0 Tx0), (Bx0 By1 Tx0)...(Bx0 By0 Tx1)]
minDistance[( bx+BLOCK_SIZE+tx )*NwhiteTiles+by]=min;
```

```c
else{ 
    //Omitted here. The part of the code to handle the last white tile goes here.
}
```
Second part of algorithm

The kernel belonging to the second part of the algorithm, this kernel sorts out the minima after the first part is completed;

```c
__global__ void mixNumberMin(float *minDistance_bw, float *minDistance_wb, float *minD, int BLOCK_SIZE, int Nb, int Nw, int NwhiteTiles, int NblackTiles) {
    // Block and thread id's
    int by=blockIdx.y;
    int tx=threadIdx.x;
    int bx=blockIdx.x;
    float min, d;
    min=200000;
    if (by==0) {
        if ( (bx*BLOCK_SIZE+tx)<Nb) {
            // Every thread computes the minima of one black point by
            // comparing the NwhiteTiles intermediate minima
            for(int i=0; i<NwhiteTiles; i++) {
                d=minDistance_bw[(bx*BLOCK_SIZE*NwhiteTiles)+tx*NwhiteTiles+i];
                if (d<min) {min=d;}
            }
            minD[bx*BLOCK_SIZE+tx]=min;
        }
    }
    else {
        if ( (bx*BLOCK_SIZE+tx)<Nw) {
            // Every thread computes the minima of one black point
            // by comparing the NwhiteTiles intermediate minima
            for(int i=0; i<NblackTiles; i++) {
                d=minDistance_wb[(bx*BLOCK_SIZE*NblackTiles)+tx*NblackTiles+i];
                if (d<min) {min=d;}
            }
            minD[Nb+bx*BLOCK_SIZE+tx]=min;
        }
    }
}
```
C.2.5 Benchmarks

In order to compare the performance of the GPU to the CPU implementation some benchmarks were done.

The hardware consisted of:

- Lenovo Thinkpad T61P laptop
- Intel Core 2 Duo T9300, 2.5 GHz CPU
- 4 GB RAM
- nVidia 570M (G84M), theoretical 91.2 GFLOPS, 256 MB DDR3 video memory
- OS: 64 bit Ubuntu 8.04 (Linux)

The CPU code consisted of a C-mex file running on a single core of the CPU, in double precision, without vector optimizations (MMX etc). On the other hand, the GPU-code consisted of a C-mex file utilizing the CUDA library (see above). Clearly, both codes could still be optimized.

The input data consisted of ten mixing snapshots at three resolutions of a $\psi_{0,1,2}$ flow. A typical example is depicted in figure C.5. As figure C.6 shows, the GPU implementation increasingly outperforms the CPU code. Since the calculation of mixing curves evolves evaluation of hundreds of snapshots, a significant speed up is achieved (e.g. 6 min vs 40 min). Also, observe that the hardware is relatively modest, and current desktop GPU’s are much more powerfull.

Since the CPU code is more accurate (double precision vs single), we also performed a rough error analysis. Figure C.7 shows the average, minimum and maximum errors between the two implementations. Clearly, such errors ($\approx 2 \times 10^{-8}$) are not significant for the application.

![Figure C.5: Typical mixing snapshot.](image)
Figure C.6: Calculation time of mixing number for ten images. Blue: GPU. Red: CPU

Figure C.7: Average, minimum and maximum error between the CPU (double precision) and GPU (single precision) algorithms.
D

Input-output linearization theory

D.1 Problem statement

Consider a dynamical system of the form:

\[
\begin{align*}
\dot{\psi} &= a(\psi) + b(\psi) F, \\
\dot{\xi} &= h(\psi),
\end{align*}
\]  

where \(\psi\) is an \(n\)-dimensional state vector, \(F\) a scalar input, and \(\xi\) a scalar output. The corresponding block diagram is shown in figure D.1. Suppose we would like to prescribe the output \(\xi\). In particular, let us design a control law for \(F\) such that \(\xi\) asymptotically tracks a reference \(r\). In this context, a basic control approach is input-output linearization.

![Figure D.1: Uncontrolled system (D.1). (1) Evolution of the state \(\psi\) as function of the input \(F\). (2) Output \(\xi\) as function of \(\psi\).](image)

D.2 Input Output Linearization

The basic idea behind input-output linearization involves two steps. First, one establishes the explicit relation between \(F\) and \(\xi\), also known as the input-output map. Second, \(F\) is defined such that all nonlinear terms in such map are cancelled. The actual stabilization and tracking of the output can then be achieved via linear control techniques.

More precisely, the first step consists in differentiating \(\xi\) until its \(\gamma\)-derivative \(\xi^{(\gamma)} = \frac{d\xi}{d\gamma}\) shows an explicit dependence on \(F\). In this spirit, we compute:

\[
\dot{\xi} = \nabla h \dot{\psi} = \nabla h (a + bF) = L_a h + L_b h F;
\]
where $\nabla$ is the transpose of the $n$-dimensional gradient operator:

$$
\nabla = \begin{bmatrix}
\frac{\partial}{\partial \psi_0} \\
\vdots \\
\frac{\partial}{\partial \psi_{n-1}}
\end{bmatrix},
$$

(D.3)

$L_a h$ is the Lie derivative of $h$ along $a$:

$$
L_a h = \nabla h_a,
$$

and $L_b h$ is the Lie derivative of $h$ along $b$:

$$
L_b h = \nabla h_b.
$$

The Lie derivatives represent how the output changes along the trajectories of the system. To simplify expressions involving iterated Lie derivatives, let:

$$
L_b L_a h = \nabla L_a h b
$$

$$
L_a^i h = L_a L_a^{i-1} h = \nabla L_a^{i-1} h a
$$

$$
L_a^0 h = h,
$$

Thus, if $L_b h = 0$, then $\dot{\xi} = L_a h$, and the first derivative of the output is independent of $F$. On the basis of the notation introduced above, we may now proceed more formally. In particular, let us introduce the following definitions.

**Definition D.2.1** (State space). The state space of an $n$-dimensional dynamical system is defined as:

$$
D = \{ \psi \in \mathbb{R}^n \}
$$

(D.4)

**Definition D.2.2** (System of relative degree $\gamma$). A dynamical system of the form (D.1) has relative degree $\gamma$ in a neighborhood $L \subset D$ if

1. $L_b L_a^{i-1} h = 0$, for $i = 1, 2, ..., \gamma - 1$;

2. $L_b L_a^{\gamma-1} h \neq 0$,

for all $\psi \in L$. □

Thus, for a system with relative degree $\gamma$, the functions $\xi, \xi^{(1)}, ..., \xi^{(\gamma-1)}$ do not depend explicitly on $F$. Nevertheless, $\xi^{(\gamma)}$ is such that:

$$
\xi^{(\gamma)} = L_a^\gamma h + L_b L_a^{\gamma-1} F,
$$

(D.5)

where the coefficient of $F$ is nonzero. Any nonlinearities in the input-output map appear in the Lie derivatives of equation (D.5).

Since $F$ is a control variable, one may choose the state-feedback $F$ as:

$$
F = \frac{1}{L_b L_a^{\gamma-1} h} [-L_a^\gamma h + \xi],
$$

(D.6)
where \( \zeta \) is a new control variable, which is valid in the domain:
\[
\mathcal{L} = \{ \psi \in \mathcal{D} | L_b L_a^{-1} h \neq 0 \}. \tag{D.7}
\]
Now, substitution of (D.6) into (D.5) leads to the (linear) input-output map:
\[
\xi^{(\gamma)} = \zeta. \tag{D.8}
\]
System (D.8) consists of \( \gamma \) first-order differential equations. To show this, let us introduce a new state vector \( q \) such that:
\[
q = \begin{bmatrix}
q_0 \\
\vdots \\
q_{\gamma-1}
\end{bmatrix} = \begin{bmatrix}
\xi^{(0)} \\
\vdots \\
\xi^{(\gamma-1)}
\end{bmatrix} \tag{D.9}
\]
In terms of (D.9) one can define a new input-output system as:
\[
\dot{q} = A q + B \zeta, \quad \xi = C q \tag{D.10}
\]
where \( \zeta \) is the input, \( \xi \) the output, and \( A, B \) and \( C \) are:
\[
A = \begin{bmatrix}
0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 1 & \ldots & 0 \\
\vdots & & & & \vdots \\
0 & \ldots & \ldots & 0 & 1
\end{bmatrix}_{[\gamma \times \gamma]}, \quad B = \begin{bmatrix}
0 \\
0 \\
\vdots \\
0 \\
1
\end{bmatrix}, \quad C = \begin{bmatrix}
1 & 0 & \ldots & 0 & 0
\end{bmatrix} \tag{D.11}
\]
To summarize, the essence of the input-output linearization is the derivation of the dynamical system (D.10). This is schematically shown in figure D.2

Figure D.2: Uncontrolled input-output linearized system. (1) Evolution of the state \( \psi \) as function of the input \( F \). (2) Output \( \xi \). (3) Linearizing state feedback \( F \). (4) Linear system (D.10), with input \( \zeta \) and output \( \xi \).

### D.3 Control law

Given a reference \( r \), the new control variable \( \zeta \) can be chosen in such a way that the difference \( \xi - r \) asymptotically converges to zero. To accomplish this, assume that the
derivatives \( r^{(1)}, \ldots, r^{(\gamma)} \) are bounded functions of time, with \( r^{(\gamma)} \) piecewise continuous. Then, consider the error vector:

\[
E = \begin{bmatrix}
q_0 - r \\
\vdots \\
q_{\gamma-1} - r^{(\gamma-1)}
\end{bmatrix}.
\] (D.12)

Thus, in terms of (D.10), the error dynamics is governed by:

\[
\dot{E} = A E + B (\zeta - r^{(\gamma)}). \tag{D.13}
\]

To stabilize equation (D.13), we may take \( \zeta = -K E + r^{(\gamma)} \), where \( K = [k_0, \ldots, k_{\gamma-1}] \). Thus:

\[
\dot{E} = (A - BK) E. \tag{D.14}
\]

System (D.14) is stable for any choice of \( K \) such that all eigenvalues of \( (A - BK) \) have a negative real part. Figure D.3 depicts this last design step in the overall control scheme.

To summarize, the controlled system is given by:

\[
\dot{\psi} = a(\psi) + b(\psi) \left( \frac{1}{L_b L_a^{\gamma-1}} h(\psi) \left[ -\frac{L_a}{L_b} \dot{h}(\psi) - K E + r^{(\gamma)} \right] \right), \tag{D.15}
\]

\[
\xi = h(\psi).
\]

In the sense of the input-output map, the control problem would thus be solved. However, careful examination of system (D.1) and figure D.3 reveals subtleties, since the states \( \psi \) may have become partly unobservable from the output \( \xi \). Thus, the states cannot always be fully reconstructed from \( \xi \). In particular, a constant output merely constrains the system states to a certain subspace of \( \mathcal{D} \). In this context, let us introduce the following definitions:

**Definition D.3.1 (Target space).** For a given reference \( r \), the target space \( \mathcal{T} \) of the dynamical system (D.15) is defined as:

\[
\mathcal{T} = \{ \psi \in \mathcal{D} | \xi = r \} \tag{\text{□}}
\]

**Definition D.3.2 (Internal space).** The internal space \( \mathcal{I} \) of system (D.15) is given by:

\[
\mathcal{I} = \mathcal{T} \cap \mathcal{L} \tag{\text{□}}
\]

Thus, the controlled system may reach \( \xi = r \) provided that \( \mathcal{I} \neq \emptyset \). Furthermore, if \( \mathcal{I} \) is a non-connected set, not every part of \( \mathcal{T} \) is reachable from every initial condition.

\[\text{\scriptsize \footnote{Depending on the relative degree \( \gamma \) of the system. For \( \gamma = n \) all the states can be completely reconstructed.}}\]


D.4 Internal dynamics

The evolution of system (D.15) in $I$ is known as internal dynamics. It evolves in an $n - \gamma$ dimensional space. For example, the internal dynamics of a system with $n = 3$ and $\gamma = 1$ evolves on a surface. On the other hand, when $\gamma = n$, the output corresponds to a point in state space. In this special case, the system has no internal dynamics.

To illustrate the relevance of the internal dynamics, consider a second order system such that:

\begin{align*}
\dot{\psi}_0 &= \psi_0 \psi_1 + F, \\
\dot{\psi}_1 &= -\psi_0^2, \\
\xi &= \frac{\psi_0}{\psi_1}.
\end{align*}

The time derivative of the output $\xi$ is given by:

\begin{align*}
\dot{\xi} &= \psi_0 - \frac{\psi_0^3}{\psi_1^2} + \frac{F}{\psi_1}.
\end{align*}

Since the first derivative is explicitly dependent on the input $F$, the system has relative degree $\gamma = 1$. Therefore, the internal dynamics evolve in a one-dimensional space. In particular, imagine that one wants to stabilize the output at $\xi = 1$. In state space, $\xi = 1$ corresponds to the subspace $\psi_0 = \psi_1$, which is simply a line. Naturally, both states may evolve to infinity without breaking the output constraint. Nevertheless, the internal dynamics on the line might drive the system to singularities (e.g. $\psi_1 = 0$), or to a region where the linearization does not hold. In the latter case, the method fails completely.

Thus, for the method to be successful, the dynamics in the internal space $I$ must not drive the system out of $I$. 

---

Figure D.3: Control of the input-output linearized system. (1) Evolution of the state $\psi$ as function of the input $F$. (2) Output $\xi$. (3) State feedback $F$ such that (4) is linear. (4) Linear system (D.10), with input $\xi$ and output $\xi$. (5) Control law $\zeta$ such that $\xi$ asymptotically tracks the reference $r(t)$. 

---

74
Linear stability analysis of controlled and uncontrolled system

E.1 Uncontrolled system

The system describing the evolutions of the amplitudes of the modes reads (2.17):

\[
\begin{align*}
\dot{\psi}_0 &= C_1 \psi_1 \psi_2 + C_2 \psi_0 + C_3 F \\
\dot{\psi}_1 &= C_4 \psi_2 \psi_0 + C_5 \psi_1 \\
\dot{\psi}_2 &= C_6 \psi_0 \psi_1 + C_7 \psi_2
\end{align*}
\]

(E.1)

where the coefficients \( C_\alpha, 1 \leq \alpha \leq 7 \), are given by:

\[
\begin{align*}
C_1 &= \frac{k(k^2 + 3)}{2(k^2 + 1)}, & C_2 &= -\frac{(k^2 + 1)}{R}, & C_3 &= -C_2, \\
C_4 &= -\frac{3k}{4}, & C_5 &= -\frac{1}{R}, \\
C_6 &= -\frac{k^3}{2(k^2 + 4)}, & C_7 &= -\frac{k^2 + 4}{R}.
\end{align*}
\]

Setting \( \dot{\psi}_\alpha \) in equation (E.1) to zero yields six equilibrium solutions, generated by:

\[
\begin{align*}
H_1 &= \text{max} \\
H_2 &= \frac{2\sqrt{6} k^2 + 4}{3} \frac{1}{k^2} \frac{1}{R}, \\
H_3 &= \frac{2}{R} \frac{(k^2 + 1)(k^2 + 4)}{k^2 \sqrt{k^2 + 3}} \sqrt{\frac{|F|}{H_2} - 1}, \\
H_4 &= \frac{\sqrt{6}}{3} \frac{k}{k^2 + 4} H_3,
\end{align*}
\]

(E.2) \hspace{2cm} (E.3) \hspace{2cm} (E.4) \hspace{2cm} (E.5)

where \( H_2, H_3, \) and \( H_4 \) only exist if \( |F| > H_2 \). Table 2.1 summarizes the possible combinations between (E.2)-(E.5). Now, we perform a linear stability analysis of the equilibria in
E.1. UNCONTROLLED SYSTEM

<table>
<thead>
<tr>
<th>Equilibrium</th>
<th>$\psi_0$</th>
<th>$\psi_1$</th>
<th>$\psi_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_+^*$</td>
<td>$H_1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$A^*$</td>
<td>$-H_1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$A_+$</td>
<td>$H_2$</td>
<td>$H_3$</td>
<td>$-H_4$</td>
</tr>
<tr>
<td>$A_+^*$</td>
<td>$H_2$</td>
<td>$-H_3$</td>
<td>$H_4$</td>
</tr>
<tr>
<td>$A_-$</td>
<td>$-H_2$</td>
<td>$-H_3$</td>
<td>$-H_4$</td>
</tr>
<tr>
<td>$A_-^*$</td>
<td>$-H_2$</td>
<td>$H_3$</td>
<td>$H_4$</td>
</tr>
</tbody>
</table>

Table E.1: Equilibria of system (E.1). The values of $H_1, ..., H_3$ are given by eqs. (E.2)-(E.5)

the case that $k = 1$ and $R = 10$. The Jacobian matrix is given by:

$$J = \begin{bmatrix} C_2 & C_1\psi_2 & C_1\psi_1 \\ C_4\psi_2 & C_5 & C_4\psi_0 \\ C_6\psi_1 & C_6\psi_0 & C_6 \end{bmatrix}$$

(E.6)

Next, we calculate the eigenvalues of (E.6) for all equilibria listed in table E.1. Figure E.1 shows the eigenvalues per equilibrium as function of the forcing. Since $A_+, A'_+, A_-, A'_-$ do not exist for $|F| < F_{bif}$, with $F_{bif} = 0.8165$, their eigenvalues are only plotted for $F > F_{bif}$. Clearly, for all $|F| < F_{bif}$ the $A_+$ and $A_-$ equilibria are stable. Moreover, for $F > F_{bif}$ $A_+^*$ is unstable and $A_+$ and $A'_+$ are stable. Likewise, for $F < -F_{bif}$ the $A_-^*$ equilibrium is unstable and $A_-$ and $A'_-$ are stable.

Figure E.1: Real part of eigenvalues for the equilibria. (a) For $A_+^* (F > 0)$ and $A_-^* (F < 0)$. (b) For $A_-$ and $A'_-$. (c) For $A_+$ and $A'_+$. 

76
E.2 Controlled system

The controlled system is given by:

\[
\dot{\psi}_0 = \frac{1}{2A_0} \left[ -2A_1 \psi_1 (C_4 \psi_2 \psi_0 + C_5 \psi_1) - 2A_2 \psi_2 (C_6 \psi_0 \psi_1 + C_7 \psi_2) \\
+ \dot{r} + k_0 (A_0 \psi_0^2 + A_1 \psi_1^2 + A_2 \psi_2^2 - r) \right] \\
\dot{\psi}_1 = C_4 \psi_2 \psi_0 + C_5 \psi_1 \\
\dot{\psi}_2 = C_6 \psi_0 \psi_1 + C_7 \psi_2.
\]  
(E.7)

Setting \( \dot{\psi} = 0 \) yields six equilibria generated by:

\[
H_1 = \sqrt{\frac{r}{A_0}}, \\
H_2 = \sqrt{\frac{C_5 C_7}{C_6 C_6}}, \\
H_3 = \sqrt{\frac{C_7 (C_4 C_6 r - A_0 C_5 C_7)}{C_6 (A_1 C_4 C_7 + C_5 A_2 C_6)}}, \\
H_4 = \sqrt{\frac{C_5 (C_4 C_6 r - A_0 C_5 C_7)}{C_4 (A_1 C_4 C_7 + C_5 A_2 C_6)}}.
\]  
(E.8) (E.9) (E.10) (E.11)

Table E.2 summarizes the six possible combinations between (5.13)-(5.16).

<table>
<thead>
<tr>
<th>Equilibrium</th>
<th>( \psi_0 )</th>
<th>( \psi_1 )</th>
<th>( \psi_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_+ )</td>
<td>( H_1 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( A_- )</td>
<td>( -H_1 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( A_+ )</td>
<td>( H_2 )</td>
<td>( H_3 )</td>
<td>( -H_4 )</td>
</tr>
<tr>
<td>( A_- )</td>
<td>( -H_2 )</td>
<td>( -H_3 )</td>
<td>( -H_4 )</td>
</tr>
<tr>
<td>( A_+ )</td>
<td>( H_2 )</td>
<td>( -H_3 )</td>
<td>( H_4 )</td>
</tr>
<tr>
<td>( A_- )</td>
<td>( -H_2 )</td>
<td>( H_3 )</td>
<td>( H_4 )</td>
</tr>
</tbody>
</table>

Table E.2: Equilibria of the controlled system. The values of \( H_1, \ldots, H_4 \) are given by eqs. (5.13)-(5.16).

Now, we perform a linear stability analysis with the setpoint \( r \) as bifurcation parameter. The Jacobian matrix is given by:

\[
J = \begin{bmatrix}
J_1 & J_2 & J_3 \\
C_4 \psi_2 & C_5 & C_4 \psi_0 \\
C_6 \psi_1 & C_6 \psi_0 & C_7
\end{bmatrix},
\]  
(E.12)

where \( J_1, J_2 \) and \( J_3 \) are:
E.2. CONTROLLED SYSTEM

\[ J_1 = \frac{1}{2} k_0 - \frac{1}{2A_0 \psi_0^2} (-2A_1 \psi_1^2 C_5 + k_0 (A_1 \psi_1^2 + A_2 \psi_2^2 - r) - 2A_2 \psi_2^2 C_7) \]

\[ J_2 = -\frac{1}{A_0 \psi_0} (A_1 C_4 \psi_2 \psi_0 + 2A_1 \psi_1 C_5 + A_2 \psi_2 C_6 \psi_0 - k_0 A_1 \psi_1) \]

\[ J_3 = -\frac{1}{A_0 \psi_0} (A_1 \psi_1 C_4 \psi_0 + A_2 C_6 \psi_0 \psi_1 + 2A_2 \psi_2 C_7 - k_0 A_2 \psi_2); \]

Figure E.2 shows the eigenvalues of the equilibria listed in table E.2 for \( R = 10, k = 1 \) and \( k_0 = -1 \). Since \( A_-, A_-', A_+ \) and \( A_+ ' \) do not exist for \( r < r_{bif} \) with \( r_{bif} = 0.66 \) the respective eigenvalues are only plotted for \( r > r_{bif} \). Clearly, for \( r < r_{bif} \) two stable solution exist: \( A_+^* \) and \( A_-^* \). On the other hand, for \( r > r_{bif} \) four stable solutions exist: \( A_-, A_-', A_+ \) and \( A_+ ' \). Moreover, figure E.3 shows \( \lambda_3 \) is dependent on the gain \( k_0 \). For increasing \( |k_0| \), \( \lambda_3 \) becomes increasingly stable. On the other hand, choosing \( k_0 > 0 \) will lead to an unstable system (discussed in chapter 5).

![Figure E.2: Real part of eigenvalues for the equilibria of controlled system. (a) For \( A_+^* \) and \( A_-^* \). (b) For \( A_-, A_-', A_+ \) and \( A_+ ' \).](image)

![Figure E.3: Real part of eigenvalues for \( A_-, A_-', A_+ \) and increasing gain \( k_0 \).](image)
In the present thesis we focused on input-output linearization. As we shall show below, system (2.17) is also full-state linearizable. Full-state linearization consists of finding an output function $y = h(\psi)$ such that the system has relative degree $\gamma = n$. The notation used below, which is different from the rest of the thesis, is equivalent to the standard notation introduced in Khalil [16].

Theorem 13.2 in [16] states the conditions for which a system is feedback linearizable:

A n-dimensional system in the form $\dot{\psi} = f(\psi) + g(\psi)F$, with $F$ the input, is feedback linearizable if and only if there is a domain $D_0 \subset D$ such that

- The matrix $\mathcal{G}(\psi) = [(\psi), ad_f f(\psi), ..., ad_f^{n-1} g(\psi)]$ has rank $n$ for all $\psi \in D_0$.
- The distribution $\mathcal{D} = \text{span}\{g, ad_f g, ..., ad_f^{n-2} g\}$ is involutive in $D_0$.

For clarity, we repeat system (2.17):

$$
\begin{align*}
\dot{\psi}_0 &= C_1 \psi_1 \psi_2 + C_2 \psi_0 + C_3 F \\
\dot{\psi}_1 &= C_4 \psi_2 \psi_0 + C_5 \psi_1 \\
\dot{\psi}_2 &= C_6 \psi_0 \psi_1 + C_7 \psi_2.
\end{align*}
$$

Let us redefine the input as $F = F_u + F_c$, where $F_c$ is a constant force, and $F_u$ is the active control input. The addition of $F_c$ ensures that the equilibrium point for $F_c = 0$ is not outside the domain of linearization. Now, we can write the system in the general form with

$$
\begin{align*}
f(\psi) &= \begin{bmatrix}
C_1 \psi_1 \psi_2 + C_2 \psi_0 + C_3 F_c \\
C_4 \psi_2 \psi_0 + C_5 \psi_1 \\
C_6 \psi_0 \psi_1 + C_7 \psi_2,
\end{bmatrix}
\end{align*}
$$

and

$$
\begin{align*}
g(\psi) &= \begin{bmatrix}
C_3 \\
0 \\
0.
\end{bmatrix}
\end{align*}
$$

Calculation of the Lie brackets for $\mathcal{G}(\psi)$ gives:

$$
[f, g] = ad_f g = \begin{bmatrix}
-C_2 C_3 \\
-C_3 C_4 \psi_2 \\
-C_6 C_3 \psi_1.
\end{bmatrix}
$$
and

\[
[f, ad_f g] = ad_f^2 g = \begin{bmatrix}
C_2^2 C_3 + C_1 C_3 C_4 \psi_2^2 + C_1 C_3 C_6 \psi_1^2 \\
-C_3 C_4 C_7 \psi_2 + C_2 C_3 C_4 \psi_2 + C_3 C_4 C_5 \psi_2 \\
-C_3 C_5 C_6 \psi_1 + C_2 C_3 C_6 \psi_1 + C_3 C_6 C_7 \psi_1
\end{bmatrix}.
\] (F5)

Next, we evaluate the conditions:

**Condition 1**

\(\mathcal{G}(\psi) = [g(\psi), ad_f g(\psi),..., ad_f^{n-1} g(\psi)]\) has rank 3 for all \(\psi\) such that \(\psi_1 \neq 0\) and \(\psi_2 \neq 0\). Since \(\mathcal{G}\) is a square matrix, full rank is equivalent to \(det(\mathcal{G}) \neq 0\). Because \(det(\mathcal{G}) = 2C_3^3 C_4 \psi_2 C_6 \psi_1 (C_5 - C_7) \neq 0\) for \(C_5 \neq C_7\) and \(\psi_2, \psi_1 \neq 0\) the first condition is satisfied.

**Condition 2**

Let \(\mathcal{D} = \text{span}\{g, ad_f g\}\), which is involutive in \(D_0\) if \([g, ad_f g] \in D_0\), which is only true if \(\text{rank}[g, ad_f g, [g, ad_f g]] = \text{rank}[g, ad_f, g]\) for all \(\psi\) in \(D_0\).

\[
[g, ad_f g] = \frac{\partial ad_f g}{\partial \psi} - \frac{\partial g}{\partial \psi} ad_f g = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.
\] (F6)

Since this is the null vector, \(\mathcal{D}\) is involutive in \(D_0\). In summary, both conditions are satisfied for \(D_0 = \{\psi \in R^3 | \psi_1 \neq 0 \text{ and } \psi_2 \neq 0\}\), and hence the system is feedback linearizable.
In this appendix a short description of the contents of the attached DVD is given. Each subdirectory contains a file `readme.txt` describing the exact contents.

- `\report` : Here one finds a digital copy of this report including source.
- `\matlab\forwardAdvect` : Forward advection files + data.
- `\matlab\backwardAdvect` : Backward advection files + data.
- `\matlab\control` : Input-output linearization simulation and analysis files.
- `\matlab\programming` : C-source files for the various mex-files.
- `\matlab\model` : Files for simulation and analysis for uncontrolled model.
- `\movies` : A small collection of movies from various simulations.
Samenvatting

Dit werk richt zich op het regelen van vloeistof menging in een lineaire serie wervels. Achtereenvolgens worden de volgende aspecten besproken:

• Wat zijn de dynamische eigenschappen van een lineaire serie wervels?
• Hoe kan vloeistof menging gekwantificeerd worden?
• Hoe kan vloeistof menging geregeld worden gebruikmakend van statistische stromings grootheden?

Ter beantwoording van de eerste vraag is er een stromingsmodel overgenomen uit de literatuur [13]. Dit model vindt zijn oorsprong in een lineaire stabiliteits analyse van de hoofdstroming. De laagste drie Fourier modes worden gebruikt om het minimale niet-lineaire systeem samen te stellen. De eerste mode beschrijft de hoofdstroming (de lineaire wervel serie), de tweede een schuif stroming, en de derde een wervel rooster. Het tijd afhankelijke gedrag van de modes, welke een functie is van de excitatie kracht, wordt beschreven door een niet-lineair systeem van differentiaal vergelijkingen. Vervolgens wordt aangetoond dat (i) voor toenemende excitatie de stroming gedomineerd wordt door de tweede en derde mode en (ii) de stromingen voor positieve en negatieve excitatie symmetrisch aan elkaar zijn.

Met betrekking tot de kwantificatie van het mengen van twee vloeistoffen is er het mixing number [28] gebruikt. Het mixing number is gebaseerd op discrete zwart-wit afbeeldingen van het process. Deze afbeeldingen kunnen gegenereerd worden met drie methodes: voorwaartse advectie, verbeterde voorwaartse advectie en achterwaartse advectie. Omdat achterwaartse advectie (op dit moment) te veel rekentijd in beslag neemt wordt in dit werk voorwaartse advectie toegepast.

Bij het beantwoorden van de laatste vraag wordt duidelijk dat het verband tussen de hydrodynamica van de stroming (snelheids veld) en de statistische beschrijving (mixing number) niet triviaal is. Om dit probleem op te lossen stellen we voor een verbinding te leggen via de globale entropie $\langle s \rangle$ en de viskeuze dissipatie $\langle \epsilon \rangle$ in de stroming. In deze context wordt een expliciete expressie afgeleid voor de entropie generatie in de lineaire wervel serie. Aangezien de expressie een directe functie is van de modes in het model, kan de stroming als een standaard ingangs-uitgangs systeem beschouwd worden.

Het model van de lineaire wervel serie met de viskeuze dissipatie als uitgang kan geregeld worden met ingangs-uitgangs linearisatie. Deze techniek stelt ons in staat het mengen van isotherme vloeistoffen te regelen door het voorschrijven van de globale entropie. In deze context bestuderen we meng protocollen zodat $\langle s \rangle \sim t^\alpha \Leftrightarrow \langle \epsilon \rangle \sim t^{\alpha - 1}$, met $0 < \alpha \leq 1$. Er wordt aangetoond dat (i) de menging monotoon toeneemt voor toenemende $\alpha$ en (ii) de mengtijd schaalt met $t_m \sim \langle \epsilon \rangle^{-1/2}$. Het laatste resultaat is nadien onderbouwd door een korte dimensie analyse.
Dankwoord

Graag wil ik van deze plaats gebruik maken om mijn ouders te bedanken voor de jarenlange steun en aanmoediging. Ze stonden altijd paraat om naar mijn problemen te luisteren en zonodig een oplossing aan te bieden. Daarnaast wil ik mijn vriendin Lidia bedanken voor de nodige ondersteuning, kritische blik en soms hoognodige afleiding: Lidia thank you.

Natuurlijk wil ik ook prof. Nijmeijer en prof. van Heijst bedanken voor de mogelijkheid tot deze bijzonder uitdagende afstudeeropdracht. Als laatste rest mij dr. Francisco Fontenele Araujo te bedanken voor zijn nimmer aflatende aanmoediging, motivatie, kritische blik en talloze belangrijke discussies.


