The Principle of Maximum Entropy Production in a Simple Model of a Convection Cell

by

David W. Hogg

Submitted to the Department of Physics
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Physics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1992

Author .....
Department of Physics
11 May 1992

Certified by .....
Michel Baranger
Professor of Physics
Thesis Supervisor

Accepted by . .....
Aron Bernstein
Chairman, Departmental Thesis Committee

The Principle of Maximum Entropy Production in a Simple Model of a Convection Cell

by

David W. Hogg

Submitted to the Department of Physics
on 11 May 1992, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Physics

Abstract

A simple, discrete, two-dimensional, computer model of a convection cell is described that is qualitatively accurate, but not designed to be quantitatively accurate. The model can be used to simulate relaxation to equilibrium as well as steady-state processes.

A principle of maximum entropy production is formulated for non-equilibrium statistical systems: close to a bifurcation, the stable branch is the one on which the system produces entropy at the greatest rate. In order to test this principle, the model is used to simulate convection and to simulate the relaxation to equilibrium of a fluid with a non-uniform initial heat distribution.

The following results are found: Close to the bifurcation from conduction-dominated to mixing-dominated behaviour, a fluid in a gravitational field with non-uniform heat distribution relaxes to equilibrium by the path that maximizes entropy production: the mixing-dominated path. Close to every investigated bifurcation in the behaviour of the simulated convection cell, the stable mode is the available steady-state mode that maximizes entropy production.

Thesis Supervisor: Michel Baranger
Title: Professor of Physics

Biography

David W. Hogg was born on 8 September 1970 in Toronto, Ontario, Canada to Peter W. Hogg and Frances L. Benson Hogg. He spent most of his childhood in Toronto, attending Whitney Public School through grade six and the University of Toronto Schools through grade thirteen. At the end of high school, he represented Canada at the Nineteenth International Physics Olympiad in Bad Ischl, Austria. He entered the Massachusetts Institute of Technology in September of 1988 and at the time of writing plans to graduate in June of 1992 with a B. S. in physics and a minor in philosophy. He plans to enter graduate school in physics at the California Institute of Technology in the fall of 1992. During the summers of 1990 and 1991 he worked at the Canadian Institute for Theoretical Astrophysics in Toronto, and he plans to work there again in the summer of 1992. At the time of writing, he has two published articles, one in educational psychology, and one in Solar System dynamics.

Acknowledgments

Above all, I would like to thank Michel Baranger for being an excellent thesis advisor. In justice, this thesis should be co-authored by him, as all the ideas in it were worked out in discussions between him and me. Of course, he is not responsible for any errors I have made.

I owe thanks to all of the physicists who listened to explanations of the work in this thesis and who commented on it, including Albert Libchaber, Michael Berry, and Edgar Knobloch. These scientists gave me new ideas, corrected many of my misconceptions, supplied me with useful references, and were very encouraging.

I am also grateful to Jack Wisdom, who suggested that I look at the work of I. Prigogine; and to S. Goerner, who put Michel Baranger and me in touch with R. Swenson.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Objectives

1.1 Introduction

One of the objectives of the research described in this thesis was to explore the behaviour of statistical systems which are not at equilibrium; this includes systems that are relaxing towards equilibrium, and systems that are in a steady non-equilibrium state. In order to perform this research, I created a very simple, discrete, mathematical model of a convection cell. The purpose of this thesis project was not to model the behaviour of a convection cell in a quantitatively accurate way. The purpose was to study a simplified but qualitatively accurate model which is well understood, and see if it behaves in a way that confirms general hypotheses that might apply to a very wide range of non-equilibrium statistical systems.

In this chapter, I will begin by introducing the principle of maximum entropy production, which is the general hypothesis that I investigated. I will then go on to discuss the system that was modelled for this purpose: the convection cell. Finally I will very briefly discuss the model itself. Following chapters will give a much more detailed description of the model and a presentation of my results.

1.2 The Principle of Maximum Entropy Production

One general principle governing statistical processes might be the following: systems which are not at equilibrium and have several possible paths or qualitatively distinct ways of getting to equilibrium or of maintaining their steady state will always choose the path or state that maximizes the rate of entropy production. This is the crudest formulation of the *principle of maximum entropy production.*

The principle of maximum entropy production has been suggested before in the literature, but, as far as I know, it has never been conclusively confirmed or refuted. Dafermos (1973; 1984) has put forward this principle in a general way as the "entropy rate admissibility criterion," and analyzed it for systems governed by "hyperbolic" conservation laws. Unfortunately, his papers are very difficult, and I discovered them too recently to have completely understood them by the time of writing. His entropy rate admissibility criterion is presented as a likely hypothesis that needs to be tested or proved. I do not know of anyone who has undertaken either task.

A principle of maximum entropy production is widely used by systems theorists, particularily R. Swenson (Swenson and Turvey, 1991), in discussing self-organization in the evolution of ecosystems and the evolution of life. However, the principle has never been rigourously related to the laws of physics, as far as I know.

Certainly the principle of maximum entropy production is not true for absolutely all physical systems. For instance, Prigogine (1955) has shown that linear systems of chemical reactions in the steady state close to equilibrium actually minimize entropy production. Therefore, the principle of maximum entropy production could only be true for non-linear systems or systems far from equilibrium. In fact, Prigogine (1980, p. 88) implies that entropy production is often maximized for systems far from equilibrium. He does not back this up with any arguments or references.

Something very close to the principle of maximum entropy production is sometimes appealed to in discussions of pattern selection in convecting systems. Rayleigh-Bénard convection can occur in many different types of patterns, such as rolls, cross-

rolls, squares and hexagons (see, *e.g.*, Knobloch, 1989; Silber and Knobloch, 1988). However, near the onset of convection (Rayleigh number close to the critical value), it has been suggested that the pattern which is stable is the one that transports the most heat from the hot lower plate to the cold upper plate (Clune and Knobloch, 1991). In a convecting system, the entropy production is proportional to the transport of heat between the hot and cold plates, so the stable pattern is the one that produces entropy at the greatest rate.

Unfortunately, I am told that this is not true far from the onset of convection (Knobloch, 1992; Libchaber, 1992). That is, when the Rayleigh number of the system is greater than its critical value by a large enough margin, the stable pattern may not be the one that transports the most heat. If the maximum-heat-transport pattern-selection rule is true at all, it is only true near the onset of convection. This suggests a more restrictive version of the principle of maximum entropy production:

The Principle of Maximum Entropy Production: Consider any non-equilibrium statistical system with variable parameter η. Imagine that there is a bifurcation in the behaviour of the system at some critical value $\eta = \eta_c$, such that for $\eta > \eta_c$ there are at least two qualitatively distinct solutions to the equations of motion. If η exceeds η_c by a small enough margin, the system will choose the branch of the bifurcation (*i.e.*, the solution to the equations of motion) that maximizes entropy production. In other words, close to any bifurcation, the stable branch is the one that maximizes entropy production.

This last formulation of the principle of maximum entropy production is the one that was tested in this thesis. I do not know whether or not this principle is obvious. I do not even know if it is true. However, the principle of maximum entropy production was confirmed in all of the tests that I performed.

1.3 The Steady-State Convection Cell

In the previous section, I said a little about convection, convection patterns, and critical Rayleigh numbers. I will now go further into this subject, as convection is

going to be the main focus of this thesis.

A convection cell is a box in a gravitational field that is completely filled with water or some other liquid. The bottom and top walls of the box are heat-conducting plates that are maintained at different temperatures; the bottom plate hotter than the top plate. If the temperature difference ΔT between the plates is small enough, then the water in the cell will remain motionless and heat will diffuse to the top of the box in a very symmetrical way. The behaviour of the system will be dominated by conduction. If ΔT is above some critical value $(\Delta T)_c$, however, convection will set in, and the water will start moving in little counter-rotating eddies, actually carrying hot water to the top of the box and cold water to the bottom. This is called *Rayleigh-Bénard convection,* after Lord Rayleigh and H. Bénard. The latter (Bénard, 1901) performed the first convection experiments, and the former (Rayleigh, 1916) was the first person to analyze them and explain the existence of a critical temperature difference $(\Delta T)_c$.

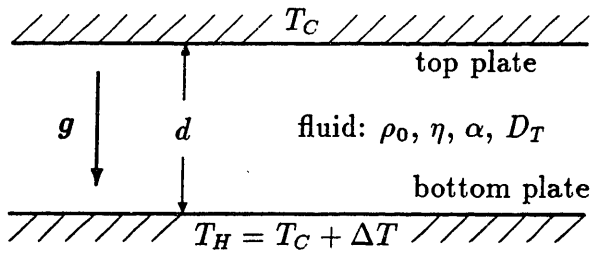

Figure 1-1: A cross-section of a convection cell

Actually, to speak more generally, temperature is not the only parameter that can be varied to bring on convection. Viscosity and system size also have critical values if the other parameters are kept fixed. Usually the system is descibed in terms of the dimensionless parameter R called the *Rayleigh number:*

$$R = \frac{\rho_0 \, g \, \alpha \, d^3}{\eta \, D_T} \Delta T \tag{1.1}$$

where ρ_0 is the mean density, g is the acceleration due to gravity, α is the expansion coefficient, d is the distance between plates, η is the viscosity, D_T is the thermal diffusivity, and ΔT is the temperature difference between the plates (Bergé, Pomeau,

Vidal, 1984, pp. 84–85). Whenever the Rayleigh number exceeds the critical value R_c, convection sets in. In the terminology of Hopf bifurcation theory, R_c is a bifurcation point, and the bifurcation to convection is supercritical (Libchaber, 1992; Tabor, 1989, pp. 197–198).

Of course there are many possible patterns of convection. As I mentioned in the previous section, these include rolls, cross-rolls, squares, and hexagons. One thing that unites these patterns is that the characteristic distance scale of all patterns is roughly d, the height of the cell (Velarde and Normand, 1980). For example, if the convection pattern is rolls, the individual rolls will tend to have a cross-section of size $d \times d$. As I said earlier, which pattern is actually selected is a difficult question except possibly when the system is close to the convection bifurcation. When it is close to the bifurcation it may simply be a question of which pattern produces entropy at the greatest rate.

The only type of convection pattern that I was able to observe was rolls, because the model I used is of a two-dimensional slice of a convection cell, and rolls form the only convection pattern with translational symmetry along the third dimension. However, as I will show, there are still many phenomena to study, even when the convection cell system is restricted to roll patterns. For instance, rolls of different aspect ratios can be solutions to the equations of motion. Why then does the system choose roll patterns with a certain aspect ratio? This question may also be answerable with the principle of maximum entropy production.

1.4 Relaxation to Equilibrium

In addition to convection experiments, there is another set of experiments that could be carried out in a fluid cell in a gravitational field, which I will refer to as *relaxation* experiments. These involve putting a small quantity of hotter fluid into the bottom of a cold fluid cell and then watching the system relax to equilibrium. That is, instead of using heat conductors at constant temperature at the top and bottom of the cell, with heat flowing through the cell, the walls of the cell could be made of non-heat-

conducting material. A small bit of heat could be added to the bottom initially and the system could be observed as it approaches equilibrium. As far as I know, no one has ever actually performed any relaxation experiments with a fluid cell.

Just as in the steady-state experiments, there should be a critical Rayleigh number R'_c, below which the behaviour of the system will be conduction dominated, and above which the behaviour will be mixing dominated. I call it "mixing," and not "convection," because the motion of the fluid will be a brief transient in which clear patterns will not develop. Also, $R'_c > R_c$ because in steady-state experiments there is a lot of time (an infinite amount) for a convection pattern to develop out of random fluctuations, whereas in relaxation experiments if the mixing does not set in quickly, heat conduction may bring the system to equilibrium before the initial fluctuations in the fluid motion can be amplified.

A relaxation experiment should provide a very simple system with a bifurcation point beyond which there are two qualitatively distinct paths to equilibrium. It is therefore an ideal system for checking the principle of maximum entropy production. In addition to modelling steady-state convection, the model presented in this thesis was used to model relaxation.

1.5 The Model

Here I will briefly introduce the model I used, leaving the details for the next chapter. The actual C code used to implement the model is given in Appendix A

The model is a two-dimensional cellular-automata-style model of a convection cell in the Boussinesq approximation. It can be used to model either steady-state or relaxation experiments.

The model is a two-dimensional cross-section of a convection cell. The hot plate is at the bottom and the cold plate is at the top. Because there is only one horizontal dimension, any patterns that are produced must be interpreted as having translational symmetry along the unmodelled axis.

It is discrete in both space and time. The model has local variables on the points

of a two-dimensional lattice: there are heats, and there are "whorls." The whorls form a stream function in the model; they simplify the laws I use regarding fluid momentum. At each time step, the variables change according to several local laws. These laws were not created by taking a discrete approximation of the differential equations that describe a convection cell. Rather they were motivated directly from simple facts about fluids. The thing that I like about the model is that its behaviour is qualitatively correct despite the fact that it is very simple and it was created without analyzing real convection cells quantitatively.

The Boussinesq approximation is the approximation in which none of the physical parameters of the system depend on temperature. Of course convection depends on the fact that fluids expand when heated; but the Boussinesq approximation is that the thermal expansion coefficient α is independent of temperature and the thermal expansion is never a significant correction to the mean density (Libchaber, 1992). In equations, this says that for all parameters μ_i except the density:

$$\frac{\partial \mu_i}{\partial T} = 0 \tag{1.2}$$

And for the density:

$$\alpha \Delta T \ll \rho_0 \tag{1.3}$$

where α is the thermal expansion coefficient, ΔT is the temperature difference and ρ_0 is the mean density.

When discussing the model, I never speak in terms of these parameters (viscosity, thermal expansion, thermal diffusivity *etc.*) because the model has a different, although related, parameter set. However, it is true that the parameters in the model are independent of temperature. That is why I say that the model is in the Boussinesq approximation.

The model can perform both steady-state and relaxation experiments, and it was used for both types. It has a large parameter space, so steady-state experiments were performed to ensure that a set of physically realistic (in a qualitative sense) parameters were being used. This parameter set was then employed in testing the principle of

maximum entropy production. The results presented in this thesis represent only the beginning of a very long possible programme of research, both into models of this type and into principles of non-equilibrium statistical mechanics such as the principle of maximum entropy production.

Chapter 2

The Model

2.1 The Lattice

The model is two-dimensional. The x-direction is parallel to the top and bottom of the cell. The y-direction is up.

The model is discrete in both space and time. The two-dimensional space is divided into cells, which are called *bins*. There are M bins in the x-direction and N bins in the y-direction. However, for reasons that will soon come apparent, there are about four times as many points in the lattice as there are bins. For any position (x,y) on the lattice, the following holds:

$$0 \leq x \leq 2M \quad \text{and} \quad 0 \leq y \leq 2N \tag{2.1}$$

Each position (x,y) on the grid can be either a bin, an edge, or a corner. Table 2.1 explains this terminology, and it is depicted in Figure 2-1.

Because whether or not a particular index is odd or even is important to the discussion, I will always use the indices m and n to represent odd numbers and the indices r and s to represent even numbers. This is also illustrated in Table 2.1.

x	y	position name	examples
odd	odd	bin	(m,n), $(15,9)$, $(1,2N-1)$
odd	even	edge, horizontal	(m,s), $(11,4)$, $(m,n+1)$
even	odd	edge, vertical	(r,n), $(6,13)$, $(2M-2,n)$
even	even	corner	(r,s), $(8,8)$, $(r+2,n-1)$

Table 2.1: Lattice position names for an arbitrary position (x,y).

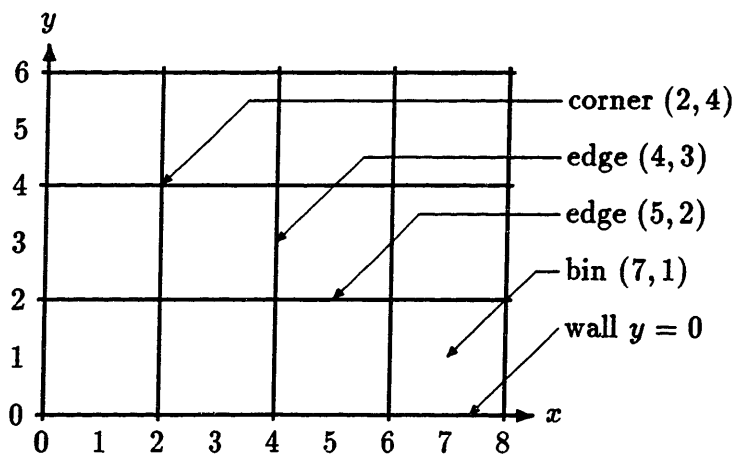

Figure 2-1: A small portion of the lattice, showing some representative position names.

2.2 Heat

In each bin (m, n) there is an amount of heat $q_{(m,n)}$. $q_{(m,n)}$ is a real number and it can be negative or positive. It represents the amount of heat contained in the liquid at that location above some reference amount (zero).

The model operates in the Boussinesq approximation, as discussed in Section 1.5. Therefore, the heat capacity of the liquid in each bin is independent of the amount of heat in that bin, so the heats $q_{(m,n)}$ can be thought of as temperatures. The field of heats is proportional to the temperature field. In fact, I may occasionally refer to the heats in the model as temperatures.

2.3 Momenta

There are momenta defined at every position in the lattice, but I will do most calculations with momenta at the edges. In fact, I will do most momentum calculations with x-components on vertical edges and y-components on horizontal edges. These are the momentum components that represent flow directly from one bin to an adjacent bin. I will call these *edge-crossing momenta.* For example $p_{x(4,5)}$ is the edge-crossing momentum from bin $(3, 5)$ to bin $(5, 5)$.

Because the liquid is modelled as incompressible, the following restriction holds on the edge-crossing momenta surrounding each bin (m, n):

$$p_{x(m+1,n)} - p_{x(m-1,n)} + p_{y(m,n+1)} - p_{y(m,n-1)} = 0 \tag{2.2}$$

Equation (2.2) is just a discrete version of the continuous equation

$$\frac{\partial p_x}{\partial x} + \frac{\partial p_y}{\partial y} = 0 \tag{2.3}$$

(2.3) is simply the equation of continuity for a fluid of constant density (Landau and Lifshitz, 1959, p. 20).

Table 2.2 shows how to use the edge-crossing momenta to calculate the x- and

y-momenta in bins and at corners.

position	momentum formulae
bin (m,n)	$p_{x(m,n)} = (1/2)(p_{x(m+1,n)} + p_{x(m-1,n)})$
	$p_{y(m,n)} = (1/2)(p_{y(m,n+1)} + p_{y(m,n-1)})$
corner (r,s)	$p_{x(r,s)} = (1/2)(p_{x(r,s+1)} + p_{x(r,s-1)})$
	$p_{y(r,s)} = (1/2)(p_{y(r+1,s)} + p_{y(r-1,s)})$

Table 2.2: Calculation of momenta in bins and at corners using edge-crossing momenta. Table 2.3 will replace these formulae once whorls have been introduced.

2.4 Whorls

The incompressibility of the fluid, as expressed by (2.2), is difficult to model with local laws. That is, an arbitrary momentum field will not satisfy (2.2). At first I thought that there might need to be a non-local law in the model that would readjust all momenta to ensure that (2.2) is always satisfied. However, the introduction of whorls renders this unnecessary. The whorls are a set of local quantities that determine the momentum field and ensure that (2.2) is always satisfied.

At each corner (r,s), there is a real number $w_{(r,s)}$ which is called the *whorl* at (r,s). A unit whorl $w_{(r,s)} = 1$ represents counter-clockwise flow of liquid through the four bins surrounding the corner (r,s). An isolated unit whorl at the corner (r,s) is equivalent to the following four momenta on the surrounding edges:

$$p_{y(r+1,s)} = 1, \quad p_{x(r,s+1)} = -1, \quad p_{y(r-1,s)} = -1, \quad p_{x(r,s-1)} = 1 \tag{2.4}$$

Figure 2-2 depicts a unit whorl.

Notice that an isolated whorl satisfies (2.2) for its four surrounding bins. In fact, any linear superposition of whorls will also satisfy (2.2) in all bins (m,n). The whorls form a complete set for describing the momenta in the model.

If one takes the continuum limit, one sees that the whorls form a scalar function

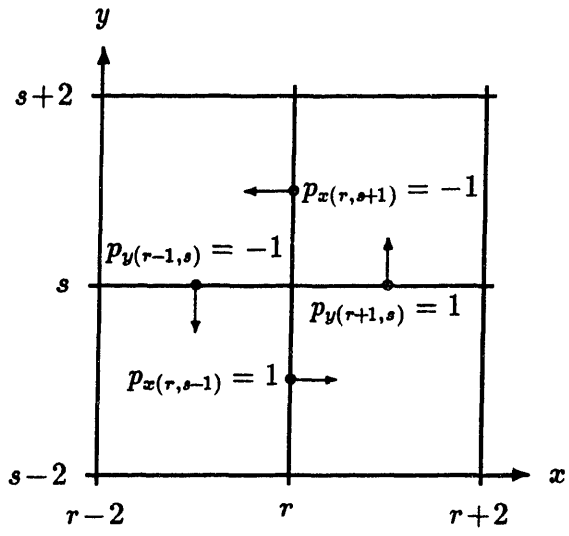

Figure 2-2: An isolated unit whorl $w_{(r,s)} = 1$

W, the gradient of which is the momentum field rotated by 90°. That is:

$$p_x = \frac{\partial W}{\partial y} \quad \text{and} \quad p_y = -\frac{\partial W}{\partial x} \tag{2.5}$$

Equations (2.5) show that the field of whorls constitutes a stream function (Berry, 1992; Landau and Lifshitz, 1959, p. 22).

Equations (2.5) ensure that (2.3) would be satisfied in the continuum limit. Also, the equations (2.5) show that the flow lines for the momentum are contours of constant W. The flow is counter-clockwise around positive "hills" of W and clockwise around negative "depressions."

Table 2.3 shows how to calculate the momenta at all positions in the lattice using adjacent whorls.

2.5 Laws in the Model

The following laws get applied to the heats $q_{(m,n)}$ and whorls $w_{(r,s)}$ in the model at each time step. Each law gets applied as a separate, independent operation, but a single time step consists of the application of all the laws in order.

The laws constitute the "physics" of the model. The list of laws given in this

position	momentum formulae
bin (m,n)	$p_{x(m,n)} = (1/2)[w_{(m+1,n+1)} - w_{(m+1,n-1)} + w_{(m-1,n+1)} - w_{(m-1,n+1)}]$ $p_{y(m,n)} = (1/2)[w_{(m-1,n-1)} + w_{(m-1,n+1)} - w_{(m+1,n-1)} - w_{(m+1,n+1)}]$
horizontal edge (m,s)	$p_{y(m,s)} = [w_{(m-1,s)} - w_{(m+1,s)}]$
vertical edge (r,n)	$p_{x(r,n)} = [w_{(r,n+1)} - w_{(r,n-1)}]$
corner (r,s)	$p_{x(r,s)} = (1/2)[w_{(r,s+2)} - w_{(r,s-2)}]$ $p_{y(r,s)} = (1/2)[w_{(r-2,s)} - w_{(r+2,s)}]$

Table 2.3: Calculation of momenta at all lattice positions using adjacent whorls. The corner and bin formulae follow from the edge formulae and the equations given in Table 2.2.

section is not exhaustive; other laws will be introduced in Sections 2.6.2 and 2.6.3. The laws are summarized in Table 2.4, where their order of application is also given.

2.5.1 Heat Diffusion

At each time step, a fraction α_q of the heat in each bin spreads into each of the four adjacent bins. That is, $4\alpha_q q_{(m,n)}$ is subtracted from $q_{(m,n)}$ and $\alpha_q q_{(m,n)}$ is added to $q_{(m+2,n)}$, $q_{(m,n+2)}$, $q_{(m-2,n)}$, and $q_{(m,n-2)}$. α_q must lie in the range

$$0 \leq \alpha_q \leq \frac{1}{4} \tag{2.6}$$

or else more heat will diffuse out of each location than is in it to begin with.

Heat diffusion simulates heat conduction. I call it "diffusion" to emphasize the symmetry between heats and whorls in the model (*cf.* Section 2.5.3). Heat diffusion is a linear process acting only on the heats.

2.5.2 Heat Transport

After heat diffusion is completed, some quantity of heat is transported across each edge in the direction of the moving liquid. At the vertical edge (r,n), a quantity of heat $\Delta_x q_{(r,n)}$ moves from bin $(r-1,n)$ to bin $(r+1,n)$. Note that $\Delta_x q_{(r,n)}$ can be

either positive or negative. If $\Delta_x q_{(r,n)}$ is negative, heat actually moves from $(r+1,n)$ to $(r-1,n)$.

$$\Delta_x q_{(r,n)} = \beta_q \, p_{x(r,n)} \frac{1}{2} \left[q_{(r-1,n)} + q_{(r+1,n)} \right] \tag{2.7}$$

$\Delta_x q_{(r,n)}$ is called the *horizontal heat transport.* β_q is called the *heat transport parameter.*

Similarly, in the y-direction there is the vertical heat transport, defined at every horizontal edge (m,s):

$$\Delta_y q_{(m,s)} = \beta_q \, p_{y(m,s)} \frac{1}{2} \left[q_{(m,s-1)} + q_{(m,s+1)} \right] \tag{2.8}$$

$\Delta_y q_{(m,s)}$ moves from $(m,s-1)$ to $(m,s+1)$ if it is positive, and from $(m,s+1)$ to $(m,s-1)$ if it is negative.

Heat transport is a non-linear process since the amount of heat transported is proportional to a product of momentum (a difference of whorls) and heat.

2.5.3 Whorl Diffusion

Just like heat diffusion (Section 2.5.1), there is whorl diffusion. This occurs at each time step after heat diffusion and heat transport have completed. A fraction α_w of the whorl at each corner (r,s) spreads into each of the four adjacent corners. Like α_q:

$$0 \leq \alpha_w \leq \frac{1}{4} \tag{2.9}$$

Whorl diffusion simulates the effect of viscosity, since any large differences in nearby whorls will get evened out by its action.

Whorl diffusion is a linear process acting on the whorls.

2.5.4 Whorl Transport

We whirl the whorl.

—Michel Baranger.

Whorl diffusion is followed by whorl transport. Consider the whorl at (r,s), and

the four surrounding whorls at corners $(r+2, s)$, $(r, s+2)$, $(r-2, s)$, and $(r, s-2)$. If $w_{(r,s)}$ is positive, then a fraction of each of the four adjacent whorls moves into the location 90° counter-clockwise from it. That is, if $w_{(r,s)}$ is positive, then the following movements of whorls take place:

$$\begin{aligned}
&\beta_w\, w_{(r,s)} \frac{1}{2}\left[w_{(r+2,s)} + w_{(r,s+2)}\right] \quad \text{moves from } (r+2, s) \quad \text{to } (r, s+2) \\
&\beta_w\, w_{(r,s)} \frac{1}{2}\left[w_{(r,s+2)} + w_{(r-2,s)}\right] \quad \text{moves from } (r, s+2) \quad \text{to } (r-2, s) \\
&\beta_w\, w_{(r,s)} \frac{1}{2}\left[w_{(r-2,s)} + w_{(r,s-2)}\right] \quad \text{moves from } (r-2, s) \quad \text{to } (r, s-2) \\
&\beta_w\, w_{(r,s)} \frac{1}{2}\left[w_{(r,s-2)} + w_{(r+2,s)}\right] \quad \text{moves from } (r, s-2) \quad \text{to } (r+2, s)
\end{aligned} \tag{2.10}$$

β_w is called the *whorl transport parameter.* If $w_{(r,s)}$ is negative, the movement is in the opposite direction. The idea is that the four whorls rotate around the central whorl in the direction of the central whorl, and that the amount that moves is proportional to the magnitude of the central whorl.

Whorl transport is a non-linear process acting on the whorls, since the amount of whorl that moves is a product of two whorls.

How whorl transport is accomplished at the top and bottom boundaries of the lattice is discussed in Section 2.6.4.

2.5.5 Gravity

Hot liquid tends to rise. This is the motivation for the law called *gravity* in the model. This law simulates the "torque" on each corner from an imbalance of heats on the left and right sides.

Consider the four bins surrounding each corner (r, s). At each time step, after the diffusion and transport operations, if there is more heat in the two bins to the right of (r, s) than there is to the left, then the whorl $w_{(r,s)}$ will be increased. If there is more heat to the left than to the right, then $w_{(r,s)}$ will be decreased. Specifically,

$w_{(r,s)}$ will change by an amount $\Delta w_{(r,s)}$ where

$$\Delta w_{(r,s)} = \gamma \left(q_{(r+1,s+1)} + q_{(r+1,s-1)} - q_{(r-1,s+1)} - q_{(r-1,s-1)} \right) \tag{2.11}$$

γ is the gravitational parameter.

Gravity is a linear process acting on the whorls, since the change in whorl is proportional only to a sum of heats.

2.5.6 Randomization

Some randomness is put into the model to allow symmetry to break. At each time step, to each corner is added a random whorl picked from a Gaussian distribution with mean zero and standard deviation σ_w.

Sometimes my results consist of comparing runs in which symmetry is allowed to break with runs in which symmetry is not allowed to break. The latter are runs in which $\sigma_w = 0$.

The Gaussian-distributed random numbers were generated with Press *et al.*'s (1988) C language routine "gasdev.c."

2.6 Boundary Conditions

Until now I have carefully avoided discussion of what happens at the walls of the container. A position (x, y) in the lattice will be called a *wall point* if

$$y = 0 \quad \text{or} \quad y = 2N \tag{2.12}$$

That is, a point is a wall point if it lies on the top or bottom boundary of the lattice.

There are also boundaries to the lattice in the x-direction, at $x = 0$ and $x = 2M$. However, I model the convection cell as being infinite in x-extent by using circular boundary conditions. That is, bin $(2M - 1, y)$ is adjacent to bin $(1, y)$ for all y, and the edges and corners with $x = 2M$ are identical with those with $x = 0$. The model

"wraps-around" in the x-direction. All laws behave accordingly.

Obviously the wall points defined by (2.12) are corners and edges by the terminology given in Table 2.1. However, they are corners at which the whorls are identically zero and edges at which no edge-crossing momenta are defined. This is because there can be no momenta "through" the walls.

The behaviour of heat at the walls determines the "version" of the model that is being used. If no heat is exchanged with the walls, then I will say that the model is in the *relaxation version.* If, on the other hand, there is heat exchange with the walls, then I will say that the model is in the *steady-state version.*

2.6.1 Relaxation Version

In the relaxation version, there is no heat exchange with the walls. The model starts in some initial heat and whorl distribution and it relaxes to equilibrium (see Section 2.7.1 for the actual initial conditions used). This process does not involve dissipation of heat, only redistribution of heat throughout the lattice by heat transport and diffusion. Equilibrium is the state in which the heat is evenly distributed and the whorls are all zero. In the relaxation version, heat diffusion and transport do not operate "through" or "into" the walls.

2.6.2 Steady-State Version and Heat Flow

In the steady-state version of the model, the bins at the top and bottom of the cell are maintained at heat values 0 and q_0, respectively, where $q_0 > 0$. That is, precisely enough heat is added to the bottom of the cell and subtracted from the top to maintain the heats in the bins adjacent to the walls at prescribed "temperatures." This is how the model is used to simulate a conventional convection cell. That is, in the steady-state version, heat diffusion and transport do not operate "through" or "into" the walls, but a new law called *heat flow* operates in addition to those previously given:

At every time step, all bins $(m, 2N-1)$ at the top of the lattice have their heats $q_{(m,2N-1)}$ set to zero, and all bins $(m, 1)$ at the bottom of the lattice have their heats

$q_{(m,1)}$ set to q_0.

2.6.3 Whorl Absorption

There is no heat diffusion into the walls, but there is whorl diffusion into the walls. It is called *whorl absorption.*

In order to model fluid-wall friction, there is a parameter η_w that is the fraction of the whorls adjacent to the walls that is absorbed by the walls at each time step. For example, the whorl $w_{(r,2)}$ adjacent to the lower wall would have an amount $\Delta w = \eta_w w_{(r,2)}$ subtracted from it at every time step. Whorl absorption is exactly like whorl diffusion, except that the whorls are dissipated from the system into the walls.

In practice, the whorl absorption and whorl diffusion laws are performed simultaneously (see the source code `whorl_diffusion.c` in Section A.17).

2.6.4 Whorl Transport at Walls

Whorls cannot transport into the walls, since no whorls are defined on the walls. That is, all whorls are identically zero along the walls. Consider a corner $(r,2)$ or $(r,2N-2)$ adjacent to the wall. Whorl transport operates around $(r,2)$ and $(r,2N-2)$, except that there are only three whorls involved instead of four. Consider the whorl transport around the corner $(r,2)$:

$$\begin{aligned}
\beta_w\, w_{(r,2)}\, \frac{1}{2}\left[w_{(r+2,2)} + w_{(r,4)}\right] &\quad \text{moves from } (r+2,2) \quad \text{to } (r,4) \\
\beta_w\, w_{(r,2)}\, \frac{1}{2}\left[w_{(r,4)} + w_{(r-2,2)}\right] &\quad \text{moves from } (r,4) \quad \text{to } (r-2,2) \\
\beta_w\, w_{(r,2)}\, \frac{1}{2}\left[w_{(r-2,2)} + w_{(r+2,2)}\right] &\quad \text{moves from } (r-2,2) \quad \text{to } (r+2,2)
\end{aligned} \tag{2.13}$$

Note that all coefficients are the same as in (2.10), but only the three non-wall adjacent corners are used. The analogous thing happens in locations $(r,2N-2)$.

2.7 Initial Conditions

2.7.1 Step Function

Step function initial conditions are used in the relaxation version of the model. All whorls $w_{(r,s)}$ are set to zero. The heats $q_{(m,n)}$ in all bins are set to zero, except for the line of bins $(m,1)$ adjacent to the lower wall $y = 0$. The bins adjacent to the lower wall are set to:

$$q_{(m,1)} = q_0 \tag{2.14}$$

where q_0 has some positive value.

2.7.2 Static Gradient

Static gradient initial conditions (or simply "gradient initial conditions") are sometimes used in the steady-state version of the model. All whorls $w_{(r,s)}$ are set to zero (hence the "static"). The heats $q_{(m,n)}$ are set as follows:

$$q_{(m,n)} = q_0 \frac{2N + 1 - n}{2N} \tag{2.15}$$

(hence the "gradient") where q_0 has some positive value. This starts the model in a state that is close to the conduction steady-state.

2.7.3 Pattern-Inducing

Sometimes, in the steady-state version, the initial conditions are set so as to excite a particular convection pattern. When this is done, the initial conditions will be specified in the discussion (see Section 3.3).

2.8 "Thermodynamic" Quantities

Aside from the actual state of the model at any time, as specified by the heats $q_{(m,n)}$ and whorls $w_{(r,s)}$, there are a couple of more coarse-grained, global, macroscopic, or

"thermodynamic" quantities that describe the general condition of the model. These quantities are some of the "measurements" that I make on the model as it is running.

2.8.1 Total Heat

The total heat Q in the model is defined in the obvious way:

$$Q = \sum_{m,n \text{ odd}} q_{(m,n)} \tag{2.16}$$

Q is be time-independent in the relaxation version of the model and time-dependent in the steady-state version. Of course, when the steady-state version is actually in a steady state, Q is time-independent.

2.8.2 Heat Entropy

The heat entropy S is a measure of how evenly distributed the heat is in the bins:

$$S = -\sum_{m,n \text{ odd}} \frac{q_{(m,n)}}{Q} \ln\left(\frac{|q_{(m,n)}|}{Q}\right) \tag{2.17}$$

Where Q is defined by (2.16). The maximum possible value $S_{\max}$ of S is $\ln(MN)$, and $S = S_{\max} = \ln(MN)$ when the heat is evenly distributed across the bins.

It might be slightly misleading to use the name "entropy," since S in (2.17) does not measure the normal thermodynamic entropy of the fluid in the cell. However, (2.17) is based on a generalized entropy formula from information theory which is justified very well in Balian (1991, § 3.1). S is really a measure of the uniformity of the heat distribution in the lattice.

2.8.3 Heat Flux

In the steady-state version of the model, there is heat flux in and out of the convection cell. This heat flux is a result of the heat flow law described in Section 2.6.2. The heat flux $(\Delta Q)_H$ entering the cell on the hot side (the bottom) at each time step is

given by:

$$(\Delta Q)_H = \sum_{m \text{ odd}} (q_0 - q_{(m,1)}) \tag{2.18}$$

While the heat flux entering the cell on the cold side (the top) is:

$$(\Delta Q)_C = \sum_{m \text{ odd}} (0 - q_{(m,2N-1)}) \tag{2.19}$$

Note that usually $(\Delta Q)_H$ will be positive while $(\Delta Q)_C$ will be negative.

When the steady-state version of the model has actually reached a steady state,

$$(\Delta Q)_H = -(\Delta Q)_C \tag{2.20}$$

2.9 Summary

Table 2.4 shows the laws that are used in the model. Table 2.5 shows the symbols that I use in discussing the model.

law name	parameter	section
heat diffusion	α_q	2.5.1
heat transport	β_q	2.5.2
whorl diffusion	α_w	2.5.3
whorl absorption at walls	η_w	2.6.3
whorl transport	β_w	2.5.4
gravity	γ	2.5.5
randomization	σ_w	2.5.6
heat flow (steady-state only)	q_0	2.6.2

Table 2.4: Laws in the model. At each time step, the laws in this table are applied to the lattice in the order shown. The only exceptions are whorl absorption, which actually takes place at the same time as whorl diffusion, and heat flow, which only acts in the steady-state version of the model.

symbol	name	section
M	number of bins, x-direction	2.1
N	number of bins, y-direction	2.1
(x, y)	lattice location	2.1
m, n	odd indices	2.1
r, s	even indices	2.1
$q_{(m,n)}$	heat (temperature) in bin at (m, n)	2.2
$p_{x(r,n)}$	x-momentum at (r, n)	2.3
$p_{y(m,s)}$	y-momentum at (m, s)	2.3
$w_{(r,s)}$	whorl at (r, s)	2.4
α_q	heat diffusion parameter	2.5.1
β_q	heat transport parameter	2.5.2
α_w	whorl diffusion parameter	2.5.3
β_w	whorl transport parameter	2.5.4
γ	gravitational parameter	2.5.5
σ_w	standard deviation of random whorls	2.5.6
q_0	lower wall heat value	2.6.2
η_w	whorl absorption at walls	2.6.3
Q	total heat	2.8.1
S	heat entropy	2.8.2
$S_{\max}$	maximum possible heat entropy	2.8.2
$(\Delta Q)_H$	heat flux from hot resevoir	2.8.3
$(\Delta Q)_C$	heat flux from cold resevoir	2.8.3

Table 2.5: Symbols used in the model discussion.

Chapter 3

Results

Roughly, the experiments that I performed with the model fall into two categories: tests to see that the model behaves in a qualitatively reasonable way, and actual results that relate to my objectives.

The symbols that I use in this chapter will be the same as those used in Chapter 2. They are summarized in Table 2.5.

3.1 Testing the Model

All of the tests of the model's qualitative accuracy were performed in the steady-state version (Section 2.6.2). This is because there is a great deal that is known about steady-state convection and I was able to use some of this knowledge to check the behaviour of the model.

3.1.1 Critical Rayleigh Number

Because the model does not represent an attempt to simulate a convection cell exactly, I have not attempted to calculate a Rayleigh number in terms of the model parameters.

However, it is true that for any setting of the other parameters, there is a critical value q_c of the lower wall heat value q_0, such that for $q_0 < q_c$ there is no convection and for $q_0 > q_c$ there is convection. This is illustrated in Figure 3-1. Of course q_c

is a function of the other parameters. This is analogous to the critical temperature difference $(\Delta T)_c$ mentioned in Section 1.3.

It is also true that for any setting of the other parameters, there is a critical value α_{wc} of the whorl diffusion parameter α_w such that convection only occurs when $\alpha_w < \alpha_{wc}$. Again, α_{wc} is a function of the other parameter settings. This is analogous to the critical fluid viscosity under which convection sets in for real convection cells.

The existence of these critical values does not show that the model is quantitatively accurate, but it does show that it has some of the important qualitative features of a convection cell.

3.1.2 Wavelength of Stable Convection Pattern

As I mentioned in Chapter 1, convecting rolls have a characteristic aspect ratio. If the separation between the top and bottom walls of the convection cell is d, rolls that develop have cross-sectional size $d \times d$. Actually, the rolls that develop adjacent to one another rotate in opposite directions, so that the repeated pattern in a convection cell is a pair of rolls, one rotating clockwise and the other counter-clockwise. The aspect ratio of this basic pattern is 2. For this reason, Bergé, Pomeau, and Vidal (1984, p. 88) suggest making convection cells with cross-sectional aspect ratio 2.

I initially ran the model with aspect ratio 2 ($M \times N = 20 \times 10$) in order to see if the same type of convection pattern appears. Figure 3-1 shows that the convection pattern that sets in does have aspect ratio 2, and it consists of two counter-rotating rolls.

When the lattice has aspect ratio 2, the aspect ratio of the convection pattern is fairly independent of the parameters. That is, so long as the parameters allow convection, the convection pattern usually has aspect ratio 2. Roughly, this is because the size of the cell and the circular boundary conditions "force" a pattern of aspect ratio 2. This is not true for higher aspect ratios.

For example, if $\eta_w = 0$, stable convection on a lattice of aspect ratio 4 has aspect ratio 4, not 2. That is, only one double-roll pattern forms, not two. On a lattice of aspect ratio 6, the stable pattern has aspect ratio 6, not 2. On the other hand, if η_w

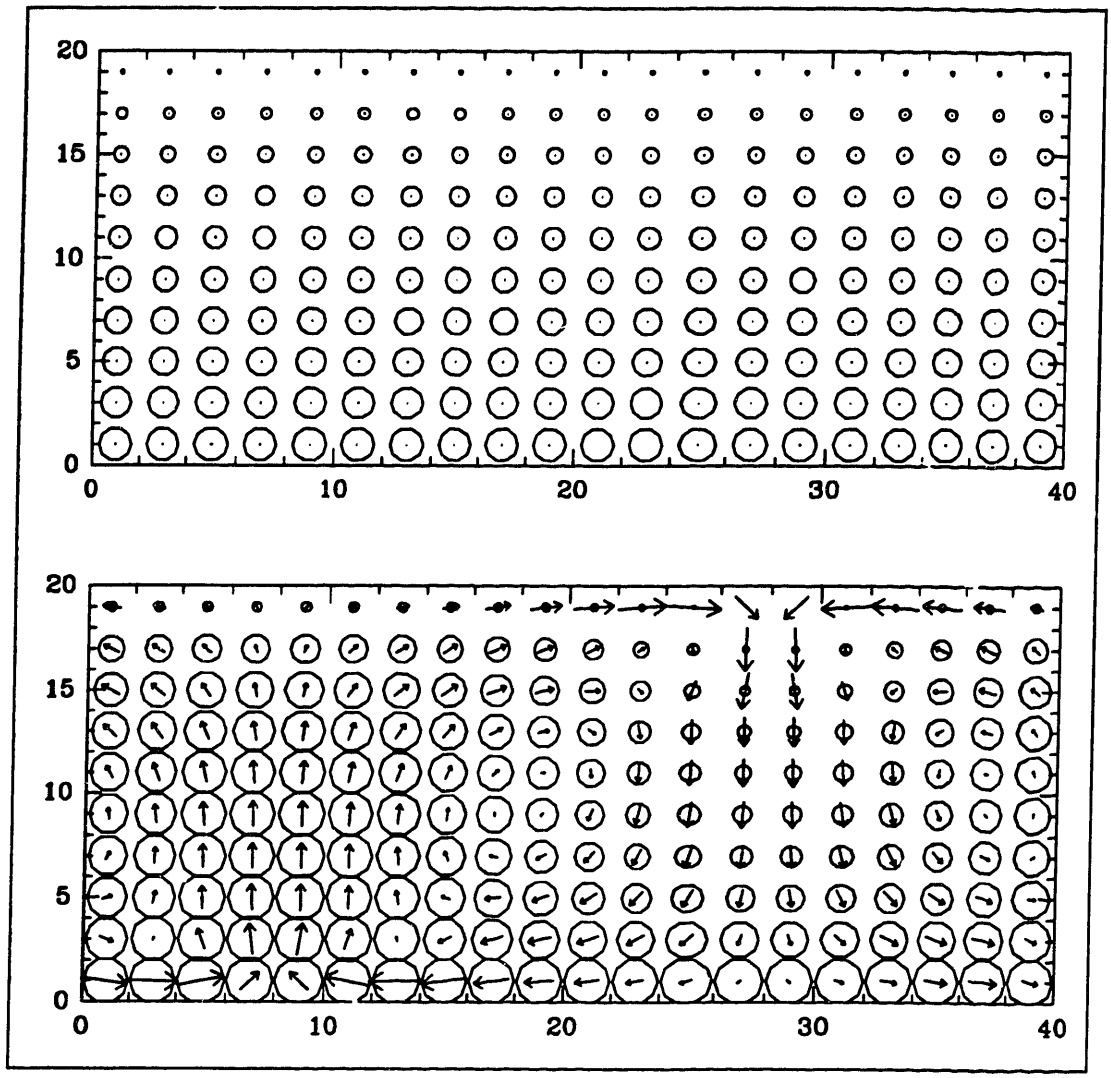

Figure 3-1: The top picture shows the output from the steady-state version of the model with $q_0 = 1.5$ (5000 time steps), and the bottom picture shows the output with $q_0 = 3.0$ (10000 time steps), establishing the existence of a critical value q_c for convection, $1.5 < q_c < 3.0$. Later tests showed $1.5 < q_c < 1.75$ (see Figure 3-5). Other parameters are set as per equation (3.1), $\sigma_w = 0.01$, $M \times N = 20 \times 10$ (remember that the lattice dimensions are actually $2M \times 2N$). The circle size represents the amount of heat $q_{(m,n)}$ at each location (m, n), and the arrow lengths and directions indicate magnitude and direction of fluid momentum.

is large enough, the stable pattern on a lattice of aspect ratio 4 might have aspect ratio 1.33 (*i. e.*, 3 double-roll patterns form) while on a lattice with aspect ratio 6 the stable pattern might have aspect ratio 1.5 (*i. e.*, 4 patterns form). I decided to fix the parameters of the model so that the aspect ratio of the convection patterns comes out to be 2 on all lattices of even aspect ratio.

I ran tests on lattices of even aspect ratios up to aspect ratio 8. The parameters that I ended up choosing were:

$$\alpha_q = \beta_q = \alpha_w = \beta_w = \gamma = 0.1, \quad \eta_w = 0.025 \tag{3.1}$$

I used these parameters in most of my experiments.

Figure 3-2 shows the results of running the model at various aspect ratios with these parameters and $q_0 = 3.0$. The convection patterns all have aspect ratio 2. Actually, later testing showed that even with these parameters, when I ran the aspect ratio 8 lattice with different random number sequences, sometimes 5 patterns with aspect ratio 8/5 developed instead of 4 patterns with aspect ratio 2. However, when the model was run with $q_0 = 2.0$, much closer to q_c ($1.5 < q_c < 1.75$), the aspect ratio 8 lattice always developed 4 patterns. I will return to this phenomenon later (Section 3.3.2).

The parameters (3.1) that I chose are such that the stable convection patterns have the same aspect ratio as real convection patterns. This suggests that the parameters are physically realistic, and that the model is qualitatively accurate.

3.1.3 Further Tests

There are many further tests that should be performed on the model that lack of time prevented me from performing. For instance, tests should be made on the size of the lattice: are there artifacts from the lattice discreteness that disappear as the number of lattice points increases? If so, what is the smallest lattice such that the lattice artifacts are negligible?

Another test would be to change the order in which the rules (heat diffusion, heat

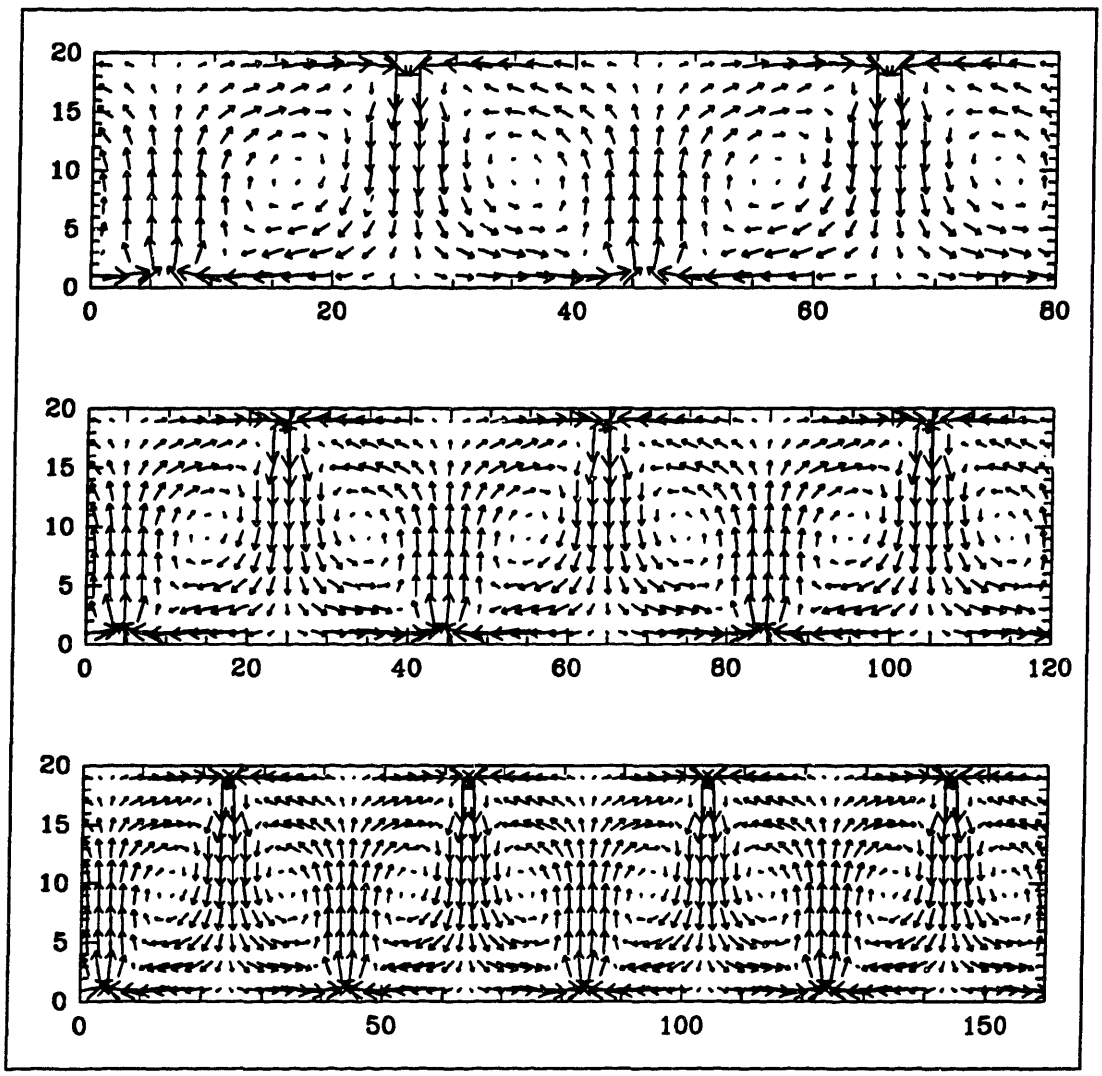

Figure 3-2: These three diagrams show convection patterns for lattices of aspect ratios 4 ($M \times N = 40 \times 10$), 6 ($60 \times 10$), and 8 ($80 \times 10$), after 10000 time steps, using the parameters in (3.1) and $\sigma_w = 0.01$. Remember that the lattice has dimensions $2M \times 2N$. Note that the diagrams have been horizontally compressed to fit into the available space, and the heats (circles) have been ommited for clarity. One can see that in each case the (uncompressed) convection pattern is of aspect ratio 2: in the top diagram there are 2 patterns, in the middle diagram there are 3 patterns, in the bottom diagram there are 4 patterns.

transport, *etc.*) are executed. In all the experiments described in this thesis, the laws were executed in the order given in Table 2.4. Ideally, the results should not depend strongly on the order of execution of the rules.

Also, all of the simulations were run in a very small volume of "parameter space." That is, I did not experiment much with varying the parameters. This should be done extensively to see if there are any anomalies, and in order to help choose the most physically realistic set of parameters.

Finally, I should have taken the continuum limit of our discrete equations. In the continuum limit, I would like to get the equations of motion for convection in the Boussinesq approximation. This would be a check to see if the model really does simulate a convection cell. It was not my intention to model a convection cell precisely, but the "accuracy" of the model is still an interesting and important question.

3.2 Results in the Relaxation Version

As I explained in Section 2.6.1, the relaxation version allows observation of a system relaxing to equilibrium. The system modelled is that of a fluid cell in a gravitational field with non-heat-conducting walls. It starts with hot water on the bottom, colder water above. These initial conditions are modelled with step function initial conditions (Section 2.7.1): all bins (m, n) have heat $q_{(m,n)} = 0$ except the bins $(m, 1)$ along the bottom of the cell which have $q_{(m,1)} = q_0$. All whorls $w_{(r,s)}$ are set to zero.

As I conjectured in Section 1.4, above a critical value of q_0 there should be two qualitatively distinct paths to equilibrium, one that is very symmetrical and conduction (heat diffusion) dominated, and one that is asymmetrical and mixing (heat transport) dominated. Furthermore, according to the principle of maximum entropy production, when q_0 is close to its critical value q_c, the stable path will be the one that maximizes the rate of entropy production.

Figure 3-3 illustrates the bifurcation in the relaxation behaviour as a function of the lower wall heat q_0. When the lower wall heat is small enough, there is only one path to equilibrium that is a solution to the equations of motion. This is the

conduction-dominated, symmetrical path that involves no fluid motion. When q_0 is above a critical value q_c ($q_c \approx 63$ for the parameters in (3.1)), there are two paths, one stable and one unstable. The unstable path is the conduction-dominated path, which involves no fluid motion. It can be produced by setting $\sigma_w = 0$ because when there is no randomization, there are no fluctuations which can be amplified into mixing. The stable path, achieved when $\sigma > 0$, is mixing dominated. It involves a magnification of the small fluid motion fluctuations from the randomization, and the subsequent mixing speeds the approach to equilibrium. Figure 3-4 compares the time evolution of the heat entropy for a run with and a run without randomization.

In Figures 3-3 and 3-4, the comparison between the two paths is made in terms of the heat entropy S (Section 2.8.2), not in terms of the real entropy of the fluid cell that is being modelled (if such an entropy exists for a non-equilibrium system; see Section 3.5.2). The heat entropy is only an indicator of how close the system is to equilibrium, and I am only using it to indicate when the system has reached equilibrium. The same total amount of "real" entropy is produced along both the conduction-dominated and mixing-dominated paths, since they both start and end in identical states. Therefore, the path which gets to equilibrium fastest is the one that produces entropy at the greatest rate.

In the region of the bifurcation to mixing-dominated behaviour (*i.e.*, with q_0 greater than, but close to, q_c) the stable path gets the system to equilibrium more quickly. Therefore the stable path is the path that maximizes the rate of entropy production, and the principle of maximum entropy production is confirmed.

3.3 Results in the Steady-State Version

In the steady-state version, the entropy that I will discuss is not the heat entropy, but rather the entropy produced as the convection cell transports heat from a hot resevoir to a cold resevoir. In all cases, I will be comparing systems with identical lower wall heats q_0, and I will be comparing them when they have reached their steady-state patterns, and so the entropy production comparison will simply be a comparison of

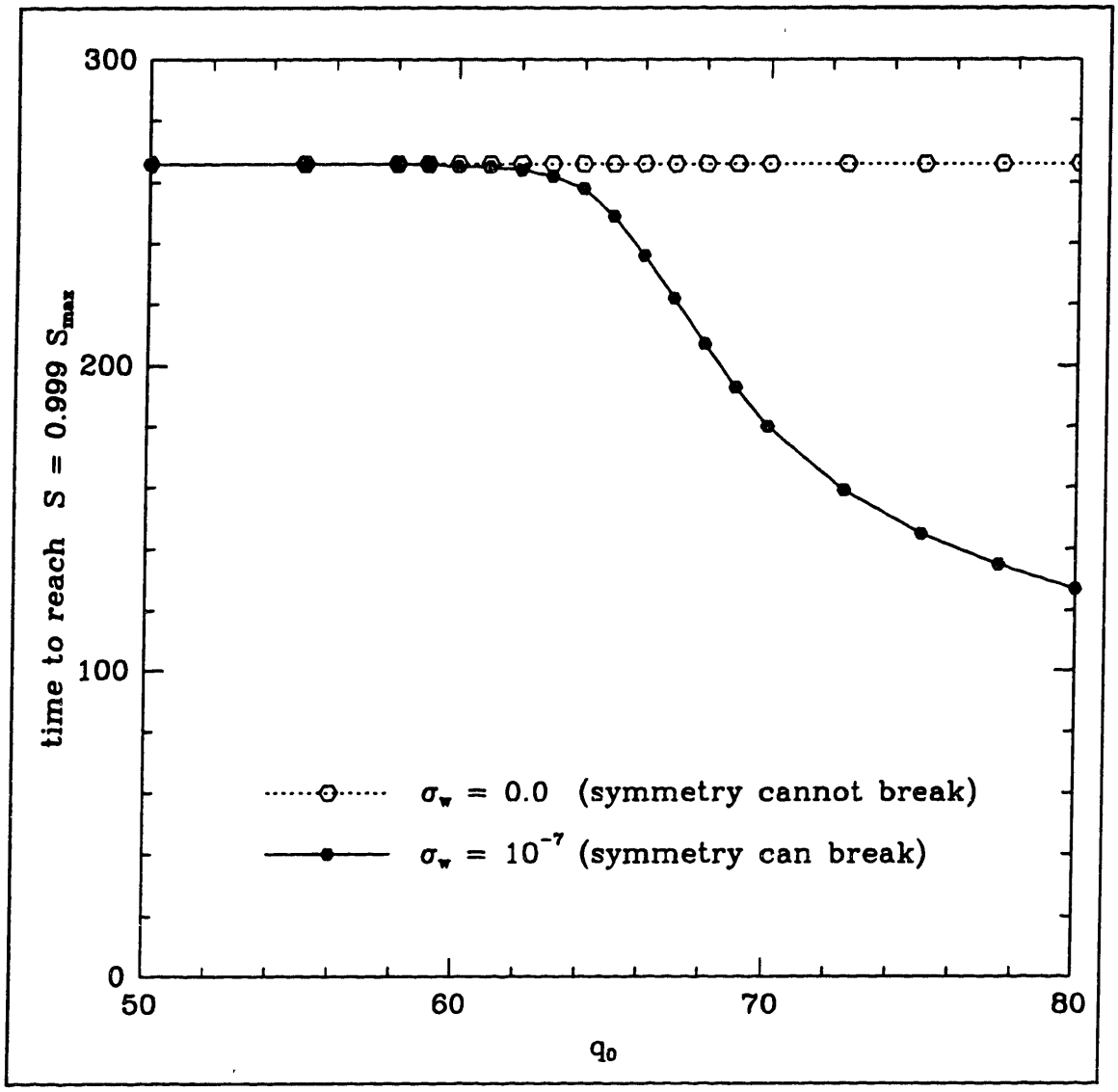

Figure 3-3: This graph is a comparison of the behaviour of the relaxation version with and without randomization for varying q_0. It plots the time it takes the system to get to a heat distribution with heat entropy $S = 0.999 S_{\max}$ (Section 2.8.2) as a function of q_0. Parameters are set as per (3.1). When there is no randomization ($\sigma_w = 0$), the only path available for the system is the symmetrical path with no fluid motion. When randomization is introduced (here $\sigma_w = 10^{-7}$), the symmetry can break, and mixing can speed the approach to equilibrium, as long as q_0 is above the critical value (here about 63). The stable path is the path chosen when randomization is introduced, and this is also the path that gets the system to equilibrium the most quickly. In both cases the total entropy production is the same, so the stable path is the one with the maximum rate of entropy production.

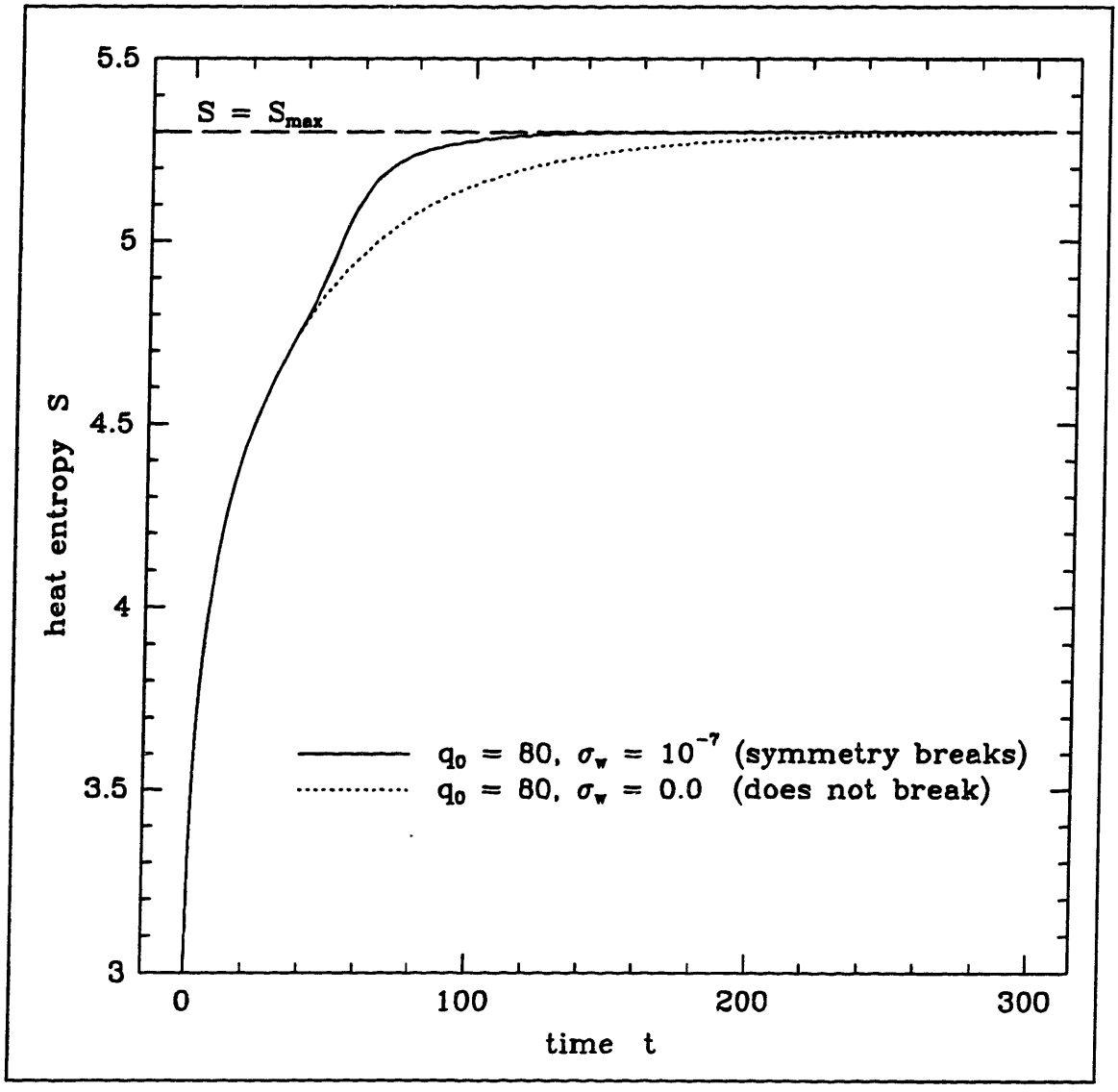

Figure 3-4: This graph is a comparison of the time evolution of the heat entropy in the relaxation version for runs with and without randomization for $q_0 = 80$ (recall that $q_c \approx 63$). Other parameters are set as per (3.1). Clearly the stable path (the one with $\sigma_w > 0$) produces heat entropy at a greater rate. See the text for a discussion of how this relates to the real entropy of the system.

the heat fluxes $(\Delta Q)_H$.

3.3.1 20 × 10 Lattice

Consider the 20 × 10 lattice originally tested in Section 3.1.1. When the parameters are set as in (3.1) and $q_0 = 3.0$, the stable steady-state behaviour mode is convection with a single convection pattern of aspect ratio 2. However, if the system starts with static gradient initial conditions (Section 2.7.2), and $\sigma_w = 0$, then there are no fluid-motion fluctuations in the system to allow it to develop convection patterns. In this discussion, I will sometimes refer to static gradient initial conditions as *zero whorl initial conditions.* With zero-whorl initial conditions and $\sigma_w = 0$, the system is "forced" into the conduction mode, even when only convection is stable.

Furthermore, although convection with a single pattern of aspect ratio 2 is stable, it is possible to excite the system into a convection mode that has two patterns, each with aspect ratio 1. To do this, I used pattern-inducing initial conditions (Section 2.7.3). The initial state was as follows: the heat distribution started as in gradient initial conditions (Section 2.7.2), but the whorls were not all set to zero. The whorls $w_{(r,s)}$ were set to:

$$\begin{aligned} w_{(r,s)} = &\quad -0.5 \quad \text{for } r = 2,4,6,8 \\ = &\quad 0.5, \quad r = 12,14,16,18 \\ = &\quad -0.5, \quad r = 22,24,26,28 \\ = &\quad 0.5, \quad r = 32,34,36,38 \\ = &\quad 0, \quad r = 0,10,20,30 \end{aligned} \tag{3.2}$$

I call these initial conditions *2-pattern pattern-inducing initial conditions.* Initial conditions (3.2) effectively start the system with a large fluctuation that is the right wavelength to excite convection in two patterns. When the model is run with initial conditions (3.2), q_0 greater than the appropriate critical value, and $\sigma_w = 0$, the model is forced into the two-pattern steady-state mode, even though only the one-pattern

mode is stable. The convection mode with two patterns does not die out in time, so it is a steady-state solution to the equations of motion, albeit an unstable solution.

Ordinarily, 1-pattern convection is triggered with static gradient initial conditions, $q_0 > q_c$, and $\sigma > 0$. However, in order to be able to make comparisons with $\sigma = 0$, I created *1-pattern pattern-inducing initial conditions* as well:

$$\begin{aligned} w_{(r,s)} &= -0.5 \quad \text{for } 2 \leq r \leq 18 \\ &= 0.5, \quad 22 \leq r \leq 38 \\ &= 0, \quad r = 0, 20 \end{aligned} \tag{3.3}$$

These act the same way as 2-pattern conditions, except that they excite the 1-pattern convection mode.

Using pattern-inducing initial conditions and $\sigma_w = 0$ to force the system into particular modes allowed comparison of the entropy production (heat flux $(\Delta Q)_H$) of unstable modes with the entropy production of stable modes. As Figure 3-5 shows, the stable mode is the one that produces the most entropy. Figure 3-6 shows the three modes that were compared. Of course, Figure 3-5 also shows that the two convection modes tested (1-pattern and 2-pattern) have different critical values q_c at which they become solutions to the equations of motion.

Figure 3-5 shows that the stable convection aspect ratio is the one that maximizes entropy production. However, it also shows that the two different aspect ratio convection patterns have different bifurcation points. 1-pattern convection becomes a solution to the equations of motion at q_c where $1.5 < q_c < 1.75$, while 2-pattern convection only becomes a solution at q_c' where $2.5 < q_c' < 2.75$. This means that the principle of maximum entropy production, as formulated in Section 1.2, might not really apply to the selection of convection pattern size. It is true that, at the bifurcation to 1-pattern convection (q_0 near q_c), the system chooses the branch that maximizes entropy production, but this is only how it chooses between 1-pattern convection and conduction, not how it chooses between 1- and 2-pattern convection when $q_0 > q_c'$.

The principle of maximum entropy production can be applied to the bifurcation

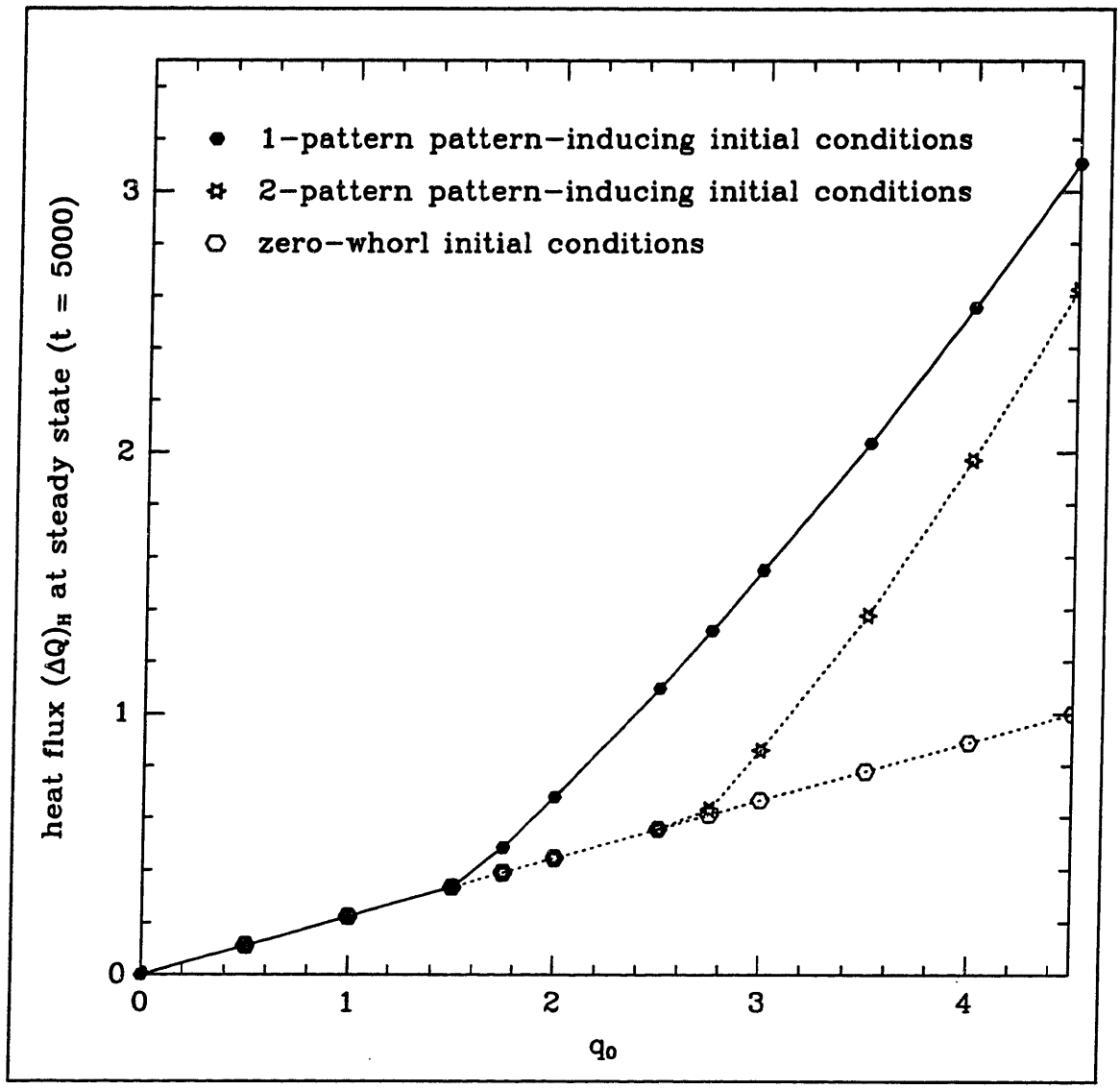

Figure 3-5: This figure compares heat fluxes $(\Delta Q)_H$ at time step 5000 in the steady-state version for three different initial conditions on a 20×10 lattice as a function of q_0. Initial conditions are described in the text in detail (see Section 2.7 and equations (3.2) and (3.3)). Roughly: the 1-pattern initial conditions (3.3) excite 1-pattern convection if such convection is a solution to the equations of motion; 2-pattern conditions (3.2) excite 2-pattern convection if it is a solution; and zero-whorl conditions (actually static gradient conditions, see Section 2.7.2 and the text) excite only the conduction mode. Pictures of the three behaviour modes are shown in Figure 3-6. Parameters are set as per (3.1) and $\sigma_w = 0.0$. Here the stable branch is indicated by a solid line, unstable behaviours are indicated with dotted lines. The bifurcation points for 1- and 2-pattern convection are different.

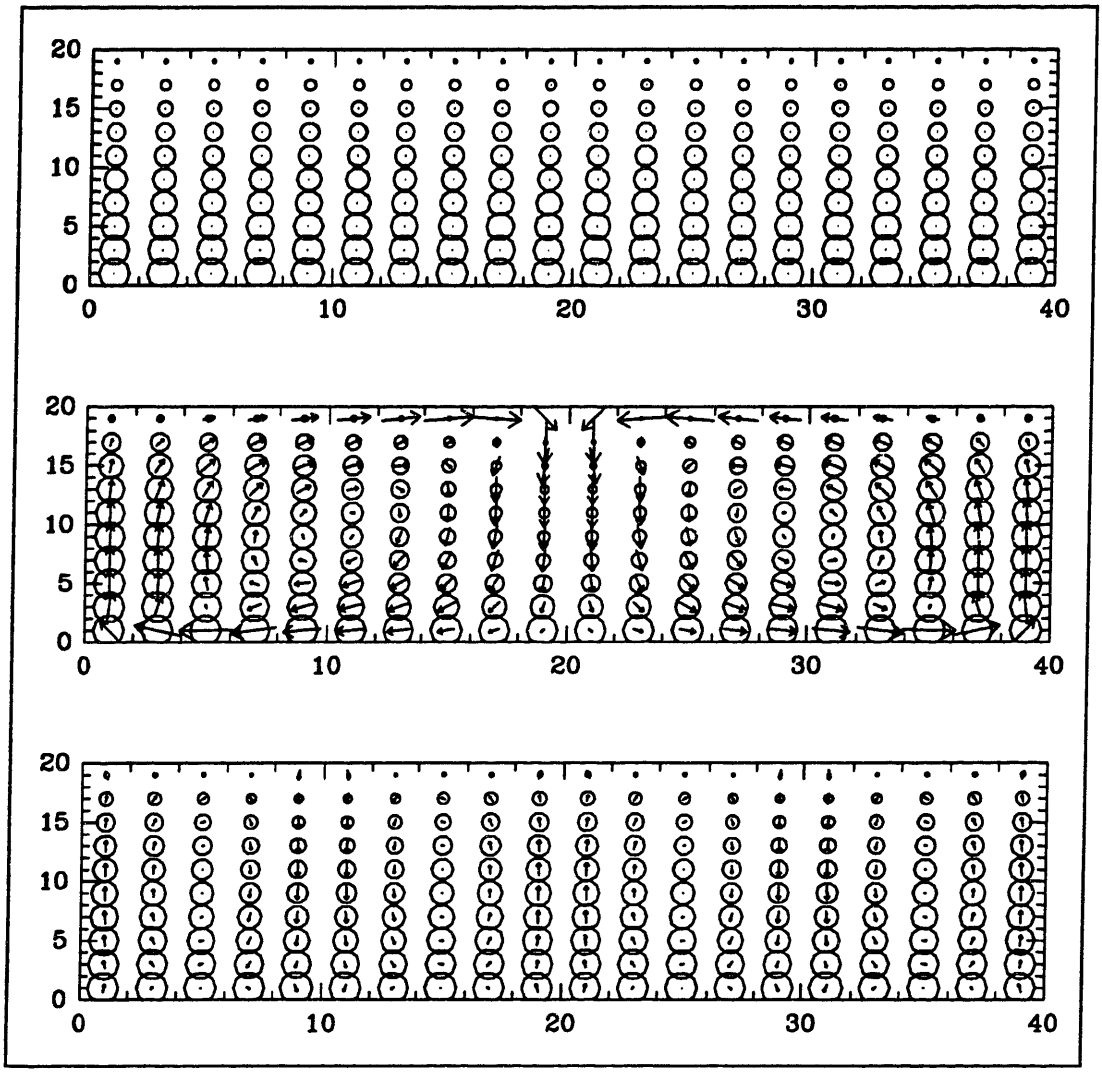

Figure 3-6: These three pictures show the three steady-state behaviour modes compared in Figure 3-5. The top picture shows conduction, the middle shows convection with one pattern, the bottom shows convection with two patterns. In each case $M \times N = 20 \times 10$, but note that the diagrams have been slightly compressed to fit in the available space and remember that the lattice has dimensions $2M \times 2N$. Parameters are set as per (3.1), $q_0 = 3.0$, and $\sigma_w = 0$. Initial conditions are zero-whorl (static gradient) at the top, 1-pattern pattern-inducing in the middle, and 2-pattern pattern-inducing at the bottom (see the text). The middle picture depicts the stable mode (convection, one pattern).

shown in Figure 3-5 between conduction and 2-pattern convection ($q_0 = q_c'$), even though neither of these steady-state modes is actually stable at this bifurcation. They can be compared by running the model with a modified randomization law. If the randomization law acts so that the randomness added to the whorl $w_{(r,s)}$ is always equal to the randomness added to the whorl $w_{(r+M,s)}$ (remember that the lattice has dimensions $2M \times 2N$), then only behaviour patterns with a certain symmetry can be excited: patterns that are repeated twice side by side. This symmetry constraint rules out 1-pattern convection, but allows for conduction and 2-pattern convection. In fact, because the lattice has periodic boundary conditions in the x-direction, when the randomization law is changed in this way, the 20×10 lattice becomes mathematically identical to a 10×10 lattice with normal randomization. I found that a 20×10 lattice run with this randomization always chooses 2-pattern convection over conduction if $\sigma_w > 0$ and $q_0 > q_c'$. Again (see Figure 3-5) the system is choosing the available steady-state mode that maximizes entropy production.

These tests show that near each bifurcation that I investigated, the 20×10 system chooses the available behaviour that maximizes entropy production.

3.3.2 80 × 10 Lattice

Recall that the aspect ratio 8 lattice ($M \times N = 80 \times 10$, parameters set as per (3.1), $q_0 = 3$, and $\sigma_w = 10^{-2}$) sometimes developed 4 convection patterns, each of aspect ratio 2, and sometimes developed 5, each of ratio 8/5. In a manner similar to that in which 1- and 2-pattern pattern-inducing initial conditions were constructed on the 20×10 lattice, 4- and 5-pattern pattern-inducing initial conditions can be constructed on the 80×10 lattice. 4-pattern conditions are just those given in (3.3) repeated 4 times in a row, and 5-pattern are the same, but with 5 repeats, and a basic length of 32 lattice units, not 40. By using the parameters given in (3.1) and $\sigma = 0$, I was able to compare the heat fluxes $(\Delta Q)_H$ of the two patterns for various values of q_0, independent of their relative stability.

Figure 3-7 shows the bifurcation diagram for the 80×10 lattice, comparing the heat fluxes for conduction, 4-pattern convection, and 5-pattern convection. The bifurcation

points for 4- and 5-pattern convection are very close, but they are not identical. For $q_c < q_0 < 2.0$, only the 4-pattern mode is stable. For $q_0 > 3.0$, the mode that is selected depends on the sequence of random numbers used by the randomization function. That is, for $q_0 > 3.0$, both 4-pattern and 5-pattern convection modes are stable. I did not find the point between 2.0 and 3.0 at which both modes become stable.

The model with an 80×10 lattice is an example of a system which only chooses the branch which maximizes entropy production close to the bifurcation point. For example, $q_0 = 3.0$ is far enough from q_c ($1.55 < q_c < 1.60$) such that 5-pattern convection, despite producing less entropy than 4-pattern convection, is a stable steady-state mode.

These tests show that near the bifurcation that I investigated, the 80×10 system also chooses the available behaviour that maximizes entropy production. However, the system does not have to be far from the bifurcation for maximum entropy production to fail as a pattern-selection rule. This is the kind of situation that motivated putting the "near the bifurcation" condition into the principle of maximum entropy production. Of course, it may be that every bifurcation I have studied has the property that far enough from it, the stable mode is no longer the one with the maximum rate of entropy production. However, it is only this bifurcation at which I actually observed this change in behaviour.

3.4 Summary

I have created a simple, discrete, two-dimensional, computer model of a convection cell, described in Chapter 2. I tested this model and found that given the right parameters, it is qualitatively accurate. I used this model to test the principle of maximum entropy production that I formulated in Section 1.2. This principle states that near to a bifurcation in the behaviour of a non-equilibrium statistical system, the stable branch is the one on which entropy is produced at the greatest rate.

In all the tests that I performed with the model, the principle of maximum entropy

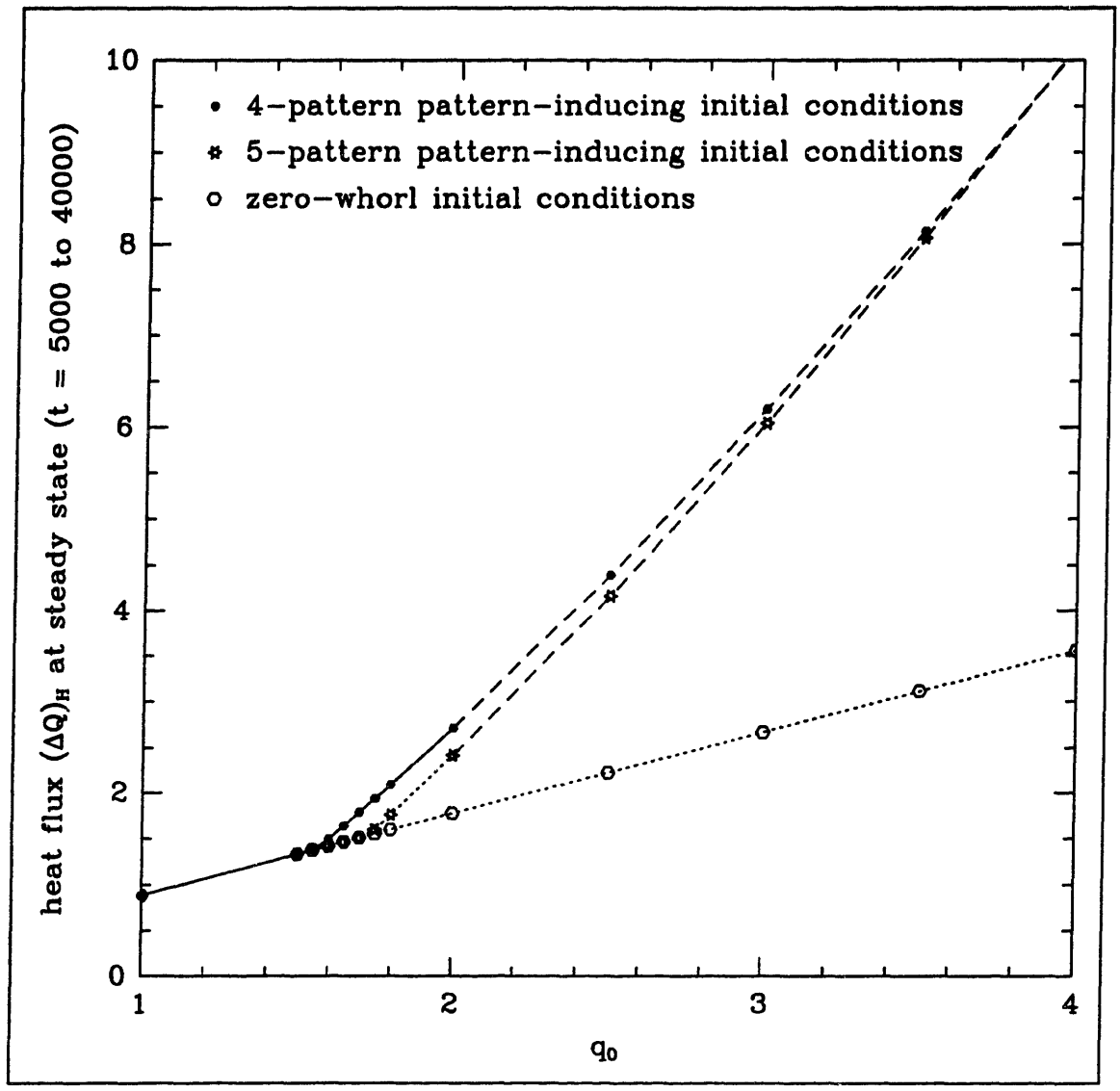

Figure 3-7: This figure compares heat fluxes $(\Delta Q)_H$ in the steady-state version for three different initial conditions on a 80×10 lattice as a function of q_0. Values for $(\Delta Q)_H$ were taken at $t = 5000$ except when $1.55 \leq q_0 \leq 1.80$, for which $t = 5000$ to $t = 40000$, depending on how long it took the model to reach a steady-state to 5 digits of accuracy. Initial conditions are described in the text. Roughly: the 4-pattern initial conditions excite 4-pattern convection if such convection is a solution to the equations of motion; 5-pattern conditions excite 5-pattern convection if it is a solution; and zero-whorl (static gradient) conditions excite only the conduction mode. Parameters are set as per (3.1) and $\sigma_w = 0.0$. Here the stable branch is shown with a solid line until $q_0 = 2.0$, beyond which the stability is unknown for $2.0 < q_0 < 3.0$ and shared for $q_0 > 3.0$ (see text).

production was confirmed. When the relaxation version of the model was used to simulate a fluid cell in a gravitational field with a non-uniform initial heat distribution, it was found that near the bifurcation to mixing-dominated behaviour, the system takes the path to equilibrium that maximizes entropy production: the mixing path. When the model was used to simulate a convection cell with aspect ratio 2, it was found that near the bifurcation to convection, the steady-state mode that is stable is the one that produces entropy at the greatest rate: convection with patterns of aspect ratio 2. Conduction, and convection patterns with aspect ratio 1, both of which had lower entropy production, are unstable behaviours. When the model was used to simulate a convection cell with aspect ratio 8, it was found that convection patterns of aspect ratio 2 maximize entropy production, and that they are the only stable mode very close to the bifurcation to convection.

3.5 Areas for Further Study

This thesis represents only the beginning of what could be a very interesting and useful research programme into the principle of maximum entropy production. Also, I believe that the convection cell model described in this thesis is new and quite unusual, and much more work could be done to test and improve it. I discussed some of the possibilites for future work on the model in Section 3.1.3. In this section I will suggest some avenues of further research for the principle of maximum entropy production.

3.5.1 More Interesting Bifurcations

The tests presented in this thesis were performed on a set of very similar bifurcations. The bifurcations were all supercritical, in that in every case, below the critical value the behaviour of the system approached a static mode, while beyond the critical value the stable mode involved periodic fluid motion (convection). It would be interesting to see if the principle of maximum entropy production can also be used to analyze subcritical bifurcations, in which the mode involving fluid motion is unstable, and

exists only below the critical value (Tabor 1989, pp. 197–199). Furthermore, every bifurcation I studied had only two branches, one stable and one unstable. It would be nice to test out the principle of maximum entropy production on bifurcations with many branches, as it is in these situations that the principle would be most useful.

3.5.2 Defining Entropies

One problem with the principle of maximum entropy production as it now stands is that it does not give a definition of the appropriate entropy to use in comparing the branches of the bifurcation. In fact, some physicists believe that entropy is not defined at all for non-equilibrium systems. If entropy is taken to be the logarithm of the volume of the accesible phase space of the system, then it is not defined for a system following a trajectory towards equilbirium. After all, a single trajectory does not represent any volume in phase space at all. For this reason, if entropy is defined outside of equilibrium, it must be something like Boltzmann's (1896) function H or an entropy from information theory that relates to the amount of information needed to specify the state of the system.

In the model, there are several possible entropies which might be relevant to the principle of maximum entropy production. There is the heat entropy introduced in Section 2.8.2, which is an information-theoretic entropy that is really a measure of the uniformity of the heat distribution. There is also an analogous whorl entropy. There is the actual entropy of the real system that is being simulated, if, as discussed above, non-equilibrium systems have entropies. And there may be another entropy that corresponds to the amount of information that is being thrown away when the actions of the various rules cancel each other out. It is not clear how these entropies are related, and probably this must be understood before it will be possible to formulate the principle of maximum entropy production in a clear and general way.

3.5.3 Theoretical Justification of the Principle of Maximum Entropy Production

In order to really establish the principle of maximum entropy production, an argument that it is true is required. I have shown that the principle makes good predictions for a small set of questions about the model presented in this thesis, and this endeavour is useful. This type of project could be repeated with different systems or models or even with different questions about this model. However, if the principle is true in general, then there must be some general argument that can be used to justify it. This is a very important research project, because if the principle is true and can be justified, it will be one of the first general principles ever found in the study of non-equilibrium statistical mechanics.

References

Balian, R. (1991). *From Microphysics to Macrophysics: Methods and Applications of Statistical Physics.* Ter Harr, D., and Gregg, J. F. translators. Springer-Verlag, New York, 1991.

Bénard, H. (1901). Les tourbillons cellulaires dans une nappe liquide transportant de la chaleur par convection en régime permanent. *Annales de Chimie et de Physique,* series 7, **23** 62–144.

Bergé, P., Pomeau, Y., and Vidal, C. (1984). *Order within chaos: towards a deterministic approach to turbulence.* Tuckerman, L. translator. John Wiley & Sons, Inc., New York NY, 1984.

Berry, M. V. (1992). Personal communication, 7 April 1992.

Boltzmann, L. (1896). *Lectures on Gas Theory.* Brush, S. G. translator. University of California Press, Berkeley and Los Angeles CA, 1964.

Clune, T., and Knobloch, E. (1991). Square pattern convection in binary fluids with experimental boundary conditions. *Physical Review A* **44** 8084–8090.

Dafermos, C. M. (1973). The Entropy Rate Admissibility Criterion for Solutions of Hyperbolic Conservation Laws. *Journal of Differential Equations* **14**, 202–212.

Dafermos, C. M. (1984). Discontinuous Thermokinetic Processes. In Truesdell, C. *Rational Thermodynamics*, 2 ed. Springer-Verlag, New York, 1984, appendix 4B, pp. 211–218.

Knobloch, E. (1989). Pattern selection in binary-fluid convection at positive separation ratios. *Physical Review A* **40** 1549–1559.

Knobloch, E. (1992). Personal communication, 10 April 1992.

Landau, L. D., and Lifshitz, E. M. (1959). *Fluid Mechanics.* Sykes, J. B., and Reid, J. H. translators. Pergamon Press, London, 1959.

Libchaber, A. (1992). Personal communication, 9 April 1992.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1988). *Numerical Recipes in C.* Cambridge University Press, Cambridge, 1988.

Prigogine, I. (1955). *Introduction to Thermodynamics of Irreversible Processes.* Charles C. Thomas publisher, Springfield IL, 1955.

Prigogine, I. (1980). *From being to becoming: time and complexity in the physical sciences.* W. H. Freeman and Co., San Francisco CA, 1980.

Rayleigh, Lord J. W. S. (1916). On Convection Currents in a Horizontal Layer of Fluid, when the Higher Temperature is on the Under Side. *Philosophical Magazine,* series 6, **32** 529–546.

Silber, M., and Knobloch, E. (1988). Pattern selection in steady binary-fluid convection. *Physical Review A* **38** 1468–1477.

Swenson, R., and Turvey, M. T. (1991). Thermodynamic Reasons for Perception-Action Cycles. *Ecological Psychology* **3**(4) 317–348.

Tabor, M. (1989). *Chaos and integrability in nonlinear dynamics: an introduction.* John Wiley & Sons, Inc., New York NY, 1989.

Velarde, M. G., and Normand, C. (1980). Convection. *Scientific American* **234** (1) 92–108.

Appendix A

The Source Code

The following pages show the computer files containing the C source code actually used to run the simulations. Although the programmes are not necessarily all that clear, the emphasis in writing them was on clarity, not computational speed. I am certain that they could be greatly speeded up with rewriting.

The functions are all written to operate on a matrix that is isomorphic to the lattice discussed in section 2.1. That is, points with odd coördinates are bins, points with even coördinates are corners, and all other points are edges (see Figure 2-1 and Table 2.1). The single lattice contains heats, whorls, and edge-crossing momenta.

The code listed here handles only gradient and step function initial conditions, not pattern-inducing ones (see Section 2.7). Experiments with pattern-inducing initial conditions were performed using modified versions of this code.

This appendix starts with the makefile, the header file, and the main function and then lists all the other functions in alphabetical order. The only files used but not given in this appendix are `gasdev.c`, `nrutil.c`, and `ran1.c`, since they are taken directly from Press *et al.* (1988). My only modification of the Press *et al.* (1988) files was to change all occurences of `float` to `double`.

A.1 Makefile

This is the makefile for the unix command **make** that is used to compile the model programme. The model programme is called **benard**. Another version, **benard2**, was used for testing and for performing runs with strange ("pattern-inducing," see Section 2.7.3) initial conditions. The code **main2.c** is not listed in this appendix. As can be seen from this makefile, the source code was in several different directories.

```
CFLAGS= -O

allocate.o:
cc $(CFLAGS) -c ../utilities/allocate.c
gasdev.o:
cc $(CFLAGS) -c ../recipes/gasdev.c
get_double.o:
cc $(CFLAGS) -c ../utilities/get_double.c
get_int.o:
cc $(CFLAGS) -c ../utilities/get_int.c
nrutil.o:
cc $(CFLAGS) -c ../recipes/nrutil.c
ran1.o:
cc $(CFLAGS) -c ../recipes/ran1.c

benard= \
allocate.o \
entropy.o \
gasdev.o \
get_double.o \
get_int.o \
gradient.o \
gravity.o \
heat_diffusion.o \
heat_flux.o \
heat_transport.o \
menu.o \
nrutil.o \
ran1.o \
randomize.o \
step_function.o \
update_momenta.o \
whorl_diffusion.o \
whorl_transport.o

benard: $(benard) main.o
cc $(CFLAGS) $(benard) main.o -lm -o benard

benard2: $(benard) main2.o
cc $(CFLAGS) $(benard) main2.o -lm -o benard2
```

```
clean:
rm *.o a.out core benard benard2 benard.out output.?
```

A.2 benard.h

This is the header file that is included at the top of most of the files with an **#include** statement.

```
/* this is the header file for the convection cell functions */

#define    MAXM  80  /* Maximum M */
#define    MAXN  40  /* Maximum N */

#define   BUFSZ  32  /* parameter buffer size in menu() */
#define MAXLINE  80  /* maximum input length in menu() */
#define  MAXOUT   3  /* maximum number of output files */

#define      PI  3.14159265

#define      STEADY  1  /* flag setting for steady-state version */
#define       RELAX  0  /*              for relaxation version */
```

A.3 main.c

This is the main function which calls the others.

```
#include "benard.h"
#include <math.h>
#include <stdio.h>
#include <strings.h>

void main()
/* This is the main programme. It handles output and it calls all of the
functions.
*/
{
  double **allocate(),entropy() ;
  void heat_diffusion(),heat_transport(),whorl_diffusion(),whorl_transport() ;
  void gravity(),update_momenta(),menu(),heat_flux(),step_function() ;
  void gradient(),randomize() ;

  double **lattice,q_0,sigma,alpha_q,beta_q,alpha_w,beta_w,gamma,eta_w=0.0 ;
  double px,py,p,theta,deltaq_c=0.0,deltaq_h=0.0 ;
```

```
    int M,N,m,n,T,t,deltaT,seed,steady,outputtimes[MAXOUT],outindex ;
    char outname[MAXLINE] ;
    FILE *outfp,*fp ;

    lattice = allocate(2*MAXM+1,2*MAXN+1) ;

  /* get parameters */
    menu("benard.par",&M,&N,&steady,&T,outputtimes,&q_0,&sigma,&seed,&alpha_q,
         &beta_q,&alpha_w,&beta_w,&gamma,&eta_w) ;
  /* set initial conditions */
    if(steady==RELAX) step_function(lattice,M,N,q_0) ;
    if(steady==STEADY) gradient(lattice,M,N,q_0) ;
    update_momenta(lattice,M,N) ;

  /* open output file for times and heat entropies */
    fp = fopen("benard.out","w") ;
    fprintf(fp,"%dx%d steady=%d q_0=%.1f sigma=%.1e seed=%d\n",
    M,N,steady,q_0,sigma,seed) ;
    fprintf(fp,"alpha_q=%.3f beta_q=%.3f alpha_w=%.3f beta_w=%.3f gamma=%.3f\n",
    alpha_q,beta_q,alpha_w,beta_w,gamma) ;

    deltaT = T / 1000 ;      /* number of steps between small outputs */
    if(deltaT<1) deltaT=1 ;
    outindex = 0 ;           /* initialize output file number */

  /* start time loop */
    for(t=0 ; t<T ; t++){

  /* output a bit every deltaT2 steps */
      if(t%deltaT == 0) fprintf(fp,"%04d  %.17e  %e  %e\n",t,
         entropy(lattice,M,N),deltaq_h,deltaq_c) ;
  /* output lots 3 times */
      if(outindex < MAXOUT && t == outputtimes[outindex]){
        outindex++ ;
        sprintf(outname,"output.%d",outindex) ;
        outfp = fopen(outname,"w") ;
        fprintf(stdout,"main: opened file %s\n",outname) ;
        fprintf(outfp,"t=%d %dx%d steady=%d q_0=%.1f sigma=%.1e seed=%d\n",
        t,M,N,steady,q_0,sigma,seed) ;
        fprintf(outfp,
        "alpha_q=%.3f beta_q=%.3f alpha_w=%.3f beta_w=%.3f gamma=%.3f\n",
        alpha_q,beta_q,alpha_w,beta_w,gamma) ;
        for(m=1 ; m<2*M ; m+=2){
for(n=1 ; n<2*N ; n+=2){
  px = 0.5*(lattice[m+1][n] + lattice[m-1][n]) ;
  py = 0.5*(lattice[m][n+1] + lattice[m][n-1]) ;
  p = sqrt(px*px + py*py) ;
  theta = atan2(py,px) ; if(theta<0.0) theta += 2*PI ;
  fprintf(outfp,"%02d  %02d  %.4e  %.4e  %.4e  %.4e  %.4e\n",
  m,n,lattice[m][n],px,py,p,theta) ;
}
      }
      fclose(outfp) ;
      fprintf(stdout,"main: closed file %s\n",outname) ;
```

```
    }

 /* do processes */
    if(steady == STEADY) heat_flux(&deltaq_h,&deltaq_c,lattice,M,N,q_0) ;
    heat_diffusion(lattice,M,N,alpha_q) ;
    heat_transport(lattice,M,N,beta_q) ;
    whorl_diffusion(lattice,M,N,alpha_w,eta_w) ; /* this includes absorption */
    whorl_transport(lattice,M,N,beta_w) ;
    gravity(lattice,M,N,gamma) ;
    if(sigma>0.0) randomize(lattice,M,N,sigma,seed) ;
 /* update edge-crossing momenta */
    update_momenta(lattice,M,N) ;

  }
 /* done */
 }
```

A.4 allocate.c

This is a function that allocates two-dimensional arrays.

```
double **allocate(M,N)
     int M,N ;
/* allocate() returns a pointer to an MxN array.
*/
{
  int i=0 ;
  double **temp ;

  temp = (double **) malloc(M * sizeof(double *)) ;

  for(i=0 ; i<M ; i++){
    temp[i] = (double *) malloc(N * sizeof(double)) ;
  }

  return temp ;
}
```

A.5 entropy.c

This function calculates the heat entropy described in Section 2.8.2.

```
#include <math.h>
#define TINY 1e-17
```

```
double entropy(L,M,N)
     double **L ;
     int M,N ;
/* entropy() returns the value of the entropy of the heat distribution
according to the formula (3.1) in Balian, "du microscopique au macroscopique,"
p. 92.
   L[0..2M][0..2N], on input, is the Lattice.
   MxN, on input, is the size of the lattice.
*/
{
  int i=0,j=0 ;
  double S=0.0,Q=0.0,temp ;

  for(i=1 ; i<2*M ; i+=2){
    for(j=1 ; j<2*N ; j+=2){
      temp=L[i][j] ;
      Q += temp ;
    }
  }

  for(i=1 ; i<2*M ; i+=2){
    for(j=1 ; j<2*N ; j+=2){
      temp=L[i][j] ;
      if(temp > TINY) S -= (log(fabs(temp/Q)) * (temp/Q)) ;
    }
  }

  return(S) ;
}
```

A.6 get_double.c

This is an input utility.

```
#include <stdio.h>

#define MAXLINE 80

void get_double(message, number)
     char *message ;
     double *number ;
/* get_double is a generalized prompting-input routine for double-precision
real numbers.
   *message, on input, is the prompt to be used, with an optional "%f" or
whatever in it to display the default value.
   *number, on input, is the default value; on output, it is the user-supplied
value, if anything is in fact supplied.
*/
```

```
{
  int i ;
  char inputline[MAXLINE] ;

  fprintf(stdout, message, *number) ;

  i = -1 ;
  do{
    inputline[++i] = getc(stdin) ;
  } while(inputline[i] != '\n') ;

  if(i > 0){
    inputline[i] = '\0' ;
    if(sscanf(inputline, "%lf", number) != 1) {
      fprintf(stderr, "get_double: error reading input\n") ;
      exit(0) ;
    }
  }

}
```

A.7 get_int.c

This is an input utility.

```
#include <stdio.h>

#define MAXLINE 80

void get_int(message, number)
     char *message ;
     int *number ;
/* get_int is a generalized prompting-input routine for double-precision real
numbers.
   *message, on input, is the prompt to be used, with an optional "%d" or
whatever in it to display the default value.
   *number, on input, is the default value; on output, it is the user-supplied
value, if anything is in fact supplied.
*/
{
  int i ;
  char inputline[MAXLINE] ;

  fprintf(stdout, message, *number) ;

  i = -1 ;
  do{
    inputline[++i] = getc(stdin) ;
  } while(inputline[i] != '\n') ;
```

```
    if(i > 0){
      inputline[i] = '\0' ;
      if(sscanf(inputline, "%d", number) != 1) {
        fprintf(stderr, "get_int: error reading input\n") ;
        exit(0) ;
      }
    }

}
```

A.8 gradient.c

This function creates gradient initial conditions, described in Section 2.7.2.

```
#include "benard.h"

void gradient(L,M,N,q_0)
     double **L,q_0 ;
     int M,N ;
/* This function sets the gradient initial conditions. This involves:
(a) setting the heats to a slope from 0 to q_0; and (b) setting the whorls to
zero.
   L[0..2M][0..2N], on output, is the initialized lattice.
   M,N, on input, are the dimensions of the lattice.
   q_0, on input, is the heat of the lower wall.
*/
{
  int m,n,r,s ;

/* initialize heats */
  for(m=1 ; m<2*M ; m+=2) for(n=1 ; n<2*N ; n+=2)
    L[m][n] = q_0 * (2*M+1-n) / (2*M) ;

/* set whorls to zero */
  for(r=0 ; r<2*M ; r+=2) for(s=0 ; s<=2*N ; s+=2) L[r][s] = 0.0 ;

/* done */
}
```

A.9 gravity.c

This function is responsible for performing the law of gravity, described in Section 2.5.5.

```
#include "benard.h"

void gravity(L,M,N,gamma)
     double **L,gamma ;
     int M,N ;
/* L[0..2M][0..2N], on input, is the lattice. On output it is the lattice
updated by one gravity step.
   M, on input, is the number of bins in the x-direction.
   N, on input, is the number of bins in the y-direction.
   gamma, on input, is the gravitational parameter.
*/
{
  int r,r1,r2,s,s1,s2 ;  /* r,s will always be even, r1,r2,s1,s2 odd */

  for(r=0 ; r<2*M ; r+=2){
    r1 = r-1 ; r2 = r+1 ;
    if(r1<0) r1 = 2*M-1 ;
    for(s=2 ; s<2*N ; s+=2){
      s1 = s-1 ; s2 = s+1 ;
      L[r][s] += gamma*(L[r2][s2] + L[r2][s1] - L[r1][s2] - L[r1][s1]) ;
    }
  }
}
```

A.10 heat_diffusion.c

This function is responsible for performing the law of heat diffusion, described in Section 2.5.1.

```
#include "benard.h"

void heat_diffusion(L,M,N,alpha_q)
     double **L,alpha_q ;
     int M,N ;
/* L[0..2M][0..2N], on input, is the lattice. On output it is the lattice
updated by one heat diffusion step.
   M, on input, is the number of bins in the x-direction.
   N, on input, is the number of bins in the y-direction.
   alpha_q, on input, is the heat diffusion parameter.
*/
{
  int m,n,m1,m2,n1,n2 ;   /* m,n,m1,m2,n1,n2 will all always be odd */
  double temp,deltaL[2*MAXM+1][2*MAXN+1] ;

/* initialize the delta-lattice to zero for odd m,n */
  for(m=1 ; m<2*M ; m+=2) for(n=1 ; n<2*N ; n+=2) deltaL[m][n] = 0.0 ;

/* calculate the changes to the lattice values */
  for(m=1 ; m<2*M ; m+=2){
```

```
      m1 = m-2 ; m2 = m+2 ;
      if(m1<0) m1 = 2*M-1 ;          /* model wraps-around in x */
      if(m2>2*M) m2 = 1 ;            /* same explanation */
      for(n=1 ; n<2*N ; n+=2){
        n1 = n-2 ; n2 = n+2 ;
        temp = alpha_q*L[m][n] ;
        deltaL[m][n] -= temp ;
        deltaL[m1][n] += temp ;      /* -ve x-direction */
        deltaL[m][n] -= temp ;
        deltaL[m2][n] += temp ;      /* +ve x-direction */
        if(n1>0){
 deltaL[m][n] -= temp ;
 deltaL[m][n1] += temp ;   /* -ve y-direction */
        }
        if(n2<2*N){
 deltaL[m][n] -= temp ;
 deltaL[m][n2] += temp ;   /* +ve y-direction */
        }
      }
    }

 /* now change the actual lattice values using the delta-lattice */
   for(m=1 ; m<2*M ; m+=2) for(n=1 ; n<2*N ; n+=2) L[m][n] += deltaL[m][n] ;

 /* done */
 }
```

A.11 heat_flux.c

This function is responsible for performing the law of heat flow, described in Section 2.6.2. It is also responsible for calculating the two heat fluxes, $(\Delta Q)_H$ and $(\Delta Q)_C$, described in Section 2.8.3.

```
#include "benard.h"

void heat_flux(dqh,dqc,L,M,N,q_0)
     double *dqh,*dqc,**L,q_0 ;
     int M,N ;
/* *dqh, on output, is the amount of heat that enters the system from the
bottom.
   *dqc, on output, is the amount of heat that enters the system from the top.
   L[0..2M][0..2N], on input, is the lattice. On output it is the lattice with
the edge heats set to 0 and q_0. This function is used for modelling steady-
state convection.
   M, on input, is the number of bins in the x-direction.
   N, on input, is the number of bins in the y-direction.
   q_0, on input, is the heat at the bottom of the convection cell.
*/
```

```
{
  int m,n ;
  double temph=0.0,tempc=0.0 ;

  n = 2*N-1 ;
  for(m=1 ; m<2*M ; m+=2){
    temph += q_0 - L[m][1] ;
    tempc += - L[m][n] ;
    L[m][1] = q_0 ;
    L[m][n] = 0.0 ;
  }
  *dqh = temph ;
  *dqc = tempc ;
}
```

A.12 heat_transport.c

This function is responsible for performing the law of heat transport, described in Section 2.5.2.

```
#include "benard.h"

void heat_transport(L,M,N,beta_q)
     double **L,beta_q ;
     int M,N ;
/* L[0..2M][0..2N], on input, is the lattice. On output it is the lattice
updated by one heat transport step.
   M, on input, is the number of bins in the x-direction.
   N, on input, is the number of bins in the y-direction.
   beta_q, on input, is the heat transport parameter.
*/
{
  int m,n,h,h1,h2,i,j,k,k1,k2 ;  /* m,n,h1,h2,i,j,k1,k2 all odd; h,k even */
  double temp,deltaL[2*MAXM+1][2*MAXN+1] ;

/* initialize the delta-lattice to zero for odd m,n */
  for(m=1 ; m<2*M ; m+=2) for(n=1 ; n<2*N ; n+=2) deltaL[m][n] = 0.0 ;

/* first do horizontal transport across vertical edges */
  for(h=0 ; h<2*M ; h+=2){
    h1 = h-1 ; h2 = h+1 ;
    if(h1<0) h1 = 2*M-1 ;    /* model wraps-around in x */
    for(i=1 ; i<2*N ; i+=2){
      temp = beta_q*L[h][i]*0.5*(L[h1][i] + L[h2][i]) ;
      deltaL[h1][i] -= temp ;
      deltaL[h2][i] += temp ;
    }
  }
```

```
 /* now vertical transport */
   for(j=1 ; j<2*M ; j+=2){
     for(k=2 ; k<2*N ; k+=2){   /* this ensures no transport across y=0 */
       k1 = k-1 ; k2 = k+1 ;
       temp = beta_q*L[j][k]*0.5*(L[j][k1] + L[j][k2]) ;
       deltaL[j][k1] -= temp ;
       deltaL[j][k2] += temp ;
     }
   }

 /* now change the actual lattice values using the delta-lattice */
   for(m=1 ; m<2*M ; m+=2) for(n=1 ; n<2*N ; n+=2) L[m][n] += deltaL[m][n] ;

 /* done */
 }
```

A.13 menu.c

This is an input function which presents the user with a set of parameters and the option to change them. It reads the parameters in from a parameter file and then when the user is done selecting parameters, it saves them back into the parameter file for next time.

```
#include "benard.h"
#include <fcntl.h>
#include <stdio.h>
#include <strings.h>

#define PERMS 0666   /* permissions for parameter file */
#define RUN  1
#define WAIT 0

void menu(parameterfile,M,N,steady,T,outtimes,q_0,sigma,seed,alpha_q,beta_q,
  alpha_w,beta_w,gamma,eta_w)
     double *q_0,*sigma,*alpha_q,*beta_q,*alpha_w,*beta_w,*gamma,*eta_w ;
     int *M,*N,*steady,*T,outtimes[],*seed ;
     char *parameterfile ;
/* menu() reads the parameter file and then presents the user with a menu for
changing the parameters. It is relatively self-explanatory. At the beginning it
reads parameters in from a paramter file, and at the end it saves the chosen
parameters for next time in the same file.
*/
{
  void get_double() ;
  void get_int() ;

  static int count=0 ;
```

```
  double parambuf[BUFSZ] ;   /* BUFSZ defined in benard.h */
  int paramfd,index1,index2,choice,status,k ;

  count++ ;
  status = WAIT ;

/* read parameter file */
  paramfd = open(parameterfile,O_RDONLY,0) ;  /* O_RDONLY in fcntl.h */
  if(paramfd == -1){
    fprintf(stderr,"menu: warning: file %s does not yet exist\n",
    parameterfile) ;
  } else{
    fprintf(stdout,"menu: opened file %s to read\n",parameterfile) ;
    read(paramfd,parambuf,(BUFSZ * sizeof(double))) ;
    close(paramfd) ;
    fprintf(stdout,"menu: closed file %s\n",parameterfile) ;
  }

  index1 = 0 ;
  index2 = 0 ;

/* read in old parameter list */
  *M = (int) parambuf[index1++] ;
  *N = (int) parambuf[index1++] ;
  *steady = (int) parambuf[index1++] ;
  *T = (int) parambuf[index1++] ;
  for(k=0;k<MAXOUT;k++) outtimes[k] = (int) parambuf[index1++] ;
  *q_0 = parambuf[index1++] ;
  *alpha_q = parambuf[index1++] ;
  *beta_q = parambuf[index1++] ;
  *alpha_w = parambuf[index1++] ;
  *beta_w = parambuf[index1++] ;
  *gamma = parambuf[index1++] ;
  *sigma = parambuf[index1++] ;
  *seed = (int) parambuf[index1++] ;
  *eta_w = parambuf[index1++] ;

  do{

/* create menu */
    fprintf(stdout,"0: run simulation\n") ;
    fprintf(stdout,"1: set lattice dimensions MxN (max %dx%d) [%dx%d]\n",
    MAXM,MAXN,*M,*N) ;
    fprintf(stdout,"2: relaxation (%d) or steady-state (%d) [%d]\n",
    RELAX,STEADY,*steady) ;
    fprintf(stdout,"3: set number of time steps T [%d]\n",*T) ;
    fprintf(stdout,"4: set %d output times",MAXOUT) ;
    for(k=0;k<MAXOUT;k++) fprintf(stdout," [%d]",outtimes[k]) ;
    fprintf(stdout,"\n") ;
    fprintf(stdout,"5: set lower wall heat q_0 [%.3f]\n",*q_0) ;
    fprintf(stdout,"6: set heat diffusion parameter alpha_q [%.3f]\n",
    *alpha_q) ;
    fprintf(stdout,"7: set heat transport parameter beta_q [%.3f]\n",*beta_q) ;
    fprintf(stdout,"8: set whorl diffusion parameter alpha_w [%.3f]\n",
```

```
      *alpha_w) ;
      fprintf(stdout,"9: set whorl transport parameter beta_w [%.3f]\n",
      *beta_w) ;
      fprintf(stdout,"10: set gravitational parameter gamma [%.3f]\n",*gamma) ;
      fprintf(stdout,"11: set random whorl standard deviation sigma [%.3e]\n",
      *sigma) ;
      fprintf(stdout,"12: set random number seed [%d]\n",*seed) ;
      fprintf(stdout,"13: set whorl absorption parameter eta_w [%.3f]\n",
      *eta_w) ;

 /* get menu choice */
      choice = 0 ;
      get_int("Menu selection? [%d] ",&choice) ;
      fprintf(stdout,"choice = %d\n",choice) ;

      switch(choice){

      case 0:
        status = RUN ;
        break ;

      case 1:
 /* set lattice dimensions M,N */
        get_int("number of bins in x-direction (M)? [%d] ",M) ;
        fprintf(stdout,"M = %d\n",*M) ;
        get_int("number of bins in y-direction (N)? [%d] ",N) ;
        fprintf(stdout,"N = %d\n",*N) ;
        break ;

      case 2:
 /* set steady-state or equilibrium */
        get_int("relaxation or steady state? [%d] ",steady) ;
        fprintf(stdout,"steady = %d\n",*steady) ;
        break ;

      case 3:
 /* set time steps T */
        get_int("number of time steps? [%d] ",T) ;
        fprintf(stdout,"T = %d\n",*T) ;
        break ;

      case 4:
 /* set output times */
        for(k=0;k<MAXOUT;k++){
 get_int("output time? [%d] ",outtimes+k) ;
 fprintf(stdout,"output time %d = %d\n",k+1,outtimes[k]) ;
        }
        break ;

      case 5:
 /* set lower wall heat q_0 */
        get_double("lower wall heat q_0? [%.3f] ",q_0) ;
        fprintf(stdout,"q_0 = %f\n",*q_0) ;
        break ;
```

```
     case 6:
 /* set heat diffusion parameter alpha_q */
       get_double("heat diffusion parameter? [%.3f] ",alpha_q) ;
       fprintf(stdout,"alpha_q = %f\n",*alpha_q) ;
       break ;

     case 7:
 /* set heat transport parameter beta_q */
       get_double("heat transport parameter? [%.3f] ",beta_q) ;
       fprintf(stdout,"beta_q = %f\n",*beta_q) ;
       break ;

     case 8:
 /* set whorl diffusion parameter alpha_w */
       get_double("whorl diffusion parameter? [%.3f] ",alpha_w) ;
       fprintf(stdout,"alpha_w = %f\n",*alpha_w) ;
       break ;

     case 9:
 /* set whorl transport parameter beta_w */
       get_double("whorl transport parameter? [%.3f] ",beta_w) ;
       fprintf(stdout,"beta_w = %f\n",*beta_w) ;
       break ;

     case 10:
 /* set gravitational parameter gamma */
       get_double("gravitational parameter? [%.3f] ",gamma) ;
       fprintf(stdout,"gamma = %f\n",*gamma) ;
       break ;

     case 11:
 /* set whorl standard deviation sigma */
       get_double("random whorl standard deviation? [%.3e] ",sigma) ;
       fprintf(stdout,"sigma = %e\n",*sigma) ;
       break ;

     case 12:
 /* set random number seed */
       get_int("random number seed? [%d] ",seed) ;
       fprintf(stdout,"seed = %d\n",*seed) ;
       break ;

     case 13:
 /* set whorl absorption parameter eta_w */
       get_double("whorl absorption parameter? [%.3f] ",eta_w) ;
       fprintf(stdout,"eta_w = %f\n",*eta_w) ;
       break ;

    }
  }while(status == WAIT) ;

 /* rewrite parameter buffer */
  parambuf[index2++] = (double) (*M) ;
```

```
    parambuf[index2++] = (double) (*N) ;
    parambuf[index2++] = (double) (*steady) ;
    parambuf[index2++] = (double) (*T) ;
    for(k=0;k<MAXOUT;k++) parambuf[index2++] = (double) (outtimes[k]) ;
    parambuf[index2++] = *q_0 ;
    parambuf[index2++] = *alpha_q ;
    parambuf[index2++] = *beta_q ;
    parambuf[index2++] = *alpha_w ;
    parambuf[index2++] = *beta_w ;
    parambuf[index2++] = *gamma ;
    parambuf[index2++] = *sigma ;
    parambuf[index2++] = (double) (*seed) ;
    parambuf[index2++] = *eta_w ;

 /* check pointers to parambuf[] */
    if(index1 != index2){
      fprintf(stderr,"menu: warning: %s",
      "may be error in reading and writing parameters\n") ;
    }

 /* re-write parameter file */
    paramfd = creat(parameterfile,PERMS) ;
    if(paramfd == -1){
      fprintf(stderr,"menu: error opening %s\n",parameterfile) ;
      exit(0) ;
    }
    fprintf(stdout,"menu: opened file %s to write\n",parameterfile) ;
    write(paramfd,((char *) parambuf),(BUFSZ * sizeof(double))) ;
    close(paramfd) ;
    fprintf(stdout,"menu: closed file %s\n",parameterfile) ;

}
```

A.14 randomize.c

This function is responsible for performing the law of randomization, described in Section 2.5.6.

```
#include "benard.h"

void randomize(L,M,N,sigma,seed)
     double **L,sigma ;
     int M,N,seed ;
/* This function adds to each whorl a randomly generated number from a
Gaussian distribution with zero mean and standard deviation sigma.
   L[0..2M][0..2N], on output, is the initialized lattice.
   M,N, on input, are the dimensions of the lattice.
   sigma, on input, is the standard deviation of the random whorls.
   seed, on input, is the initial seed for the random number generator.
```

```
 */
 {
   double gasdev() ;   /* from Press et. al., Numerical Recipes */
   static int sed=0,flag=0 ;
   int r,s ;

 /* set seed on first call */
   if(flag==0){
     sed = seed ;
     flag++ ;
   }
 /* add random values to whorls */
   for(r=0 ; r<2*M ; r+=2) for(s=2 ; s<2*N ; s+=2)
     L[r][s] += sigma * gasdev(&sed) ;

 /* done */
 }
```

A.15 step_function.c

This function sets up step function initial conditions, described in Section 2.7.1.

```
#include "benard.h"

void step_function(L,M,N,q_0)
     double **L,q_0 ;
     int M,N ;
/* This function sets the step function initial conditions. This involves:
(a) setting the heats to zero except for those along the lower wall, which get
set to q_0; and (b) setting whorls to zero.
   L[0..2M][0..2N], on output, is the initialized lattice.
   M,N, on input, are the dimensions of the lattice.
   q_0, on input, is the heat of the lower wall.
*/
{
  int m,n,r,s ;

/* initialize heats to zero and q_0 */
  for(m=1 ; m<2*M ; m+=2){
    L[m][1] = q_0 ;
    for(n=3 ; n<2*N ; n+=2) L[m][n] = 0.0 ;
  }

/* set whorls to zero */
  for(r=0 ; r<2*M ; r+=2) for(s=0 ; s<=2*N ; s+=2) L[r][s] = 0.0 ;

/* done */
}
```

A.16 update_momenta.c

This function converts the whorl field into a field of edge-crossing momenta, and stores those momenta in the appropriate places in the lattice.

```
#include "benard.h"

void update_momenta(L,M,N)
     double **L ;
     int M,N ;
/* update_momenta() sets all edge-crossing momenta to agree with the whorls.
   L[0..2M[0..2N], on input, is the lattice with correct whorls and heats; on
output it is the lattice with correct whorls, heats and edge-crossing momenta.
   M, on input, is the number of bins in the x-direction.
   N, on input, is the number of bins in the y-direction.
*/
{
  int m,m1,n,r,s ;   /* m,n are always odd; r,s always even */

/* do horizontal edges (m,s) */
  for(m=1 ; m<2*M ; m+=2){
    m1 = m+1 ; if(m1==2*M) m1 = 0 ;      /* model wraps-around in x */
    for(s=0 ; s<=2*N ; s+=2){
      L[m][s] = L[m-1][s] - L[m1][s] ;
    }
  }

/* do vertical edges (r,n) */
  for(r=0 ; r<2*M ; r+=2){
    for(n=1 ; n<2*N ; n+=2){
      L[r][n] = L[r][n+1] - L[r][n-1] ;
    }
  }

/* done */
}
```

A.17 whorl_diffusion.c

This function is responsible for performing the law of whorl diffusion, described in Section 2.5.3. It also performs whorl absorption, described in Section 2.6.3.

```
#include "benard.h"

void whorl_diffusion(L,M,N,alpha_w,eta_w)
```

```
      double **L,alpha_w,eta_w ;
      int M,N ;
 /* L[0..2M][0..2N], on input, is the lattice. On output it is the lattice
 updated by one whorl diffusion step, plus one whorl absorption step.
    M, on input, is the number of bins in the x-direction.
    N, on input, is the number of bins in the y-direction.
    alpha_w, on input, is the whorl diffusion parameter.
    eta_w, on input, is the whorl absorption at walls parameter.
 */
 {
   int r,s,r1,r2,s1,s2 ;   /* r,s,r1,r2,s1,s2 will all always be even */
   double temp,deltaL[2*MAXM+1][2*MAXN+1] ;

 /* initialize the delta-lattice to zero for even r,s */
   for(r=0 ; r<2*M ; r+=2) for(s=2 ; s<2*N ; s+=2) deltaL[r][s] = 0.0 ;

 /* calculate the changes to the lattice values */
   for(r=0 ; r<2*M ; r+=2){
     r1 = r-2 ; r2 = r+2 ;
     if(r==0) r1 = 2*M-2 ;          /* model wraps-around in x */
     if(r2==2*M) r2 = 0 ;           /* same explanation */
     deltaL[r][2] -= eta_w*L[r][2] ;           /* whorl absorption at bottom */
     deltaL[r][2*N-2] -= eta_w*L[r][2*N-2] ;   /* whorl absorption at top */
     for(s=2 ; s<2*N ; s+=2){
       s1 = s-2 ; s2 = s+2 ;
       temp = alpha_w*L[r][s] ;
       deltaL[r][s] -= temp ;
       deltaL[r1][s] += temp ;      /* -ve x-direction */
       deltaL[r][s] -= temp ;
       deltaL[r2][s] += temp ;      /* +ve x-direction */
       if(s1>1){
deltaL[r][s] -= temp ;
deltaL[r][s1] += temp ;   /* -ve y-direction */
       }
       if(s2<2*N-1){
deltaL[r][s] -= temp ;
deltaL[r][s2] += temp ;   /* +ve y-direction */
       }
     }
   }

 /* now change the actual lattice values using the delta-lattice */
   for(r=0 ; r<2*M ; r+=2) for(s=2 ; s<2*N ; s+=2) L[r][s] += deltaL[r][s] ;

 /* done */
 }
```

A.18 whorl_transport.c

This function is responsible for performing the law of whorl transport, described in Section 2.5.4.

```
#include "benard.h"

void whorl_transport(L,M,N,beta_w)
     double **L,beta_w ;
     int M,N ;
/* L[0..2M][0..2N], on input, is the lattice. On output it is the lattice
updated by one whorl transport step.
   M, on input, is the number of bins in the x-direction.
   N, on input, is the number of bins in the y-direction.
   beta_w, on input, is the whorl transport parameter.
*/
{
  double deltaL[2*MAXM+1][2*MAXN+1],temp ;
  int r,r1,r2,s,s1,s2 ;   /* r,r1,r2,s,s1,s2 all even at all times */

/* initialize the delta-lattice to zero for even r,s */
  for(r=0 ; r<=2*M ; r+=2) for(s=0 ; s<=2*N ; s+=2) deltaL[r][s] = 0.0 ;

/* first do non-edge cases */
  for(r=0 ; r<(2*M-1) ; r+=2){
    r1 = r-2 ; r2 = r+2 ;
    if(r1<0) r1 = 2*M-2 ;       /* model wraps-around in x */
    if(r2==2*M) r2 = 0 ;        /* model wraps-around in x */
    for(s=4 ; s<(2*N-3) ; s+=2){
      s1 = s-2 ; s2 = s+2 ;
      temp = beta_w*L[r][s]*0.5*(L[r2][s] + L[r][s2]) ;
      deltaL[r2][s] -= temp ;
      deltaL[r][s2] += temp ;
      temp = beta_w*L[r][s]*0.5*(L[r][s2] + L[r1][s]) ;
      deltaL[r][s2] -= temp ;
      deltaL[r1][s] += temp ;
      temp = beta_w*L[r][s]*0.5*(L[r1][s] + L[r][s1]) ;
      deltaL[r1][s] -= temp ;
      deltaL[r][s1] += temp ;
      temp = beta_w*L[r][s]*0.5*(L[r][s1] + L[r2][s]) ;
      deltaL[r][s1] -= temp ;
      deltaL[r2][s] += temp ;
    }
  }

/* now do edge cases */
  s = 2*M-2 ; s1 = 2*M-4 ;
  for(r=0 ; r<(2*M-1) ; r+=2){
    r1 = r-2 ; r2 = r+2 ;
    if(r1<0) r1 = 2*M-2 ;       /* model wraps-around in x */
    if(r2==2*M) r2 = 0 ;        /* model wraps-around in x */
```

```
 /* do the bottom wall (s=2) */
     temp = beta_w*L[r][2]*0.5*(L[r2][2] + L[r][4]) ;
     deltaL[r2][2] -= temp ;
     deltaL[r][4] += temp ;
     temp = beta_w*L[r][2]*0.5*(L[r][4] + L[r1][2]) ;
     deltaL[r][4] -= temp ;
     deltaL[r1][2] += temp ;
     temp = beta_w*L[r][2]*0.5*(L[r1][2] + L[r2][2]) ;
     deltaL[r1][2] -= temp ;
     deltaL[r2][2] += temp ;
 /* do the top wall (s=2M-2) */
     temp = beta_w*L[r][s]*0.5*(L[r2][s] + L[r1][s]) ;
     deltaL[r2][s] -= temp ;
     deltaL[r1][s] += temp ;
     temp = beta_w*L[r][s]*0.5*(L[r1][s] + L[r][s1]) ;
     deltaL[r1][s] -= temp ;
     deltaL[r][s1] += temp ;
     temp = beta_w*L[r][s]*0.5*(L[r][s1] + L[r2][s]) ;
     deltaL[r][s1] -= temp ;
     deltaL[r2][s] += temp ;
   }

 /* now change the actual lattice values using the delta-lattice */
   for(r=0 ; r<2*M ; r+=2) for(s=2 ; s<2*N ; s+=2) L[r][s] += deltaL[r][s] ;

 /* done */
 }
```