NORTHWESTERN UNIVERSITY

Instabilities and Spatio-Temporal Chaos of Hexagons in the Presence of Rotation

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree
DOCTOR OF PHILOSOPHY
Field of Applied Mathematics

By
Filip D. Sain

EVANSTON, ILLINOIS
December 2000

ABSTRACT

Instabilities and Spatio-Temporal Chaos of Hexagons in the Presence of Rotation

Filip D. Sain

The dynamics of hexagon patterns in rotating systems are investigated within the framework of modified Swift-Hohenberg equations that can be considered as simple models for rotating convection with broken up-down symmetry, e.g. non-Boussinesq Rayleigh-Bénard or Marangoni convection. In the weakly nonlinear regime a linear stability analysis of the hexagons reveals long- and short-wave instabilities, which can be steady or oscillatory. The oscillatory short-wave instabilities can lead to stable hexagon patterns that are periodically modulated in space and time, or to a state of spatio-temporal chaos with a Fourier spectrum whose overall pattern roughly precesses on in time. The chaotic state can exhibit bistability with the steady hexagon pattern. There exist regimes in which hexagons of any wavenumber are unstable.

For Sisyphus.

Acknowledgments

This thesis would never have seen the light of day without a lot of help. I would like to thank the following:

My parents for obvious reasons - namely everything.

My advisor Hermann Riecke for his instruction and patience.

My friends Jeff, Val, Catherine, James, and my brother Stefan for their support and help.

The professors of the Northwestern University Applied Math Department.

The ever-helpful Judy and Edla, who also shared my interests in Pita Inn and Halloween, respectively.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis studies phenomenological and derived models for rotating convection [4]. The phenomenological models were inspired by experiments done in rotating Rayleigh-Bénard convection. The model derived from the governing equations concerns rotating long-wave Marangoni convection. The first part of the introduction describes the physical problem and experimental setup, and also briefly discusses the mathematical modeling of these experiments. The second part of the introduction describes previous work done, upon which the current work is based.

1.1 The Physical Problem

Understanding rotating convection is critical to understanding flows in the atmosphere, oceans, earth's mantle, sun, etc. At this point, a thorough understanding of these flows is far away (witness the accuracy of weather forecasts). As a small step towards this understanding, much research has been done on (rotating) convection in a thin fluid layer. The thinness of the fluid layer restricts the motion of the fluid sufficiently for analytical techniques to be useful in trying to understand the flow. That theory and experiment happen to agree well in this system in comparison to other physical systems also makes its study appealing.

The essence of the experimental setup is a thin layer of fluid either between two solid plates or between a plate on bottom and another fluid (e.g. air) on top. The phenomenological models studied here concern themselves with the first case, the derived model with the second. Since the basic phenomena are sufficiently similar in the two experiments, the case of the fluid between two plates will be described.

A thin fluid layer is a fluid layer of high aspect ratio. In other words, it is a fluid layer where the horizontal dimension of the fluid d_h is much greater than the vertical dimension (thickness) d_v of the fluid,

$$\frac{d_h}{d_v} \gg 1. \tag{1.1}$$

The plates bordering the fluid on top and on bottom are held at constant temperatures, T_{top} and T_{bottom}, respectively. Then $\Delta T = T_{bottom} - T_{top} \geq 0$ is the control parameter of the system. For nonzero ΔT, a heat transfer from the bottom of the fluid layer to the top occurs. If ΔT is small, the heat is transported by conduction, and no bulk movement of the fluid takes place. Conduction is the base state. Above some threshold $\Delta T = \Delta T_c$, the conduction state becomes unstable. Convection results, with the relatively cold and dense fluid at the top of the layer descending and replacing the relatively warm and less dense fluid below, and in turn warming up and rising once more to the top of the fluid layer, only to repeat the cycle. Rotation about the vertical axis of the fluid layer may also be introduced.

Above and close to the convection threshold $\Delta T = \Delta T_c$, one of a few standard patterns of convection often occurs. The pattern selected depends on suitable dimensionless quantities describing the fluid (the Rayleigh number, Prandtl number, etc.), the strength (or absence) of rotation in the system, the boundary conditions, and of course the control parameter ΔT. Commonly seen patterns include rolls, hexagons, squares, rhombs, and rectangles. Experimental pictures of rolls and hexagons are shown in Fig.1.1a and Fig.1.2a,b. The flow for rolls and hexagons is diagramed in Fig.1.1b and Fig.1.2c. In Fig.1.1a, the snapshots are looking down on a circular fluid layer; the white areas indicate cold fluid moving downwards, the black areas indicate warm fluid moving upwards [1]. In Fig.1.1a no orientation of rolls covers the whole fluid layer, rather there are patches of different orientations. The rolls prefer to be perpendicular to the boundary. Fig.1.1b diagrams a section of a roll patch, showing two rolls between the top and the bottom plates. A roll pair consists of one period of the periodic structure, i.e. a clockwise and a coun-

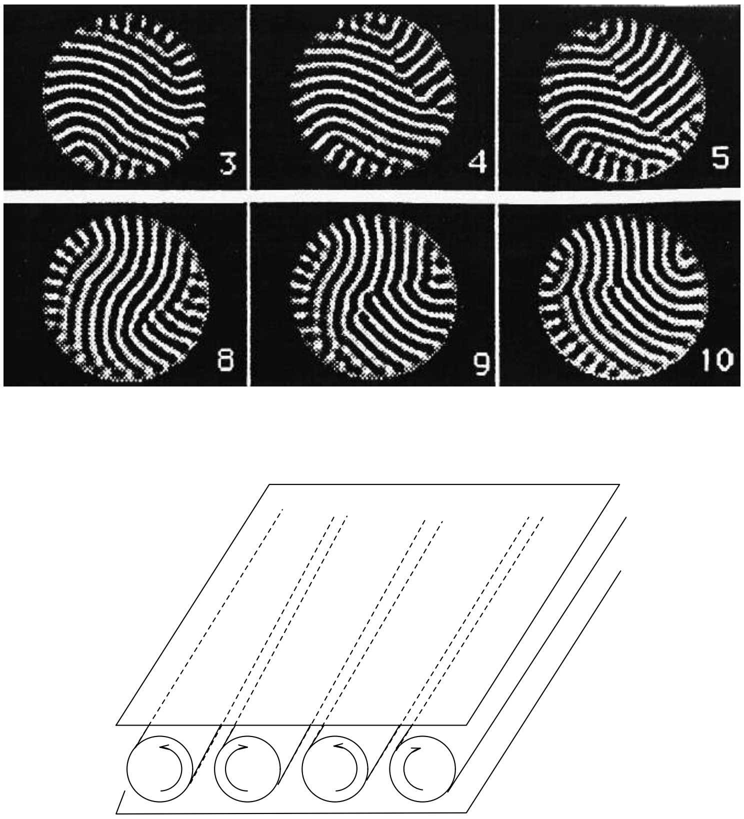


Figure 1.1: *a)* Experimental picture of the Küppers-Lortz instability [1]. *b)* Diagram of roll pair convection cells.

terclockwise component (clockwise and counterclockwise here refer to directions in Fig.1.1b). The rolls are essentially horizontal cylinders, with the arrows indicating direction of the fluid flow. Fig.1.2a shows an almost perfect hexagon pattern in non-Boussinesq Rayleigh-Bénard convection viewed from above [2]. The Boussinesq approximation assumes that the density of the fluid with temperature varies only in the buoyancy term. The hexagonal convection cells form a honeycomb pattern. A detail of Fig.1.2a is shown in Fig.1.2b. The honeycomb pattern is more clearly visible. The fluid flow is down in the middle of the hexagon cell, and up along the boundaries of the hexagon cell (Fig.1.2c). A pattern such as Fig.1.2a is simply many hexagon cells tiled together. Square, rhombic, and rectangle patterns are analogous to the hexagon pattern.

Two experimental results in particular motivated the present work: the Küppers-Lortz instability [1,5,6] or rolls and the aforementioned existence of hexagons. The Küppers-Lortz instability occurs for roll solutions under sufficient rotation. Rolls of one orientation become unstable to rolls oriented at about $60°$ to the original. These new rolls are in turn unstable, and a continuously changing pattern evolves. In fact, Fig.1.1a is an experimental picture of this instability [1]. The pictures in the time series are in the rotating frame of reference. In two separate instances (frames 3,4, and 5, and frames 8,9, and 10), rolls of one orientation become unstable with respect to rolls oriented at about $60°$ to the original. In practice, as noted above, there are patches of rolls of several orientations in the fluid layer at any give time [7–9]. This is also seen in numerical simulations of model equations [10–12]. The Küppers-Lortz instability manifests itself by a patch of rolls of one orientation being taken over by a patch rotated at about $60°$ to the first. The second result, the existence of stable hexagon solutions in thin fluid layer convection without rotation, illustrated in Fig.1.2a,b, has already been noted above.

Prompted by these two results, the question asked in this thesis, in terms of the physical problem, is whether rotation can induce complex dynamics in a hexagon planform, like it does in a roll planform. Experimentally, transitions to spatio-temporal chaos from hexagon patterns have been investigated in parametrically driven surface waves [13, 14] as well as in a chemical system [15].

1.2 Model Equations

We seek to investigate whether rotation can induce complex dynamics in a hexagon planform using model equations that relatively simply describe the bulk motion of the fluid. On a continuum level, the governing equations of the system are the Navier-Stokes, continuity, and heat equations. However, we are interested in describing the flow on a macroscopic level, where equations much simpler than the continuum equations can realistically describe the bulk motion of the fluid. In the case of thin fluid layer convection, the essential feature of these simplified

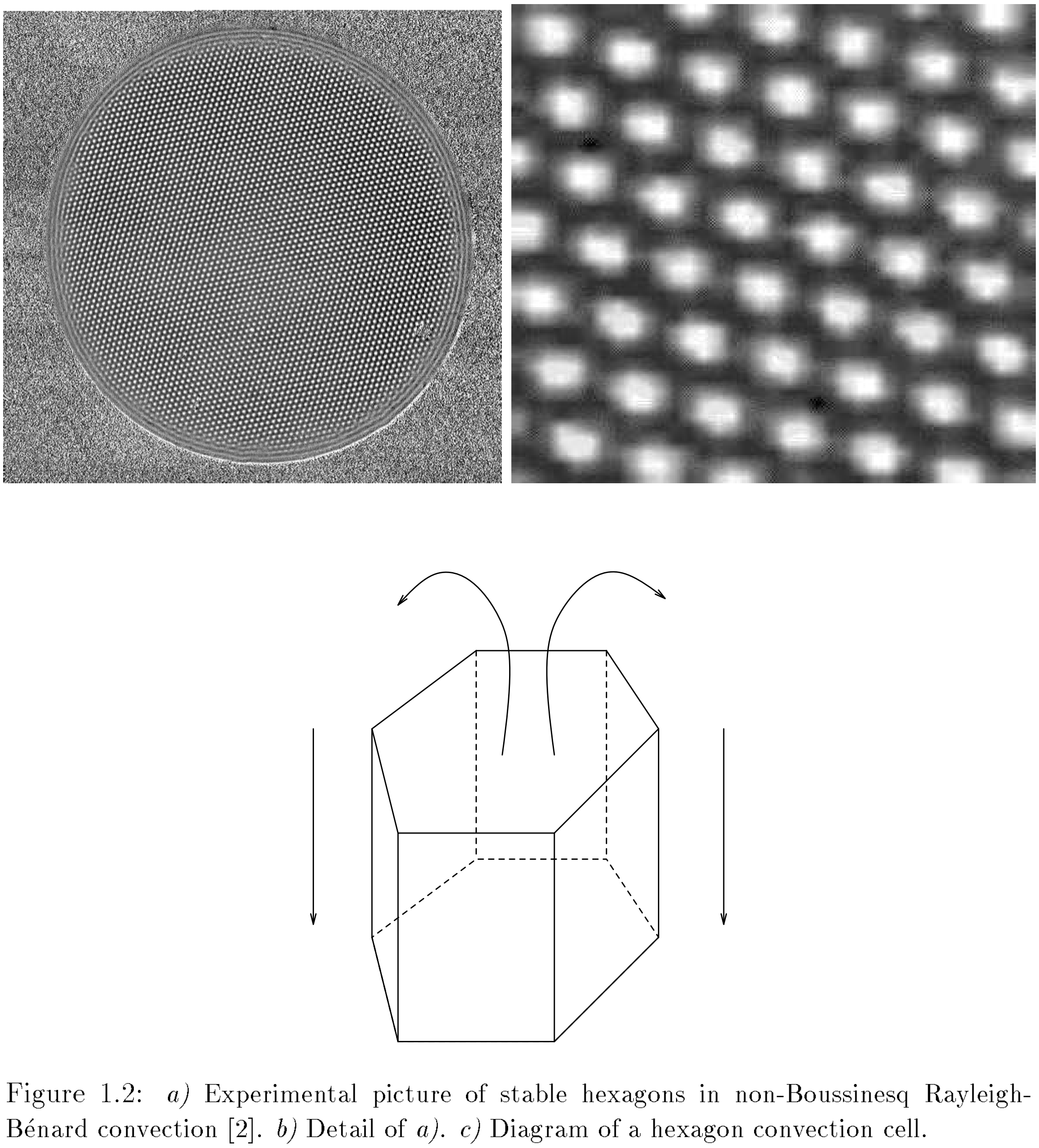


Figure 1.2: *a)* Experimental picture of stable hexagons in non-Boussinesq Rayleigh-Bénard convection [2]. *b)* Detail of *a)*. *c)* Diagram of a hexagon convection cell.

macroscopic equations is the slaving of the dependence of the motion in the vertical dimension to the motion in the horizontal dimensions. Thus we expect a model dependent upon the two horizontal spatial variables, and of course time.

The physical problem as described above is a guide to the phenomena the model equations must describe. Model equations generally consider the fluid layer to be infinite in the horizontal dimensions, and thus there are no horizontal boundary conditions. The equations need to be able to describe both roll and hexagonal convection cells, as well as the Küppers-Lortz instability. A three-mode model for rolls with the modes oriented at 120° to each other in complex Fourier space satisfies these requirements [16–20]. Generally, one obtains six equations, for the three amplitudes and the three phases of the three complex Fourier modes. Such a model can, for example, be obtained from the governing equations through a center manifold reduction [20]. In such three-mode models, the Küppers-Lortz instability leads to a structurally stable heteroclinic orbit connecting the three sets of rolls [17,19]. If the up-down symmetry of the system is broken, thus making stable hexagons possible, the heteroclinic cycle resulting from the Küppers-Lortz instability transforms into a periodic orbit of oscillating hexagons [20–22]. Breaking the up-down symmetry corresponds to a non-Boussinesq fluid in the physical problem. This periodic orbit arises from a Hopf bifurcation, which replaces the steady instability of the hexagonal pattern that leads to rolls [20, 21]. This is discussed in more detail in the next section, on the hexagon subspace results. Large scale drift flow effects driven by the curvature of rolls can strongly affect the stability of roll convection [23,24]. Hexagon patterns are much stiffer than rolls, and so these effects are ignored.

However, the three-mode model, by its very definition, restricts itself to considering modes only within a small subspace of the two-dimensional spatial Fourier transform. While the three-mode model can nicely model the switching from one roll to another rotated at about 60° to the original, the rotation of a hexagon solution by 60° simply results in the original hexagon state. Thus we consider modes outside the three-mode model. One possibility is to extend the three amplitude equations into Ginzburg-Landau equations by allowing slow spatial variation of the amplitudes [3,25]. However, this approach is inadequate since it can only consider long-wave perturbations, which we show is inadequate to describe behavior examined in this thesis.

Thus we are led to the consideration of models of the Swift-Hohenberg [26] type. Swift-Hohenberg models have been used to model several different dynamical systems [12,22,26–30]. They preserve the isotropy of the system and allow arbitrary perturbations of the hexagon solution. Additionally, the model equations considered must meet some further criteria in order to realistically model rotating convection. The model equations need to break the chiral symmetry to model rotation. Also, they need to break the up-down symmetry to model a non-Boussinesq fluid and allow for the possibility of a stable hexagon solution. A modified Swift-Hohenberg equation similar to the first two equations studied in this thesis (2.1,2.10) is studied in [22]. Direct numerical simulations of the equation in a circular thin fluid layer yielded a rigid rotation of the hexagon pattern. This phenomenon is similar to the phenomena described in this thesis. However, there are significant differences. The rigid rotation is in a regime with a sufficiently high rotation rate that there are no stable rolls due to the Küppers-Lortz instability. The work in this thesis is done in parameter regimes where stable rolls do exist (cf. Fig.1.3). It should also be noted that [22] shows rigidly rotating hexagons, different from the switching of roll orientations due to the Küppers-Lortz instability. Additionally, the rigidly rotating hexagons were influenced by the boundaries of the fluid container, whereas an infinite horizontal layer is simulated here by the periodic boundary conditions. Still, the results of the simulations in [22] indicate that a search for complex dynamics from a hexagon planform may be fruitful.

The first two modified Swift-Hohenberg equations studied (2.1,2.10) are phenomenological equations modeling rotating non-Boussinesq Rayleigh-Bénard convection. This part of the thesis is published [31]. Although we consider them as phenomenological equations, equations of this type can be obtained from the governing equations as order parameter equations [27, 32–34].

The last equation studied in this thesis (5.1) is a long-wave equation for

Marangoni convection with poorly conducting boundaries in the presence of rotation. The equation (5.1) was derived from the governing equations [35]. Previous work documented in the literature includes a general stability analysis for Marangoni convection with poorly conducting boundaries on top and on bottom in the absence of rotation [36]. Model equations for long-wave convection have been derived from the governing Navier-Stokes equations in other works, for buoyancy-driven convection with asymmetric boundary conditions [37] and for Marangoni convection without rotation [38,39]. Weakly nonlinear Marangoni convection in the presence of rotation has also been studied [40].

We study the linear stability of weakly nonlinear hexagon planform solutions of the three modified Swift-Hohenberg-type model equations to modes rotated with respect to the hexagon pattern. The analysis results in a 6 by 6 matrix whose eigenvalues, which determine the linear stability of the hexagons, are computed numerically for parameters which vary within appropriate ranges. Additionally, we perform direct numerical simulations of the model equations in order to study the nonlinear evolution of the instabilities. Persistent dynamics have been found to develop from oscillatory instabilities of the hexagon solutions. For some parameter values hexagon solutions of all wavenumbers are unstable, leading to spatio-temporal chaos. For parameters where the hexagon solutions are stable for a range of wavenumbers, bistability between the hexagon solution and persistent dynamics has been found.

1.3 The Hexagon Subspace

This thesis is based largely on the work of J. W. Swift [20]. Soward [21] independently reached similar conclusions. This section summarizes the relevant results presented in [20].

Amplitude equations for three complex amplitudes are obtained from the governing equations for convection in a rotating fluid layer via a center manifold reduction. The equations for the three phases can be separated out. The only relevant quantity is the sum of the phases, which approaches a fixed point on its manifold, leading to the three coupled real amplitude equations that are studied:

$$\begin{aligned} A_t &= RA + \xi BC - (A^2 + h_2 B^2 + h_3 C^2)A \\ B_t &= RB + \xi CA - (B^2 + h_2 C^2 + h_3 A^2)B \\ C_t &= RC + \xi AB - (C^2 + h_2 A^2 + h_3 B^2)C \end{aligned} \tag{1.2}$$

The dependence of the solutions on the coefficients of the amplitude equations is studied. No restrictions are placed on the values of the coefficients. In this thesis, the weakly nonlinear analysis of the model Swift-Hohenberg type equations leads to amplitude equations from which we obtain the hexagon solution whose linear stability is studied. Starting from the model equations (2.1,2.10) restricts the possible ranges and relationships of the coefficients of the amplitude equations. The amplitude equations obtained from the model equations in this thesis are equivalent to (1.2) modulo a rescaling of the amplitudes to make the first cubic coefficient -1. In terms of (1.2), the relevant restrictions are $\xi \neq 0$ (corresponding to a breaking of the up-down symmetry) and $h_2 + h_3 > 2$. Rotation is proportional to $h_2 - h_3$. For the case $h_2 + h_3 > 2$, two bifurcation diagrams, without and with rotation, are explained in this section, and are shown in Fig.1.3.

We define the types of solutions of (1.2) that appear in Fig.1.3. A roll solution consists of one nonzero amplitude, i.e. $A \neq 0$, $B = C = 0$. A hexagon solution consists of three equal nonzero amplitudes, i.e. $A = B = C \neq 0$. A general solution consists of three nonequal nonzero amplitudes, i.e. $A \neq B$, $B \neq C$, $A \neq C$, where all three amplitudes are nonzero. The limit cycle is a time-periodic solution in which the three amplitudes oscillate with fixed relative phases.

We assume the quadratic coefficient is small, in order to have weak subcriticality of the hexagon solution. Swift notes that a small ξ guarantees that the bifurcations described occur, at small amplitude where the equations are valid, while in practice the descriptions of the bifurcations are often qualitatively correct even for larger ξ. The bifurcation parameter is the reduced Rayleigh number R, which is proportional to the temperature difference ΔT across the fluid layer. The following

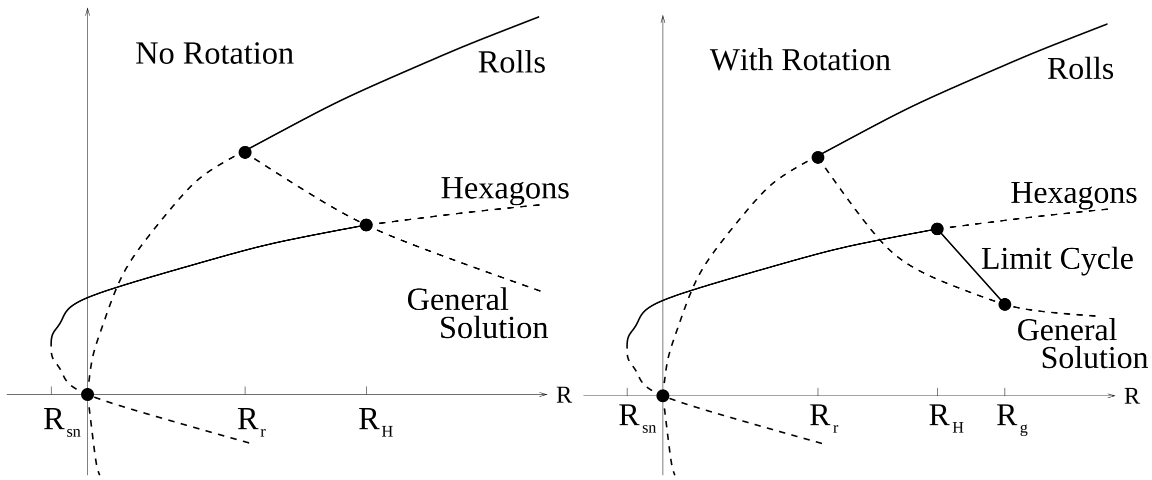


Figure 1.3: Bifurcation Diagrams. The hexagon solution gains stability through a saddle node bifurcation. *a)* Without rotation the hexagons become unstable through a transcritical bifurcation involving a general solution, where two of the three non-zero amplitudes are equal. *b)* With rotation the hexagons become unstable to a limit cycle through a Hopf bifurcation. The limit cycle in turn disappears through a collision with the general solution, now with three unequal non-zero amplitudes.

descriptions apply to Fig.1.3. The roll solution bifurcates supercritically from the trivial solution at $R = 0$ and is unstable at onset. For some value of $R = R_r > 0$, the roll solution becomes stable, through a pitchfork bifurcation which creates a general solution. The hexagon solution is weakly subcritical, and comes into existence through a saddle-node bifurcation at $R = R_{sn} < 0$. One branch of the hexagons is stable at the saddle-node bifurcation, the other unstable. At the point $R = R_H > 0$, the hexagons lose stability. The three general solutions (cyclic permutations of the three amplitudes of the general solution) pass through the hexagon solution at $R = R_H$, resulting in a transcritical bifurcation. For large enough R, the only stable solution is the roll solution. If there is no rotation, all eigenvalues are real, and all instabilities steady.

Introducing rotation changes the bifurcation diagram slightly but significantly. Within these equations (1.2), the values of R_{sn} and R_H for which the hexagon becomes stable and unstable do not change with the introduction of rotation. The transcritical bifurcation at $R = R_H$ splits into two bifurcations - a Hopf and a global bifurcation. The Hopf bifurcation occurs at $R = R_H$ and leads to the instability of the hexagon solution to a stable limit cycle. The limit cycle in turn loses stability through a global bifurcation with the general solutions. The limit cycle is annihilated through a bifurcation in which the limit cycle connects the three general solutions in a heteroclinic cycle at $R = R_g > R_H$. For large enough rotation rates, the limit cycle is stable for all $R > R_H$, as the global bifurcation at R_g is pushed to infinity. Also, as rotation increases, R_r is pushed to infinity, which corresponds to the Küppers-Lortz instability. Otherwise, if the Küppers-Lortz instability is not reached, the only stable solution for large enough values of R is the roll solution.

Swift additionally computed the values of the bifurcation parameter where the saddle node and Hopf bifurcations occur:

$$\begin{aligned} R_{sn} &= \frac{-\xi^2}{4(1+h_2+h_3)} \\ R_H &= \frac{2\xi^2(h_2+h_3+4)}{(h_2+h_3-2)^2} \end{aligned} \tag{1.3}$$

Note that the coefficients h_2 and h_3 always occur in the combination $h_2 + h_3$, indicating that the rate of rotation does not affect R_{sn} and R_H.

In terms of Swift's work, this thesis asks if there are instabilities of hexagons for $R < R_H$ to perturbations outside the hexagonal subspace.

1.4 Organization of Thesis

The thesis is organized as follows. Chapter 2 presents the two phenomenological models studied, and describes the weakly nonlinear analysis method which is used in the study of both models. Chapter 3 presents the results of the weakly nonlinear analysis for, as well as results of direct numerical simulation of, the first phenomenological model. Chapter 4 presents the results of the weakly nonlinear and numerical analyses of the second model. Chapter 5 presents the derived model for rotating Marangoni convection, as well as some analysis. Conclusions are presented in Chapter 6.

Chapter 2

Methodology

The first two models studied in this thesis are models for non-Boussinesq Rayleigh-Bénard convection in the presence of rotation. Both models are based on the Swift-Hohenberg equation [26]

$$\partial_t \psi \quad = \quad R\psi - (\nabla^2 + 1)^2 \psi - \psi^3.$$

The minimal model (2.1) is intended to be the minimal modification of a Swift-Hohenberg equation that exhibits the instability of hexagons to perturbations outside the hexagonal subspace. The extended model (2.10) includes the terms of the minimal model and adds some more terms, allowing for more varied behavior. The Fourier expansion used in the studies of the two models is the same, and will be explained in this chapter. The illustration of the logic behind this expansion draws on examples from both models, which is why it is presented here in a separate chapter. Each section in this chapter is divided into two subsections, one explaining the calculation in the case of the minimal model, the other for the case of the extended model. The two subsections of each section are essentially parallel, with occasional reference from one subsection to the other. In the interest of keeping the discussions of the minimal model (2.1) and the extended model (2.10) as parallel as possible, the discussions contain some redundancies and formalisms.

2.1 The Hexagon Solution

2.1.1 The Minimal Model

We first consider the minimal model for $\psi = \psi(x, y, t)$,

$$\partial_t \psi \quad = \quad R\psi - (\nabla^2 + 1)^2 \psi - \psi^3 + \alpha \psi^2 + \gamma \; \hat{e}_z \cdot (\nabla \psi \times \nabla(\nabla^2 \psi)). \qquad (2.1)$$

Here $\hat{e}_z$ is the unit vector perpendicular to the (x, y)-plane. The bifurcation parameter is the reduced Rayleigh number R, proportional to the temperature difference ΔT across the fluid layer. The ψ^3 term provides nonlinear saturation of the solution. We include the term $\alpha\psi^2$ in order to break the up-down symmetry $\psi \to -\psi$ without breaking the chiral symmetry and thus have the possibility of stable hexagons in the absence of rotation. To break the chiral symmetry nonlinear gradient terms have to be introduced. At leading order in the amplitude and the gradients this introduces an additional quadratic term $\gamma \; \hat{e}_z \cdot (\nabla \psi \times \nabla(\nabla^2 \psi))$. It should be noted that this quadratic gradient term has the somewhat non-generic feature that it generates harmonics only from the interaction of modes with wavevectors of different magnitude. The consequences of this non-genericity will be discussed shortly.

We perform a weakly nonlinear analysis of (2.1). The first step is to find the hexagon solution whose stability we are going to study. We start with a small amplitude expansion of ψ on a hexagonal lattice,

$$\begin{aligned} \psi \quad = \quad & \epsilon \left\{ A e^{ikx} + B e^{ik(-x/2 + y\sqrt{3}/2)} + C e^{ik(-x/2 - y\sqrt{3}/2)} \right\} \qquad (2.2) \\ & + \epsilon^2 \left\{ D_{000} + D_{200} e^{i2kx} + D_{020} e^{ik(-x + \sqrt{3}y)} + D_{002} e^{ik(-x - \sqrt{3}y)} \right. \\ & \quad \left. + D_{1\bar{1}0} e^{ik(3x/2 - y\sqrt{3}/2)} + D_{10\bar{1}} e^{ik(3x/2 + y\sqrt{3}/2)} + D_{01\bar{1}} e^{ik\sqrt{3}y} \right\} \\ & + c.c. + h.o.t., \end{aligned}$$

where $\epsilon \ll 1$. The wavenumber of the pattern k is considered to be close to 1. The Fourier amplitudes are functions of the slow time $\tau_2 = \epsilon^2 t$. The subscripts on the harmonics D_{lmn} indicate how the Fourier mode is expressed in terms of the $O(\epsilon)$ modes; the first, second, and third subscripts indicate how many times e^{ikx},

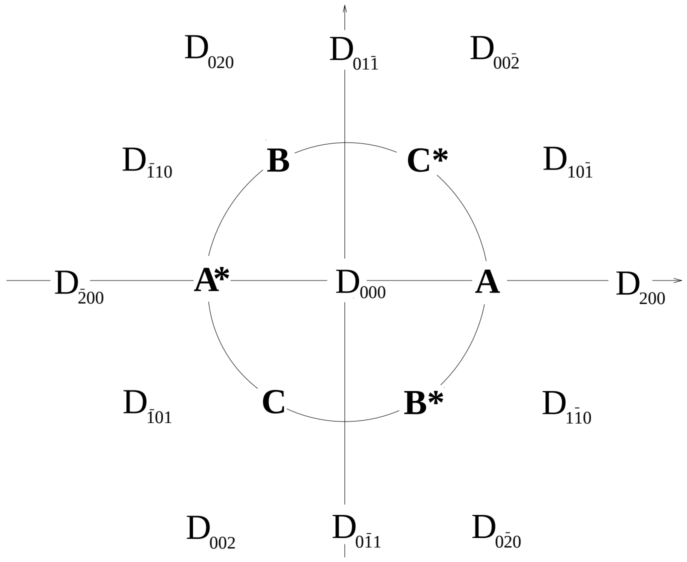


Figure 2.1: Fourier space diagram showing the critical unit circle and location of the modes used to find the hexagon solution (cf. (2.24)).

$e^{ik(-x/2+y\sqrt{3}/2)}$, and $e^{ik(-x/2-y\sqrt{3}/2)}$ are factors of the Fourier mode, respectively. Bars over the subscripts indicate complex conjugates. For example, $D_{10\bar{1}}e^{ik(3x/2+y\sqrt{3}/2)}$ contains one e^{ikx}-mode and one complex conjugate of the $e^{ik(-x/2-y\sqrt{3}/2)}$-mode (cf. Fig 2.1).

We substitute (2.2) into (2.1) and expand in powers of ϵ, keeping only terms of $O(\epsilon^3)$ or less. In order to obtain valid amplitude equations from which to get a hexagon solution, all behavior must be at a single order of ϵ; thus we scale the parameters so that all behavior occurs at $O(\epsilon^3)$. First, we find the neutral curve for the linear stability of the basic solution $\psi = 0$ from the linear part of (2.1) to be $R = (k^2-1)^2$. We expand the bifurcation parameter R about the neutral curve:

$$R_2 = \frac{(R-(k^2-1)^2)}{\epsilon^2}.$$

Here the usual scaling of $k^2 - 1 = O(\epsilon)$ is not done. Thus, if the usual expansion of R about $R = 0$ were done, the equations at $O(\epsilon)$ and $O(\epsilon^2)$ would not be trivial for $k \neq k_c = 1$, and the homogeneous scaling of the hexagon subspace equations at $O(\epsilon^3)$ would be destroyed. Hence, to ensure the homogeneous scaling at $O(\epsilon^3)$, the expansion of R is done about the neutral curve $R = (k^2 - 1)^2$. Even though we do not formally expand k about $k_c = 1$, we do assume that k is close enough to 1 for all scalings dependent upon this closeness to be valid. Additionally, an assumption of weak subcriticality is also necessary for obtaining the homogeneous scaling. Hence, we scale

$$\zeta = \alpha/\epsilon \equiv O(1), \tag{2.3}$$

effectively making $\alpha = O(\epsilon)$.

Having scaled everything properly, we now proceed with the expansion into powers of ϵ. The $O(\epsilon)$ equations are trivial. At $O(\epsilon^2)$, we have only the linear terms of the D_{lmn} amplitudes, whose solution is thus trivial:

$$D_{lmn} = 0. \tag{2.4}$$

It is at $O(\epsilon^3)$ that all relevant behavior is found. Here we have the amplitude equations for A, B, and C,

$$\begin{aligned}
\frac{\partial A}{\partial \tau_2} &= R_2 A + 2\,\zeta B^* C^* + \left(3|A|^2 + 6|B|^2 + 6|C|^2\right) A,\\
\frac{\partial B}{\partial \tau_2} &= R_2 B + 2\,\zeta C^* A^* + \left(3|B|^2 + 6|C|^2 + 6|A|^2\right) B,\\
\frac{\partial C}{\partial \tau_2} &= R_2 C + 2\,\zeta A^* B^* + \left(3|C|^2 + 6|A|^2 + 6|B|^2\right) C.
\end{aligned} \tag{2.5}$$

The chiral symmetry of hexagon subspace is not broken in (2.5). Since the amplitude equations (2.5) result from the substitution of (2.2) into (2.1), the chiral symmetry could only be broken by the cubic cross-coupling terms, as illustrated below. Since these cubic cross-coupling are equal, the amplitude equations (2.5) cannot include the effects of rotation, and thus cannot exhibit the Küppers-Lortz instability.

As mentioned above the only term in (2.1) which breaks the chiral symmetry, $\gamma\ \hat{e}_z \cdot (\nabla\psi \times \nabla(\nabla^2\psi))$, is not a general rotation term. This is because it requires taking the cross product of two modes of unequal Fourier wavelength to be nonzero. To illustrate, the substitution of

$$\psi = A e^{i(q_1 x + p1 y)} + B e^{i(q_2 x + p_2 y)} + c.c.$$

into $\hat{e}_z \cdot (\nabla\psi \times \nabla(\nabla^2\psi))$ yields

$$(q_2p_1 - q_1p_2)(q_2^2 + p_2^2 - q_1^2 - p_1^2)AB(e^{i((q_2+q_1)x+(p_2+p_1)y)} + e^{i((q_2-q_1)x+(p_2-p_1)y)}), \quad (2.6)$$

and similar terms involving the complex conjugates of A and B, with the first factor of the product identical. All these terms (2.6) are nonzero only if the squares of the wavenumbers of the two modes are unequal, $q_2^2+p_2^2 \neq q_1^2+p_1^2$. To get a nonzero value for (2.6) at lowest order within the hexagon subspace, the cross product of a hexagon mode and a quadratic harmonic of a hexagon mode must be considered. Since the coefficient α of ψ^2 is $O(\epsilon)$, making the amplitudes of the quadratic harmonics trivial as shown in (2.4), any contributions from the term $\hat{e}_z\cdot(\nabla\psi\times\nabla(\nabla^2\psi))$ to the hexagon subspace amplitude equations are of higher order. Hence, the chiral symmetry of the amplitude equations (2.5) is not broken. In the extended model, which includes another quadratic term as well as a cubic rotation term, we do have rotation within the hexagon amplitude equations (cf. 2.14).

The amplitude of the hexagon solution, found by setting the time derivatives on the left equal to zero and $A = B = C$ in (2.5) and substituting to express the solution in terms of the original variables is

$$\mathcal{A} = \mathcal{B} = \mathcal{C} = \frac{-\alpha - \sqrt{\alpha^2 - 15(R - (k^2-1)^2)}}{15}. \quad (2.7)$$

From (1.3), the saddle node and Hopf bifurcation points are

$$R_{sn} = (k^2-1)^2 - \frac{\alpha^2}{15}, \quad (2.8)$$

$$R_H = (k^2-1)^2 + \frac{16\,\alpha^2}{3}. \quad (2.9)$$

2.1.2 The Extended Model

We now consider the extended model

$$\begin{aligned} \partial_t\psi &= R\psi - (\nabla^2+1)^2\psi - \psi^3 + \alpha\psi^2 + \beta(\nabla\psi)^2 \quad (2.10) \\ &\quad + \gamma\,\hat{e}_z\cdot(\nabla\psi\times\nabla(\nabla^2\psi)) \\ &\quad + g_2\,\hat{e}_z\cdot(\nabla\times(\nabla\psi)^2\nabla\psi) + g_3(\nabla\cdot(\nabla\psi)^2\nabla\psi), \end{aligned}$$

Again $\hat{e}_z$ is the unit vector perpendicular to the (x, y)-plane, and R, the reduced Rayleigh number, is the bifurcation parameter. There are two non-rotational quadratic terms, $\alpha\psi^2$ and $\beta(\nabla\psi)^2$, each of which individually can ensure the possibility of stable hexagons by breaking the up-down symmetry $\psi \to -\psi$. As in the minimal model, the quadratic term $\gamma\, \hat{e}_k \cdot (\nabla\psi \times \nabla(\nabla^2\psi))$ breaks the chiral symmetry. Additionally, a cubic term $g_2\, \hat{e}_k \cdot (\nabla \times (\nabla\psi)^2\nabla\psi)$ also breaks the chiral symmetry. The term $g_3(\nabla \cdot (\nabla\psi)^2\nabla\psi)$ influences the angle of the new rolls in the Küppers-Lortz instability of rolls [10]. A large value of g_3 can lead to a pattern of stable squares.

To find the steady hexagon solution, we use the same expansion as in the case of the minimal model, (2.2). As before, we substitute (2.2) into (2.10) and expand in powers of ϵ, keeping terms of $O(\epsilon^3)$ or less. In order for all behavior to be at $O(\epsilon^3)$, we again expand the bifurcation parameter R about the neutral curve,

$$R_2 = \frac{(R - (k^2 - 1)^2)}{\epsilon^2}. \tag{2.11}$$

The assumption of weak transcriticality is again necessary. However, it is not necessary in the case of this model to force both quadratic coefficients, α and β, to be of $O(\epsilon)$. It is sufficient for their combined effect in the hexagon subspace amplitude equations to be of $O(\epsilon)$, hence

$$\alpha + \frac{\beta k^2}{2} = \epsilon\zeta. \tag{2.12}$$

We can now proceed with the expansion. Again, at $O(\epsilon)$, all equations are trivial. At $O(\epsilon^2)$, we must again solve for the quadratic harmonics. In the case of the extended model, unlike the minimal model, these solutions are not trivial:

$$\begin{aligned}
D_{000} &= -\frac{\beta\, k^2 AA^* + \beta\, k^2 BB^* + \beta\, k^2 CC^*}{k^4 - 2\,k^2} \\
D_{200} &= \frac{3\,\beta\, k^2 A^2}{-30\,k^4 + 12\,k^2} \\
D_{020} &= \frac{3\,\beta\, k^2 B^2}{-30\,k^4 + 12\,k^2} \\
D_{002} &= \frac{3\,\beta\, k^2 C^2}{-30\,k^4 + 12\,k^2}
\end{aligned} \tag{2.13}$$

$$\begin{aligned} D_{1\bar{1}0} &= \frac{2\,\beta\,k^2 A B^*}{4\,k^2 - 8\,k^4} \\ D_{10\bar{1}} &= \frac{2\,\beta\,k^2 A C^*}{4\,k^2 - 8\,k^4} \\ D_{01\bar{1}} &= \frac{2\,\beta\,k^2 B C^*}{4\,k^2 - 8\,k^4} \end{aligned}$$

The complex conjugates of these equations are also valid and necessary, of course. So it is again at $O(\epsilon^3)$ that all relevant behavior is found:

$$\begin{aligned} \frac{\partial A}{\partial \tau_2} &= R_2 A + 2\,\zeta B^* C^* + \left(f_1 |A|^2 + f_2 |B|^2 + f_3 |C|^2\right) A, \\ \frac{\partial B}{\partial \tau_2} &= R_2 B + 2\,\zeta C^* A^* + \left(f_1 |B|^2 + f_2 |C|^2 + f_3 |A|^2\right) B, \\ \frac{\partial C}{\partial \tau_2} &= R_2 C + 2\,\zeta A^* B^* + \left(f_1 |C|^2 + f_2 |A|^2 + f_3 |B|^2\right) C, \end{aligned} \tag{2.14}$$

where

$$\begin{aligned} f_1 &= 3 + 3\,g_3\,k^4 - \frac{\beta^2 k^4}{k^4 - 2\,k^2} - \frac{3\,\beta^2 k^4}{4\,k^2 - 10\,k^4}, \\ f_2 &= 6 + 3\,g_3\,k^4 - \frac{\beta^2 k^4}{k^4 - 2\,k^2} - \frac{\beta^2 k^4}{k^2 - 2\,k^4} - \frac{\Gamma \beta\,k^6 \sqrt{3}}{2\,k^2 - 4\,k^4} - g_2\,k^4 \sqrt{3}, \\ f_3 &= 6 + 3\,g_3\,k^4 - \frac{\beta^2 k^4}{k^4 - 2\,k^2} - \frac{\beta^2 k^4}{k^2 - 2\,k^4} + \frac{\Gamma \beta\,k^6 \sqrt{3}}{2\,k^2 - 4\,k^4} + g_2\,k^4 \sqrt{3}. \end{aligned} \tag{2.15}$$

In contrast to the minimal model, in the extended model the cubic cross-coupling coefficients f_2 and f_3 are not necessarily equal. When $f_2 = f_3$, there is no rotation in the amplitude equations (2.14). The presence of rotation in the amplitude equations is manifested by $f_2 \neq f_3$. Rotation in the model equation (2.1) is a result of $\gamma \neq 0$ or $g_2 \neq 0$, and this may not necessarily be reflected in the amplitude equations (2.14). Additionally, since the rotation affects f_2 and f_3 with terms of opposite sign, it is always the case that $f_2 + f_3$ does not depend on either γ or g_2.

The amplitude of the hexagon solution is found by setting the time derivatives equal to zero and $A = B = C$ in (2.14) and substituting to express the solution in terms of the original variables,

$$\mathcal{A} = \mathcal{B} = \mathcal{C} = \frac{\alpha + \frac{\beta k^2}{2} + \sqrt{(\alpha + \frac{\beta k^2}{2})^2 + (R - (k^2 - 1)^2)(f_1 + f_2 + f_3)}}{f_1 + f_2 + f_3}, \tag{2.16}$$

where the f_i are as in (2.15). Again, one can apply (1.3) to obtain

$$R_{sn} = (k^2-1)^2 - \frac{(2\,\alpha+\beta\,k^2)^2}{\left(-\frac{18\,\beta^2 k^4}{-15\,k^4+6\,k^2}+60+36\,g_3\,k^4-\frac{12\,\beta^2 k^4}{k^4-2\,k^2}-\frac{32\,\beta^2 k^4}{4\,k^2-8\,k^4}\right)}, \tag{2.17}$$

$$R_H = (k^2-1)^2 + \frac{2\,(2\,\alpha+\beta\,k^2)^2}{\left(24-\frac{8\,\beta^2 k^4}{4\,k^2-8\,k^4}-\frac{6\,\beta^2 k^4}{k^4-2\,k^2}+18\,g_3\,k^4-\frac{18\,\beta^2 k^4}{-15\,k^4+6\,k^2}\right)\left(6-\frac{8\,\beta^2 k^4}{4\,k^2-8\,k^4}+\frac{9\,\beta^2 k^4}{-15\,k^4+6\,k^2}\right)^2}.$$

Note that because the saddle node and Hopf bifurcations equations (1.3) contain h_2 and h_3 (equivalent to f_2 and f_3) only in the combination h_2+h_3, the rotation in the amplitude equations of the extended model, as modeled by the coefficients g_2 and γ, is not relevant to the calculation of the saddle node and Hopf bifurcation points. This is true even if rotation is manifest in the amplitude equations (2.14).

2.2 The Perturbation Expansion

We now perturb the weakly nonlinear hexagon subspace solutions of the model equations in order to study the linear stability of the hexagon solutions. This perturbation analysis preserves the isotropy of the system, which the numerical analysis does not. The anisotropy of the numerical analysis will be discussed later. For both the minimal and extended models, the linear stability calculation results in a 6 by 6 matrix for the coupled linear equations governing the growth of the perturbation modes. In this section we describe the reasons for using the perturbation expansion that is used by starting with a naive perturbation expansion and demonstrating its inadequacy.

2.2.1 The Minimal Model

To study the stability of the hexagon solution $H(x,y)$ a general Floquet ansatz with modulation wavevector $(\tilde{q},\tilde{p})$ is made,

$$\psi = H(x,y) + \delta e^{i\tilde{q}x+i\tilde{p}y} P(x,y,t), \tag{2.18}$$

where $P(x,y,t)$ has the spatial periodicity of the hexagonal lattice and $\delta \ll 1$. Being interested in the weakly nonlinear regime, we expand $H(x,y)$ and $P(x,y,t)$ in ϵ and obtain

$$\begin{aligned}
\psi \;=\; & \epsilon\left\{Ae^{ikx} + Be^{ik(-x/2+y\sqrt{3}/2)} + Ce^{ik(-x/2-y\sqrt{3}/2)}\right\} \\
& + \epsilon\delta\left\{P_{000}e^{i(qx+py)}\right\} \\
& + \epsilon^2\left\{D_{000} + D_{200}e^{i2kx} + D_{020}e^{ik(-x+\sqrt{3}y)} + D_{002}e^{ik(-x-\sqrt{3}y)}\right. \\
& \quad \left. + D_{1\bar{1}0}e^{ik(3x/2-y\sqrt{3}/2)} + D_{10\bar{1}}e^{ik(3x/2+y\sqrt{3}/2)} + D_{01\bar{1}}e^{ik\sqrt{3}y}\right\} \\
& + \epsilon^2\delta\left\{H_{0\bar{1}0}e^{i((q+k/2)x+(p-k\sqrt{3}/2)y)} + H_{100}e^{i((q+k)x+py)} + H_{00\bar{1}}e^{i((q+k/2)x+(p+k\sqrt{3}/2)y)}\right. \\
& \quad \left. + H_{\bar{1}00}e^{i((q-k)x+py)} + P_{010}e^{i((q-k/2)x+(p+k\sqrt{3}/2)y)} + P_{001}e^{i((q-k/2)x+(p-k\sqrt{3}/2)y)}\right\} \\
& + c.c. + h.o.t.
\end{aligned} \tag{2.19}$$

with the amplitudes of the perturbation modes depending on time. Here the notation for the $O(\epsilon)$ and $O(\epsilon^2)$ modes is as before. The subscripts on the $O(\epsilon^2\delta)$ amplitudes indicate how many times the $e^{i(qx+py)}$-mode (i.e. P_{000}) was multiplied by the indicated Fourier components that make up the original hexagon solution to create the harmonic mode (cf. 2.2). The reason for labelling some of the $O(\epsilon^2\delta)$ modes as H and some as P will become apparent below. We assume that the P and H amplitudes depend on three time scales, e.g. $P_{000} = P_{000}(t,\tau_1,\tau_2)$, where $\tau_1 = \epsilon t$ and $\tau_2 = \epsilon^2 t$. In the expansion (2.19) we have combined the modulation wavevector $(\tilde{q},\tilde{p})$ with the basic wavevector $(k,0)$ of the hexagon pattern into the perturbation wavevector $(q = k+\tilde{q}, p = \tilde{p})$. Thus, long-wave perturbations are characterized by $|\tilde{q},\tilde{p}|$ being small, i.e. (q,p) being close to $(k,0)$. Further, we define θ to be the angle between the wavevectors associated with the modes $P_{000}e^{i(qx+py)}$ and Ae^{ikx}, i.e.

$$\theta = arctan\left(\frac{p}{q}\right). \tag{2.20}$$

We substitute (2.19) into (2.1) and expand in powers of ϵ and δ, keeping only up to cubic order in ϵ and linear order in δ. At $O(\epsilon^2)$ and $O(\epsilon^3)$ we obviously get the same equations as before, and the same hexagon solution. At $O(\epsilon\delta)$, we have

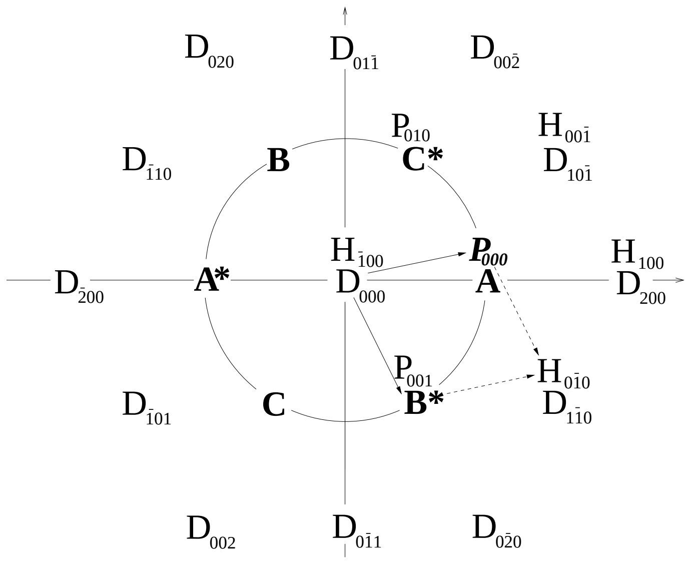


Figure 2.2: Fourier space diagram showing the critical unit circle and location of the modes used in the finite angle expansion of ψ (cf. (2.24)). Indicated with arrows is an example of the combination of an $O(\epsilon\delta)$ perturbation mode and an $O(\epsilon)$ original hexagon mode driving a $O(\epsilon^2\delta)$ higher harmonic mode.

a nontrivial equation for P_{000}:

$$\frac{\partial P_{000}}{\partial t} = (-q^4 + k^4 - 2\,k^2 + 2\,p^2 - 2p^2q^2 - p^4 + 2q^2)P_{000}. \tag{2.21}$$

We are assuming that both the hexagon wavenumber k and the perturbation wavenumber $\sqrt{q^2+p^2}$ are close to $k_c = 1$, meaning that the terms on the right hand side of (2.21) which appear to be $O(\epsilon\delta)$ are actually of higher order, specifically $O(\epsilon^3\delta)$.

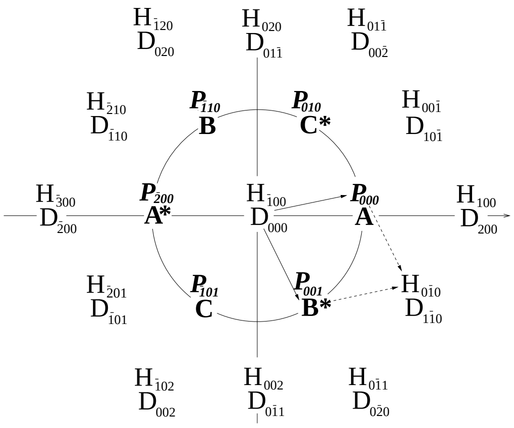
The perturbation expansion as written in (2.19) is flawed, however. At $O(\epsilon^2\delta)$, we get equations which we are to solve for the modes of that order. The equations for P_{010} and P_{001},

$$\frac{\partial P_{010}}{\partial t} = -\left(p^2 + p\sqrt{3} + q^2 - q\right)^2 P_{010} + BP_{000}f(q,p,k=1;\alpha,\gamma), \tag{2.22}$$
$$\frac{\partial P_{001}}{\partial t} = -\left(p^2 - p\sqrt{3} + q^2 - q\right)^2 P_{001} + CP_{000}f(q,p,k=1;\alpha,\gamma), \tag{2.23}$$

are to be solved for P_{010} and P_{001}, respectively, by setting the derivatives on the left hand side to zero, thereby slaving these modes to P_{000}. But (2.22) and (2.23) cannot be solved for P_{010} and P_{001} when $(q,p) \to (1,0)$, i.e. $\theta \to 0$ and $\sqrt{q^2+p^2} \to 1$, because the coefficients of the amplitudes P_{010} and P_{001} go to zero, and the solutions for the amplitudes diverge. In other words, when $(q,p) \to (1,0)$, the modes associated with P_{010} and P_{001} approach the critical circle in Fourier space and their linear damping goes to zero, making the solutions for the amplitudes diverge. Since these modes are no longer strongly damped, their corresponding amplitudes P_{010} and P_{001} can no longer be adiabatically eliminated. The divergence of the amplitudes, and hence of the growth rate of the instability for $\theta = 0$ is shown graphically for the case of the extended model in Fig.2.5, which has the same divergence when the expansion (2.19) is used.

In order to avoid the singularities, P_{010} and P_{001} are allowed to be of $O(\epsilon\delta)$, that is, of the same order as the original perturbation mode P_{000}. This makes P_{010} and P_{001} independent perturbation modes. Then at $O(\epsilon^2\delta)$ they drive the modes $P_{\bar{1}01}$ and $P_{\bar{1}10}$ leading to singularities in these modes for $\theta \to 0$, analogous to the equations (2.22) and (2.23). Hence, modes $P_{\bar{1}01}$ and $P_{\bar{1}10}$ must also be allowed to be of $O(\epsilon\delta)$. In fact, all modes that come close to the critical circle as $e^{i(qx+py)}$ approaches e^{ix} must be kept at $O(\epsilon\delta)$. In practice, this includes just one more mode, $P_{\bar{2}00}e^{i((q-2k)x+py)}$. They are labelled P_{lmn}. This finally leads to the following expansion for ψ (cf. Fig. 2.3),

$$
\begin{aligned}
\psi \;=\;& \epsilon\left\{Ae^{ikx} + Be^{ik(-x/2+y\sqrt{3}/2)} + Ce^{ik(-x/2-y\sqrt{3}/2)}\right\} \qquad (2.24)\\
&+ \epsilon\delta\left\{P_{000}e^{i(qx+py)} + P_{010}e^{i((q-k/2)x+(p+k\sqrt{3}/2)y)} + P_{001}e^{i((q-k/2)x+(p-k\sqrt{3}/2)y)}\right.\\
&\left. + P_{\bar{1}10}e^{i((q-3k/2)x+(p+k\sqrt{3}/2)y)} + P_{\bar{1}01}e^{i((q-3k/2)x+(p-k\sqrt{3}/2)y)} + P_{\bar{2}00}e^{i((q-2k)x+py)}\right\}\\
&+ \epsilon^2\left\{D_{000} + D_{200}e^{i2kx} + D_{020}e^{ik(-x+\sqrt{3}y)} + D_{002}e^{ik(-x-\sqrt{3}y)}\right.\\
&\left. + D_{1\bar{1}0}e^{ik(3x/2-y\sqrt{3}/2)} + D_{10\bar{1}}e^{ik(3x/2+y\sqrt{3}/2)} + D_{01\bar{1}}e^{ik\sqrt{3}y}\right\}\\
&+ \epsilon^2\delta\left\{H_{\bar{1}00}e^{i((q-k)x+py)} + H_{100}e^{i((q+k)x+py)} + H_{00\bar{1}}e^{i((q+k/2)x+(p+k\sqrt{3}/2)y)}\right.\\
&+ H_{0\bar{1}1}e^{i(qx+(p-k\sqrt{3})y)} + H_{002}e^{i((q-k)x+(p-k\sqrt{3})y)} + H_{\bar{1}02}e^{i((q-2k)x+(p-k\sqrt{3})y)}\\
&+ H_{\bar{2}01}e^{i((q-5k/2)x+(p-k\sqrt{3}/2)y)} + H_{\bar{3}00}e^{i((q-3k)x+py)} + H_{\bar{2}10}e^{i((q-5k/2)x+(p+k\sqrt{3}/2)y)}\\
&+ H_{\bar{1}20}e^{i((q-2k)x+(p+k\sqrt{3})y)} + H_{020}e^{i((q-k)x+(p+k\sqrt{3})y)} + H_{01\bar{1}}e^{i(qx+(p+k\sqrt{3})y)}
\end{aligned}
$$



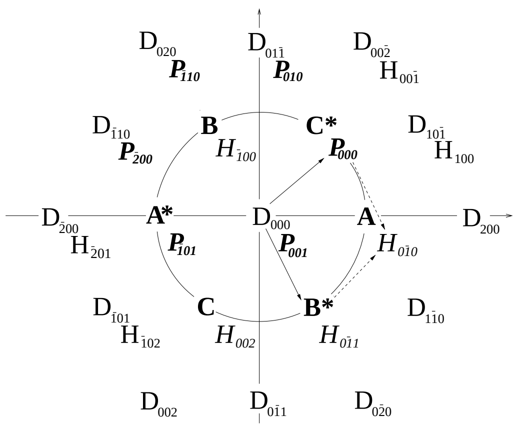


Figure 2.3: Fourier space diagram showing the critical unit circle and location of the modes used in the final expansion of ψ (cf. (2.24)). Indicated with arrows is an example of the combination of an $O(\epsilon\delta)$ perturbation mode and an $O(\epsilon)$ original hexagon mode driving a $O(\epsilon^2\delta)$ higher harmonic mode.

Figure 2.4: Fourier space diagram where $\theta \to 60°$. Here the modes $H_{0\bar{1}0}$, $H_{0\bar{1}1}$, H_{002}, and $H_{\bar{1}00}$ approach the critical unit circle despite being of $O(\epsilon^2\delta)$, and thus drive a singularity when solved for.

$$+ \ H_{0\bar{1}0}e^{i((q+k/2)x+(p-k\sqrt{3}/2)y)}\Big\} + c.c. + h.o.t.$$

The amplitudes labelled $H_{\tilde{l}\tilde{m}\tilde{n}}$ are the amplitudes of those harmonics of the original perturbation mode P_{000} and the hexagon modes which are not near the critical circle $k = k_c$ when θ is near zero (cf. Fig.2.3).

Certain values of (q, p) drive a resonance that is still not accounted for in the scaling of the modes. For $k = 1$, as θ approaches $60°$ the expansion breaks down again because the modes corresponding to $H_{0\bar{1}0}, H_{0\bar{1}1}, H_{\bar{1}00}$, and H_{002} are now near the critical circle (cf. Fig. 2.4), but are still kept at $O(\epsilon^2\delta)$ in the expansion (2.24). Thus, when one attempts to adiabatically eliminate one of these modes as $\theta \to 60°$, one gets equations analogous to (2.22) and (2.23) and solving for the desired amplitudes is impossible, since the solutions diverge. Similarly, the expansion breaks down as θ approaches $-60°$. Due to the $60°$-periodicity of the expansion, we may restrict ourselves to θ between $-30°$ and $30°$, and thus avoid having to do an altered calculation for these situations. If $k = 0.888$, a resonance occurs already for $\theta = \pm 30°$. This is due to the mode H_{002} being close to the critical circle. Since $k = 0.888$ is smaller than any hexagon wavenumber we consider, restricting θ to the range $-30°$ to $30°$ is adequate.

The extended model (2.10) uses the same expansion (2.24) as the minimal model (2.1). Fig.2.5 demonstrates that the influence of the singularities at $\theta = 0$ and $\theta = 60°$ when using (2.19) is so great that the eigenvalue calculations become unreliable over the whole range $0° < \theta < 60°$.

To summarize, in the expansion (2.24) all modes are retained that are excited at $O(\epsilon^2)$ or $O(\epsilon^2\delta)$ by the perturbation modes or the hexagon modes. This is sufficient to determine the growth rates of the perturbation modes at $O(\epsilon^3\delta)$. The expansion remains well ordered as long as the hexagon wavenumber and the perturbation wavenumber are close to 1 and the angle θ is between $-30°$ and $+30°$.

Now, having the final expansion we are going to use, we substitute said expansion (2.24), into (2.1), and expand in powers of ϵ and δ. Again, the equations for the the pure ϵ orders are just those from which we got the hexagon solution. At $O(\epsilon\delta)$, we again have equation (2.21) and five other equations analogous to it, for the other five P modes:

$$
\begin{aligned}
\frac{\partial P_{000}}{\partial t} &= \left(k^2 - 2 + q^2 + p^2\right)\left(k^2 - q^2 - p^2\right) P_{000} \\
\frac{\partial P_{010}}{\partial t} &= \left(2\,k^2 - qk + pk\sqrt{3} + q^2 + p^2 - 2\right)\left(-q^2 - p^2 + qk - pk\sqrt{3}\right) P_{010} \\
\frac{\partial P_{001}}{\partial t} &= \left(2\,k^2 - qk - pk\sqrt{3} + q^2 - 2 + p^2\right)\left(-q^2 - p^2 + qk + pk\sqrt{3}\right) P_{001} \\
\frac{\partial P_{\bar{1}10}}{\partial t} &= -\left(2\,k^2 - pk\sqrt{3} - 3\,qk + q^2 + p^2\right)\left(q^2 - 2 + p^2 - 3\,qk - pk\sqrt{3} + 4\,k^2\right) P_{\bar{1}10} \\
\frac{\partial P_{\bar{1}01}}{\partial t} &= -\left(2\,k^2 + pk\sqrt{3} - 3\,qk + q^2 + p^2\right)\left(q^2 - 2 + p^2 - 3\,qk + pk\sqrt{3} + 4\,k^2\right) P_{\bar{1}01} \\
\frac{\partial P_{\bar{2}00}}{\partial t} &= -\left(q^2 - 2 + p^2 - 4\,qk + 5\,k^2\right)\left(q^2 + p^2 - 4\,qk + 3\,k^2\right) P_{\bar{2}00}
\end{aligned}
\tag{2.25}
$$

The first equation of (2.25) is identical to (2.21). All the terms on the right hand sides of equations (2.25) are considered to be of higher order, specifically of $O(\epsilon^3\delta)$, due to the proximity of $\sqrt{q^2 + p^2}$ to 1.

At $O(\epsilon^2\delta)$, we have 13 equations for the H_{lmn} amplitudes. Like the D_{lmn}

amplitudes, these are futher away from the critical unit circle, and therefore strongly linearly damped, and as such are adiabatically eliminated. The equations are of the form,

$$H_{\tilde{l}\tilde{m}\tilde{n}} = \frac{\gamma P_{lmn} \mathcal{A} f(k,q,p)}{g(k,q,p)}, \tag{2.26}$$

where $g(k,q,p)$ is related to the wavenumber of the Fourier mode associated with the amplitude, P_{lmn} is an appropriate P amplitude, and $\mathcal{A}$ is replaced by $\mathcal{B}$ or $\mathcal{C}$ as appropriate. The H modes are eliminated in favor of the P modes and the hexagon modes A, B, and C. They are then substituted for in the equations at $O(\epsilon^3\delta)$.

Additionally, at $O(\epsilon^2\delta)$ there are terms of the form

$$\frac{\partial P_{lmn}}{\partial \tau_1} = \frac{\gamma P_{\tilde{l}\tilde{m}\tilde{n}} \mathcal{A} f(k,q,p)}{g(k,q,p)}. \tag{2.27}$$

Note that the subscripts lmn and $\tilde{l}\tilde{m}\tilde{n}$ as used in this thesis are not related across different equations or sentences, but were merely used to denote generic subscripts. The right hand sides of equations (2.27) are considered to be of higher order, specifically $O(\epsilon^3\delta)$. For $\theta \to 0$, the factor $f(k,q,p)$ is small, making the term on the right hand side of (2.27) of $O(\epsilon^3\delta)$. For finite θ, the modes $P_{\tilde{l}\tilde{m}\tilde{n}}$ are small, again pushing the right hand side to $O(\epsilon^3\delta)$.

All quadratic terms (2.26) and (2.27) are driven by the rotation term $\gamma\ \hat{e}_z \cdot (\nabla\psi \times \nabla(\nabla^2\psi))$, since α is of $O(\epsilon)$. As noted above, the solutions for the H amplitudes in (2.26) are substituted into the $O(\epsilon^3\delta)$ equations. The equations (2.27) remain at $O(\epsilon^2\delta)$. However, like the terms at $O(\epsilon\delta)$ (equation (2.21)), the terms in (2.27) are considered small, and thus actually of $O(\epsilon^3\delta)$. However, this case is a bit more complex. It turns out that despite the assumption of weak subcriticality, (2.3), the coefficients of the quadratic terms are not small in general. While the coefficients go to zero for $\theta \to 0$ and $\zeta = 0$, they are of $O(1)$ for finite angles θ. For finite angles and k close to 1, however, the P amplitudes to which H is coupled are small rendering the quadratic terms again of the same order as the cubic terms. Thus the consideration of the terms in the $O(\epsilon^2\delta)$ equations (2.27) as being $O(\epsilon^3\delta)$ is justified.

At $O(\epsilon^3\delta)$ we thus obtain six evolution equations for the P modes. However, there is relevant information at all three orders kept in the calculation, including $O(\epsilon\delta)$ (equation (2.21)), $O(\epsilon^2\delta)$ (equation (2.27)). To obtain a unified description, we reconstitute the perturbation equations at all three orders by substituting in for all scaled amplitudes, e.g. $A = \mathcal{A}/\epsilon$, $P_{lmn} = \mathcal{P}_{lmn}/\epsilon\delta$, etc., as well as the scaled variables, $\zeta = \alpha/\epsilon$ and $R_2 = (R-(k^2-1)^2)/\epsilon^2$, and the scaled times, $\partial_T = \partial_t + \epsilon\partial_{\tau_1} + \epsilon^2\partial_{\tau_2}$. This leads to six equations of the form

$$\begin{aligned} \partial_T \mathcal{P}_{000} &= (R-(q^2+p^2-1)^2)\mathcal{P}_0 \\ &\quad + \mathcal{B}^*\mathcal{P}_{010}f_{010}(q,p,k;\alpha,\gamma) + \mathcal{C}^*\mathcal{P}_{001}f_{001}(q,p,k;\alpha,\gamma) \\ &\quad + \sum_{lmn} f_{lmn}(q,p,k;\alpha,\gamma)|\mathcal{A}|^2\mathcal{P}_{lmn}. \end{aligned} \tag{2.28}$$

The equations for the $\mathcal{P}_{lmn}$ amplitudes follow from symmetry. This generates a full 6 by 6 matrix, whose eigenvalues give us the linear growth rates of the perturbations $\mathcal{P}_{lmn}$. It is the stability of the initial hexagon to the perturbations $\mathcal{P}_{lmn}$ according to the eigenvalues of this matrix that we study.

2.2.2 The Extended Model

The perturbation analysis of the extended model parallels that of the minimal model. The main difference is in the dependence of the coefficients of the amplitude equations on the coefficients of (2.10). Again, we make a general Floquet ansatz (cf. (2.18)) with modulation wavevector $(\tilde{q},\tilde{p})$. With the extended model we illustrate the effect of the singularity at $\theta = 0°$ graphically. The finite angle expansion (2.19) is invalid for the same reasons as in the case of the minimal model. In the case of the extended model, (2.19) leads to the following equations at $O(\epsilon^2\delta)$:

$$\frac{\partial P_{010}}{\partial t} = -(p^2+p\sqrt{3}+q^2-q)^2 P_{010} + BP_{000}f_{010}(q,p,k=1;\alpha,\beta,\gamma,g_2,g_3), \tag{2.29}$$
$$\frac{\partial P_{001}}{\partial t} = -(p^2-p\sqrt{3}+q^2-q)^2 P_{001} + CP_{000}f_{010}(q,p,k=1;\alpha,\beta,\gamma,g_2,g_3). \tag{2.30}$$

These equations are the same as (2.22) and (2.23) except for the dependence of f_{010} and f_{001} on the parameters of (2.10).

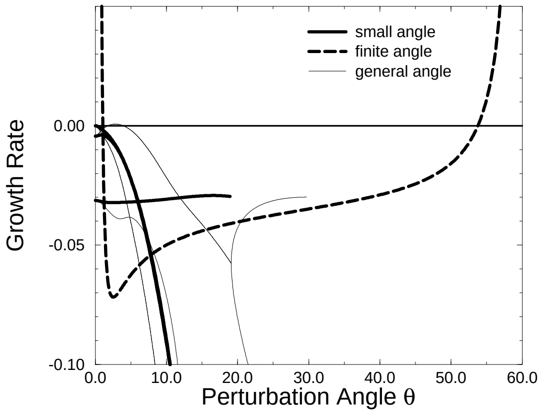


Figure 2.5: An illustration of the inadequacy of the separate small angle and finite angle calculations. The parameters are $R = 0.08$, $k = 1.00 = \sqrt{q^2 + p^2}$, $\alpha = 0$, $\beta = 0.2$, $\gamma = 6$, $g_2 = 0$, $g_3 = 2$.

For the parameters $R = 0.08$, $k = 1.00 = \sqrt{q^2 + p^2}$, $\alpha = 0$, $\beta = 0.2$, $\gamma = 6$, $g_2 = 0.0$, $g_3 = 2$, the singularity is shown in the graph Fig.2.5. The finite angle calculation, based on (2.19), blows up at $\theta = 0°$ and $\theta = 60°$, as discussed above. As can be seen in Fig.2.5, the dependence of the finite angle expansion eigenvalues on θ is very weak, and the singularities do indeed dominate the calculation. A potential correction for the singularity at $\theta = 0$ was a small angle expansion, with asymptotic matching connecting the two solutions. The small angle expansion was done using (2.24), but expanding the perturbation wavenumber (q, p) about $(1, 0)$, scaled with ϵ. We, however, were unable to match the solutions. The calculation done in the end, using (2.24) to do a general angle stability analysis for all θ, is also shown. The general angle calculation does not agree very well with the small angle calculation except in a very small neighborhood of $\theta = 0$. In fact, the small angle calculation misses the instability found by the general angle calculation for $\theta \approx 4°$. Additionally, the finite angle calculation also does not agree well with the general angle calculation. Hence, we feel justified in using (2.24) to perform a general angle stability calculation as discussed.

Again, we have the same restrictions in the use of (2.24) as in the minimal model, namely $k > 0.888$ and $-30° < \theta < 30°$. Again, there is relevant behavior at three orders, $O(\epsilon\delta)$, $O(\epsilon^2\delta)$, and $O(\epsilon^3\delta)$. The same substitutions are used to reconstitute the equations as are used in the minimal model, with the exception of $\zeta = (\alpha + \beta/2)/\epsilon$. The result is six equations of the form

$$\begin{aligned} \partial_T \mathcal{P}_{000} &= (R - (q^2 + p^2 - 1)^2)\mathcal{P}_{000} \\ &+ \mathcal{B}^* \mathcal{P}_{010} f_{010}(q, p, k; \alpha, \beta, \gamma) + \mathcal{C}^* \mathcal{P}_{001} f_{001}(q, p, k; \alpha, \beta, \gamma) \\ &+ \sum_{lmn} f_{lmn}(q, p, k; \alpha, \beta, \gamma, g_2, g_3) |\mathcal{A}|^2 \mathcal{P}_{lmn}. \end{aligned} \tag{2.31}$$

The equations for the $\mathcal{P}_{lmn}$ amplitudes again follow from symmetry, leading to a full 6 by 6 matrix for the linear growth rates of the perturbations $\mathcal{P}$.

Chapter 3

The Minimal Model

3.1 Long-wave Analysis

Using the same expansion (2.24) for ψ as in the general expansion, we perform a long-wave analysis. This expansion is valid for (q, p) near $(k, 0)$, and $k \approx k_c = 1$. For the minimal model, a long-wave expansion leads to a relatively simple set of two equations, which can be analyzed much more easily than the six coupled equations determining the general linear stability of the hexagon solution. We expand the wavenumbers of both the hexagon solution and the perturbation modes around the critical wavenumber $k_c = 1$,

$$\begin{aligned} k &= 1 + \epsilon k_1, \\ q &= 1 + \epsilon m, \\ p &= \epsilon n. \end{aligned} \tag{3.1}$$

We substitute (2.24) and (3.1) into (2.1) and expand as before in powers of ϵ and δ. Due to the expansion of the wavenumbers about the critical wavenumber 1, both $O(\epsilon)$ and $O(\epsilon\delta)$ are trivial. Again due to the long-wave expansion, the equations at $O(\epsilon^2)$ and $O(\epsilon^2\delta)$ are trivial, and we have trivial solutions for all the adiabatically eliminated quadratic harmonic modes,

$$D_{lmn} = 0,$$

$$H_{\tilde{l}\tilde{m}\tilde{n}} = 0.$$

In this long-wave expansion, all the behavior is at $O(\epsilon^3)$ and $O(\epsilon^3\delta)$, and so, in contrast to the general expansion, reconstitution by substituting for the scaled amplitudes is not necessary to combine terms which are apparently but not really of different orders. However, the scalings are removed from the hexagon solutions and the evolution equations for the long-wave perturbation amplitudes. After removal of the scalings, the long-wave expansion gives the same hexagon solution (2.7) as the general expansion. We remove the scalings with the substitutions

$$\begin{aligned} \zeta &= \frac{\alpha}{\epsilon}, \\ R_2 &= \frac{R-(k^2-1)^2}{\epsilon^2}, \\ m &= \frac{q-1}{\epsilon}, \\ n &= \frac{p}{\epsilon}, \\ k_1 &= \frac{k-1}{\epsilon}. \end{aligned} \tag{3.2}$$

At order $O(\epsilon^3\delta)$, we again get six linear equations for the six P amplitudes, with each amplitude cross-coupled with every other amplitude. Due to the linearity of the equations, we assume $P_{lmn} \sim e^{\sigma t}$, where σ is the linear growth rate of P. Above, we expanded k, q, and p about the critical wavenumber $k_c = 1$ because the long-wave expansion is only valid near the critical wavenumber. Also, the long-wave expansion consists of keeping the perturbation wavenumbers near the hexagon wavenumber. Additionally, the growth rate of the perturbation σ is to be expanded about $\sigma_c = 0$, because for the perturbation wavenumber equalling the hexagon wavenumber (which is what we are expanding around) the growth rate $\sigma_c = 0$ exactly. Thus, we have the following,

$$\begin{aligned} q &= k + \nu m_1, \\ p &= \nu n_1, \\ \sigma &= \nu^2 \sigma_2, \end{aligned} \tag{3.3}$$

where $\nu \ll 1$ is another small parameter, independent of ϵ and δ. Substituting these into the reconstituted equations for the P_{lmn} amplitudes, we expand in ν. We take the determinant of the 6 by 6 matrix whose eigenvalues give us the solutions

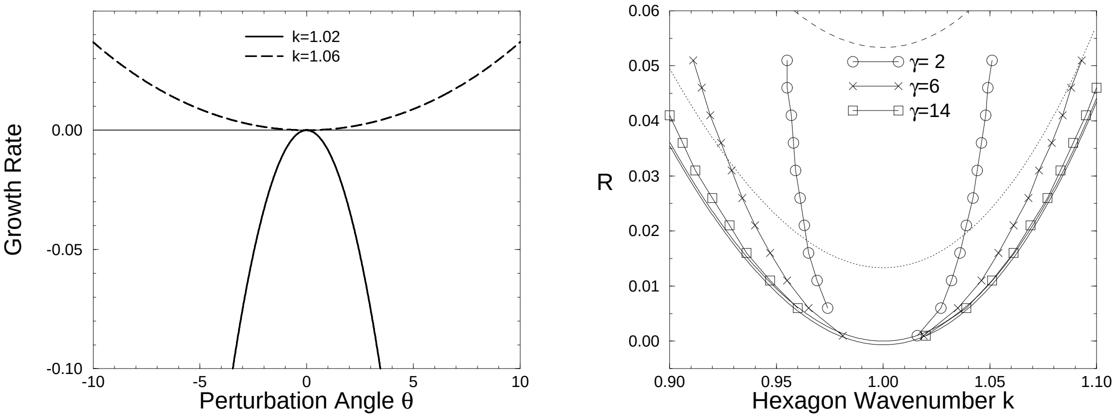


Figure 3.1: *a)* Results of the long-wave calculation. In each case of k, the two real eigenvalues are visually indistinguishable from one another. The parameters are $R = 0.02$, $\alpha = 0.1$, $\gamma = 6$. *b)* The long-wave stability limits for $\gamma = 2$, $\gamma = 6$, and $\gamma = 14$. As before, the two solid lines near each other are the neutral and saddle node curves, the dashed line near the top of the figure is the Hopf curve, and the dotted line in the middle is the lower stability limit of the rolls.

for the growth rate σ (in the case of expansion in ν, we find σ_2). There are only two relevant eigenvalues at $\theta = 0$ (specifically, the translation eigenvalues, see next section), and so expansion in powers of ν yields a quadratic equation at first order. The equations for the first three powers of ν are trivial, and at fourth order in ν we have a quadratic equation for σ_2 resulting in the two eigenvalues sought. From this quadratic equation we can very easily see some properties of the long-wave expansion, mostly concerning what occurs for $k_c = 1$. The equation at bandcenter is

$$0 = {\sigma_2}^2 - \tau\sigma_2 + \Delta, \tag{3.4}$$

where

$$\begin{aligned} \tau &= \left(\frac{9}{4}\,\frac{\mathcal{A}\,(q^2+p^2)\,\gamma^2}{3\,\mathcal{A}-2\,\alpha} - 4\,q^2 - 4\,p^2\right), \qquad (3.5)\\ \Delta &= \frac{81}{64}\,\frac{\mathcal{A}^2\,(q^2+p^2)^2\,\gamma^4}{(3\,\mathcal{A}-2\,\alpha)^2} - \frac{9}{2}\,\frac{\mathcal{A}\,(q^2+p^2)^2\,\gamma^2}{3\,\mathcal{A}-2\,\alpha} + 3\,\left(q^2+p^2\right)^2, \end{aligned}$$

and $\mathcal{A}$ denotes the hexagon solution (2.7). From (3.4), the long-wave eigenvalues at bandcenter are

$$\lambda_{1,2} = \frac{1}{2}\left(\tau \pm \sqrt{\tau^2 - 4\Delta}\right) = \frac{\tau}{2} \pm (q^2 + p^2). \tag{3.6}$$

We note that

$$\tau^2 - 4\Delta = 4(q^2 + p^2)^2 > 0, \tag{3.7}$$

thus both eigenvalues at bandcenter are real.

Note that the Hopf bifurcation (1.3) occurs for $3\mathcal{A} - 2\alpha = 0$. Since we consider values of the bifurcation parameter R below the Hopf curve, i.e. $R < R_H$, we restrict $3\mathcal{A} - 2\alpha < 0$. It is clear that with this restriction $\tau < 0$ and $\Delta > 0$ always (cf. (3.5)), making the bandcenter always long-wave stable. There is a long-wave instability at bandcenter above the Hopf curve, but not below.

Away from bandcenter, the dependence of τ and Δ on the hexagon wavenumber k cannot be eliminated, allowing for the possibility of complex eigenvalues away from bandcenter. The value of the rotation term coefficient $\gamma = \gamma_c$ for which we get complex eigenvalues is found by solving $\tau^2 - 4\Delta = 0$,

$$\gamma_c = \pm\frac{1}{6}\frac{(3\mathcal{A} - 2\alpha)(-15\mathcal{A}^2 + \alpha\mathcal{A} + 8(k-1)^2)}{(k-1)\mathcal{A}\sqrt{(\alpha + 12\mathcal{A})(15\mathcal{A} - \alpha)}}. \tag{3.8}$$

Note the $(k-1)$ factor in the denominator, indicating that this solution for γ_c diverges at bandcenter.

Since we have only a quadratic equation (3.4) for σ_2, the two long-wave eigenvalues provide an easily calculated outer stability boundary for the hexagon solution, as shown in Fig.3.1. Fig.3.1a shows the eigenvalues as a function of θ for $R = 0.02$, $\alpha = 0.1$, and $\gamma = 6$, and two values of the hexagon wavenumber $k = 1.02$ and $k = 1.06$, where $\sqrt{q^2 + p^2} = k$. For each k, the two real eigenvalues are indistinguishable from one another in the figure. Both eigenvalues are zero for $\theta = 0$, since they correspond to translations in the horizontal plane, to which the solution is neutrally stable. As noted above, at bandcenter $k = 1.00$, the hexagon solution is stable to long-wave perturbations. This is further illustrated in Fig.3.1b, where we see the stability boundaries for various values of rotation γ. The stronger the rotation, the further away from bandcenter the long-wave stability boundary is. For values of the bifurcation parameter R further away from the neutral curve, finite angle perturbations become relevant, and limit the stability region of the hexagons more stringently than the long-wave expansion would. However, near the neutral curve, the long-wave stability limit is relevant, and is a good check on the linear stability calculation for general θ which follows.

3.2 Classification of Eigenvalues

The general weakly nonlinear stability analysis expansion (2.24) leads to 6 eigenvalues. To provide a context we first discuss the special case that the perturbation modes coincide with the original hexagon modes (i.e. $(q, p) = (k, 0)$). The analysis is then equivalent to that of the amplitude equations (1.2), which can be rewritten as equations for three real amplitudes and three phases. Due to translation symmetry in two directions, two of the phases decouple and the eigenvalues corresponding to the translation modes are identically zero. All of the relevant behavior is contained in the three real amplitude equations and one phase equation. As long as the up-down symmetry is broken sufficiently strongly the remaining phase relaxes quickly to a stable fixed point, and only three real amplitude equations are needed, corresponding to three eigenvalues instead of six. Hence comparison with the systems of Swift [20] and Soward [21] is justified.

As noted, in this thesis we are interested in perturbation modes that differ in their wavevector from the hexagon modes (i.e. $(q, p) \neq (k, 0)$) and consider the dependence of the eigenvalues on (q, p). Thus, we deal with six branches of eigenvalues. Away from the saddle-node bifurcation and for sufficiently strongly broken up-down symmetry only four branches are relevant. Two of them are associated with the instability of hexagons to either the general solution (without rotation) or the oscillating hexagons (with rotation). We call the eigenvalues on these branches the 'Hopf eigenvalues.' The other two branches are associated with the translation modes of the hexagons; we call the eigenvalues on these branches 'translation

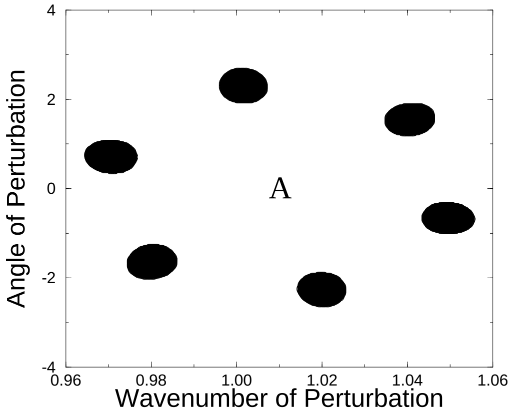


Figure 3.2: Fourier space diagram with the shaded areas indicating perturbations $\mathcal{P}_{000}$ to which the hexagons in the minimal model are unstable for $R = 0.0294$, $k = 1.01$, $\alpha = 0.1$, $\gamma = 6$. The sixfold symmetry is due to the rotation symmetry. In this case the instability first occurs at a finite angle θ.

eigenvalues.' Note that they are identically zero only for $(q,p) = (k,0)$, i.e. for $\sqrt{q^2+p^2} = k$ and $\theta = 0$. For other perturbation wavevectors they are non-zero. In fact, in the presence of rotation the translation eigenvalues need not even be real but can form a complex pair. In the following we discuss the dependence of these four eigenvalues on the magnitude and, in particular, on the orientation of the wavevector of the perturbation. As discussed in the previous chapter, we restrict ourselves to the range $-30° < \theta < 30°$ for the angle between the perturbation mode P_{000} and mode A.

In Fourier space, the growth rates exhibit a sixfold symmetry of areas of instability, centered around the wavevectors of the hexagon modes $(k, \theta = 2\pi n/6)$, due to the rotation symmetry of the system (cf. [41]). Fig.3.2 shows the instability regions for $\mathcal{P}_{000}$ in the vicinity of mode $\mathcal{A}$ for $k = 1.01$. In Fourier space (cf. Fig. 2.3), a rotation of $\mathcal{P}_{000}$ about $\mathcal{A}$ by 60° is equivalent to a rotation of the system about the origin by 60° in the same direction. The clockwise rotation of $\mathcal{P}_{000}$ about $\mathcal{A}$ by 60° makes its position equivalent to that of $\mathcal{P}_{010}$ before the rotation, the position of $\mathcal{P}_{010}$ after the rotation equivalent to that of $\mathcal{P}_{\bar{1}10}$ before the rotation, and so on.

Hence, the stability calculation exhibits a sixfold symmetry of unstable perturbation modes.

We consider only those parameter values for which the hexagons are stable to perturbations within the hexagonal lattice. This means that we consider values of the bifurcation parameter R between the saddle-node and the Hopf bifurcation points, namely R_{sn} and R_H. Hence, the Hopf eigenvalues have negative real part for $(q,p) = (k,0)$ in all our calculations. Hence, we are searching for parameter values where the Hopf eigenvalues have negative real part at $(q,p) = (k,0)$, but either the Hopf or the translation eigenvalues have positive real part for $(q,p) \neq (k,0)$.

3.3 Linear Stability Results

In the minimal model (2.1) we only found side-band instabilities involving the two branches of perturbations that are connected with the translation modes, i.e. the growth rates of the perturbations vanish as the perturbation wavevector (q,p) approaches that of a hexagon mode.

For small R the instabilities are long-wave and the wavevector of the most unstable perturbation is close to that of a hexagon mode, i.e. the instabilities arise at a small angle θ between the perturbation modes and the hexagon modes. The eigenvalues are real for sufficiently small rotation rates (Fig.3.4a), but become complex as the rotation coefficient $|\gamma|$ is increased. In these illustrations the magnitude of the perturbation wavevector is chosen to be equal to that of the hexagons, $\sqrt{q^2+p^2} = k$. The corresponding pairs of real eigenvalues that are to become complex conjugate pairs under the influence of rotation at first do so at small but nonzero θ (Fig.3.3). As rotation increases, the range of θ for which we have complex conjugate eigenvalues does as well; however, the possibility of real eigenvalues exists for any rotation rate, with the real eigenvalues generally occurring for larger $|\theta|$. For larger rotation rates, the translation eigenvalues are complex for a relatively large neighborhood of $\theta = 0$, and the long-wave instabilities are oscillatory, as in Fig.3.4b.

Further away from threshold the instabilities become short-wave, as in

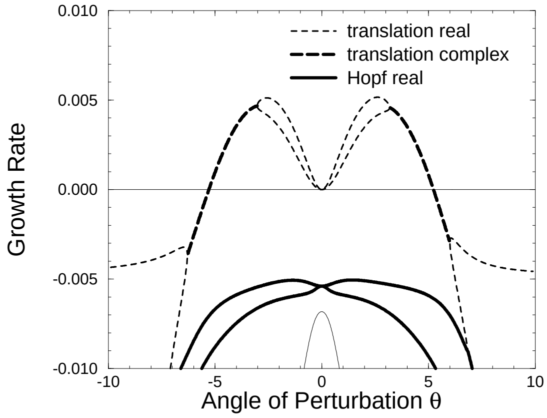


Figure 3.3: Without rotation, i.e. $\gamma = 0$, all six eigenvalues are real for all values of θ. As $|\gamma|$ increases, complex conjugate pairs of eigenvalues begin to appear. This usually occurs first for small but nonzero values of θ, as illustrated. The eigenvalue represented by the thin solid line near the bottom of the figures is never relevant. The parameters are $R = 0.01$, $k = 1.045 = \sqrt{q^2 + p^2}$, $\alpha = 0.1$, $\gamma = 0.5$.

Fig.3.4c. For small angles the perturbations are damped. Only for larger angles the real part of the growth rate (dashed line) becomes positive. In this case the relevant eigenvalues are complex. It is worth mentioning that the translation branch and the Hopf branch (solid line) influence each other in this case for small angles. This is discussed in more detail in the chapter on the extended model (cf. Fig.4.3 below).

The general stability limits obtained for (2.1) are shown in Fig.3.5 for a range of rotation rates γ. In these calculations the perturbation wavenumber is allowed to vary within the range $0.9 \leq \sqrt{q^2 + p^2} \leq 1.1$ and the angle within the range $-30° \leq \theta \leq 30°$. We note that variation of the non-rotational quadratic terms, which in the case of the minimal model is only α, simply varies the strength of the subcriticality of the hexagon solution and shifts the instability of the hexagons to rolls at $R = R_H$ up and down, thus setting the over-all scale for the stability range of the hexagons; no qualitative changes in the stability limits were found. The solid lines in Fig.3.5 indicate the neutral curve and the locus where the hexagons first arise in a saddle-node bifurcation. The dashed line indicates where the hexagons

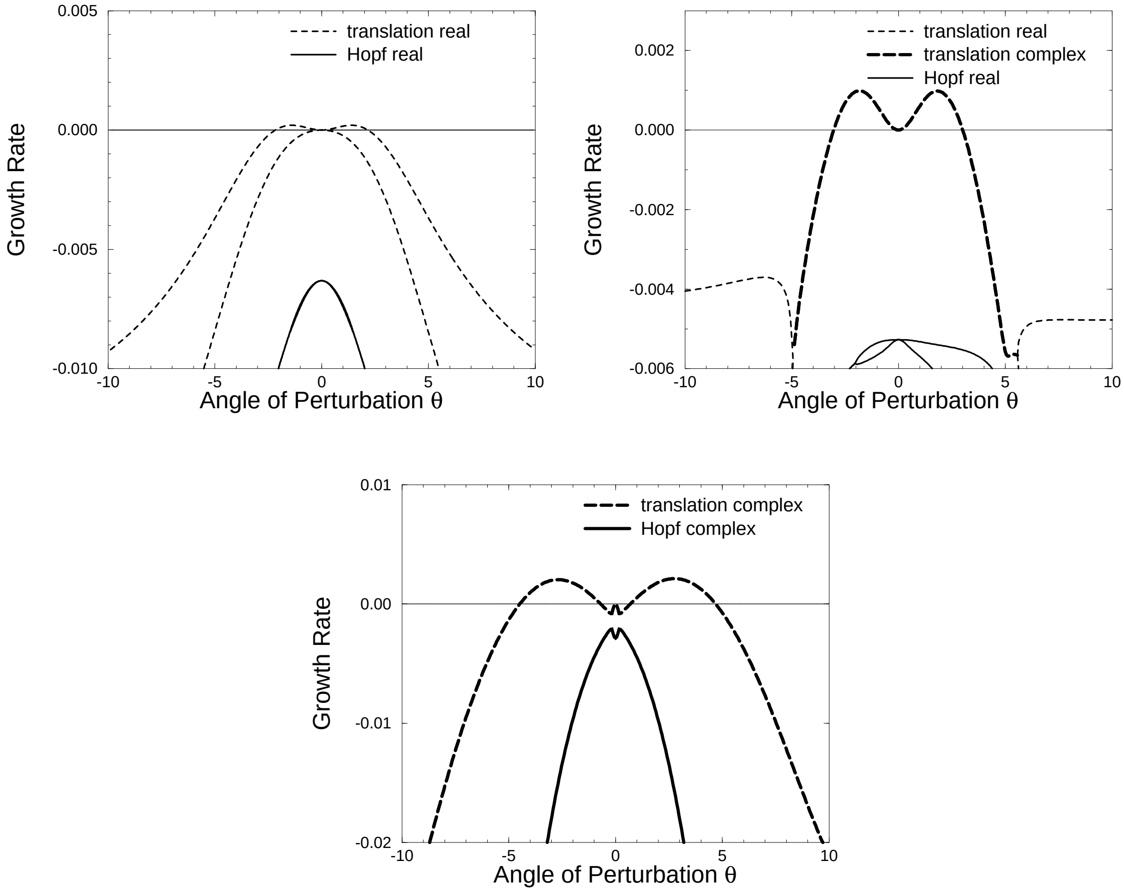


Figure 3.4: *a)* A steady long-wave instability for $R = 0.02$, $k = 1.027 = \sqrt{q^2 + p^2}$, $\alpha = 0.1$, and $\gamma = 0.1$. *b)* An oscillatory long-wave instability for $R = 0.005$, $k = 1.03 = \sqrt{q^2 + p^2}$, $\alpha = 0.1$, and $\gamma = 2$. *c)* A short-wave instability driven by the translation modes in the minimal model for $R = 0.04$, $k = 1.01 = \sqrt{q^2 + p^2}$, $\alpha = 0.1$, and $\gamma = 6$.

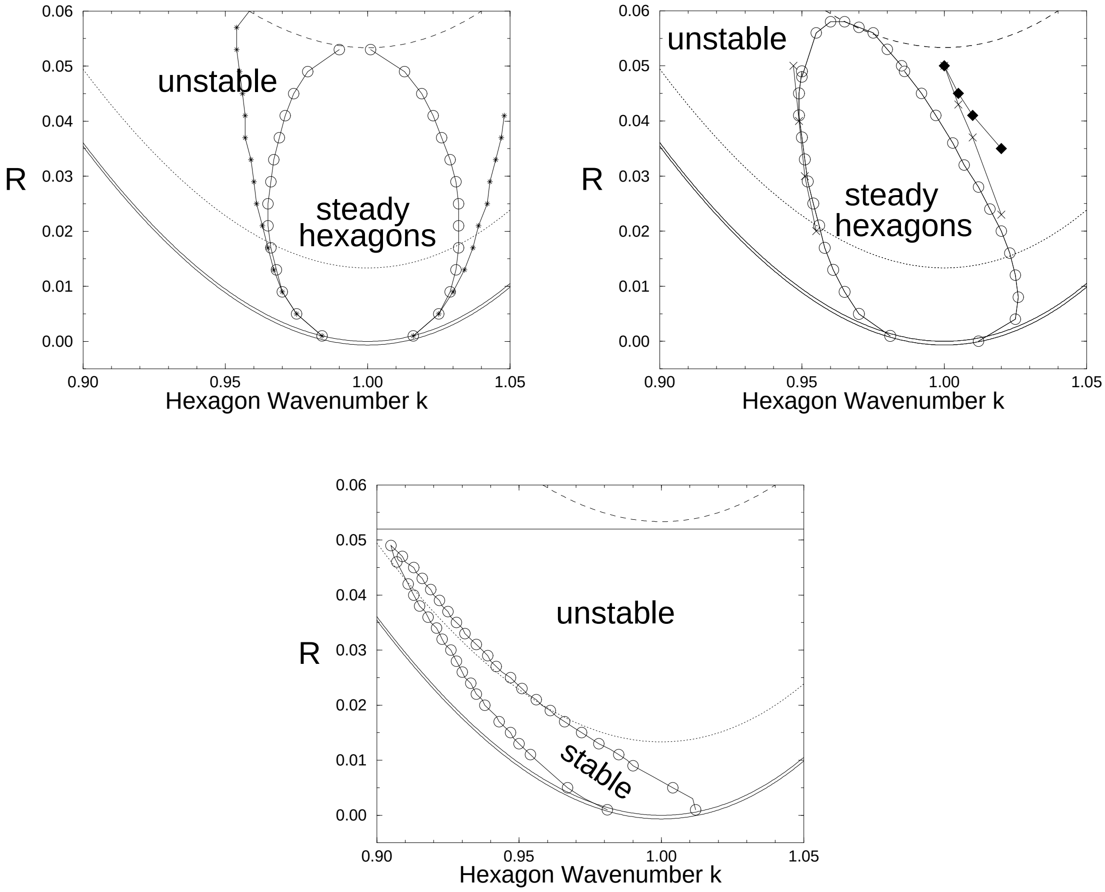


Figure 3.5: Stability regions for the minimal model for various values of γ. Open circles denote sideband instability in all cases. Solid lines: neutral curve and saddle-node bifurcation. Dashed line: instability of hexagons to rolls. Dotted line: lower stability boundary of the rolls (cf. Fig. 1.3). *a)* $\gamma = 2$. Solid symbols denote long-wave instability. *b)* $\gamma = 6$. Crosses denote the numerical linear stability limit of the hexagons, diamonds denote the stability limit of the modulated hexagons. *c)* $\gamma = 14$. Line denotes value of R of numerical investigation (see Fig.3.14 below).

become unstable to rolls $R = R_H$. In the case of the minimal model, the rotation coefficient γ does not affect the hexagon subspace equations, and thus the instability of hexagons to rolls at $R = R_H$ is always a transcritical bifurcation of the hexagons with the general solution. The dotted line indicates the lower stability limit of rolls with respect to hexagons. The case of low rotation $\gamma = 2$ is shown in Fig.3.5a. The side-band instability of the hexagons is shown with open circles. Its long-wave limit is given with small, solid symbols. As mentioned above, the long-wave result gives the correct stability limit for small R. For larger R, however, the short-wave instability sets in first. As the rotation rate is increased the stability limits become more asymmetrical as shown in Fig.3.5b. Also shown in this case (as crosses on both sides of the stability region) are the stability limits as they are obtained in full numerical simulations of (2.1). The weakly nonlinear and numerical stability boundaries agree better for values of R near the neutral curve. The diamonds denote the stability boundary of the modulated hexagon solution, which will be discussed in the section on numerical results. For sufficiently large rotation rates the stability region of the hexagons detaches from the line that indicates the transition to the general solution. Such a regime is shown in Fig.3.5c for $\gamma = 14$. Thus, over a range of the control parameter R the hexagons are still stable with respect to rolls, but nevertheless at no wavenumber are the hexagons stable with respect to side-band instabilities. This suggests the possibility of persistent dynamics, which is discussed below. A comparison may be made between the general angle and long-wave stability limits for the cases $\gamma = 6$ and $\gamma = 14$, as shown in Fig.3.5b,c and Fig.3.1b. The long-wave instabilities are relevant for R near the neutral curve in all cases. But further away from the neutral curve, the short-wave instabilities become relevant, especially for the right side of the region of stable hexagons.

3.4 Numerical Simulations

To support the analytical calculation, and to study the nonlinear behavior ensuing from the instabilities, numerical simulations of (2.1) were performed. A

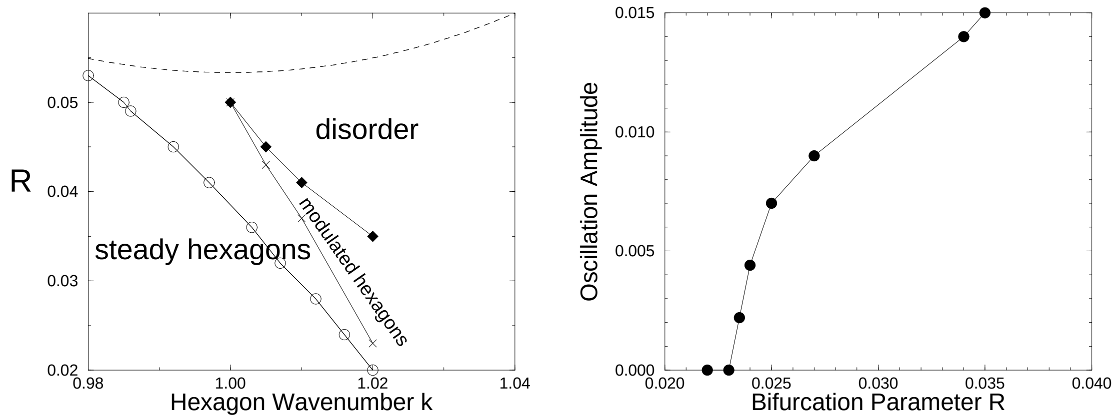


Figure 3.6: *a)* A detail of stability diagram Fig.3.5b. Circles: stability boundary of hexagons from weakly nonlinear analysis; crosses: numerical stability boundary of hexagons on the 128 by 128 lattice; solid diamonds: stability boundary of the modulated hexagon state. *b)* The supercritical bifurcation of the modulated hexagons from the stable hexagon state, occurring at the lined denoted by the crosses in *a*. Shown is the amplitude of the oscillations of the perturbation mode labeled P in the other figures.

Runge-Kutta method with an integrating factor that computes the linear derivative terms exactly was used. Derivatives were computed in Fourier space, using a two-dimensional complex fast Fourier transform (FFT).

The numerical simulations were done in a rectangular 32π by $64\pi/\sqrt{3}$ box of aspect ratio $2/\sqrt{3}$ with periodic boundary conditions. The aspect ratio of $2/\sqrt{3}$ was used to allow for regular hexagonal patterns. For the small amplitudes investigated here it was sufficient to use a 128 by 128 lattice keeping only up to the third harmonics of the modes on the critical circle. This allowed a larger number of modes within the critical annulus, reducing the effects of anisotropy.

Simple two-dimensional aliasing results in the higher harmonics of all modes (except those purely horizontal, vertical, or on a diagonal in Fourier space) being aliased to wavevectors with a different direction. This introduces a slight artificial anisotropy in the numerical system. The effect was most noticeable in our investigations of the heteroclinic cycle that results from the Küppers-Lortz instability of rolls within the three-mode model of Busse and Clever [16], or May and Leonard

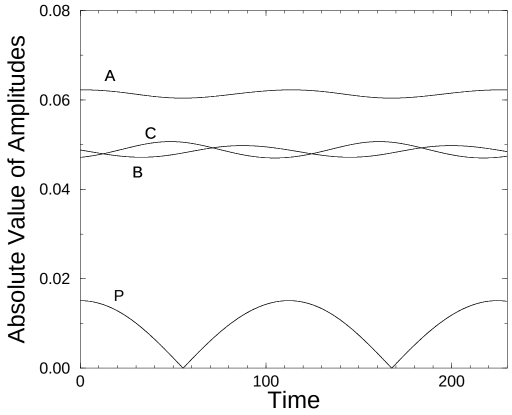


Figure 3.7: The time dependence of the absolute values of the hexagon solution amplitudes as they are modulated by the steady oscillations of the perturbation mode, whose time dependence is also shown.

[19]. Due to the anisotropy the system ends up preferentially in the same roll state independent of initial conditions. However, outside of such sensitive areas, the overall effect of the anisotropy is small. The code was checked against previous results [19, 20, 22].

In all cases investigated, if the instability is associated with a real eigenvalue it does not saturate and eventually leads again to steady hexagons with a wavevector in the stable region. This is true for the usual long-wave case close to onset as well as for the case when the instability arises first at a finite modulation wavevector.

The oscillatory instabilities exhibit more interesting behavior. While they also can take the hexagon pattern back into the stable wavenumber band, over a range of parameters they lead in a supercritical bifurcation to modulated hexagons (cf. Fig.3.6b). They are quite similar to the oscillating hexagons identified earlier by Swift [20] and Soward [21] in that the hexagons are temporally modulated in time with modes corresponding to different orientations being phase-shifted with respect to each other. In the modulated hexagons the oscillations occur, however, predominantly in three side-band perturbation modes, which implies also a spatial modulation of the hexagon pattern. A typical time sequence over one period

$T = 225$ is displayed in Fig.3.7 and Fig.3.8. Fig.3.7 shows the three hexagon modes A, B, and C as well as the main numerical perturbation mode P. Mode P is very close to A in Fourier space (Fig.3.9); in fact P is the next grid point in the Fourier space grid used in the numerical calculation, thus generating the longest modulation wavelength possible. Due perhaps to the differing relations of the main perturbation modes to the main hexagon modes and the critical circle (Fig. 3.9, discussed below), resulting in different linear growth rates for the perturbation modes, the effect of the modulation on the hexagon amplitudes A, B, and C varies. The three hexagon amplitudes all oscillate in time (Fig.3.7), but they oscillate about different values, and out of phase with each other. The amplitude of the oscillations of any of the hexagon amplitudes is small compared to the amplitude of the oscillations of the perturbation amplitude P. Note that the amplitude of the perturbation mode is actually alternately positive and negative; its magnitude is plotted here. We see the correlation between the oscillation of mode P in Fig.3.7 and the four panels of Fig.3.8. Since the wavevector (q, p) of the sideband perturbation is close to that of the steady hexagons, only one modulation wavelength fits into the system. Fig.3.8 also shows that the spatio-temporal modulation is in the form of a standing rather than a traveling wave. When the amplitude of the time dependence of amplitude P is large, at $t = 0$ and $t = 110$, the zero level contours connect near the points in space where the amplitude of the modulation is greatest. This state is shown in the first and third panels of Fig.3.8. In the first panel, the spatial amplitude of the modulation is greatest in the middle of the domain, and in the third panel it is greatest at the top and bottom boundaries of the domain. When the amplitude of the time dependence of P is near zero, as at $t = 50$ and $t = 170$, the zero level contours show a hexagonal pattern over the whole spatial domain (panels two and four). The spatial Fourier transform of the modulated hexagons is shown in Fig.3.9. Visible are the amplitudes of the hexagon modes and their quadratic harmonics. Near the hexagon modes are vertical bands of excited modes; these are the perturbation amplitude P (which is the next grid point to the hexagon amplitude A) and the harmonics of P and the hexagon amplitudes. This relatively

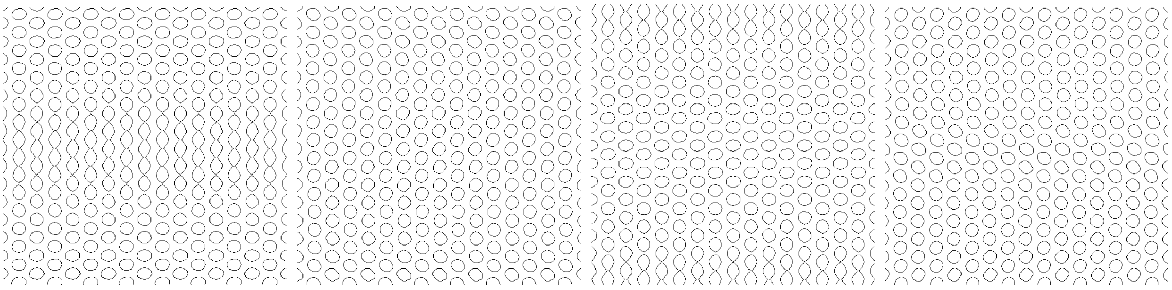


Figure 3.8: A real-space illustration of the stable oscillations of the perturbation mode in the modulated hexagon regime for $R = 0.0355$, $k = 1.02$, $\alpha = 0.1$, $\beta = 0$, $\gamma = 6$, $g_2 = 0$ at times $t = 0$, $t = 50$, $t = 110$, $t = 170$. Indicated in the figure are the zero level contours of the solution.

simple Fourier spectrum can be contrasted with the relative disorder of the Fourier spectra in Fig.3.10c and Fig.3.14a. The modulated hexagons are stable only in a small region beyond the stability limit of the steady hexagons as delimited by the crosses and diamonds in Fig.3.5b and in Fig.3.6a.

When the wavenumber k or the control parameter R is increased beyond the line marked by diamonds in Fig.3.5b and Fig.3.6a the modulated hexagons become unstable and the modulation pattern breaks down, leading to a temporally and spatially chaotic state. Typical snapshots are displayed in Fig.3.10a,b. They suggest a strong signature of a hexagonal symmetry and a predominant orientation of the disordered pattern which rotates in time. In Fig.3.10a we see a state which is dominated by three sets of rolls, oriented at 120° to each other. The orientations of the dominant rolls as they appear in the real space diagram (*not* the Fourier space diagram) are shown above each zero-level contour panel. In the areas of the real space domain where the amplitudes of the three dominant rolls are fairly equal, the zero-level contours show a hexagonal cell structure. In areas of the real space domain where one of the three roll amplitudes is dominant, the zero-level contours indicate a roll structure. In Fig.3.10b we have the zero-level contours at a time shortly after that of Fig.3.10a, where the solution is again dominated by three sets of rolls oriented at 120° to each other. As shown by the diagrams of the dominant rolls, the rolls of Fig.3.10b are rotated by 30° with respect to those of Fig.3.10a. This

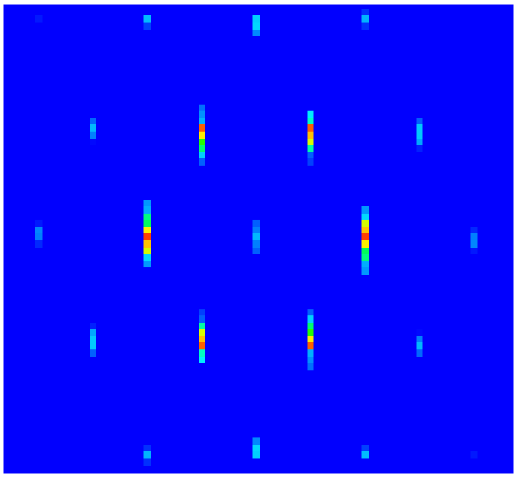


Figure 3.9: The Fourier spectrum of the modulated hexagon state on a logarithmic scale, corresponding to Fig.3.8. The only Fourier modes that are not miniscule are those of the hexagon solution and their harmonics within the hexagon subspace, as well as the perturbation modes and the harmonics generated from the interaction of the perturbation mode and the hexagon modes.

indicates that the underlying roll structure of the real space patterns is rotating in time. However, the real-space pattern is not rotating rigidly. To understand the rotation of the dominant underlying roll structure, we consider the complex spatial Fourier transform of the solution. The Fourier spectrum of Fig.3.10a is pictured in Fig.3.10c, which shows six dominant peaks (see also Fig.3.15 below). The temporal evolution of the Fourier spectrum is depicted in Fig.3.11 and, on a longer time scale, Fig.3.12a. Plotted is a space-time diagram of the radially integrated power spectrum, that is, the integral of the Fourier spectrum along the radial line defined by θ, from the $k = 0.7$ to $k = 1.3$. The horizontal axis is the the angle θ, the vertical axis is time. Since the solution is real, the Fourier spectrum is symmetric about any line through the origin. Hence, only the range $-90^o < \theta < 90^o$ is shown. White denotes a high intensity, black the lowest. In Fig.3.11, the sixfold symmetry is rotating regularly in time on a large scale. However, all quantities are in fact chaotic, as may be seen from a more detailed examination of Fig.3.11. The rotation of the spectrum is in fact seen to be surprisingly pronounced and steady over long periods of time, which are intermittently interrupted by periods during which the peaks in the spectrum are less pronounced (Fig.3.12a). Fig.3.12a spans a representative $T_{shown} = 32,000$ time units, out of a total run of $T_{max} = 278,880$ time units. The space-time autocorrelation $\mathcal{G}$ function of Fig.3.12a is given in Fig.3.12b, using the function

$$\mathcal{G}(g, g) = \int_{-\infty}^{\infty} g(\tau + t) g(\tau) d\tau. \tag{3.9}$$

The azimuthal decay of $\mathcal{G}$ is seen to be quite slow. This can be seen more clearly in Fig.3.12c, which shows a cut through Fig.3.12b for $\theta = 0$. The initial azimuthal decay away from $t = 0$ is clear; however, further away the autocorrelation regains strength. We suspect this is due to the periodicity assumed when computing the fast Fourier transform of Fig.3.12a. We do not expect that the sixfold symmetry of the power spectrum persists as the system size is increased. In fact, the highest peaks in the power spectrum decrease by roughly a factor of two if the linear dimensions of the system are doubled. However, Fig.3.12 indicates that the disordered state at least *locally* exhibits a structure with hexagonal symmetry, which rotates in time.

Another illustration of this phenomenon is provided by Fig.3.10d, which is a time series of the original hexagon Fourier modes, the perturbation mode, and another Fourier mode as the steady oscillations of the modulated hexagon state break down. To construct the figure, a stable modulated hexagon state is established, and then the bifurcation parameter R is increased past the stability boundary of the modulated hexagons, denoted by the diamonds in Fig.3.6a. Initially, the modulation of the hexagons continues at the higher value of R; however, this modulation is unstable. The unstable steady oscillations of the perturbation mode break down as other excited modes other than the principal oscillating perturbation P mode grow and destroy the pattern. We see the Fourier modes then peak regularly, corresponding to the rotation of the Fourier spectrum of the solution as discussed above.

The modulated hexagons and the disordered hexagons shown in Fig.3.8 and Fig.3.10 are obtained in the minimal model (2.1) and arise from the short-wave instability connected with the translation mode. Very similar results are also

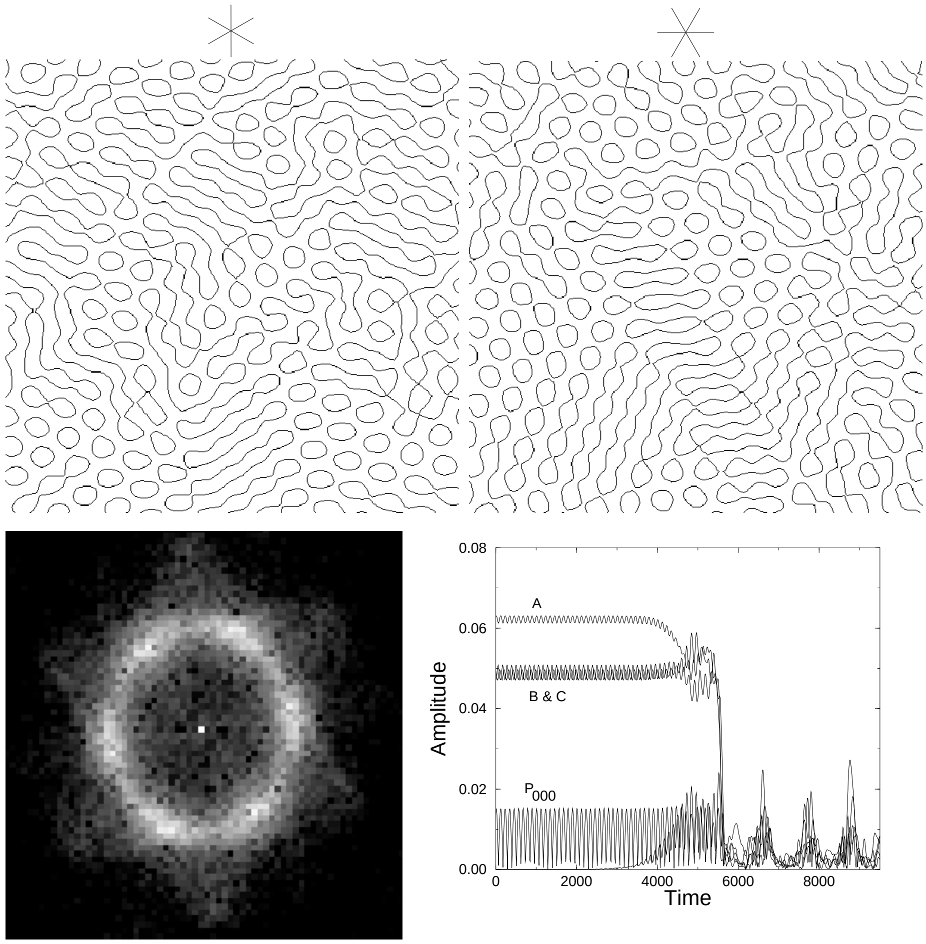


Figure 3.10: *a)* Zero-level contours in real space for time $t = 118,580$. Above the panel are the directions of the three dominant rolls as they appear in real space (*not* Fourier space). *b)* Zero-level contours at a time when the power spectrum has rotated by about 30^o. *c)* Power spectrum on a logarithmic scale of pattern at time $t = 118,580$ for $\alpha = 0.1$, $\gamma = 6$, $R = 0.036$, $k = 1.02$. Note the residual sixfold symmetry. *d)* Time plot of hexagon amplitudes, along with oscillating perturbation mode P_{000}, and another Fourier mode (not corresponding to any of the modes we consider analytically), which gets excited and destroys the modulation.

obtained in the more general model (2.10) where they can also arise from the short-wave Hopf mode. Although the Hopf and the translation modes may have similar short-wave behavior in these cases, their long-wave behaviors are different. Since the parameters under consideration are below Swift's Hopf curve, the real part of the Hopf eigenvalues approach a negative value as $(q,p) \to (1,0)$. The real part of the translation eigenvalues always approaches zero as $(q,p) \to (1,0)$. Hence, for the nonlinear evolution of the instability the long-wave behavior of the branch of destabilizing modes is apparently not relevant. This seems to corroborate the apparent occasionally indistinguishability of the the translation and Hopf eigenvalue branches, as discussed in the next chapter (cf. Fig.4.3).

An interesting aspect of the disordered state of Fig.3.10 is that it exists stably for identical values of the control parameter as the steady hexagons and the modulated hexagons (cf. Fig.3.6). This coexistence of ordered and disordered states is reminiscent of the situation in Rayleigh-Bénard convection with small Prandtl number where the state of spiral-defect chaos coexists with that of straight parallel rolls [42–44].

Fig.3.10c is significant in a slightly different way as well. Almost all the modes near the critical circle are excited, indicating that a purely long-wave stability analysis of the hexagon solution would not capture all the behavior seen in the analysis of (2.1). A long-wave analysis would be valid only for perturbations near the hexagon subspace modes in Fourier space, and thus short-wave instabilities would not be detected properly. An illustration of this is in Echebarria and Riecke [3], who use a Ginzburg-Landau approach to study the long-wave stability of a hexagon state. In doing a numerical simulation of the Ginzburg-Landau type model, they find that an instability that originates at a perturbation number near the hexagon wavenumber (analogous to $(q,p) \approx (k,0)$), which is allowed in the long-wave model, nonlinearly excites short-wave modes, which the long-wave model cannot resolve. This is illustrated in Fig.3.13, which shows the spatial Fourier transform of two snapshots in time of a numerical simulation of the Ginzburg-Landau type model equation used. Shown is only a small part of the spatial Fourier transform, near

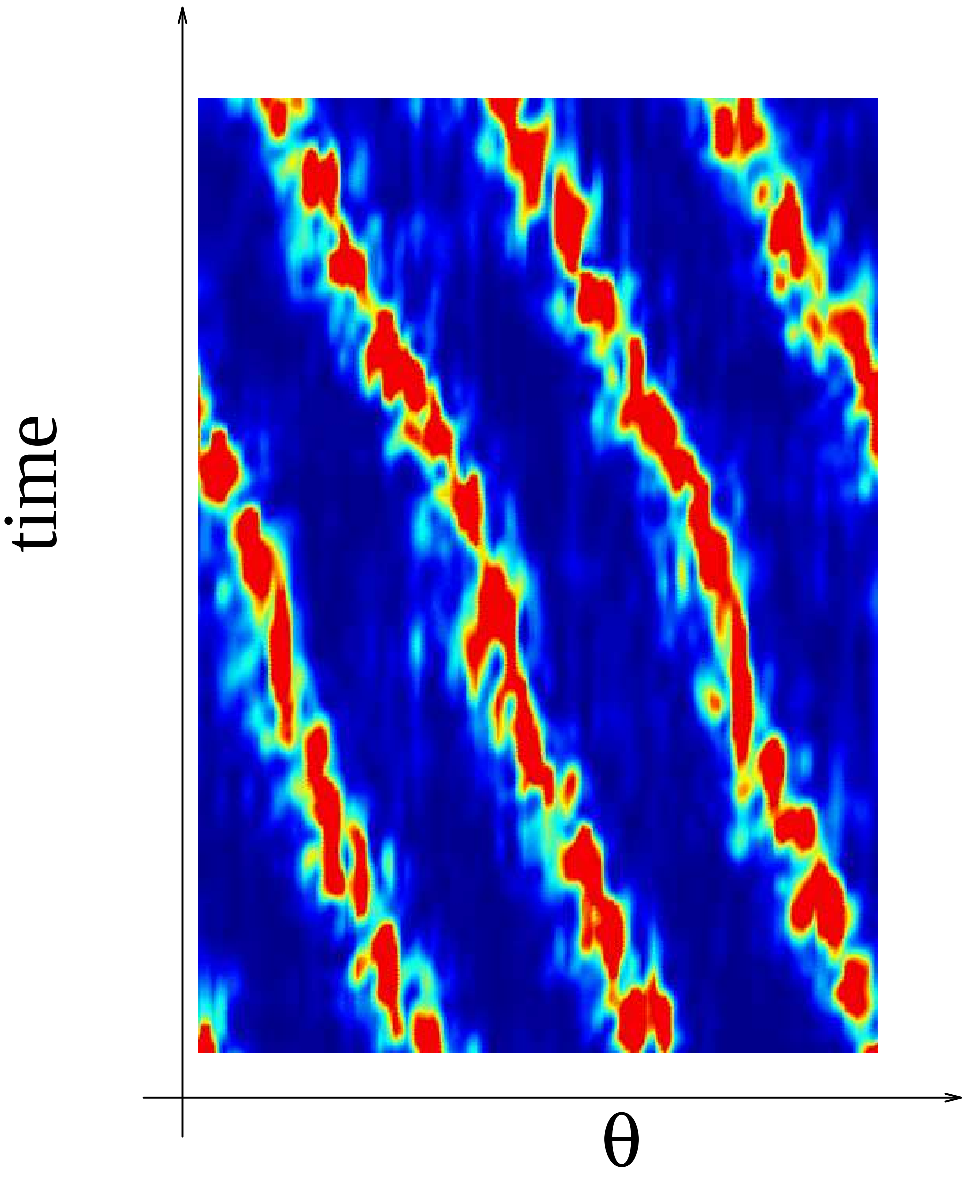


Figure 3.11: Temporal evolution of radially integrated power spectrum in the disordered domain $\alpha = 0.1$, $\gamma = 6$, $R = 0.036$, $k = 1.02$ (cf. Fig.3.6). This is a detail of Fig. 3.12a.

the hexagon mode A, which is in the middle of the figure. In the first snapshot, the modes near the hexagon mode are excited, an instability that can be resolved by the long-wave model. In the second snapshot, later in the nonlinear development of the instability, the excited modes extend the height of the Fig.3.13. Since Fig.3.13 is however just a small part of the Fourier transform near the hexagon mode A, this is equivalent to all the modes near the critical circle being excited. This nonlinear development cannot be resolved by the long-wave Ginzburg-Landau model. This illustrates an advantage of Swift-Hohenberg-type models, such as used in this thesis, over long-wave models, which cannot resolve short-wave phenomena.

It is less surprising to find complex dynamics in parameter regimes in which no stable steady hexagons exist. Such a case is shown in Fig.3.14. The parameters are as in Fig.3.5c with R (marked by a line in Fig.3.5c) chosen in the gap between the stable steady hexagons and the transition line to unmodulated oscillating hexagons. In this case the disorder is sufficiently strong that the power spectrum is essentially isotropic for the system sizes investigated (Fig.3.14a). We see a typical real-space snapshot in Fig.3.14b. The existence of a dominant wavenumber can be seen in the Fourier spectrum as well as the real solution. The real solution consists of a number of cell- and roll-like structures which are continuously created and destroyed in a disordered fashion. In Fig3.14c we see the time series of the amplitudes of the hexagon modes A, B, and C, as well as a couple of perturbation modes - the same modes as shown in Fig.3.10d. In contrast to the case of modulated hexagons, here we see no structure associated with the rotation of the sixfold symmetry of the Fourier spectrum, but merely the lack of correlation between the amplitudes associated with an isotropic Fourier spectrum.

From an experimental point of view an important question is whether the interesting dynamic states identified here are actually accessible even if the initial conditions cannot be prepared carefully. After all, they occur in parameter regimes in which the system could also evolve into a roll pattern (note the position of the stability line of rolls in Fig3.5). We have therefore performed a few simulations starting from random initial conditions. For the parameters in the stability gap

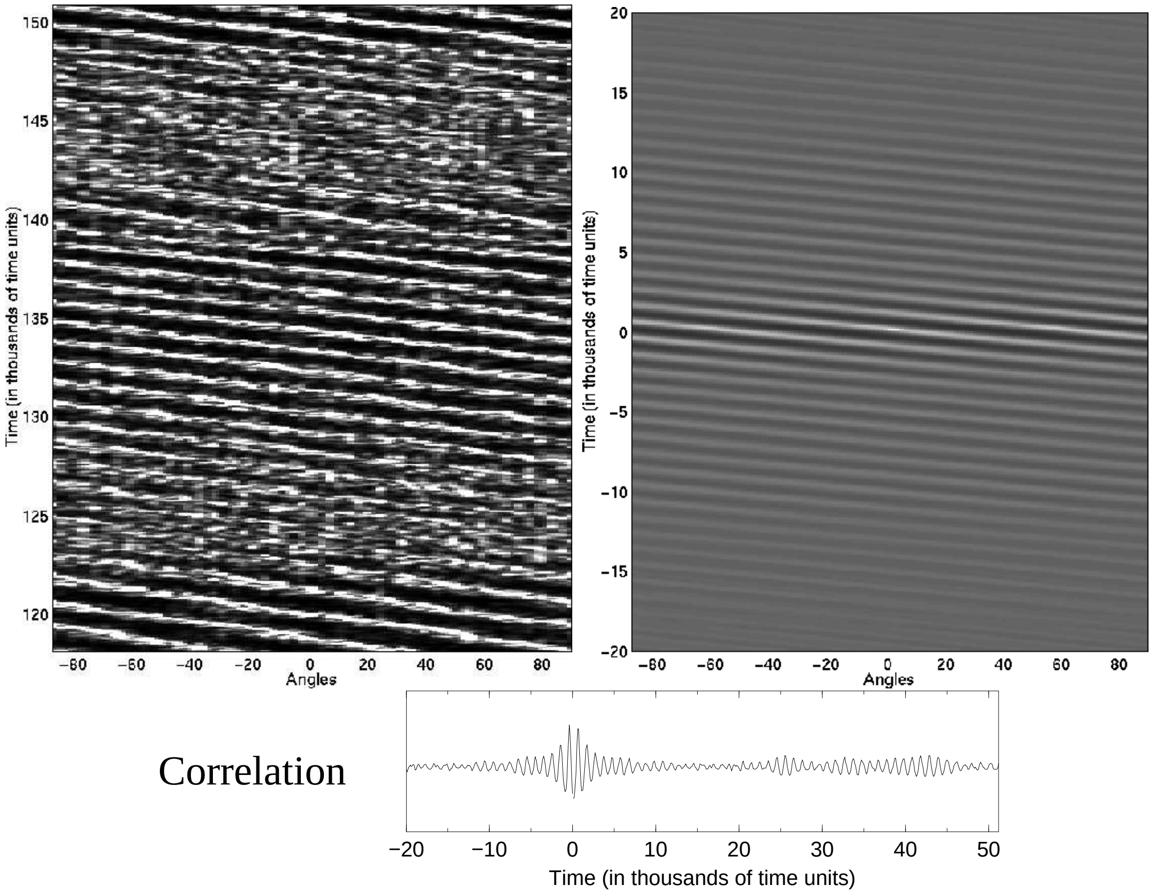


Figure 3.12: *a)* Temporal evolution of radially integrated power spectrum and *b)* space-time autocorrelation thereof in the disordered domain $\alpha = 0.1$, $\gamma = 6$, $R = 0.036$, $k = 1.02$ (cf. Fig.3.6). *c)* A cut through the autocorrelation function for $\theta = 0$. The time scale is the same as in *b*. We see that for longer times, i.e. correlation lengths of longer than 20,000 time units, the autocorrelation gains strength again. We suspect that this is due to the assumption of periodicity in using the fast Fourier transform of *a*.

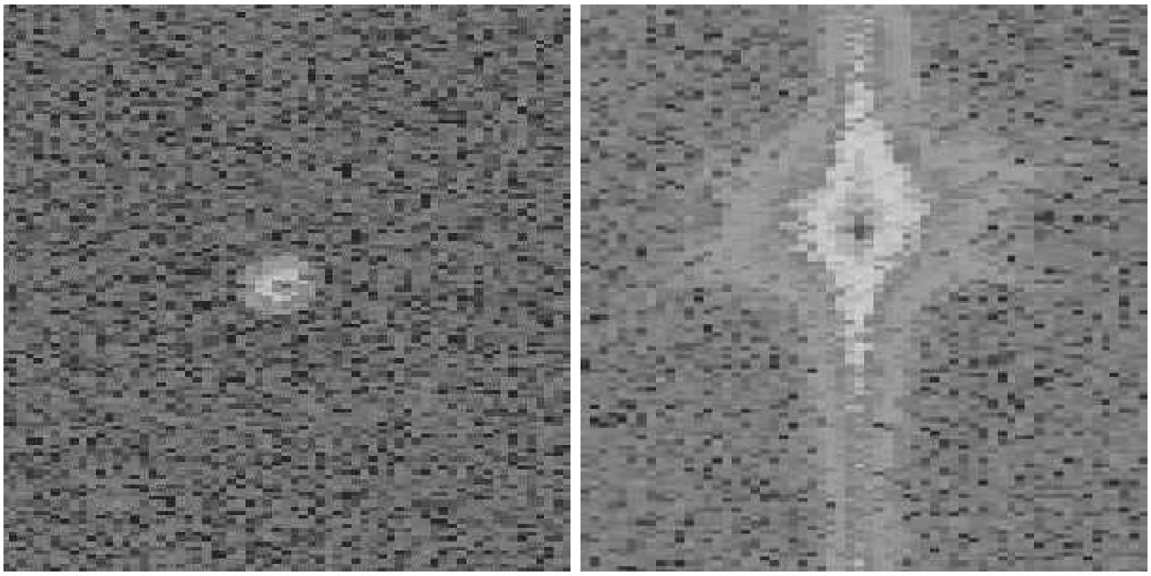


Figure 3.13: Fourier space diagrams from the direct numerical simulation of a long-wave Ginzburg-Landau type model [3]. The two snapshots in time are of the neighborhood of a hexagon solution mode. Figure courtesy of [3].

of Fig.3.5c the system evolves directly to a disordered state like that shown in Fig.3.14. Fig.3.15 shows the evolution in time of a simulation starting from small random initial conditions for parameter values for which steady and modulated hexagons coexist stably with the disordered state (cf. Fig.3.10). The spectrum is initially essentially isotropic (bottom panel), reflecting the random initial conditions, and then evolves into a more ordered spectrum (top panel) corresponding to the rotating state discussed above. These runs suggest that the states found here should be experimentally accessible. Of course, so far no information is available as to what extent the Swift-Hohenberg models discussed here describe a concrete physical system and whether the parameter regimes in which the dynamical states arise are accessible experimentally.

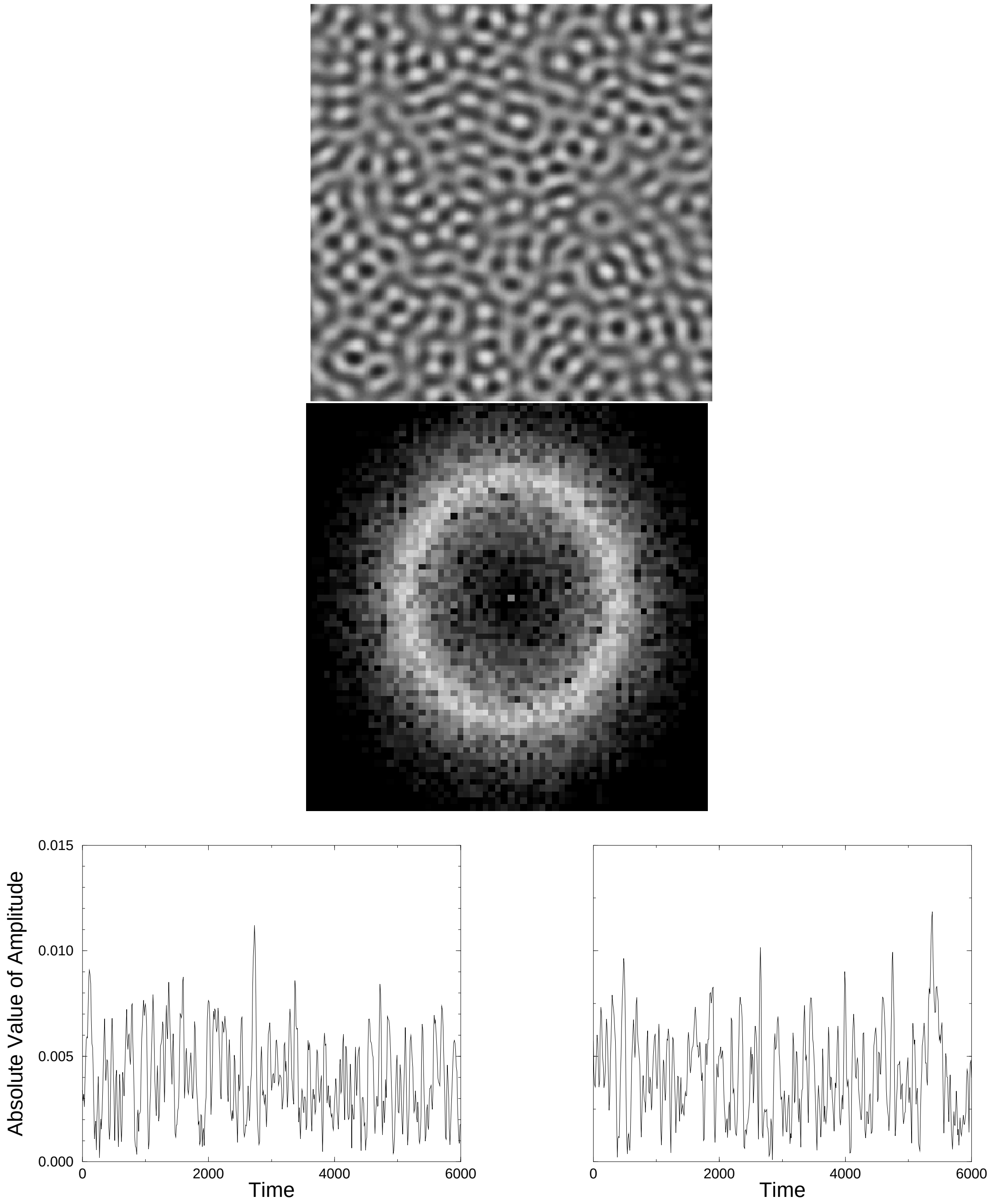


Figure 3.14: *a)* Power spectrum on a logarithmic scale and *b)* real space of disordered state at $\alpha = 0.1$, $\gamma = 14$, $R = 0.052$, at time $t = 22400$ in the minimal model (cf. Fig.3.5c). *c)* Time plot of the amplitudes A, on the left, and P, on the right. Note lack of structure as compared with Fig.3.10d.

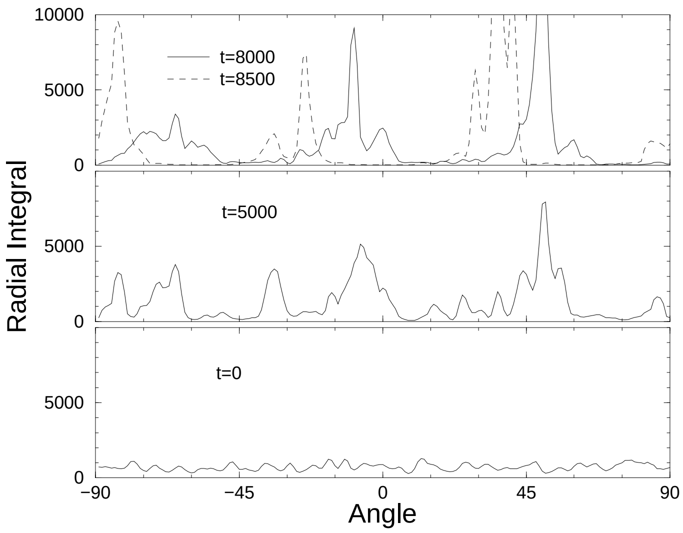


Figure 3.15: Temporal evolution of radially integrated power spectrum starting from random initial conditions for $\alpha = 0.1$, $\gamma = 6$, $R = 0.03$ (cf. Fig.3.6).

Chapter 4

Extended Model

4.1 Results

In the extended model (2.10), we go directly to the general weakly nonlinear expansion (2.24), without doing any further work with a long-wave expansion. Again, the weakly nonlinear expansion (2.24) leads to six eigenvalues, and we search for instabilities below the Hopf bifurcation at $R = R_H$. For an introduction to the terminology used in the following section, see the section on the Linear Stability Results in Ch.3.

As in the minimal model, the growth rates in the extended model exhibit a sixfold symmetry in Fourier space of areas of instability, centered around $(k, \theta = 0)$ (Fig.3.2 and Fig.4.1). This is of course due to the sixfold rotation symmetry of the system (cf. Sushchik and Tsimring [41]).

4.1.1 Types of Instabilities

Again, we consider only those parameter values for which the initial hexagons are stable to perturbations within the hexagonal lattice, i.e. $R_{sn} < R < R_H$. As in the minimal model, we illustrate the types of instabilities for the case $\sqrt{q^2 + p^2} = k$.

Nonzero-Angle Hopf. A Nonzero-Angle Hopf (NZAH) instability (Fig.4.2a) occurs when the complex conjugate pair of Hopf eigenvalues have a positive real part for a range of nonzero θ while a negative real part for $\theta = 0$. For the ranges of parameters considered, the maximal growth rate for the NZAH instability generally occurred for $|\theta| < 10°$, although positive growth rates for larger $|\theta|$ are possible as

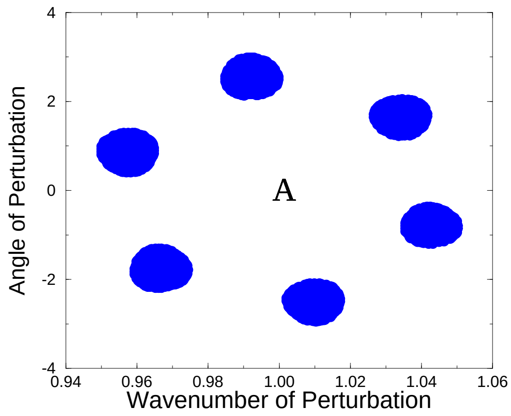


Figure 4.1: Fourier space diagram with the shaded ares indicating perturbations P_{000} to which the hexagons are unstable for $R = 0.0717$, $k = 1.00$, $\alpha = 0$, $\beta = 0.2$, $\gamma = 6$, $g_2 = 0.4$, $g_3 = 2$. The sixfold symmetry is due to the rotation symmetry.

well.

Translation. In the case $\sqrt{q^2+p^2} = k$, the translation eigenvalues are both zero for $\theta = 0$. In the presence of rotation, the translation eigenvalues may become a pair of complex conjugate eigenvalues for some range of $\theta \neq 0$. If this pair of complex conjugate eigenvalues crosses the imaginary axis for some θ, we have a manifestation of the instability driven by the translation modes, as shown in Fig.4.2b. In the case $\sqrt{q^2+p^2} \neq k$, the restriction that the translation eigenvalues are both zero for $\theta = 0$ does not hold. In this case, the translation eigenvalues can be real or complex at $\theta = 0$, with positive or negative real part.

Branchswitching. As R and k vary, a transition between the NZAH and the translation-mode instability regimes may occur in essentially one of two ways. The first is simply for the NZAH instability to become damped while the translation-mode instability becomes excited. The second, more interesting way is to go through a branch-switching transition. If we start in the NZAH regime (Fig.4.2a), we see that for these parameters (specifically $R = 0.09$), the two eigenvalue branches intersect for $\theta \approx 0.35°$. For $R = 0.083$ (Fig.4.2c), what was clearly an intersection of two branches of eigenvalues looks more like two eigenvalues meeting but not intersecting.

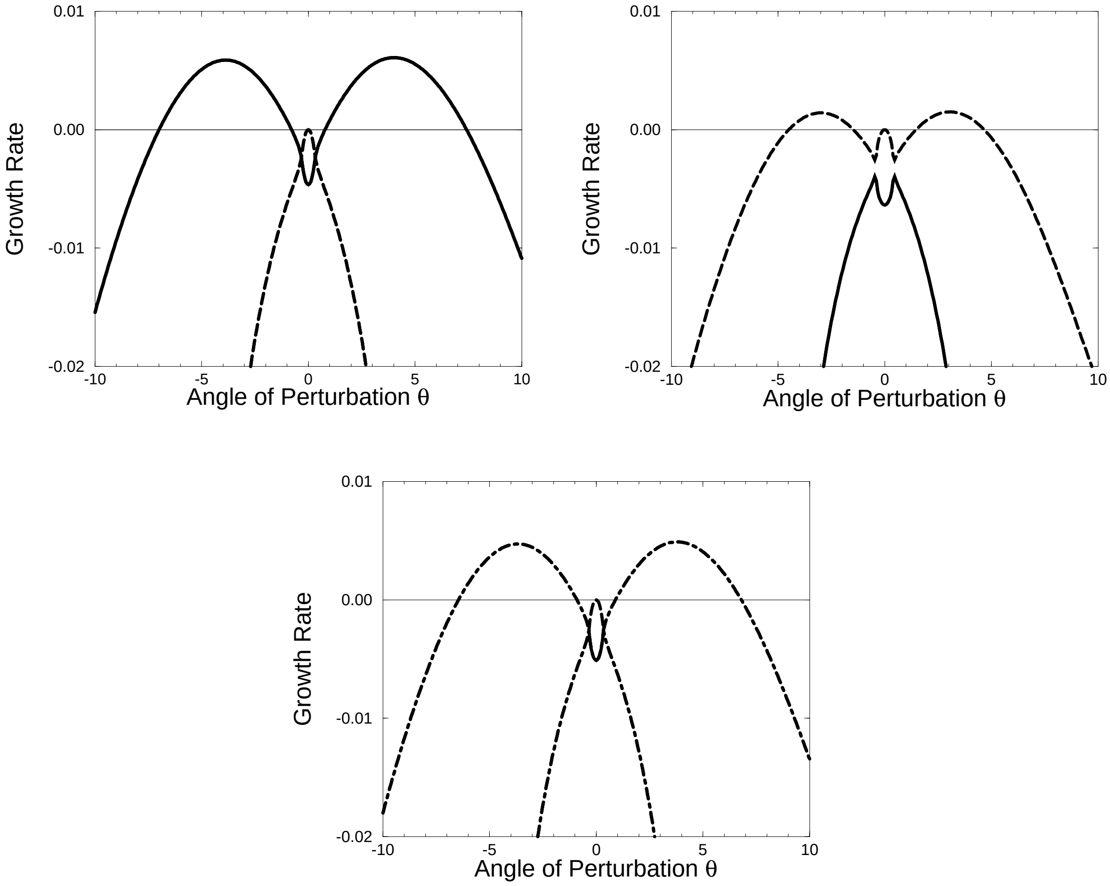


Figure 4.2: Illustrations of the types of instabilities for $k = 1.015 = \sqrt{q^2 + p^2}$, $\alpha = 0$, $\beta = 0.2$, $\gamma = 6$, $g_2 = 0.4$, $g_3 = 2$. The Hopf eigenvalues are denoted by the solid line, and the translation eigenvalues by the dashed line. *a)* The Nonzero-Angle Hopf instability with the Hopf eigenvalues positive for nonzero θ for $R = 0.090$. *b)* The translation-mode instability for $R = 0.060$. *c)* Branchswitching for $R = 0.083$. The dot-dashed lines indicate the branches whose classification is uncertain.

At this point, it is unclear whether the branch of eigenvalues that is unstable for a range of nonzero θ belongs to the translation or the Hopf eigenvalues. For $R = 0.06$, we have finished passing through the branch switching transition and are in the translation-mode instability regime (Fig.4.2b). The unstable eigenvalue branches have thus passed from being clearly connected to the Hopf eigenvalues to being clearly connected to the translation eigenvalues.

A closer look at this branchswitching is provided by Fig.4.3. Specifying the exact point at which the branchswitching takes place is difficult. In Fig.4.2c, the eigenvalues for $R = 0.083$ were shown; in Fig.4.3, a closeup of the eigenvalues near $\theta = 0$, the branchswitching seems to occur between $R = 0.065$ and $R = 0.060$. In both the real and imaginary graphs of Fig.4.3, the panels from top to bottom are for $R = 0.065$, $R = 0.063$, and $R = 0.060$, with $k = 1.015 = \sqrt{q^2 + p^2}$, $\alpha = 0$, $\beta = 0.2$, $\gamma = 6$, $g_2 = 0.4$, $g_3 = 2$. The real and imaginary parts of the translation and Hopf eigenvalues are identifiable through their behavior at $\theta = 0$. The real parts of the translation eigenvalues equal zero for $\theta = 0$. The real parts of the Hopf eigenvalues are negative for $\theta = 0$. The imaginary part of the translation eigenvalues is also zero for $\theta = 0$, whereas the imaginary part of the Hopf eigenvalues is nonzero for $\theta = 0$. The small range of θ around ± 0.42 for which there is a pair of real eigenvalues rather than a pair of complex conjugate eigenvalues is not important and can be ignored.

For $R = 0.065$, the graph of the real part of the eigenvalues seems to indicate that the Hopf and translation eigenvalues cross at $\theta \approx 0.40°$ without interacting. This is confirmed in the panel showing the imaginary part of the eigenvalues. The imaginary part of the Hopf eigenvalues, nonzero at $\theta = 0$, is concave down at $\theta = 0$, but seems to turn around and increase as $\theta \to \pm 1°$, without interacting with the translation eigenvalues. For $R = 0.063$, the real picture seems to show no intersection of the Hopf and translation eigenvalue branches. The imaginary parts of the eigenvalues seem to approach each other much more closely around $\theta = 0.4°$, and it is not quite clear whether they actually intersect or not. For $R = 0.060$, there is again clearly no intersection of the real branches of the eigenvalues. However, the imaginary branches of the eigenvalues are now clearly intersecting. Thus the

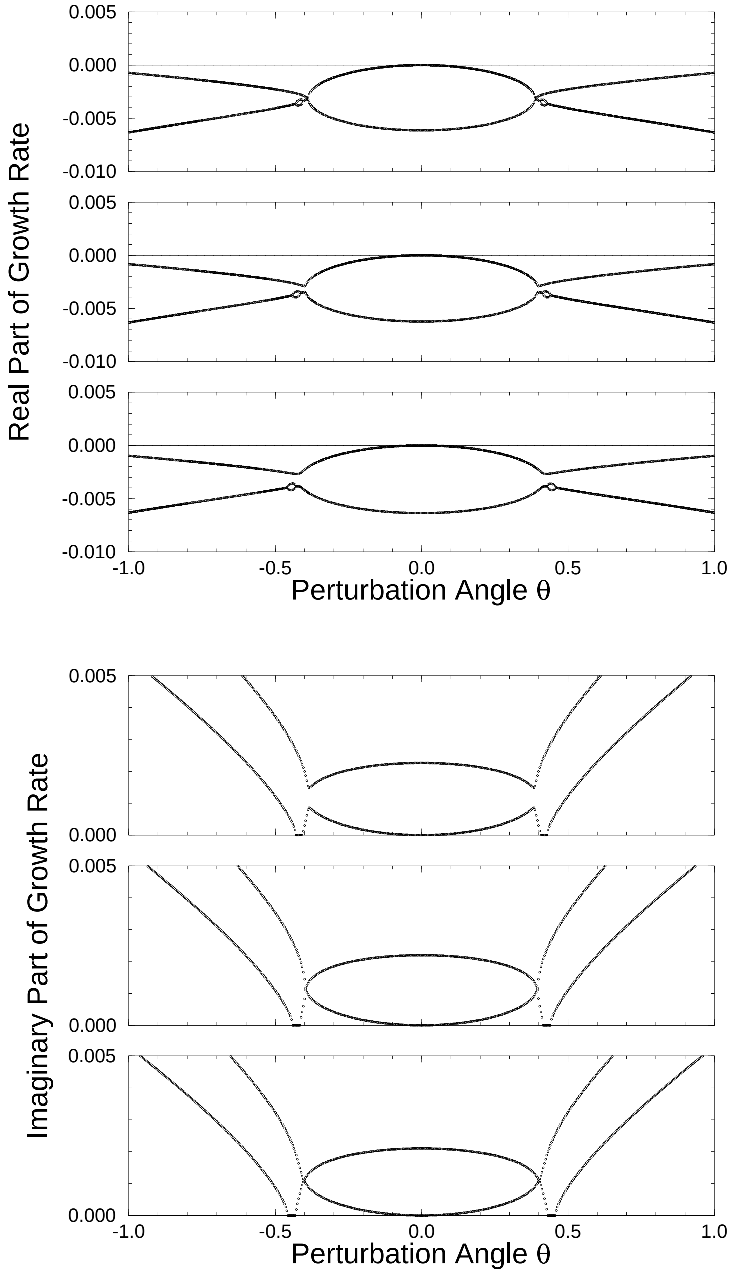


Figure 4.3: A closer look at the branch-switching transition for $k = 1.015 = \sqrt{q^2 + p^2}$, $\alpha = 0$, $\beta = 0.2$, $\gamma = 6$, $g_2 = 0.4$, $g_3 = 2$. From top to bottom, the values of the bifurcation parameter are $R = 0.065$, $R = 0.063$, $R = 0.060$.

eigenvalue branches for $|\theta| > 0.4°$ seem to have switched from Hopf to translation mode branches, and vice versa. The question of when to associate the eigenvalues for $|\theta| > 0.4°$ with the Hopf and when to associate them with the translation modes is not clearly answered in Fig.4.3. The real parts of the eigenvalues would suggest that the branchswitching occurs between $R = 0.065$ and $R = 0.063$, while the imaginary parts suggest that it occurs between $R = 0.063$ and $R = 0.060$. What is clear, however, is that some sort of transition does take place, because the Hopf and translation eigenvalues are clearly distinguishable for large enough or small enough values of R in this example.

The branchswitching phenomenon is not understood by us. It suggests that at the points at which branchswitching occurs, the eigenvectors associated with the Hopf and translation eigenvalues are very close to each other. In the bigger picture, the branchswitching also suggests that there is no clear difference between the Hopf and translation eigenvalues. The branchswitching thus supports the observation made in the previous chapter that a similar nonlinear evolution can arise from the NZAH and translation instabilities, namely modulated hexagons.

Real Translation Instability. In the absence of rotation, all eigenvalues are real. As rotation is increased, the translation eigenvalues become complex for a range of θ. In the case of Fig.4.4 the translation eigenvalues are a complex conjugate pair for small $|\theta|$. For larger $|\theta|$ a real translation eigenvalue becomes positive, driving a steady instability. Here the only rotation term is g_2; one can get qualitatively similar instabilities with only γ providing the rotation. However, rotation driven by γ does not generally have a real eigenvalue becoming positive for such a large $|\theta|$ as we see in Fig.4.4.

4.1.2 Regions of Stability

We now discuss the wave number range over which the hexagons are stable as a function of the control parameter R. To study the influence of the various terms on the stability of the hexagons, we first consider a reference case where

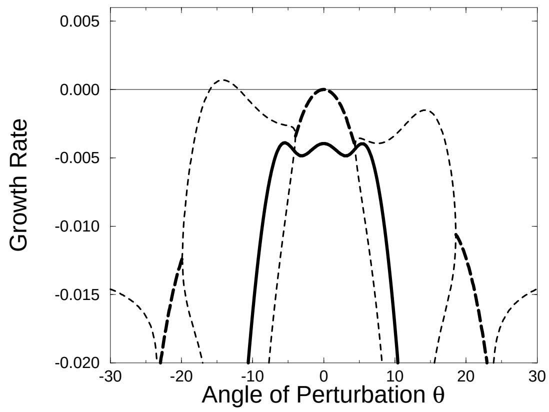


Figure 4.4: One of the real eigenvalues (thin dashed line) resulting from the splitting of the complex conjugate pair (thick dashed line) of translation eigenvalues is positive for $R = 0.035$, $k = 0.938 = \sqrt{q^2+p^2}$, $\alpha = 0$, $\beta = 0.2$, $\gamma = 0$, $g_2 = 5$, $g_3 = 0$. Note that individual eigenvalues can change from being real to being part of a complex conjugate pair of eigenvalues as θ is varied.

$\alpha = 0, \beta = 0.2, \gamma = 6, g_2 = 0, g_3 = 2$. From this state we vary the coefficients one by one. Negative values of the parameters yield the same behavior due to symmetries.

In the bifurcation diagrams (Fig. 1.3), increasing β pushes the Hopf bifurcation R_H (1.3) to higher R and the hexagons' saddle node R_{sn} to lower R. Similarly, increasing (decreasing) β enlarges (diminishes) the stability region in the R-k plane (Fig.4.5a). The Hopf curve approaches the neutral curve as β approaches zero, making the hexagons unstable for $\beta = 0$. Additionally, the range of wavenumbers for which the Hopf bifurcation forms a part of the stability boundary approaches zero as β approaches zero, and grows as β grows.

As stated above, neither R_H nor R_{sn} depend on γ or g_2. However, for $\sqrt{q^2+p^2} \neq k$ or $\theta \neq 0$, varying γ does affect the real parts of the eigenvalues of the 6 by 6 matrix. As γ increases, the stability region moves toward lower hexagon wavenumbers k. Also, the portion of the stability boundary along which the NZAH instability is relevant increases, while the portion along which the Hopf instability is relevant decreases. If γ is large enough, the stability region shrinks and leans so

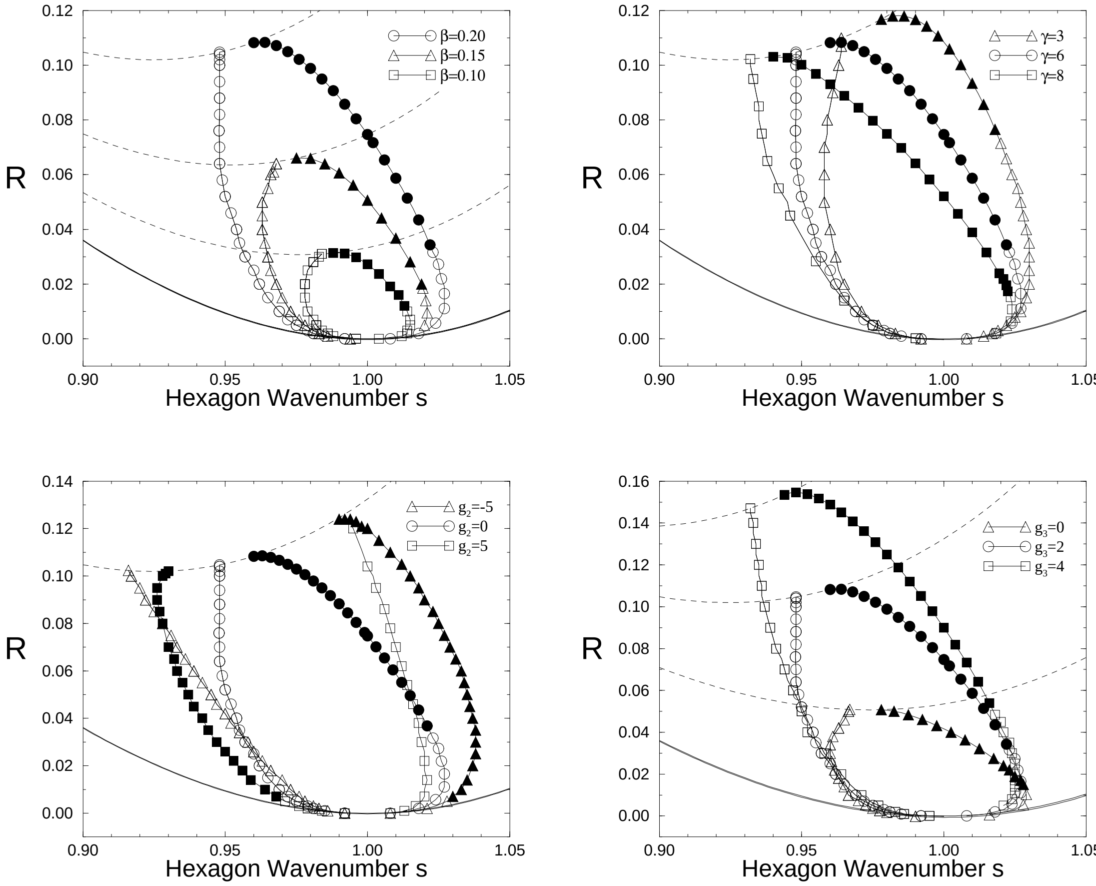


Figure 4.5: Stability regions for various values of the coefficients of the extended model. The dashed lines indicate R_H for various values of the coefficients. The solid line indicates both the neutral stability and saddle node (R_{sn}) curves. The solid symbols indicate the Nonzero-Angle Hopf (NZAH) instability, while the open ones indicate the translation instability. *a)* Vary β with $\alpha = 0$, $\gamma = 6$, $g_2 = 0$, and $g_3 = 2$ held constant. *b)* Vary γ with $\alpha = 0$, $\beta = 0.2$, $g_2 = 0$, $g_3 = 2$. *c)* Vary g_2 with $\alpha = 0$, $\beta = 0.2$, $\gamma = 6$, $g_3 = 2$. *d)* Vary g_3 with $\alpha = 0$, $\beta = 0.2$, $\gamma = 6$, $g_2 = 0$.

far to smaller wavenumbers that the Hopf instability at $R = R_H$ does not form part of the stability boundary, as will be discussed below.

The other rotation term, g_2, can represent rotation in the same or the opposite direction as γ. To consider both possibilities, we vary g_2 from -5 to 5. Again, R_H and R_{sn} do not depend on g_2. Varying g_2 from -5 to 5 does not change the size of the stability region much. However, the branch-switching transition moves from the right side of the stability region for $g_2 = -5$ to the left side of the stability region for for $g_2 = 5$ (Fig.4.5c). At $g_2 = -5$, the branch-switching transition is seen at about $(k, R) = (1.03, 0.007)$. It moves counterclockwise around the stability region, reaching the top of the stability region for approximately $g_2 = -\frac{1}{2}\beta\gamma = 0.6$. For this g_2, $f_2(k = 1, \beta, \gamma, g_2, g_3) = f_3(k = 1, \beta, \gamma, g_2, g_3)$, i.e., within the amplitude equation (2.14) the two terms arising from rotation cancel each other for $k = 1$. As g_2 approaches 5, the branch-switching transition approaches $(k, R) = (0.97, 0.007)$, continuing around the stability region as g_2 increases still further.

Similar to increasing β, increasing g_3 raises the Hopf bifurcation at $R = R_H$, making the stability region grow as well (Fig. 4.5d). The range of wavenumbers for which the Hopf bifurcation forms part of the stability boundary changes, shifting slightly to smaller wavenumber as g_3 increases; however, the width of this range of wavenumbers does not change significantly as g_3 changes.

To investigate the relevance of real eigenvalues in the presence of rotation, we let $\alpha = 0$, $\beta = 0.2$, $\gamma = 0$, $g_3 = 0$, and add rotation through the g_2 term (Fig. 4.7). In the absence of rotation ($g_2 = 0$), all eigenvalues are real. As g_2 increases from zero, the stability region grows larger. For sufficiently large g_2, the stability boundary for some finite R is determined by a complex conjugate pair of eigenvalues, while the stability boundary for R near zero is determined by a real eigenvalue (cf. Fig. 4.7). As g_2 increases further, the part of the stability boundary determined by the complex conjugate pair of eigenvalues grows. Also as g_2 increases, the real eigenvalue which determines the rest of the stability boundary becomes positive only for larger $|\theta|$, with $|\theta| \approx 30°$ for $g_2 = 5$.

A qualitative difference arises for sufficiently large γ. The stability region

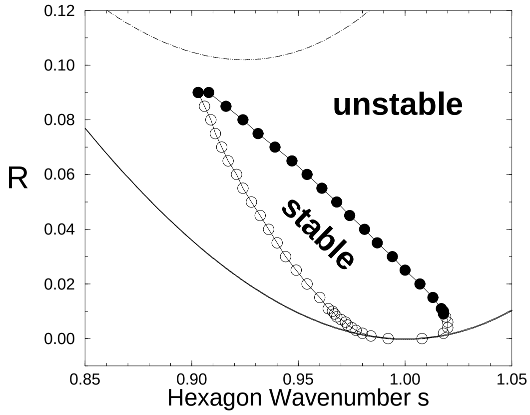


Figure 4.6: Stability region for $\alpha = 0$, $\beta = 0.2$, $\gamma = 12$, $g_2 = 0$, $g_3 = 2$. For these parameters, the Hopf instability at R_H does not form part of the stability boundary of the hexagons. There exist values of R for which no hexagon is stable. The Hopf instability is excited only for hexagons otherwise already unstable.

leans so far to the left that the maximal R for which a stable wavenumber exists is lower than the minimal R for which a wavenumber is unstable to the Hopf instability at $R = R_H$ (Fig. 4.6). Hence the Hopf instability is preempted by other instabilities over the whole range of wavelengths considered, and becomes irrelevant to the stability of hexagons in this parameter regime. Increasing γ excites the NZAH instability, which occurs on the right side of the stability region. The stability boundary for the NZAH instability moves to the left for increasing γ, until it eventually crosses the stability boundary for the translation-mode instability, preempting the Hopf instability. The value of γ necessary to reach this state is smaller for larger g_3. No simple solution is stable in this case and complex dynamics are expected.

4.2 Numerical Results

The numerical method used for the extended model is the same as that used for the minimal model, and is described above. The results of the numerical simulation of the extended model are very similar to the results presented for

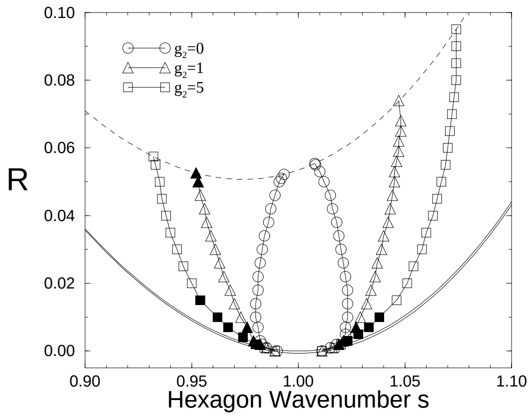


Figure 4.7: Stability regions with the Hopf instability curve for various values of g_2 with $\alpha = 0$, $\beta = 0.2$, $\gamma = 0$, $g_3 = 0$. For $g_2 = 0$, there is no rotation in the system, and all eigenvalues are real. For $g_2 \neq 0$, the solid symbols indicate an oscillatory instability, while the open symbols indicate an instability involving a real eigenvalue.

the minimal model. These results include modulated hexagons, the breakdown of the modulated hexagons, as well as the spatio-temporal chaos state with isotropic Fourier spectrum. These are described in more detail in this section.

In the numerical computations, we restrict α and β to be such that $\alpha+\beta/2$ is small. This restriction leads to a much better agreement between the amplitudes of steady solutions as computed numerically and as calculated in the weakly nonlinear analysis, thereby supporting the scaling of α with respect to β (cf. (2.12)). In this regime, the numerical analysis confirmed the stability boundaries calculated above.

The modulated hexagon state found in the minimal model exists in the extended model as well, for different sets of parameters. In fact, the modulated hexagons are stable on the right side of the stability region of the hexagons for a range of different parameters. The Hopf bifurcation at finite angle is found to be supercritical for the parameters investigated. As in the minimal model, if R is close enough to the stability boundary, the amplitude of the oscillations of the perturbation mode saturates, and steady oscillations result. We see this for the parameters $R = 0.087$, $\alpha = 0$, $\beta = 0.2$, $\gamma = 6$, $g_2 = 0$, $g_3 = 2$, and $k = 1$, in

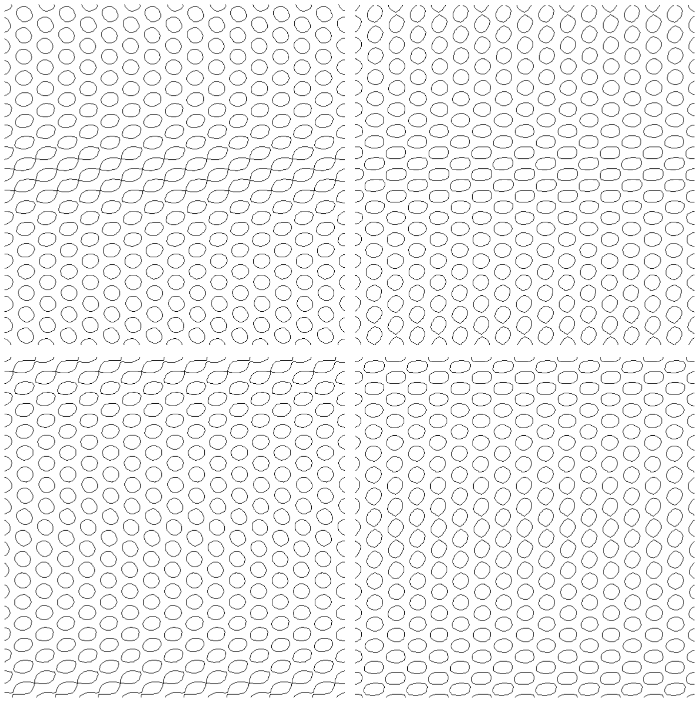
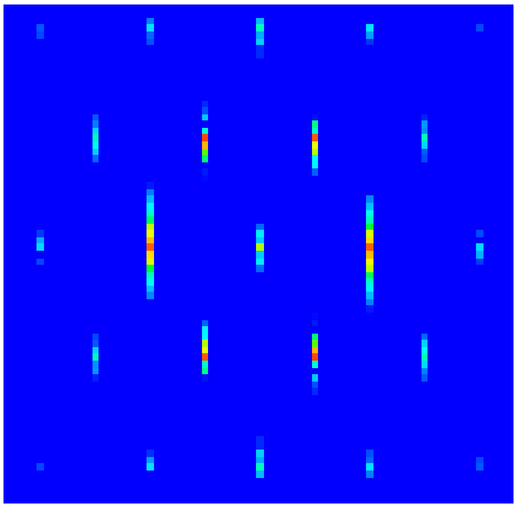


Figure 4.8: A real space illustration of the stable oscillations of the perturbation mode for $R = 0.087$, $k = 1.00$, $\alpha = 0$, $\beta = 0.2$, $\gamma = 6$, $g_2 = 0$, $g_3 = 2$, at times $t = 41851$, $t = 41904$ (top), $t = 41968$, $t = 42019$ (bottom). Indicated in the figure are the zero level contours of the solution. The perturbation mode is very close to the original hexagon mode e^{ix}, and so there is modulation in the pattern.



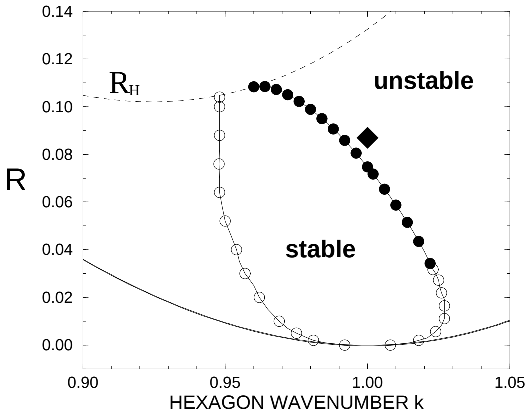


Figure 4.9: *a)* A representative Fourier spectrum for the steady oscillation state for $R = 0.087$, $k = 1.00$, $\alpha = 0$, $\beta = 0.2$, $\gamma = 6$, $g_2 = 0$, $g_3 = 2$. (cf. Fig.4.8). *b)* The black diamond indicates the parameters at which the numerical simulation was done.

Fig.4.8. The stability boundaries for these parameters are shown in Fig.4.9b. The amplitude of the steady oscillations of the perturbation mode is again much smaller than the amplitudes of the hexagon modes, yielding the zero level curves of Fig.4.8. Since the perturbation mode $e^{i(qx+py)}$ differs slightly from the hexagon mode e^{ikx}, the oscillations are undulated in space and constitute a standing wave. The Fourier spectrum of the modulated hexagon state is shown in Fig.4.9a. Compare to Fig.3.9, the corresponding Fourier spectrum for the modulated hexagon state described in the minimal model. In both cases, the only excited modes are the hexagon modes, the perturbation mode, and the harmonics of these modes. For R further away from the stability limit, the stable oscillations break down, and a disordered state ensues. For the parameters given in Fig.4.8, $R = 0.087$ is very near the upper stability boundary of the modulated hexagon state; for $R = 0.088$ the modulated hexagons are unstable. The instability of the modulated hexagons indicates a secondary bifurcation branching off of the steadily oscillating solution.

The translation-mode instability also involves a complex conjugate pair of unstable eigenvalues, and so the perturbation again oscillates with a growing

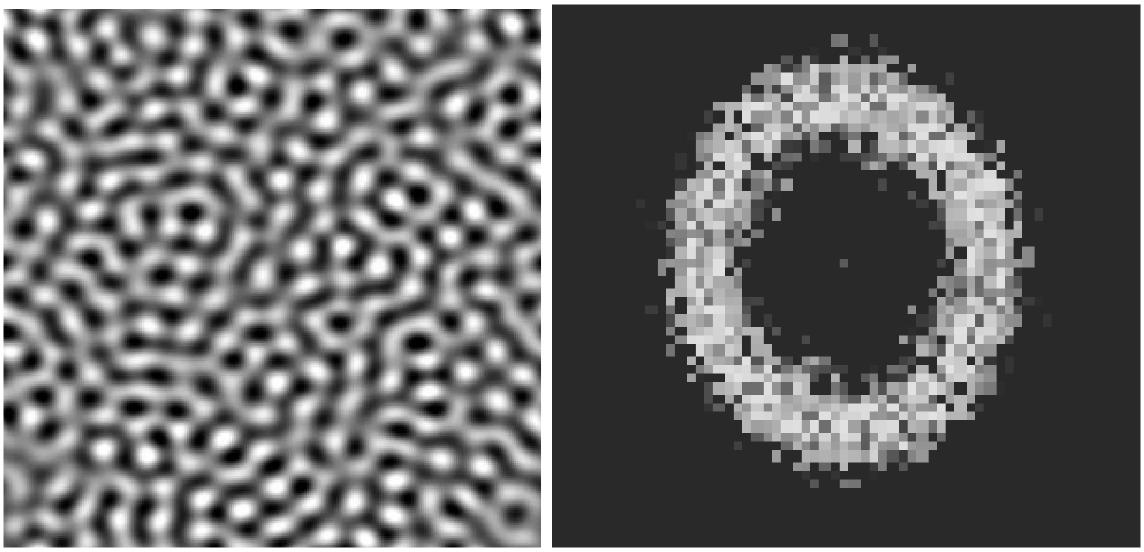


Figure 4.10: *a)* A representative snapshot of the solution in the disordered state for $R = 0.098$, $k = 0.88$, $\alpha = 0$, $\beta = 0.2$, $\gamma = 12$, $g_2 = 0$, $g_3 = 2$. This is a value of R for which no hexagon is stable and which is lower than the Hopf bifurcation value R_H for any wavenumber k (cf. Fig.4.6). *b)* The Fourier spectrum of the solution, on a logarithmic scale. Note the isotropy of the the Fourier spectrum.

amplitude (cf. Fig.4.2b). However, for the parameters investigated, which were on the left side of the stabiliy region, no steady oscillations were found near the onset of the instability. Instead, the growth of the perturbation led to the breakdown of the hexagon pattern, and from there to a disordered state.

In the case where the instability is due to a real eigenvalue being positive at finite θ (cf. Fig.4.4), the perturbation mode grew without oscillations, as expected. The perturbation quickly led to a new stable hexagon solution with a different wavenumber that is within the stability boundaries for the same value of R.

When γ is large enough, the stability region does not contact the Hopf bifurcation, and there are values of R below R_H for which there is no stable stationary state (cf. Fig.4.6). This corresponds to Fig.3.5c in the minimal model. In this case a time-dependent disordered pattern ensues, shown in Fig.4.10 (cf. Fig.3.14). As in the minimal model, this state consists of a cellular structure in which some cells are connected by roll-like structures. These connections change persistently in an irregular manner. In addition, individual cells are continually being created and destroyed. The total number of cells for the system size investigated, 32π by $64\pi/\sqrt{3}$, varies between 20 and 30.

Chapter 5

Long-wave Marangoni Convection

5.1 Introduction

The equations studied in the previous two chapters, (2.1) and (2.10), were phenomenological model equations for Rayleigh-Bénard convection. As noted earlier, however, equations of the Swift-Hohenberg type can also be derived from the governing equations of fluid dynamics, the Navier-Stokes equations, under certain conditions and in certain limits. The equation studied in this chapter was derived from the Navier-Stokes equations by Mancho and Riecke [35] for rotating Marangoni convection with poorly-conducting boundary conditions [38, 39] in the longwave limit:

$$\begin{aligned}\psi_t \;=\; & -b\psi - \lambda\nabla^4\psi - \mu\nabla^2\psi - g_3(\nabla\cdot(\nabla\psi)^3) \\ & - \alpha\nabla^2(\nabla\psi)^2 - \beta\nabla\cdot(\nabla^2\psi)(\nabla\psi) \\ & - \gamma\,\hat{e}_z\cdot(\nabla\psi\times\nabla(\nabla^2\psi)) - g_2\,\hat{e}_z\cdot(\nabla\times(\nabla\psi)^3).\end{aligned} \tag{5.1}$$

The convection is surface-tension driven, with an assumption of no displacement of the free upper surface of the fluid. The Prandtl number is infinite.

Despite the number of terms, there are only three independent parameters in the problem. The first, the rescaled Marangoni number μ, is also the bifurcation parameter. The second is the coefficient of the linear term, b, which is related to the Biot number. The third is the rotation rate Ω. The rest of the coefficients are functions of Ω. The Taylor Series expansions of the coefficients about $\Omega = 0$ are as follows:

$$
\begin{aligned}
\lambda &= \frac{1}{15} - \frac{7}{1200}\Omega^2 + O(\Omega^4), \\
\alpha &= \frac{1}{10} + \frac{1}{1575}\Omega^2 + O(\Omega^4), \\
\beta &= \frac{3}{5} + \frac{1}{1575}\Omega^2 + O(\Omega^4), \\
\gamma &= -\frac{233}{525}\Omega + \frac{404}{72765}\Omega^3 + O(\Omega^5), \\
g_2 &= \frac{8}{35}\Omega - \frac{122}{1576575}\Omega^3 + O(\Omega^5). \\
g_3 &= -\frac{48}{35} - \frac{104}{121275}\Omega^2 + O(\Omega^4),
\end{aligned}
\tag{5.2}
$$

Thus the rotation term coefficients γ and g_2 are odd in Ω, while all the other coefficients are even in Ω.

The main difference between this equation and the model equations (2.1) and (2.10) is in the quadratic terms, whose coefficients are α and β in each case. The terms whose coefficients are γ, g_2, and g_3 are identical in (5.1) and (2.10). However, the quadratic terms $\alpha\psi^2$ of (2.1) and $\alpha\psi^2$ and $\beta(\nabla\psi)^2$ of (2.10) are simpler than the terms $\alpha\nabla^2(\nabla\psi)^2$ and $\beta\nabla\cdot(\nabla^2\psi)(\nabla\psi)$ of (5.1). In (5.1), all terms quadratic in ψ have fourth order derivatives. It must also be noted that in contrast to (2.1) and (2.10), the critical wavenumber k_c of (5.1) is not fixed, and varies with the coefficients of the terms linear in ψ, namely b and λ. Note that since λ is a function of the rotation Ω, the shape of the neutral curve and the critical wavenumber k_c depend on Ω as well.

Stated explicitly,

$$k_c = \sqrt[4]{\frac{b}{\lambda}}, \tag{5.3}$$

$$\mu_c = 2\sqrt{b\lambda}\,. \tag{5.4}$$

One notices that both b and λ are under the square root in the expressions for both k_c and μ_c. Whereas b is an independent parameter, λ is a function of Ω (cf. (5.2)). This has the effect of limiting the value of $|\Omega|$ allowed, since for $b > 0$, $\lambda > 0$

is necessary. Solving $\lambda = 0$ in (5.2) gives a value of $\Omega_{max} \approx 3.38$, which is the maximum allowable value for $|\Omega|$. If $\Omega > \Omega_{max}$, then $\lambda < 0$, and (5.1) is ill-posed. The neutral curve μ_n, saddle node μ_{sn}, and Hopf μ_H points are given by

$$\mu_n = \frac{b}{k^2} + \lambda k^2, \tag{5.5}$$

$$\mu_{sn} = \mu_n - \frac{1}{12}\frac{k^4(\beta-\alpha)^2(b-3\lambda k^4)(-4\lambda k^4+b)}{Q+10\beta^2k^4b}, \tag{5.6}$$

$$\mu_H = \mu_n + \frac{1}{3}\frac{(\beta-\alpha)^2(Q+11\beta^2k^4b)(b-3\lambda k^4)(-4\lambda k^4+b)}{k^4b^2\beta^4}, \tag{5.7}$$

where

$$Q = -21g_3b\lambda k^4 + 3g_3b^2 + 36g_3\lambda^2k^8 - 36\beta^2k^8\lambda.$$

Given the similarities between the two equations, we first attempted a weakly nonlinear analysis of (5.1) using the same expansion (2.24) as used in the studies of (2.1) and (2.10). However, this weakly nonlinear analysis had a very limited region of validity, which was confined to the close proximity of the neutral curve and the critical wavenumber. We note that the neutral curve of (5.1) is very shallow compared to the neutral curves of the two model equations. Because of this, the weakly nonlinear analysis must consider a comparatively greater range of perturbation wavenumbers $\sqrt{q^2+p^2}$ than was the case with the phenomenological equations. If this greater range of perturbation wavenumbers is not considered, an incomplete picture of the stability boundaries of the hexagon solution results. However, in considering this greater range of perturbation wavenumbers the weakly nonlinear analysis comes upon more singularities in the calculation, again due to the adiabatic elimination of modes close to the critical circle. This further limits the region of validity of the weakly nonlinear analysis (Fig.5.1). The weakly nonlinear model is only valid near the neutral curve, where it agrees with the Galerkin calculation (discussed in the next section). The two calculations begin to disagree near $\mu = 0.10$. Below $\mu = 0.10$, the instabilities are long-wave in general. Above $\mu = 0.10$, the instabilities are generally short-wave. The two vertical lines in Fig.5.1 are the limits of validity of the weakly nonlinear expansion. The dotted line is the limit of validity

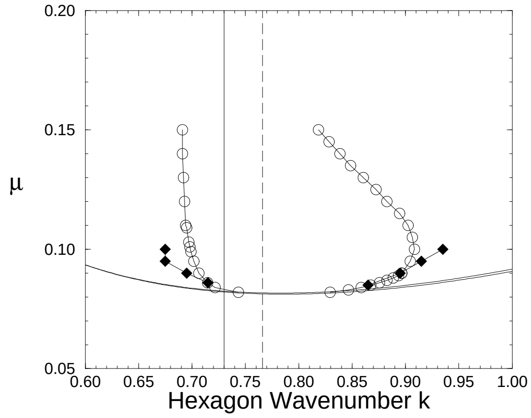


Figure 5.1: The (untrustworthy) stability boundaries as computed by the weakly nonlinear expansion (2.24) for $b = 0.025$, $\Omega = 0$. The critical wavenumber is $k_c = 0.78$. The two vertical lines are the lowest wavenumber k limit of validity of the weakly nonlinear expansion. The dotted line is the limit of validity if the largest perturbation angle allowed is $\theta_{max} = 30°$, the dashed line if $\theta_{max} = 20°$. The Hopf curve at R_H occurs for larger μ than shown here.

if the largest perturbation angle allowed is $\theta_{max} = 30°$, the dashed line if $\theta_{max} = 20°$. The weakly nonlinear calculations are considered valid to the right of the lines. The stability limits for the hexagon solution as computed disregarding the singularities are shown. As it turns out, the short-wave stability limit on the right is not reliable either, as the instabilities are frequently driven by singularities. The only reliable part of the stability lines shown is that generated by the long-wave instabilities, near the neutral curve, which was checked against the Galerkin calculation. To resolve this problem in the context of the weakly nonlinear expansion, it would be necessary to keep all modes which in (2.24) are considered to be $O(\epsilon^2\delta)$ (i.e. modes H_{lmn}) at $O(\epsilon\delta)$. Keeping so many modes as independent perturbation modes would make a practical application of the weakly nonlinear analysis unfeasible.

This analysis of the problems of the weakly nonlinear analysis does however point towards a feasible solution, in the form of a Galerkin approach. With a numerical Galerkin method, it is possible to keep a far greater number of modes as independent perturbation modes, up to 7, 9, or 11 harmonics of the original hexagon solution, and an equal number of perturbation modes. This can get around the practical problem of keeping many modes at $O(\epsilon\delta)$ encountered by the weakly nonlinear analysis. In the next section, the methodology used to find the linear stability of the hexagon solution of (5.1) is described, and in the final section of the chapter the results of this linear stability analysis as well as results of direct numerical simulations of (5.1) are presented.

5.2 Methodology

This section covers some technical aspects of the numerical Galerkin calculation.

Since the weakly nonlinear analysis cannot effectively consider a larger number of harmonics of the hexagon solution, we used a Galerkin procedure. To find the hexagon solution, the weakly nonlinear analysis uses only the hexagon solution modes and their second harmonics. The weakly nonlinear analysis considers the harmonics of the hexagon modes to be strongly damped, and slaves these harmonics to the hexagon modes. The Galerkin procedure does not do this slaving, but rather solves for all modes simultaneously, in our case numerically, using Newton's method. By solving numerically from the outset, rather than doing an analytical weakly nonlinear calculation, we are able to consider simultaneously a large number of harmonics of the original hexagon modes. This effectively is the same as keeping all harmonics at $O(\epsilon\delta)$ in the weakly nonlinear calculation. Thus the Galerkin procedure can get around the problems encountered by the weakly nonlinear analysis in the study of (5.1).

Similarly to (2.24), ψ is expanded in Fourier space. But in contrast to (2.24), the Galerkin expansion does not include any scalings,

$$\psi(x,y,t) = \sum_{n=-N}^{N} a_n(t) e^{i(n_x,n_y)\cdot(x,y)} \,. \tag{5.8}$$

The modes chosen to approximate ψ are the hexagonal modes and their harmonics (cf. Fig.5.2a). Hence, n_x is a multiple of $1/2$, while n_y is a multiple of $\sqrt{3}/2$. Let h

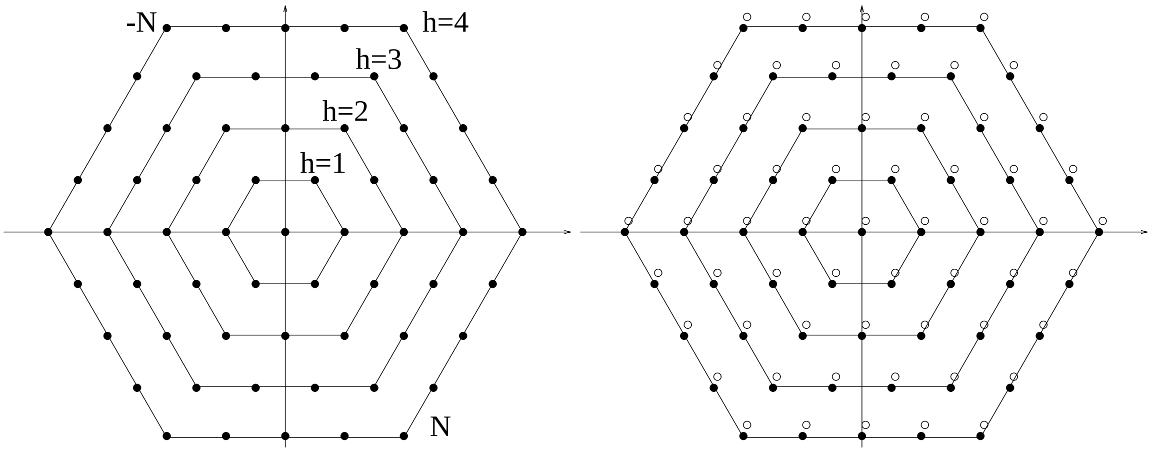


Figure 5.2: *a)* The Fourier modes used in the Galerkin calculation. The smallest hexagon shown, labeled $h = 1$, consists of the modes making up the hexagon solution of (5.1), and is considered to be close to the critical circle. The total number of modes used in the Galerkin calculation is simply a function of the number of harmonics h, namely $3h(h+1)+1$. The ordering of modes into a one-dimensional array for the purposes of the Galerkin calculation starts at the top of the figure, working downwards by going left to right in the rows of modes, going from $-N$ to N. *b)* The open circles indicate perturbation modes for a given perturbation wavevector. The perturbation modes are analogous to the perturbation modes P_{lmn} and their harmonics $H_{\tilde{l}\tilde{m}\tilde{n}}$ in Fig.2.3. There is a one-to-one correspondence between the perturbation modes and the hexagon modes.

be the number of harmonics considered. The wavevector (n_x, n_y) can be written as a piecewise defined function of h, which is not stated explicitly in this thesis. The number of modes $2N+1$ can of course also be expressed as a function of the number of harmonics, $2N + 1 = 3h(h + 1) + 1$. We consider the case $h = 0$ when the only mode considered is the constant mode ($N = 0$), $h = 1$ when only the constant mode and the hexagon modes themselves are considered ($N = 3$), $h = 2$ when the second harmonics of the hexagon modes are considered ($N = 9$), and so on. Exactly which modes are included, given h, can be seen from Fig.5.2a. When $h = 1$, the origin and the modes on the smallest hexagon shown (these are the original hexagon modes) are included. When $h = 2$, the next larger hexagon is included as well, these modes being the quadratic harmonics of the original hexagon modes. The included modes for higher h are obvious.

The equation (5.8) sets up an array of Fourier amplitudes that approximates the solution ψ of (5.1) that we seek. Let us call this array $\overline{\psi}$. Substitution of (5.8) into (5.1) would result in a series of coupled equations for the different Fourier amplitudes $a_n(t)$. The goal is to form these coupled equations into a matrix equation for $\overline{\psi}$ and use Newton's method to converge to a solution. Denote the n-th iteration of the Newton's method solution as $\overline{\psi}^n$. At each Newton iteration, we solve the matrix equation for the difference $(\overline{\psi}^{n+1} - \overline{\psi}^n)$,

$$\overline{\overline{A}}(\overline{\psi}^{n+1} - \overline{\psi}^n) = F(\overline{\psi}^n), \tag{5.9}$$

where $\overline{\overline{A}}$ is a $3h(h+1)+1$ by $3h(h+1)+1$ matrix. Thus we are going to approximate a solution to the partial differential equation (5.1) with the array of Fourier amplitudes to which the iteration scheme (5.9) converges.

Some comments on the construction of equation (5.9) follow. The simplest method of explanation is to consider the contribution of some terms of the expansion of $F(\overline{\psi}^{n+1})$ to (5.9) as examples. First, consider the simple nonlinear term $(\overline{\psi}^{n+1})^3$,

$$\begin{aligned}(\overline{\psi}^{n+1})^3 &= [\overline{\psi}^n + (\overline{\psi}^{n+1} - \overline{\psi}^n)]^3 \\ &= (\overline{\psi}^n)^3 + 3(\overline{\psi}^n)^2(\overline{\psi}^{n+1} - \overline{\psi}^n) + h.o.t.\end{aligned}$$

The higher order terms are ignored. The term $(\overline{\psi}^n)^3$ is part of the right hand side of (5.9) $F(\overline{\psi}^n)$. Since $\overline{\psi}$ is an array of Fourier modes, multiplication of these arrays involves a convolution. The most efficient way to perform the convolution is to translate it into a matrix multiplication. The term $3(\overline{\psi}^n)^2(\overline{\psi}^{n+1} - \overline{\psi}^n)$ contributes to the left hand side of (5.9), and so the matrix multiplication equivalent of the convolution is necessary to properly construct the matrix $\overline{\overline{A}}$. Thus it was useful to use matrix multiplication to do the convolutions necessary to calculate $F(\overline{\psi}^n)$ as well. Constructing the convolution matrix for a particular multiplication involves placing the correct Fourier amplitudes $a_n(t)$ in the correct elements of a $3h(h+1)+1$ by $3h(h+1)+1$ matrix. In practical terms, this is most easily implemented by creating a model matrix $\overline{\overline{M}}$, which maps out where each Fourier amplitude $a_n(t)$ is to go in the matrix, at the beginning of the code, to be invoked any time a convolution involving $\overline{\psi}$ or any of its derivatives is needed. Thus, the contribution of $(\overline{\psi}^n)^3$ to $F(\overline{\psi}^n)$ is calculated. The contribution of the term $-3(\overline{\psi}^n)^2(\overline{\psi}^{n+1} - \overline{\psi}^n)$ to the matrix $\overline{\overline{A}}$ also involves the construction of a convolution matrix. First the array representing $-3(\overline{\psi}^n)^2$ is created, and then the matrix representing multiplication of the difference vector $(\overline{\psi}^{n+1} - \overline{\psi}^n)$ by this array is created, again using $\overline{\overline{M}}$. We have thus accounted for the contribution of the term $(\overline{\psi}^{n+1})^3$ to (5.9).

A more involved example is the quadratic rotation term,

$$\hat{e}_z \cdot (\nabla\overline{\psi}^{n+1} \times \nabla(\nabla^2\overline{\psi}^{n+1})) = \overline{\psi}_x^{n+1}\overline{\psi}_{xxy}^{n+1} + \overline{\psi}_x^{n+1}\overline{\psi}_{yyy}^{n+1} - \overline{\psi}_y^{n+1}\overline{\psi}_{xxx}^{n+1} - \overline{\psi}_y^{n+1}\overline{\psi}_{xyy}^{n+1} \tag{5.10}$$

For the sake of brevity, we will only consider the first term of the four; all other terms, both in (5.10) and in the other terms of (5.1), are treated analogously. Expanding $\overline{\psi}_x^{n+1}\overline{\psi}_{xxy}^{n+1}$, we get

$$\begin{aligned} \overline{\psi}_x^{n+1}\overline{\psi}_{xxy}^{n+1} &= \overline{\psi}_x^{n}\overline{\psi}_{xxy}^{n} + \overline{\psi}_x^{n}(\overline{\psi}_{xxy}^{n+1} - \overline{\psi}_{xxy}^{n}) + \overline{\psi}_{xxy}^{n}(\overline{\psi}_x^{n+1} - \overline{\psi}_x^{n}) \\ &= \overline{\psi}_x^{n}\overline{\psi}_{xxy}^{n} + \overline{\psi}_x^{n}(\overline{\psi}^{n+1} - \overline{\psi}^{n})_{xxy} + \overline{\psi}_{xxy}^{n}(\overline{\psi}^{n+1} - \overline{\psi}^{n})_x \,. \end{aligned} \tag{5.11}$$

The first term in the last line contributes to $F(\overline{\psi}^n)$, the last two terms involve $(\overline{\psi}^{n+1} - \overline{\psi})$ and thus contribute to the left hand side of (5.9). For all three terms, there are essentially two parts to their proper implementation in (5.9). The first part is the conversion of the convolution resulting from the multiplication of one Fourier array by another into a matrix multiplication, which, as in the previous example, is done by invoking the model matrix $\overline{\overline{M}}$. The second part concerns the derivatives of $\overline{\psi}^n$ and $(\overline{\psi}^{n+1} - \overline{\psi}^n)$. Finding the derivatives of a Fourier array simply involves multiplying each array element, which is really a Fourier amplitude $a_n(t)$, by the appropriate factor, which depends on the wavenumber of the Fourier mode associated with said amplitude. Finding the derivatives of the difference $(\overline{\psi}^{n+1} - \overline{\psi}^n)$ must be done in considering the contributions of the last two terms of (5.11) to the matrix $\overline{\overline{A}}$ of (5.9). First the matrix representing the multiplication of the appropriate derivative of the difference by the appropriate derivative of $\overline{\psi}^n$ is constructed using $\overline{\overline{M}}$. The same factors which multiplied the Fourier arrays to obtain their derivatives now simply multiply the diagonal elements of this matrix to respresent the appropriate derivatives of the difference $(\overline{\psi}^{n+1} - \overline{\psi}^{n})$.

Hence, for any linear or nonlinear term in (5.1), we can create an appropriate matrix equation for ψ approximated by (5.8). Multiplying these matrices by the appropriate coefficient and adding them together, we arrive at the matrix equation (5.9) at the given iteration. Thus we construct (5.9), solve it, get the difference $(\overline{\psi}^{n+1} - \overline{\psi}^{n})$, and simply add this to $\overline{\psi}^{n}$ to get $\overline{\psi}^{n+1}$. Convergence using the Newton's method rarely takes more than ten iterations. The initial guess for the Newton's method generally consisted of a common nonzero amplitude for the hexagon modes, and zero amplitudes for the harmonics.

Once the Newton iterations converge to a solution, we can investigate the question of the stability of said hexagon solution. This stability investigation entails constructing another $3h(h+1)+1$ by $3h(h+1)+1$ matrix equation describing the linear evolution of small perturbations. Denote this perturbation matrix as $\overline{\overline{P}}$. Finding the eigenvalues of $\overline{\overline{P}}$ will give us the sought linear stability of the hexagon solution. The perturbation modes used (cf. Fig.5.2b) can be thought of as analogous to the perturbation modes in (2.24) and Fig.2.3. As illustrated by Fig.5.2b, there is a perturbation mode for every hexagon mode or harmonic in the Galerkin calculation, leading to the $3h(h+1)+1$ by $3h(h+1)+1$ matrix $\overline{\overline{P}}$. Creating $\overline{\overline{P}}$ is almost exactly the same as creating $\overline{\overline{A}}$. The only difference is in computing the impact of taking the derivatives of the perturbation modes, which is analogous to taking the derivatives of the difference $(\overline{\psi}^{n+1} - \overline{\psi}^{n})$ in the Newton method. The Fourier modes in the array of any iteration of $\overline{\psi}$ have the save wavenumbers, namely those of the hexagon solution and its harmonics. The perturbation modes and their harmonics, however, have various wavenumbers, depending upon the perturbation vector. Hence, the difference between $\overline{\overline{P}}$ and $\overline{\overline{A}}$ is in the factors which multiply the main diagonal elements of the matrix, and that represent taking the derivatives of the perturbation array or the difference $(\overline{\psi}^{n+1} - \overline{\psi}^{n})$, respectively.

Given the perturbation matrix, the linear stability of the hexagon solution is determined by the eigenvalue(s) of $\overline{\overline{P}}$ with the largest real part. It must be kept in mind that $\overline{\overline{P}}$ is a $3h(h+1)+1$ by $3h(h+1)+1$ matrix, and that to determine the stability for even one hexagon solution of a particular choice of wavenumber k and bifurcation parameter μ requires a representative scan of the possible perturbation wavevectors. Thus, for the linear stability calculation to be practical, finding the eigenvalue(s) with the largest real part must be done quickly, making the computation of all the eigenvalues of $\overline{\overline{P}}$ for each choice of perturbation wavevector impractical. The method tried initially was inverse iteration, in an attempt to find only the eigenvalue(s) with the largest real part. However, inverse iteration proved to not be robust enough, and had an especially difficult time when, as the perturbation wavevector was modified, the eigenvalue(s) with the largest real part switched from one branch of eigenvalues to another (i.e. from the translation eigenvalues to the Hopf eigenvalues). The method finally used to find the eigenvalue(s) with the largest real part was the Matlab routine *eigs*. The routine *eigs* computes the eigenvalues of a real matrix using Arnoldi methods. For the computations in this paper, options of the routine *eigs* which return up to six eigenvalues with largest real part were chosen. The routine *eigs* proved to be both robust and relatively fast.

5.3 Results

Similarly to the results of the studies of the phenomenological equations (2.1) and (2.10), the study of (5.1) has two main divisions. First are the results of the Galerkin analysis described in the last section, which provide a stability diagram for the linear stability of the hexagon solution. Second are the results of the direct numerical simulation of (5.1) (cf. [45]).

The Galerkin analysis makes a linear stability calculation of the hexagon solution feasible, as noted in the previous section. However, the calculation is still unwieldy, which limits the results that can be obtained on practical grounds. In practice, the relevant limitation is imposed on the range of the bifurcation parameter μ, which, as shall be seen, is also the case for the direct numerical simulations of

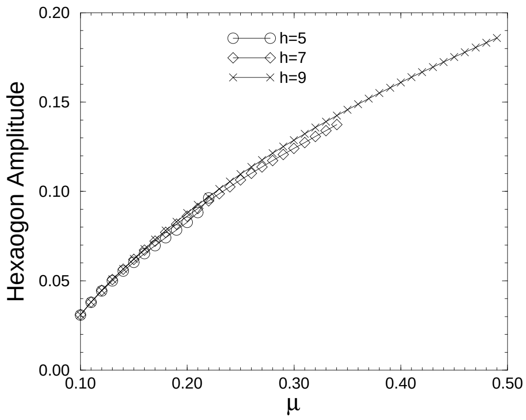


Figure 5.3: Values of the hexagonal solution as calculated by the Galerkin code for the parameters $b = 0.025$, $\Omega = 0$, $k = 0.85$. The number of harmonics kept in the Galerkin solution is denoted by h. For reference, the critical wavenumber for these parameters is $k_c = 0.78$. For $h = 5$, the highest value of the bifurcation parameter μ for which the Newton Galerkin method converges to a nontrivial solution is $\mu = \mu_{max} = 0.22$. For $h = 7$, $\mu_{max} = 0.35$, and for $h = 9$, $\mu_{max} > 0.49$.

(5.1). In this analysis, we generally keep $b = 0.025$, and vary μ and Ω. The first concern is of course the ability to find a solution for the given values of μ, Ω, and chosen hexagon solution wavenumber k. For any Ω and k, increasing μ increases the number of harmonics h needed in the Galerkin analysis to converge to a solution. In Fig.5.3, hexagon solutions for various numbers of harmonics h included are plotted against the bifurcation parameter μ, for $b = 0.025$, $\Omega = 0$, and $k = 0.85$. For reference, the critical wavenumber for these parameters is $k_c = 0.78$. In each case of h shown in Fig.5.3, first the hexagon solution for $\mu = 0.10$ was found using Newton's method. The initial guess was all amplitudes equal to zero, except the amplitudes of the original hexagon modes, which were given reasonable values. Then μ was increased in increments of 0.01. The solution for each new value of μ was found by Newton's method with the initial guess being the solution for the previous value of μ. As μ increases, the solutions for the various values of h agree less. For $\mu = 0.22$, the solution for $h = 5$ begins to vary sharply from the previous pattern, and for $\mu = 0.23$, the Galerkin analysis for $h = 5$ converges to the trivial solution, and is

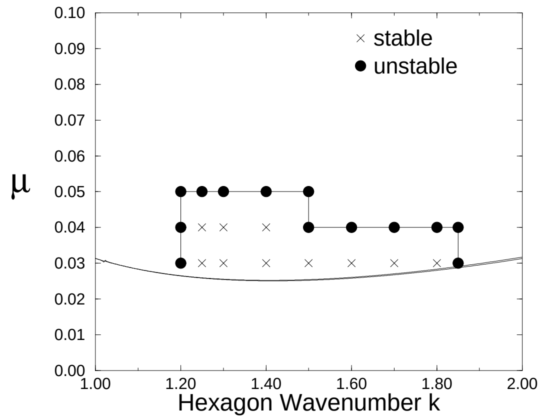


Figure 5.4: Parameter values for which the hexagon solution of (5.1) is stable (crosses) and unstable (disks) for $\Omega = 3$, $b = 0.025$. The neutral and saddle node curves are the solid curves at the bottom of the figure, almost indistinguishable from each other. The Hopf curve is not visible. The critical wavenumber is $k_c = 1.41$.

not shown. A similar effect occurs for $h = 7$ at about $\mu = 0.35$. For $h = 9$, the Galerkin method converges until $\mu = 0.49$.

However, finding a hexagon solution is only the first half of the problem - one must also consider the stability calculation. It is in the stability calculation, more specifically, in the repeated finding of the largest eigenvalue(s) of the perturbation matrices, that the number of harmonics h used really has an effect on the practicality of the analysis. For small Ω, the stability region of the hexagons extends to relatively high values of μ, and the number of harmonics h necessary to obtain accurate linear stability calculations for these larger values of μ is forbidding, considering the computing power available. However, for $\Omega = 3$, the stability region is restricted to relatively low values of μ, and hence a good approximation to the stability boundary can be given, as shown in Fig.5.4. The critical wavenumber is $k_c = 1.41$. One should note that the relevant instabilities of the hexagon solution usually occurred for perturbation angles $|\theta|$ near the maximum considered value of $\theta_{max} = 30°$.

The calculation of both the hexagon and roll solutions by direct numerical

type of solution	Ω	k	number of harmonics	μ_{max}
roll	0	0.85	8	.13
roll	0	0.85	16	.28
roll	0	0.85	32	.31
roll	0	0.85	64	.31
roll	2	1.02	8	.11
roll	2	1.02	16	.23
roll	2	1.02	32	.24
roll	2	1.02	64	.24
hexagon	0	0.85	8	.12
hexagon	0	0.85	16	.23
hexagon	0	0.85	32	.31
hexagon	0	0.85	64	.31
hexagon	2	1.02	8	.14
hexagon	2	1.02	16	.19
hexagon	2	1.02	32	.19
hexagon	2	1.02	64	.19

Table 5.1: The highest value of the bifurcation parameter μ_{max} for which the direct numerical simulation of (5.1) finds a solution, depending upon the type of solution, rotation rate Ω, wavenumber of the solution k, and the number of harmonics of the main solution mode(s) used in the simulation. In all cases, $b = 0.025$.

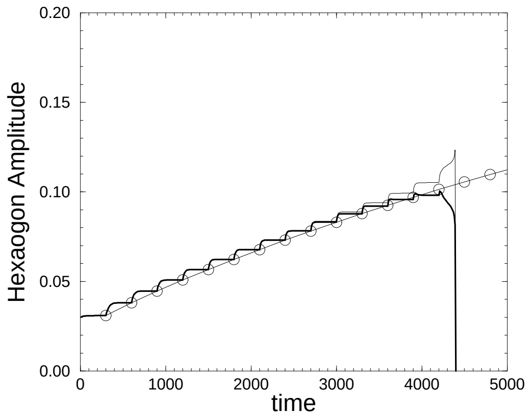


Figure 5.5: A time plot of the solution as μ increases step by step showing the blowup of the solution for $\mu = 0.25$. The thin solid line is mode A of the direct numerical simulation, the thick solid line is modes B and C. Plotted along is the Galerkin solution with nine harmonics kept (cf. Fig.5.3). The parameters are $b = 0.025$, $\Omega = 0$, $k = 0.85$. Sixteen harmonics of the hexagon solution were kept in the numerical simulation.

simulation of (5.1) is also limited in μ. The same code used in the direct numerical simulations of (2.1) and (2.10) is used in the simulation of (5.1). The code was discussed in Chapter 3. Again, as is the case with the Galerkin analysis, the number of harmonics of the hexagon modes included in the calculation determines how large a μ is allowed. Table 5.1 lists the highest value, μ_{max}, for which the numerical method finds the given type of solution, given Ω, the solution wavenumber k, and the number of harmonics of the solution modes used in the simulation. All data is for $b = 0.025$. The greater number of harmonics used, the coarser the numerical grid is in Fourier space, and thus the fewer relevant perturbations that can be considered. For both roll and hexagon solutions, and for both $\Omega = 0$ and $\Omega = 2$, increasing the number of harmonics included in the numerical grid in Fourier space helped increase μ_{max} only up to a threshold. At this μ_{max} the solution always diverges. Decreasing the timestep does not alter μ_{max}. Thus, the threshold of μ_{max} for a given set of parameters indicates that the limit of μ_{max} is not the result of an inadequate pseudospectral expansion of the solution, or of insufficiently small timesteps, but is

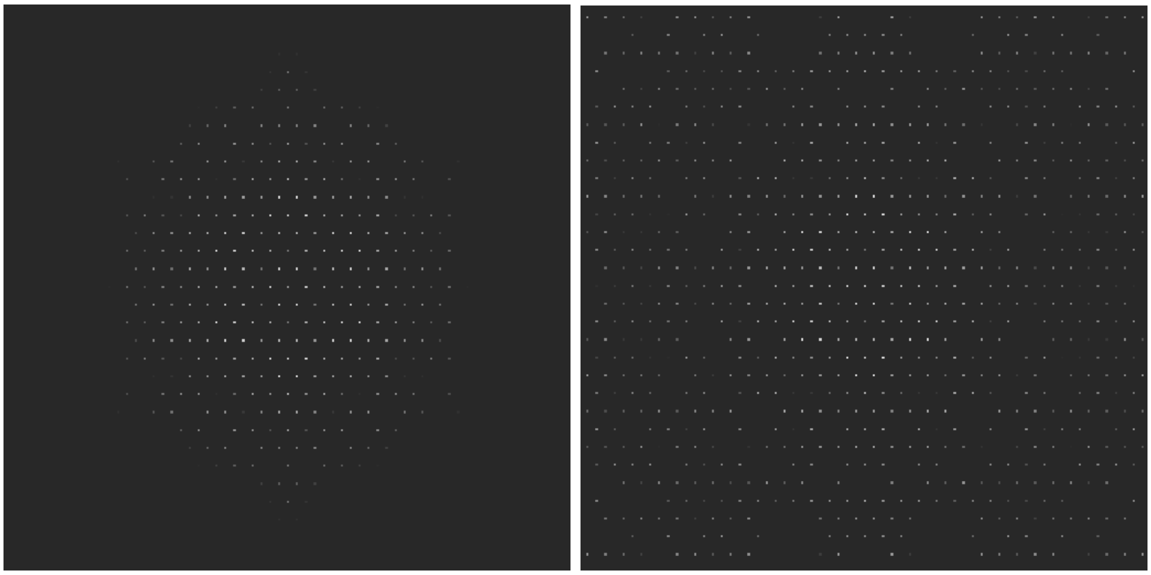


Figure 5.6: The Fourier spectrum of the hexagon solution on a logarithmic scale for $\mu = 0.24$, $\Omega = 0$, $k = 0.85$, $b = 0.025$, and time *a)* t=3500, *b)* t=3680. This corresponds to the blowup in Fig.5.5. Frame *a* shows the Fourier spectrum of the solution at the point where the bifurcation parameter μ is changed from $\mu = 0.23$ to $\mu = 0.24$. In the middle of each frame, we see the hexagon modes centered around the Fourier space origin. As can be seen from Fig.5.5, for lower values of μ the hexagon solution exists (true stability cannot be concluded from this simulation). For $\mu = 0.24$, the highest harmonics of the hexagon solution become relevant, and their growth leads to the blowup, as shown in *b*.

instead a threshold of stability of the numerical method. The cause of this instability of the method is not clear.

Fig.5.5 shows a comparison of the Galerkin solution with $h = 9$ from Fig.5.3 and the solution from the numerical simulation of (5.1) for corresponding values of μ. The thin line is mode A of the direct numerical simulation, the thick line is modes B and C. Their disagreement is due to the anisotropy of the Fourier grid (discussed in Chapter 3); more harmonics kept in the simulation make the agreement better. For the range of μ for which there is a numerical hexagon solution, the two agree very well. The parameters are $b = 0.025$, $\Omega = 0$, $k = 0.85$, where sixteen harmonics of the hexagon solution were kept in the numerical simulation. The sharp jumps in the numerical solution coincide with changes in the value of μ used in the simulation. The blowup is detailed in Fig.5.6, with the two frames showing the Fourier spectrum of the hexagon solution on a logarithmic scale at two times, namely when the bifurcation parameter is first changed from $\mu = 0.23$ to $\mu = 0.24$, and just before the actual blowup of the solution. The parameters are $\Omega = 0$, $k = 0.85$, $b = 0.025$. In Fig.5.6a, we see the Fourier spectrum of a hexagon solution, with the original hexagon modes close to the Fourier space origin, and a certain number of excited harmonics. In Fig.5.6b, even higher harmonics have become excited, causing the solution to diverge. It is this excitement of higher harmonics that we surmise limits the validity of the direct numerical simulations to $\mu < \mu_{max}$. Another manifestation of this excitement of the higher harmonics is shown in Fig.5.7. A cut of the real space hexagon solution is shown for the last two time steps before blowup. The bottom panel, which is a detail of the top panel, shows the influence of the higher harmonics on the solution - the peaks and troughs of the real space solution become sharper. At the next timestep there is no solution. All Fourier amplitudes return the value NaN, for "not a number".

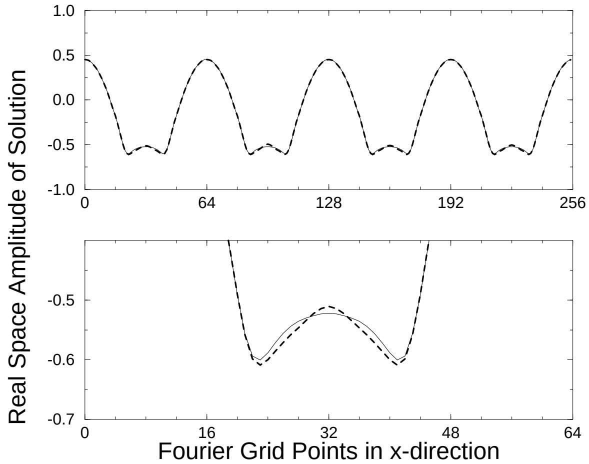


Figure 5.7: A cut of the real space hexagon solution in the x direction in the last two timesteps before blowup. The parameters are again $\mu = 0.24$, $\Omega = 0$, $k = 0.85$, $b = 0.025$. The solid line is the solution at the second timestep before blowup, and the dashed line is the solution at the timestep just before blowup. The bottom panel is a detail of the top. The dashed line has sharper corners than the solid line, an indication that the strength of high frequency modes is increasing. This is consistent with Fig.5.6, which indicates the highest harmonics of the hexagon solution becoming excited and leading to blowup.

Chapter 6

Conclusions

This thesis studied three equations. The first two were phenomenological equations modeling Rayleigh-Bénard convection, the second of these (2.10) an extension of the first (2.1). The third equation (5.1) was derived from the governing equations of surface-tension driven rotating Marangoni convection with poorly conducting boundary conditions in the long-wave limit. For all three equations, we studied the stability of a weakly nonlinear hexagonal planform solution with respect to perturbations outside of the subspace of Fourier space spanned by the modes corresponding to the hexagon solution itself. For all three equations, instabilities to these perturbations were found for lower values of the bifurcation parameter than instabilities to perturbations within said hexagon subspace.

For both the minimal (2.1) and extended (2.10) phenomenological models of Rayleigh-Bénard convection, the instabilities were long-wave at onset of convection. Further away from onset, the instabilities became short-wave. For sufficient rotation, the instabilities could become oscillatory, and for sufficiently high rotation, all instabilities did so.

Direct numerical simulation of the minimal (2.1) and extended (2.10) phenomenological equations was used to examine the nonlinear evolution of instabilities. The steady instabilities examined lead to relaxation to a stable hexagon state. Oscillatory instabilities could lead to persistent dynamics, which were bistable with the stable hexagon solution for some parameter values. Near the linear stability boundary of the hexagons, the oscillatory instability could lead to steady oscillations of modulated hexagons, while further away from the stability boundary the oscillations broke down leading to spatio-temporal chaos with a residual sixfold symmetry. For parameter values for which the hexagons are unstable at all wavenumbers, persistent dynamics involving spatio-temporal chaos with an isotropic Fourier spectrum were found. Furthermore, these simulations indicated the necessity of using a model equation that preserves the isotropy of the physical system. This would allow for short-wave perturbations of the hexagon solution, as a long-wave model could not resolve the nonlinear evolution of the instabilities of the hexagons.

To compare the minimal (2.1) and extended (2.10) phenomenological models with experimental results would be the next step in their study. This would require that the coefficients of the model equations be correlated with the physical parameters of the governing equations. Equations similar to (2.1) and (2.10) have been obtained from the governing equations as order parameter equations; however, no rigorous derivation of the studied phenomenological equations from the governing equations is known to us. A comparison between solutions generated by amplitude equations and by direct numerical simulations of the governing equations showed good agreement [36]. This indicates that a properly obtained Swift-Hohenberg type model can provide an accurate representation of the convection.

The model equation (5.1) for rotating Marangoni convection proved more difficult to study. The regimes of validity for both the weakly nonlinear and the numerical analyses of (5.1) were limited to the region close to the neutral curve. A Galerkin expansion was used to find the stability boundary of the hexagon solution for a high rotation rate. In this case all instabilities were oscillatory. In the absence of rotation the Galerkin and weakly nonlinear expansions agreed near the neutral curve. Further away from the neutral curve, the Galerkin expansion required that more harmonics of the hexagon solution be kept in the expansion for the stability calculation to be valid. Direct numerical simulations of (5.1) also required a large number of harmonics for the hexagon solution to exist. If the bifurcation parameter μ exceeded a threshold μ_{max}, the simulated solution diverged.

The values of the coefficients of the derived equation for long-wave Marangoni convection (5.1) correspond to specific values of the parameters of the governing equations. Further work would include the direct numerical simulations of (5.1). Again, parameter values where persistent dynamics occur would be sought. The direct numerical simulations of (5.1) could then easily be compared with experimental results in the appropriate parameter regimes of the governing equations.

Bibliography

[1] F. Zhong and R. Ecke, Chaos **2**, 163 (1992).

[2] E. Bodenschatz *et al.*, Physica D **61**, 77 (1992).

[3] B. Echebarria and H. Riecke, Physica D **139**, 97 (2000).

[4] M. Cross and P. Hohenberg, Rev. Mod. Phys. **65**, 851 (1993).

[5] G. Küppers and D. Lortz, J. Fluid Mech. **35**, 609 (1969).

[6] L. Ning and R. Ecke, Phys. Rev. E **47**, R2991 (1993).

[7] F. Zhong, R. Ecke, and V. Steinberg, Physica D **51**, 596 (1991).

[8] Y. Hu, R. Ecke, and G. Ahlers, Phys. Rev. Lett. **74**, 5040 (1995).

[9] Y. Hu, R. Ecke, and G. Ahlers, Phys. Rev. E **55**, 6928 (1997).

[10] M. Cross, D. Meiron, and Y. Tu, Chaos **4**, 607 (1994).

[11] Y. Tu and M. Cross, Phys. Rev. Lett. **69**, 2515 (1992).

[12] H. Xi, J. Gunton, and G. Markish, Physica A **204**, 741 (1994).

[13] A. Kudrolli and J. Gollub, Physica D **97**, 133 (1996).

[14] N. Tufillaro, R. Ramshankar, and J. Gollub, Phys. Rev. Lett. **62**, 422 (1989).

[15] Q. Ouyang and H. Swinney, Chaos **1**, 411 (1991).

[16] F. H. Busse and R. M. Clever, in *Recent Developments in Theoretical and Experimental Fluid Mechanics*, edited by U. Müller, K. G. Roesner, and B. Schmidt (Springer, Berlin, 1979).

[17] F. Busse and K. Heikes, Science **208**, 173 (1980).

[18] K. Heikes and F. Busse, Ann. N.Y. Acad. Sc. **357**, 28 (1980).

[19] R. May and W. Leonard, SIAM J. Appl. Math. **29**, 243 (1975).

[20] J. Swift, in *Contemporary Mathematics Vol. 28* (American Mathematical Society, Providence, 1984), p. 435.

[21] A. Soward, Physica D **14**, 227 (1985).

[22] J. Millàn-Rodriguez *et al.*, Phys. Rev. A **46**, 4729 (1992).

[23] E. Siggia and A. Zippelius, Phys. Rev. Lett. **47**, 835 (1981).

[24] P. Manneville, J. de Physique **44**, 759 (1983).

[25] J. Bragard and M. Velarde, J. Fluid Mech. **368**, 165 (1998).

[26] J. Swift and P. Hohenberg, Phys. Rev. A **15**, 319 (1977).

[27] M. Neufeld, R. Friedrich, and H. Haken, Z. Phys. B **92**, 243 (1993).

[28] C. Kubstrup, H. Herrero, and C. Perez-Garcia, Phys. Rev. E **54**, 1560 (1996).

[29] H. Sakaguchi and H. Brand, J. Phys. II **7**, 1325 (1997).

[30] C. Crawford and H. Riecke, Physica D **129**, 83 (1999).

[31] F. Sain and H. Riecke, Physica D **144**, 124 (2000).

[32] M. Bestehorn and R. Friedrich, in *A Perspective Look at Nonlinear Media*, edited by J. Parisi, S. Müller, and W. Zimmermann (Springer, Berlin, 1998), p. 31.

[33] M. Bestehorn and R. Friedrich, Phys. Rev. E **59**, 2642 (1999).

[34] H. Haken, *Advanced Synergetics : Instability Hierarchies of Self- Organizing Systems and Devices* (Springer-Verlag, Berlin, 1983).

[35] A. Mancho, F. Sain, and H. Riecke, unpublished .

[36] M. Bestehorn, Phys. Rev. E **48**, 3622 (1993).

[37] S. Cox, SIAM J. Appl. Math. **58**, 1338 (1998).

[38] E. Knobloch, Physica D **41**, 450 (1990).

[39] V. Gertsberg and G. Sivashinsky, Prog. Theor. Phys. **66**, 1219 (1981).

[40] D. Riahi, Int. J. Eng. Sci. **32**, 877 (1994).

[41] M. Sushchik and L. Tsimring, Physica D **74**, 90 (1994).

[42] S. Morris, E. Bodenschatz, D. Cannell, and G. Ahlers, Phys. Rev. Lett. **71**, 2026 (1993).

[43] W. Decker, W. Pesch, and A. Weber, Phys. Rev. Lett. **73**, 648 (1994).

[44] R. Cakmur, D. Egolf, B. Plapp, and E. Bodenschatz, Phys. Rev. Lett. **79**, 1853 (1997).

[45] L. Shtilman and G. Sivashinsky, Physica D **52**, 477 (1991).

Appendix A

Galerkin Code

```
c This program calculates the roll and hexagon solutions for the
c Marangoni model equation (5.1), and computes the solutions' stability.
c It uses the Galerkin method, with only the modes found on the
c hexagonal grid.
c For reference in understanding the structure of this program, the
c reader should see Fig.5.2 which show the hexagonal
c subspace modes and the perturbations to those modes.
c The hexagonal subspace solution (be it hexagons or rolls) is computed
c using either Newton iteration or Crank-Nicolson timestepping.
c The number of harmonics used is n, and the number of modes used is
c 3*n*(n+1)+1.
c After convergence to a solution, the perturbations are added and the
c eigenvalues found.
      program main
      integer n
      parameter (m=500)
      integer i
      real*8 waveno(m,3)
      integer matplan(m,m)
      integer corners(50,50,3),rows(50,2)
      complex*16 u(m)
      integer method,ictype
```

```
      real*8 mubot,mutop,muinc
      real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      real*8 time,dt,theta,r
      integer nsteps
      common /runpar/ time,dt,theta,r,nsteps
c *** default parameters of Ana's equation***
      mu = 0.11
      beta = 0.025
      kappa = 0.6d0
      g2 = 0.d0
      g3 = -48.d0/35.d0
      nu = 0.1d0
      gamma = 0.d0
      lambda = 1.d0/15.d0
      q = 0.85d0
c These are Swift-Hohenberg parameters
c mu = 2.0
c beta = 0.9
c kappa = 0.0d0
c g2 = 0.d0
c g3 = 0.d0
c nu = 0.d0
c gamma = 0.d0
c lambda = 1.d0
c q = 1.d0
c *** default run parameters ***
      nsteps = 15
      time = 0.d0
      dt = 0.1d0
```

```
      theta = 0.5d0
      n = 3
      mubot = 0.1d0
      mutop = 0.35d0
      muinc = 0.01d0
      method = 1
      ictype = 2
      r = .1
c *** MAIN MENU ***
   20 write(*,*)
      write(*,*) '*** Main Menu ***'
      write(*,*) 'Enter choice:'
      write(*,*) '1: Change equation parameter.'
      write(*,*) '2: Run current params in time.'
      write(*,*) '3: Change run parameter.'
      if (ictype.eq.1) write(*,*) '4: Compute IC (now: rolls).'
      if (ictype.eq.2) write(*,*) '4: Compute IC (now: up-hex).'
      if (ictype.eq.3) write(*,*) '4: Compute IC (now: down-hex).'
      write(*,*) '5: Loop over mu.'
      if (method.eq.1) write(*,*) '6: Change method (now: Newton).'
      if (method.eq.2) write(*,*) '6: Change method (now: C-N).'
      if (ictype.eq.1) write(*,*) '7: Choose IC (now: rolls).'
      if (ictype.eq.2) write(*,*) '7: Choose IC (now: up-hex).'
      if (ictype.eq.3) write(*,*) '7: Choose IC (now: down-hex).'
      write(*,*) '8: Compute stability.'
      write(*,*) '9: Quit.'
      read(*,*,err=99) choice
      if (choice.eq.1) then
         write(*,*) 'Enter choice and value:'
         write(*,*) '1: mu = ',mu
```

```
write(*,*) '2: beta = ',beta
write(*,*) '3: kappa = ',kappa
write(*,*) '4: g2 = ',g2
write(*,*) '5: g3 = ',g3
write(*,*) '6: nu = ',nu
write(*,*) '7: gamma = ',gamma
write(*,*) '8: lambda = ',lambda
write(*,*) '9: wavenum q= ',q
read(*,*,err=99) choice,value
if (choice.eq.0) then
   goto 20
end if
if (choice.eq.1) then
   mu = value
   goto 20
end if
if (choice.eq.2) then
   beta = value
   goto 20
end if
if (choice.eq.3) then
   kappa = value
   goto 20
end if
if (choice.eq.4) then
   g2 = value
   goto 20
end if
if (choice.eq.5) then
   g3 = value
```

```
      goto 20
    end if
    if (choice.eq.6) then
      nu = value
      goto 20
    end if
    if (choice.eq.7) then
      gamma = value
      goto 20
    end if
    if (choice.eq.8) then
      lambda = value
      goto 20
    end if
    if (choice.eq.9) then
      q = value
      goto 20
    end if
  end if
  if (choice.eq.2) then
    write(*,*)
    write(*,*) 'Running ....'
    write(*,*)
    call timeloop(n,u,method,corners,rows,waveno,matplan)
    go to 20
  end if
  if (choice.eq.3) then
    write(*,*) 'Enter choice and value:'
    write(*,*) '1: nsteps = ',nsteps
    write(*,*) '2: time = ',time
```

```
write(*,*) '3: dt = ',dt
write(*,*) '4: theta = ',theta
write(*,*) '5: r = ',r
write(*,*) '6: n = ',n
write(*,*) '7: mubot = ',mubot
write(*,*) '8: mutop = ',mutop
write(*,*) '9: muinc = ',muinc
read(*,*,err=99) choice,value
if (choice.eq.0) then
   goto 20
end if
if (choice.eq.1) then
   nsteps = value
   goto 20
end if
if (choice.eq.2) then
   time = value
   goto 20
end if
if (choice.eq.3) then
   dt = value
   goto 20
end if
if (choice.eq.4) then
   theta = value
   goto 20
end if
if (choice.eq.5) then
   r = value
   goto 20
```

```
    end if
    if (choice.eq.6) then
      n = value
      goto 20
    end if
    if (choice.eq.7) then
      mubot = value
      goto 20
    end if
    if (choice.eq.8) then
      mutop = value
      goto 20
    end if
    if (choice.eq.9) then
      muinc = value
      goto 20
    end if
  end if
  if (choice.eq.4) then
    do i = 1, 3*n*(n+1)+1
      waveno(i,1) = 0.d0
      waveno(i,2) = 0.d0
      waveno(i,3) = 0.d0
    end do
    call wavenum(n,waveno,corners,rows,matplan)
    call ic(n,u,r,ictype)
    goto 20
  end if
  if (choice.eq.5) then
    write(*,*)
```

```
         write(*,*) 'Running ....'
         write(*,*)
         call muloop(n,u,mubot,mutop,muinc,method,corners,rows,waveno
     *                                ,matplan)
         goto 20
      end if
      if (choice.eq.6) then
         write(*,*) 'Which method to use?'
         write(*,*) ' 1: Newton'
         write(*,*) ' 2: Crank-Nicolson'
         write(*,*) 'Enter choice:'
         read(*,*,err=99) method
         goto 20
      end if
      if (choice.eq.7) then
         write(*,*) 'Which initial condition?'
         write(*,*) ' 1: Rolls'
         write(*,*) ' 2: Up-Hexagons'
         write(*,*) ' 3: Down-Hexagons'
         write(*,*) 'Enter choice:'
         read(*,*,err=99) ictype
         goto 20
      end if
      if (choice.eq.8) then
         write(*,*)
         write(*,*) 'Running ....'
         write(*,*)
         call perturbloop(n,u,corners,rows,waveno,matplan)
         go to 20
      end if
```

```
      if (choice.eq.9) then
         goto 100
      end if
      99 write(*,*) '****************************'
      write(*,*) '* ERROR : input not accepted *'
      write(*,*) '****************************'
      go to 20
  100 end
c————————————————————————————————————————
c computes initial condition in hexagonal subspace
c ictype determines rolls or hex
      subroutine ic(n,u,r,ictype)
      implicit none
      integer n,ictype
      complex*16 u(3*n*(n+1)+1)
      integer i
      real*8 roll,hex,r
      real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      open(91,file='ic.dat')
      do i = 1, 3*n*(n+1)+1
         u(i) = 0.d0
      end do
c Assign initial values to roll and hex - can use
c theoretical values or just put in constants.
      roll = .1d0
      hex = .1d0
      if (ictype.eq.1) then
         u((3*n*(n+1)+2)/2+1) = roll
         u((3*n*(n+1)+2)/2-1) = roll
```

```
      u((3*n*(n+1)+2)/2-2*n) = 0.d0
      u((3*n*(n+1)+2)/2+2*n) = 0.d0
      u((3*n*(n+1)+2)/2-2*n-1) = 0.d0
      u((3*n*(n+1)+2)/2+2*n+1) = 0.d0
    elseif (ictype.eq.2) then
      u((3*n*(n+1)+2)/2+1) = hex
      u((3*n*(n+1)+2)/2-1) = hex
      u((3*n*(n+1)+2)/2-2*n) = hex
      u((3*n*(n+1)+2)/2+2*n) = hex
      u((3*n*(n+1)+2)/2-2*n-1) = hex
      u((3*n*(n+1)+2)/2+2*n+1) = hex
    elseif (ictype.eq.3) then
      u((3*n*(n+1)+2)/2+1) = hex
      u((3*n*(n+1)+2)/2-1) = hex
      u((3*n*(n+1)+2)/2-2*n) = hex
      u((3*n*(n+1)+2)/2+2*n) = hex
      u((3*n*(n+1)+2)/2-2*n-1) = hex
      u((3*n*(n+1)+2)/2+2*n+1) = hex
    end if
    do i = 1, 3*n*(n+1)+1
      write(91,9100) dreal(u(i))
    end do
    close(91)
9100 format(f15.8)
    return
    end
c------------------------------------------------------------
c Finds stable hexagonal subspace solutions for a range of mu.
    subroutine muloop(n,u,mubot,mutop,muinc,method,corners,rows
*                            ,waveno,matplan)
```

```
      implicit none
      integer n
      complex*16 u(3*n*(n+1)+1)
      integer i,method
      real*8 mubot,mutop,muinc
      real*8 waveno(3*n*(n+1)+1,3)
      integer matplan(3*n*(n+1)+1,3*n*(n+1)+1)
      integer corners(2*n+1,2*n+1,3),rows(2*n+1,2)
      real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      real*8 time,dt,theta,r
      integer nsteps
      common /runpar/ time,dt,theta,r,nsteps
c open(25,file='mode.ratios.dat')
      open(26,file='modes.final.dat')
      do mu=mubot,mutop,muinc
        call timeloop(n,u,method,corners,rows,waveno,matplan)
        write(26,2500) mu
        write(26,2600) dreal(u((3*n*(n+1)+2)/2+1))
        write(26,2600) dreal(u((3*n*(n+1)+2)/2-1))
        write(26,2600) dreal(u((3*n*(n+1)+2)/2-2*n))
        write(26,2600) dreal(u((3*n*(n+1)+2)/2+2*n))
        write(26,2600) dreal(u((3*n*(n+1)+2)/2-2*n-1))
        write(26,2601) dreal(u((3*n*(n+1)+2)/2+2*n+1))
        call flush1(26)
      end do
2101 format(f18.8,$)
2102 format(f18.8)
2500 format(f10.4,$)
2501 format(f10.4)
```

```
2600 format(f15.12,$)
2601 format(f15.12)
      close(26)
      return
      end
c————————————————————————————————————————
c This is the master subroutine for finding a hexagonal subspace
c solution. It goes through iterations if
c using Newton, timesteps if using CN.
c Each iteration or timestep is set up as a matrix equation to
c to be solved.
      subroutine timeloop(n,u,method,corners,rows,waveno,matplan)
      implicit none
      integer n
      complex*16 u(3*n*(n+1)+1),rhs(3*n*(n+1)+1)
      complex*16 lhs(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lhstemp(3*n*(n+1)+1,3*n*(n+1)+1)
      real*8 newsum,oldsum
      integer info,ipiv(3*n*(n+1)+1),timesteps,i,method
      real*8 waveno(3*n*(n+1)+1,3)
      integer matplan(3*n*(n+1)+1,3*n*(n+1)+1)
      integer corners(2*n+1,2*n+1,3),rows(2*n+1,2)
      real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      real*8 time,dt,theta,r
      integer nsteps
      common /runpar/ time,dt,theta,r,nsteps
      open(21,file='modes.dat') !time series of all modes
      open(22,file='lhs.dat') !elements of matrix A
      open(23,file='rhs.dat') !elements of vector b
```

```
          open(24,file='basic.modes.dat') !time series of basic hex modes
          newsum = dreal(u((3*n*(n+1)+2)/2+1))
     *                     +dreal(u((3*n*(n+1)+2)/2-1))
     *                     +dreal(u((3*n*(n+1)+2)/2-2*n))
     *                     +dreal(u((3*n*(n+1)+2)/2+2*n))
     *                     +dreal(u((3*n*(n+1)+2)/2-2*n-1))
     *                     +dreal(u((3*n*(n+1)+2)/2+2*n+1))
          do timesteps=1,nsteps
            call makerhs(n,u,rhs,method,corners,rows,waveno,matplan)
            call writerhs(n,rhs)
            call makelhs(n,u,lhs,method,corners,rows,waveno,matplan)
            call writelhs(n,lhs)
            call writemodes(n,time,u)
            call zgesv(3*n*(n+1)+1,1,lhs,3*n*(n+1)+1,ipiv,rhs,
     *                     3*n*(n+1)+1,info)
            do i=1,3*n*(n+1)+1
              if (method.eq.1) u(i) = u(i) + rhs(i)
              if (method.eq.2) u(i) = rhs(i)
            end do
            time = time + dt
            oldsum = newsum
            newsum = dreal(u((3*n*(n+1)+2)/2+1))
     *                     +dreal(u((3*n*(n+1)+2)/2-1))
     *                     +dreal(u((3*n*(n+1)+2)/2-2*n))
     *                     +dreal(u((3*n*(n+1)+2)/2+2*n))
     *                     +dreal(u((3*n*(n+1)+2)/2-2*n-1))
     *                     +dreal(u((3*n*(n+1)+2)/2+2*n+1))
            if (dabs(newsum-oldsum).le. 0.000000005) go to 40
          end do
2100 format(f8.4)
```

```
2101 format(f18.12,$)
2102 format(f18.12)
  40 close(22)
     close(23)
     return
     end
c———————————————————————————————————
c This writes out the values of all the hexagon subspace modes
c at each timestep. Useful only for small n.
     subroutine writemodes(n,time,u)
     implicit none
     integer n
     complex*16 u(3*n*(n+1)+1)
     real*8 time
     integer i
     write(21,2100) time
     do i = 1,3*n*(n+1)
        write(21,2101) dreal(u(i))
     end do
     write(21,2102) dreal(u(3*n*(n+1)+1))
     write(21,*)
     write(24,2101) time
     write(24,2101) dreal(u((3*n*(n+1)+2)/2+1))
     write(24,2101) dreal(u((3*n*(n+1)+2)/2-1))
     write(24,2101) dreal(u((3*n*(n+1)+2)/2-2*n))
     write(24,2101) dreal(u((3*n*(n+1)+2)/2+2*n))
     write(24,2101) dreal(u((3*n*(n+1)+2)/2-2*n-1))
     write(24,2102) dreal(u((3*n*(n+1)+2)/2+2*n+1))
     call flush1(21)
     call flush1(24)
```

```
2100 format(f8.4)
2101 format(f18.12,$)
2102 format(f18.12)
2103 format(f11.4,$)
2104 format(f11.4)
      return
      end
c————————————————————————————————————————
c Prints out matrix A of matrix equation A*x = b,
c the matrix equation generated at each iteration/timestep.
      subroutine writelhs(n,lhs)
      implicit none
      integer n
      complex*16 lhs(3*n*(n+1)+1,3*n*(n+1)+1)
      integer i,j
      real*8 time,dt,theta,r
      integer nsteps
      common /runpar/ time,dt,theta,r,nsteps
      open(81,file='checkzeros.dat')
      write(22,2100) time
      do j = 1,3*n*(n+1)+1
        do i = 1,3*n*(n+1)
          write(22,2101) dreal(lhs(j,i))
          if (lhs(j,i).ne.0) write(81,*) j,i
        end do
        write(22,2102) dreal(lhs(j,3*n*(n+1)+1))
        if (lhs(j,3*n*(n+1)+1).ne.0) write(81,*) j,3*n*(n+1)+1
      end do
      write(22,*)
      close(81)
```

```
2100 format(f8.4)
2101 format(f18.8,$)
2102 format(f18.8)
      return
      end
c-----------------------------------------------------------
c Prints out vector b of matrix equation A*x = b,
c the matrix equation generated at each iteration/timestep.
      subroutine writerhs(n,rhs)
      implicit none
      integer n
      complex*16 rhs(3*n*(n+1)+1)
      integer i
      real*8 time,dt,theta,r
      integer nsteps
      common /runpar/ time,dt,theta,r,nsteps
      write(23,2100) time
      do i = 1,3*n*(n+1)
        write(23,2101) dreal(rhs(i))
      end do
      write(23,2102) dreal(rhs(3*n*(n+1)+1))
      write(23,*)
      call flush1(23)
2100 format(f8.4)
2101 format(f18.12,$)
2102 format(f18.12)
      return
      end
c-----------------------------------------------------------
c Makes vector b of matrix equation A*x = b,
```

```
c the matrix equation generated at each iteration/timestep.
      subroutine makerhs (n,u,rhs,method,corners,rows,waveno,matplan)
      implicit none
      integer n
      complex*16 u(3*n*(n+1)+1),uxx(3*n*(n+1)+1)
      complex*16 uxxxx(3*n*(n+1)+1)
      complex*16 rhs(3*n*(n+1)+1)
      complex*16 uyy(3*n*(n+1)+1),uyyyy(3*n*(n+1)+1)
      complex*16 uxxyy(3*n*(n+1)+1)
      complex*16 ux(3*n*(n+1)+1),uxxx(3*n*(n+1)+1)
      complex*16 uy(3*n*(n+1)+1),uyyy(3*n*(n+1)+1)
      complex*16 ux2(3*n*(n+1)+1),uy2(3*n*(n+1)+1)
      complex*16 uxx2(3*n*(n+1)+1)
      complex*16 uxy2(3*n*(n+1)+1),uyy2(3*n*(n+1)+1)
      complex*16 uxxy(3*n*(n+1)+1),uxyy(3*n*(n+1)+1)
      complex*16 uxxuyy(3*n*(n+1)+1)
      complex*16 uxuxxx(3*n*(n+1)+1),uxuxxy(3*n*(n+1)+1)
      complex*16 uxuxyy(3*n*(n+1)+1),uxuyyy(3*n*(n+1)+1)
      complex*16 uyuxxx(3*n*(n+1)+1),uyuxxy(3*n*(n+1)+1)
      complex*16 uyuxyy(3*n*(n+1)+1),uyuyyy(3*n*(n+1)+1)
      complex*16 uxuyuxx(3*n*(n+1)+1),uxuyuxy(3*n*(n+1)+1)
      complex*16 uxuyuyy(3*n*(n+1)+1)
      complex*16 ux2uxx(3*n*(n+1)+1),ux2uxy(3*n*(n+1)+1)
      complex*16 ux2uyy(3*n*(n+1)+1)
      complex*16 uy2uxx(3*n*(n+1)+1),uy2uxy(3*n*(n+1)+1)
      complex*16 uy2uyy(3*n*(n+1)+1)
      complex*16 luxmat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxuymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxxmat(3*n*(n+1)+1,3*n*(n+1)+1)
```

```
complex*16 lux2mat(3*n*(n+1)+1,3*n*(n+1)+1)
complex*16 luy2mat(3*n*(n+1)+1,3*n*(n+1)+1)
complex*16 u3(3*n*(n+1)+1),u2(3*n*(n+1)+1)
complex*16 uxy(3*n*(n+1)+1),uxuy(3*n*(n+1)+1)
integer i,j,method
real*8 waveno(3*n*(n+1)+1,3)
integer matplan(3*n*(n+1)+1,3*n*(n+1)+1)
integer corners(2*n+1,2*n+1,3),rows(2*n+1,2)
real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
real*8 time,dt,theta,r
integer nsteps
common /runpar/ time,dt,theta,r,nsteps
call vectcub(n,u,u3,corners,rows,matplan)
call vectsq(n,u,u2,corners,rows,matplan)
call diffx(n,u,ux,waveno)
call diff2x(n,u,uxx,waveno)
call diff3x(n,u,uxxx,waveno)
call diff4x(n,u,uxxxx,waveno)
call diffy(n,u,uy,waveno)
call diff2y(n,u,uyy,waveno)
call diff3y(n,u,uyyy,waveno)
call diff4y(n,u,uyyyy,waveno)
call diffx(n,uy,uxy,waveno)
call diff2y(n,uxx,uxxyy,waveno)
call diff2y(n,ux,uxyy,waveno)
call diff2x(n,uy,uxxy,waveno)
call vectsq(n,uxx,uxx2,corners,rows,matplan)
call vectsq(n,uxy,uxy2,corners,rows,matplan)
call vectsq(n,uyy,uyy2,corners,rows,matplan)
```

```
      call vectsq(n,ux,ux2,corners,rows,matplan)
      call vectsq(n,uy,uy2,corners,rows,matplan)
      call vecttomat(n,ux,luxmat,corners,rows,matplan)
      call vecttomat(n,uy,luymat,corners,rows,matplan)
      call vecttomat(n,uxx,luxxmat,corners,rows,matplan)
      call vecttomat(n,ux2,lux2mat,corners,rows,matplan)
      call vecttomat(n,uy2,luy2mat,corners,rows,matplan)
      call matvectmult(n,luxmat,uy,uxuy)
      call vecttomat(n,uxuy,luxuymat,corners,rows,matplan)
      call matvectmult(n,luxxmat,uyy,uxxuyy)
      call matvectmult(n,luxmat,uxxx,uxuxxx)
      call matvectmult(n,luxmat,uxxy,uxuxxy)
      call matvectmult(n,luxmat,uxyy,uxuxyy)
      call matvectmult(n,luxmat,uyyy,uxuyyy)
      call matvectmult(n,luymat,uxxx,uyuxxx)
      call matvectmult(n,luymat,uxxy,uyuxxy)
      call matvectmult(n,luymat,uxyy,uyuxyy)
      call matvectmult(n,luymat,uyyy,uyuyyy)
      call matvectmult(n,luxuymat,uxx,uxuyuxx)
      call matvectmult(n,luxuymat,uxy,uxuyuxy)
      call matvectmult(n,luxuymat,uyy,uxuyuyy)
      call matvectmult(n,lux2mat,uxx,ux2uxx)
      call matvectmult(n,lux2mat,uxy,ux2uxy)
      call matvectmult(n,lux2mat,uyy,ux2uyy)
      call matvectmult(n,luy2mat,uxx,uy2uxx)
      call matvectmult(n,luy2mat,uxy,uy2uxy)
      call matvectmult(n,luy2mat,uyy,uy2uyy)
      do i = 1,3*n*(n+1)+1
         if (method.eq.1) rhs(i) = (0.000001d0-beta)*u(i)
     *                  - mu*(uxx(i)+uyy(i))
```

```
*                    - lambda*(uxxxx(i) + uyyyy(i) + 2.d0*uxxyy(i))
*                    -(2.d0*kappa+nu)
*                    *(uxx2(i)+uyy2(i)+uxuxxx(i)+uxuxyy(i)
*                    +uyuxxy(i)+uyuyyy(i))
*                    - 4.d0*kappa*uxy2(i) - 2.d0*nu*uxxuyy(i)
*                    - gamma*(uyuxxx(i)+uyuxyy(i)-uxuxxy(i)-uxuyyy(i))
*                    - 2.d0*g2*(uxuyuxx(i)+uy2uxy(i)-ux2uxy(i)-uxuyuyy(i))
*                    - g3*(4.d0*uxuyuxy(i)+3.d0*ux2uxx(i)+3.d0*uy2uyy(i)
*                    +ux2uyy(i)+uy2uxx(i))
             if (method.eq.2) rhs(i) = u(i)/dt+(1.d0-theta)*(-beta*u(i)
*                    - mu*(uxx(i)+uyy(i))
*                    - lambda*(uxxxx(i) + uyyyy(i) + 2.d0*uxxyy(i)))
*                    +(2.d0*theta-1.d0)*((2.d0*kappa+nu)
*                                    *(
*                    uxx2(i)+uyy2(i)+uxuxxx(i)+uxuxyy(i)
*                    +uyuxxy(i)+uyuyyy(i))
*                    +4.d0*kappa*uxy2(i) + 2.d0*nu*uxxuyy(i)
*                    +gamma*(uyuxxx(i)+uyuxyy(i)-uxuxxy(i)-uxuyyy(i))
*                                    )
*                    + (3.d0*theta - 1.d0)*(
*                    2.d0*g2*(uxuyuxx(i)+uy2uxy(i)-ux2uxy(i)-uxuyuyy(i))
*                    +g3*(4.d0*uxuyuxy(i)+3.d0*ux2uxx(i)+3.d0*uy2uyy(i)
*                    +ux2uyy(i)+uy2uxx(i))
*                                    )
          end do
          return
          end
c----------------------------------------------------------------------
c Makes matrix A of matrix equation A*x = b,
c the matrix equation generated at each iteration/timestep.
```

```
      subroutine makelhs (n,u,lhs,method,corners,rows,waveno,matplan)
      implicit none
      integer n
      complex*16 u(3*n*(n+1)+1),ux(3*n*(n+1)+1),uxx(3*n*(n+1)+1)
      complex*16 uxxx(3*n*(n+1)+1),lhs(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 u2(3*n*(n+1)+1)
      complex*16 uy(3*n*(n+1)+1),uyy(3*n*(n+1)+1),uyyy(3*n*(n+1)+1)
      complex*16 ux2(3*n*(n+1)+1),uy2(3*n*(n+1)+1),uxy(3*n*(n+1)+1)
      complex*16 uxuy(3*n*(n+1)+1)
      complex*16 uxxy(3*n*(n+1)+1),uxyy(3*n*(n+1)+1)
      complex*16 uxuxx(3*n*(n+1)+1),uxuxy(3*n*(n+1)+1)
      complex*16 uxuyy(3*n*(n+1)+1)
      complex*16 uyuxx(3*n*(n+1)+1),uyuxy(3*n*(n+1)+1)
      complex*16 uyuyy(3*n*(n+1)+1)
      complex*16 luxmat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxuymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lux2mat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luy2mat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxxmat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luyymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxxxmat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxxymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxyymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luyyymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxuxxmat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxuxymat(3*n*(n+1)+1,3*n*(n+1)+1),
     *           luxuyymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luyuxxmat(3*n*(n+1)+1,3*n*(n+1)+1)
```

```
      complex*16 luyuxymat(3*n*(n+1)+1,3*n*(n+1)+1),
     *           luyuyymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 ltemp(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lkappanu(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lkappa(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lnu(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lgamma(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lg2(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lg3(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lu2mat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 dx(3*n*(n+1)+1),dxx(3*n*(n+1)+1)
      complex*16 dxxx(3*n*(n+1)+1),dxxxx(3*n*(n+1)+1)
      complex*16 dy(3*n*(n+1)+1),dyy(3*n*(n+1)+1),dyyy(3*n*(n+1)+1)
      complex*16 dyyyy(3*n*(n+1)+1)
      complex*16 dxy(3*n*(n+1)+1),dxxyy(3*n*(n+1)+1)
      complex*16 dxyy(3*n*(n+1)+1),dxxy(3*n*(n+1)+1)
      complex*16 lumat(3*n*(n+1)+1,3*n*(n+1)+1)
      integer i,j,method
      integer corners(2*n+1,2*n+1,3),rows(2*n+1,2)
      real*8 waveno(3*n*(n+1)+1,3)
      integer matplan(3*n*(n+1)+1,3*n*(n+1)+1)
      real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      real*8 time,dt,theta,r
      integer nsteps
      common /runpar/ time,dt,theta,r,nsteps
      do j = 1,3*n*(n+1)+1
        do i = 1,3*n*(n+1)+1
          lhs(i,j) = 0.d0
          lkappanu(i,j) = 0.d0
```

```
      lkappa(i,j) = 0.d0
      lnu(i,j) = 0.d0
      lgamma(i,j) = 0.d0
      lg2(i,j) = 0.d0
      lg3(i,j) = 0.d0
      ltemp(i,j) = 0.d0
   end do
end do
call vectsq(n,u,u2,corners,rows,matplan)
call vecttomat(n,u2,lu2mat,corners,rows,matplan)
call diffx(n,u,ux,waveno)
call diff2x(n,u,uxx,waveno)
call diff3x(n,u,uxxx,waveno)
call diffy(n,u,uy,waveno)
call diff2y(n,u,uyy,waveno)
call diff3y(n,u,uyyy,waveno)
call diffx(n,uy,uxy,waveno)
call diff2y(n,ux,uxyy,waveno)
call diff2x(n,uy,uxxy,waveno)
call vectsq(n,ux,ux2,corners,rows,matplan)
call vectsq(n,uy,uy2,corners,rows,matplan)
call vecttomat(n,ux,luxmat,corners,rows,matplan)
call vecttomat(n,uy,luymat,corners,rows,matplan)
call vecttomat(n,ux2,lux2mat,corners,rows,matplan)
call vecttomat(n,uy2,luy2mat,corners,rows,matplan)
call vecttomat(n,uxx,luxxmat,corners,rows,matplan)
call vecttomat(n,uxy,luxymat,corners,rows,matplan)
call vecttomat(n,uyy,luyymat,corners,rows,matplan)
call vecttomat(n,uxxx,luxxxmat,corners,rows,matplan)
call vecttomat(n,uxxy,luxxymat,corners,rows,matplan)
```

```
call vecttomat(n,uxyy,luxyymat,corners,rows,matplan)
call vecttomat(n,uyyy,luyyymat,corners,rows,matplan)
call matvectmult(n,luxmat,uy,uxuy)
call matvectmult(n,luxmat,uxx,uxuxx)
call matvectmult(n,luymat,uxx,uyuxx)
call matvectmult(n,luxmat,uyy,uxuyy)
call matvectmult(n,luymat,uyy,uyuyy)
call matvectmult(n,luxmat,uxy,uxuxy)
call matvectmult(n,luymat,uxy,uyuxy)
call vecttomat(n,uxuy,luxuymat,corners,rows,matplan)
call vecttomat(n,uxuxx,luxuxxmat,corners,rows,matplan)
call vecttomat(n,uyuxx,luyuxxmat,corners,rows,matplan)
call vecttomat(n,uxuyy,luxuyymat,corners,rows,matplan)
call vecttomat(n,uyuyy,luyuyymat,corners,rows,matplan)
call vecttomat(n,uxuxy,luxuxymat,corners,rows,matplan)
call vecttomat(n,uyuxy,luyuxymat,corners,rows,matplan)
do i= 1,3*n*(n+1)+1
  dx(i) = (0.d0,1.d0) *q* waveno(i,2)
  dxx(i) = -1.d0 *q*q* ((waveno(i,2))**2)
  dxxx(i) = (0.d0,-1.d0)*q*q*q* ((waveno(i,2))**3)
  dxxxx(i) = q*q*q*q*((waveno(i,2))**4)
  dy(i) = (0.d0,1.d0) *q* waveno(i,3)
  dyy(i) = -1.d0 *q*q* ((waveno(i,3))**2)
  dyyy(i) = (0.d0,-1.d0)*q*q*q* ((waveno(i,3))**3)
  dyyyy(i) = q*q*q*q*((waveno(i,3))**4)
  dxy(i) = -q*q* waveno(i,2) * waveno(i,3)
  dxxyy(i) = q*q*q*q*((waveno(i,2))**2)*((waveno(i,3))**2)
  dxyy(i) =(0.d0,-1.d0)*q*q*q* waveno(i,2) *((waveno(i,3))**2)
  dxxy(i) =(0.d0,-1.d0)*q*q*q*((waveno(i,2))**2)* waveno(i,3)
end do
```

```
call multderiv(n,luxxmat,dxx,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lkappanu(i,j)=lkappanu(i,j)+2.d0*ltemp(i,j)
  end do
end do
call multderiv(n,luyymat,dyy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lkappanu(i,j)=lkappanu(i,j)+2.d0*ltemp(i,j)
  end do
end do
call multderiv(n,luxmat,dxxx,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
  end do
end do
call multderiv(n,luxxxmat,dx,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
  end do
end do
call multderiv(n,luxmat,dxyy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
  end do
end do
```

```
call multderiv(n,luxyymat,dx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luymat,dxxy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luxxymat,dy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luymat,dyyy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luyyymat,dy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
   end do
end do
```

```
call multderiv(n,luxymat,dxy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lkappa(i,j)=lkappa(i,j)+2.d0*ltemp(i,j)
   end do
end do
call multderiv(n,luxxmat,dyy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lnu(i,j)=lnu(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luyymat,dxx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lnu(i,j)=lnu(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luymat,dxxx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lgamma(i,j)=lgamma(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luxxxmat,dy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lgamma(i,j)=lgamma(i,j)+ltemp(i,j)
   end do
end do
```

```
call multderiv(n,luymat,dxyy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lgamma(i,j)=lgamma(i,j)+ltemp(i,j)
  end do
end do
call multderiv(n,luxyymat,dy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lgamma(i,j)=lgamma(i,j)+ltemp(i,j)
  end do
end do
call multderiv(n,luxmat,dxxy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lgamma(i,j)=lgamma(i,j)-ltemp(i,j)
  end do
end do
call multderiv(n,luxxymat,dx,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lgamma(i,j)=lgamma(i,j)-ltemp(i,j)
  end do
end do
call multderiv(n,luxmat,dyyy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lgamma(i,j)=lgamma(i,j)-ltemp(i,j)
  end do
end do
```

```
call multderiv(n,luyyymat,dx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lgamma(i,j)=lgamma(i,j)-ltemp(i,j)
   end do
end do
call multderiv(n,luxuymat,dxx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luxuxxmat,dy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luyuxxmat,dx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luy2mat,dxy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)+ltemp(i,j)
   end do
end do
```

```
call multderiv(n,luyuxymat,dy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)+2.d0*ltemp(i,j)
   end do
end do
call multderiv(n,lux2mat,dxy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)-ltemp(i,j)
   end do
end do
call multderiv(n,luxuxymat,dx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)-2.d0*ltemp(i,j)
   end do
end do
call multderiv(n,luxuymat,dyy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)-ltemp(i,j)
   end do
end do
call multderiv(n,luxuyymat,dy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)-ltemp(i,j)
   end do
end do
```

```
call multderiv(n,luyuyymat,dx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)-ltemp(i,j)
   end do
end do
call multderiv(n,luxuymat,dxy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg3(i,j)=lg3(i,j)+4.d0*ltemp(i,j)
   end do
end do
call multderiv(n,luxuxymat,dy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg3(i,j)=lg3(i,j)+4.d0*ltemp(i,j)
   end do
end do
call multderiv(n,luyuxymat,dx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg3(i,j)=lg3(i,j)+4.d0*ltemp(i,j)
   end do
end do
call multderiv(n,lux2mat,dxx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg3(i,j)=lg3(i,j)+3.d0*ltemp(i,j)
   end do
end do
```

```
call multderiv(n,luxuxxmat,dx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg3(i,j)=lg3(i,j)+6.d0*ltemp(i,j)
   end do
end do
call multderiv(n,luy2mat,dyy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg3(i,j)=lg3(i,j)+3.d0*ltemp(i,j)
   end do
end do
call multderiv(n,luyuyymat,dy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg3(i,j)=lg3(i,j)+6.d0*ltemp(i,j)
   end do
end do
call multderiv(n,lux2mat,dyy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg3(i,j)=lg3(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luxuyymat,dx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg3(i,j)=lg3(i,j)+2.d0*ltemp(i,j)
   end do
end do
```

```
      call multderiv(n,luy2mat,dxx,ltemp)
      do i=1,3*n*(n+1)+1
         do j=1,3*n*(n+1)+1
            lg3(i,j)=lg3(i,j)+ltemp(i,j)
         end do
      end do
      call multderiv(n,luyuxxmat,dy,ltemp)
      do i=1,3*n*(n+1)+1
         do j=1,3*n*(n+1)+1
            lg3(i,j)=lg3(i,j)+2.d0*ltemp(i,j)
         end do
      end do
      do i = 1,3*n*(n+1)+1
         if (method.eq.1) lhs(i,i) = 0.000001d0+beta
     *                   + mu*(dyy(i)+dxx(i))
     *                   + lambda*(dxxxx(i)+dyyyy(i)+2.d0*dxxyy(i))
         if (method.eq.2) lhs(i,i) = 1.d0/dt+theta*(beta
     *                   + mu*(dyy(i)+dxx(i))
     *                   + lambda*(dxxxx(i)+dyyyy(i)+2.d0*dxxyy(i)))
      end do
      do j = 1,3*n*(n+1)+1
         do i = 1,3*n*(n+1)+1
            if (method.eq.1) lhs(i,j) = lhs(i,j)
     *                   +(2.d0*kappa+nu)*lkappanu(i,j)
     *                   + 4.d0*kappa*lkappa(i,j)+2.d0*nu*lnu(i,j)
     *                   + gamma*lgamma(i,j)
     *                   + 2.d0*g2*lg2(i,j)
     *                   + g3*lg3(i,j)
            if (method.eq.2) lhs(i,j) = lhs(i,j)
     *                   +theta*(
```

```
*                    (2.d0*kappa+nu)*lkappanu(i,j)
*                    + 4.d0*kappa*lkappa(i,j)+2.d0*nu*lnu(i,j)
*                    + gamma*lgamma(i,j)
*                    + 2.d0*g2*lg2(i,j)
*                    + g3*lg3(i,j))
        end do
      end do
      return
      end
c-----------------------------------------------
c This loops through variations in perturbation angle theta,
c writing out the real and imaginary parts of the eigenvalue
c with largest real part, which is given by invit.
      subroutine perturbloop(n,u,corners,rows,waveno,matplan)
      implicit none
      integer n
      complex*16 u(3*n*(n+1)+1)
      complex*16 lhs(3*n*(n+1)+1,3*n*(n+1)+1)
      real*8 lhsreal(3*n*(n+1)+1,3*n*(n+1)+1)
      integer info,i,j,k
      real*8 waveno(3*n*(n+1)+1,3)
      integer matplan(3*n*(n+1)+1,3*n*(n+1)+1)
      integer corners(2*n+1,2*n+1,3),rows(2*n+1,2)
      real*8 px,py,maxeval,mineval
      real*8 work(4*3*n*(n+1)+1)
      real*8 vr(1,3*n*(n+1)+1),vl(1,3*n*(n+1)+1)
      real*8 wr(3*n*(n+1)+1),wi(3*n*(n+1)+1)
      real*8 fv1(3*n*(n+1)+1)
      complex*16 evector(3*n*(n+1)+1),evalue
      integer iv1(3*n*(n+1)+1)
```

```
      real*8 z(3*n*(n+1)+1,3*n*(n+1)+1),esize,ran0
      real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      open(71,file='evalues.dat')
c set up first guess to eigenvector, and normalize it.
      do k = 1,3*n*(n+1)
        evector(k) = dcmplx((ran0(mod(k,10))-.5d0),
     *                  (ran0(10-mod(k,10))-.5d0))
      end do
      evector(3*n*(n+1)+1)=1.d0
      esize = 0.d0
      do k = 1,3*n*(n+1)+1
        esize=esize+evector(k)*evector(k)
      end do
      esize=sqrt(esize)
      do k = 1,3*n*(n+1)+1
        evector(k) = evector(k)/esize
      end do
c play around with the initial value of evalue
      evalue = (0.0d0,0.0d0)
      do i=0,60
      py = i/60.d0/sqrt(3.d0)/q
      px = sqrt(q*q-py*py)-q
      call perturblhs(n,u,lhs,px,py,corners,rows,waveno,matplan)
      do j = 1, 3*n*(n+1)+1
        do k = 1, 3*n*(n+1)+1
          write(78,7103) j,k,dreal(lhs(j,k))
        end do
      end do
      call invit(3*n*(n+1)+1,evalue,evector,lhs)
```

```
      write(71,7102) datan(py/q)*180.d0/3.14159265359, dreal(evalue)
*                          , dimag(evalue)
      call flush1(71)
      evalue = evalue + (0.15d0,-0.02d0)
      end do
      close(71)
7100 format(f8.4)
7101 format(f18.12,$)
7102 format(f8.2,2f18.8)
7103 format(2i,f18.8)
      return
      end
c————————————————————————————————————
c Makes matrix A of matrix equation A*x = b,
c this time not for iteration/timestepping, but for finding
c the eigenvales.
      subroutine perturblhs (n,u,lhs,px,py,corners,rows,waveno,matplan)
      implicit none
      integer n
      complex*16 u(3*n*(n+1)+1),ux(3*n*(n+1)+1),uxx(3*n*(n+1)+1)
      complex*16 uxxx(3*n*(n+1)+1),lhs(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 u2(3*n*(n+1)+1)
      complex*16 uy(3*n*(n+1)+1),uyy(3*n*(n+1)+1),uyyy(3*n*(n+1)+1)
      complex*16 ux2(3*n*(n+1)+1),uy2(3*n*(n+1)+1),uxy(3*n*(n+1)+1)
      complex*16 uxuy(3*n*(n+1)+1)
      complex*16 uxxy(3*n*(n+1)+1),uxyy(3*n*(n+1)+1)
      complex*16 uxuxx(3*n*(n+1)+1),uxuxy(3*n*(n+1)+1)
      complex*16 uxuyy(3*n*(n+1)+1)
      complex*16 uyuxx(3*n*(n+1)+1),uyuxy(3*n*(n+1)+1)
      complex*16 uyuyy(3*n*(n+1)+1)
```

```
      complex*16 luxmat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxuymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lux2mat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luy2mat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxxmat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luyymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxxxmat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxxymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxyymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luyyymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxuxxmat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luxuxymat(3*n*(n+1)+1,3*n*(n+1)+1),
     *           luxuyymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luyuxxmat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luyuxymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 luyuyymat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 ltemp(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lkappanu(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lkappa(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lnu(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lgamma(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lg2(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lg3(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lu2mat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 dx(3*n*(n+1)+1),dxx(3*n*(n+1)+1)
      complex*16 dxxx(3*n*(n+1)+1),dxxxx(3*n*(n+1)+1)
      complex*16 dy(3*n*(n+1)+1),dyy(3*n*(n+1)+1),dyyy(3*n*(n+1)+1)
      complex*16 dyyyy(3*n*(n+1)+1)
```

```
complex*16 dxy(3*n*(n+1)+1),dxxyy(3*n*(n+1)+1)
complex*16 dxyy(3*n*(n+1)+1),dxxy(3*n*(n+1)+1)
complex*16 lumat(3*n*(n+1)+1,3*n*(n+1)+1)
integer i,j,method
integer corners(2*n+1,2*n+1,3),rows(2*n+1,2)
real*8 waveno(3*n*(n+1)+1,3),px,py
integer matplan(3*n*(n+1)+1,3*n*(n+1)+1)
real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
real*8 time,dt,theta,r
integer nsteps
common /runpar/ time,dt,theta,r,nsteps
do j = 1,3*n*(n+1)+1
  do i = 1,3*n*(n+1)+1
    lhs(i,j) = 0.d0
    lkappanu(i,j) = 0.d0
    lkappa(i,j) = 0.d0
    lnu(i,j) = 0.d0
    lgamma(i,j) = 0.d0
    lg2(i,j) = 0.d0
    lg3(i,j) = 0.d0
    ltemp(i,j) = 0.d0
  end do
end do
call vectsq(n,u,u2,corners,rows,matplan)
call vecttomat(n,u2,lu2mat,corners,rows,matplan)
call diffx(n,u,ux,waveno)
call diff2x(n,u,uxx,waveno)
call diff3x(n,u,uxxx,waveno)
call diffy(n,u,uy,waveno)
```

```
call diff2y(n,u,uyy,waveno)
call diff3y(n,u,uyyy,waveno)
call diffx(n,uy,uxy,waveno)
call diff2y(n,ux,uxyy,waveno)
call diff2x(n,uy,uxxy,waveno)
call vectsq(n,ux,ux2,corners,rows,matplan)
call vectsq(n,uy,uy2,corners,rows,matplan)
call vecttomat(n,ux,luxmat,corners,rows,matplan)
call vecttomat(n,uy,luymat,corners,rows,matplan)
call vecttomat(n,ux2,lux2mat,corners,rows,matplan)
call vecttomat(n,uy2,luy2mat,corners,rows,matplan)
call vecttomat(n,uxx,luxxmat,corners,rows,matplan)
call vecttomat(n,uxy,luxymat,corners,rows,matplan)
call vecttomat(n,uyy,luyymat,corners,rows,matplan)
call vecttomat(n,uxxx,luxxxmat,corners,rows,matplan)
call vecttomat(n,uxxy,luxxymat,corners,rows,matplan)
call vecttomat(n,uxyy,luxyymat,corners,rows,matplan)
call vecttomat(n,uyyy,luyyymat,corners,rows,matplan)
call matvectmult(n,luxmat,uy,uxuy)
call matvectmult(n,luxmat,uxx,uxuxx)
call matvectmult(n,luymat,uxx,uyuxx)
call matvectmult(n,luxmat,uyy,uxuyy)
call matvectmult(n,luymat,uyy,uyuyy)
call matvectmult(n,luxmat,uxy,uxuxy)
call matvectmult(n,luymat,uxy,uyuxy)
call vecttomat(n,uxuy,luxuymat,corners,rows,matplan)
call vecttomat(n,uxuxx,luxuxxmat,corners,rows,matplan)
call vecttomat(n,uyuxx,luyuxxmat,corners,rows,matplan)
call vecttomat(n,uxuyy,luxuyymat,corners,rows,matplan)
call vecttomat(n,uyuyy,luyuyymat,corners,rows,matplan)
```

```
call vecttomat(n,uxuxy,luxuxymat,corners,rows,matplan)
call vecttomat(n,uyuxy,luyuxymat,corners,rows,matplan)
do i= 1,3*n*(n+1)+1
   dx(i) = (0.d0,1.d0) *q* (waveno(i,2)+px)
   dxx(i) = -1.d0 *q*q* ((waveno(i,2)+px)**2)
   dxxx(i) = (0.d0,-1.d0)*q*q*q* ((waveno(i,2)+px)**3)
   dxxxx(i) = q*q*q*q*((waveno(i,2)+px)**4)
   dy(i) = (0.d0,1.d0) *q* (waveno(i,3)+py)
   dyy(i) = -1.d0 *q*q* ((waveno(i,3)+py)**2)
   dyyy(i) = (0.d0,-1.d0)*q*q*q* ((waveno(i,3)+py)**3)
   dyyyy(i) = q*q*q*q*((waveno(i,3)+py)**4)
   dxy(i) = -(q*waveno(i,2)+px) * (q*waveno(i,3)+py)
   dxxyy(i)= ((q*waveno(i,2)+px)**2)*((q*waveno(i,3)+py)**2)
   dxyy(i) =(0.d0,-1.d0)* (q*waveno(i,2)+px)*((q*waveno(i,3)+py)**2)
   dxxy(i) =(0.d0,-1.d0)*((q*waveno(i,2)+px)**2)*(q*waveno(i,3)+py)
end do
call multderiv(n,luxxmat,dxx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+2.d0*ltemp(i,j)
   end do
end do
call multderiv(n,luyymat,dyy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+2.d0*ltemp(i,j)
   end do
end do
call multderiv(n,luxmat,dxxx,ltemp)
do i=1,3*n*(n+1)+1
```

```
    do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
    end do
end do
call multderiv(n,luxxxmat,dx,ltemp)
do i=1,3*n*(n+1)+1
    do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
    end do
end do
call multderiv(n,luxmat,dxyy,ltemp)
do i=1,3*n*(n+1)+1
    do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
    end do
end do
call multderiv(n,luxyymat,dx,ltemp)
do i=1,3*n*(n+1)+1
    do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
    end do
end do
call multderiv(n,luymat,dxxy,ltemp)
do i=1,3*n*(n+1)+1
    do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
    end do
end do
call multderiv(n,luxxymat,dy,ltemp)
do i=1,3*n*(n+1)+1
```

```
    do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
    end do
end do
call multderiv(n,luymat,dyyy,ltemp)
do i=1,3*n*(n+1)+1
    do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
    end do
end do
call multderiv(n,luyyymat,dy,ltemp)
do i=1,3*n*(n+1)+1
    do j=1,3*n*(n+1)+1
      lkappanu(i,j)=lkappanu(i,j)+ltemp(i,j)
    end do
end do
call multderiv(n,luxymat,dxy,ltemp)
do i=1,3*n*(n+1)+1
    do j=1,3*n*(n+1)+1
      lkappa(i,j)=lkappa(i,j)+2.d0*ltemp(i,j)
    end do
end do
call multderiv(n,luxxmat,dyy,ltemp)
do i=1,3*n*(n+1)+1
    do j=1,3*n*(n+1)+1
      lnu(i,j)=lnu(i,j)+ltemp(i,j)
    end do
end do
call multderiv(n,luyymat,dxx,ltemp)
do i=1,3*n*(n+1)+1
```

```
  do j=1,3*n*(n+1)+1
    lnu(i,j)=lnu(i,j)+ltemp(i,j)
  end do
end do
call multderiv(n,luymat,dxxx,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lgamma(i,j)=lgamma(i,j)+ltemp(i,j)
  end do
end do
call multderiv(n,luxxxmat,dy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lgamma(i,j)=lgamma(i,j)+ltemp(i,j)
  end do
end do
call multderiv(n,luymat,dxyy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lgamma(i,j)=lgamma(i,j)+ltemp(i,j)
  end do
end do
call multderiv(n,luxyymat,dy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lgamma(i,j)=lgamma(i,j)+ltemp(i,j)
  end do
end do
call multderiv(n,luxmat,dxxy,ltemp)
do i=1,3*n*(n+1)+1
```

```
    do j=1,3*n*(n+1)+1
      lgamma(i,j)=lgamma(i,j)-ltemp(i,j)
    end do
  end do
  call multderiv(n,luxxymat,dx,ltemp)
  do i=1,3*n*(n+1)+1
    do j=1,3*n*(n+1)+1
      lgamma(i,j)=lgamma(i,j)-ltemp(i,j)
    end do
  end do
  call multderiv(n,luxmat,dyyy,ltemp)
  do i=1,3*n*(n+1)+1
    do j=1,3*n*(n+1)+1
      lgamma(i,j)=lgamma(i,j)-ltemp(i,j)
    end do
  end do
  open(76,file='luxdyyy.dat')
  do i=1,n*n
    do j=1,n*n-1
      write(76,2101) dreal(ltemp(i,j))
    end do
    write(76,2102) dreal(ltemp(i,n*n))
    write(76,*)
  end do
  close(76)
  call multderiv(n,luyyymat,dx,ltemp)
  do i=1,3*n*(n+1)+1
    do j=1,3*n*(n+1)+1
      lgamma(i,j)=lgamma(i,j)-ltemp(i,j)
    end do
```

```
end do
call multderiv(n,luxuymat,dxx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luxuxxmat,dy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luyuxxmat,dx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luy2mat,dxy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)+ltemp(i,j)
   end do
end do
call multderiv(n,luyuxymat,dy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg2(i,j)=lg2(i,j)+2.d0*ltemp(i,j)
   end do
```

```
end do
call multderiv(n,lux2mat,dxy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lg2(i,j)=lg2(i,j)-ltemp(i,j)
  end do
end do
call multderiv(n,luxuxymat,dx,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lg2(i,j)=lg2(i,j)-2.d0*ltemp(i,j)
  end do
end do
call multderiv(n,luxuymat,dyy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lg2(i,j)=lg2(i,j)-ltemp(i,j)
  end do
end do
call multderiv(n,luxuyymat,dy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lg2(i,j)=lg2(i,j)-ltemp(i,j)
  end do
end do
call multderiv(n,luyuyymat,dx,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lg2(i,j)=lg2(i,j)-ltemp(i,j)
  end do
```

```
end do
call multderiv(n,luxuymat,dxy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg3(i,j)=lg3(i,j)+4.d0*ltemp(i,j)
   end do
end do
call multderiv(n,luxuxymat,dy,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg3(i,j)=lg3(i,j)+4.d0*ltemp(i,j)
   end do
end do
call multderiv(n,luyuxymat,dx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg3(i,j)=lg3(i,j)+4.d0*ltemp(i,j)
   end do
end do
call multderiv(n,lux2mat,dxx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg3(i,j)=lg3(i,j)+3.d0*ltemp(i,j)
   end do
end do
call multderiv(n,luxuxxmat,dx,ltemp)
do i=1,3*n*(n+1)+1
   do j=1,3*n*(n+1)+1
      lg3(i,j)=lg3(i,j)+6.d0*ltemp(i,j)
   end do
```

```
end do
call multderiv(n,luy2mat,dyy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lg3(i,j)=lg3(i,j)+3.d0*ltemp(i,j)
  end do
end do
call multderiv(n,luyuyymat,dy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lg3(i,j)=lg3(i,j)+6.d0*ltemp(i,j)
  end do
end do
call multderiv(n,lux2mat,dyy,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lg3(i,j)=lg3(i,j)+ltemp(i,j)
  end do
end do
call multderiv(n,luxuyymat,dx,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lg3(i,j)=lg3(i,j)+2.d0*ltemp(i,j)
  end do
end do
call multderiv(n,luy2mat,dxx,ltemp)
do i=1,3*n*(n+1)+1
  do j=1,3*n*(n+1)+1
    lg3(i,j)=lg3(i,j)+ltemp(i,j)
  end do
```

```
      end do
      call multderiv(n,luyuxxmat,dy,ltemp)
      do i=1,3*n*(n+1)+1
        do j=1,3*n*(n+1)+1
          lg3(i,j)=lg3(i,j)+2.d0*ltemp(i,j)
        end do
      end do
      do i = 1,3*n*(n+1)+1
        lhs(i,i) = beta
*                 + mu*(dyy(i)+dxx(i))
*                 + lambda*(dxxxx(i)+dyyyy(i)+2.d0*dxxyy(i))
      end do
      do j = 1,3*n*(n+1)+1
        do i = 1,3*n*(n+1)+1
          lhs(i,j) = lhs(i,j)
*                 +(2.d0*kappa+nu)*lkappanu(i,j)
*                 + 4.d0*kappa*lkappa(i,j)+2.d0*nu*lnu(i,j)
*                 + gamma*lgamma(i,j)
*                 + 2.d0*g2*lg2(i,j)
*                 + g3*lg3(i,j)
        end do
      end do
      do j = 1,3*n*(n+1)+1
        do i = 1,3*n*(n+1)+1
          lhs(i,j) = -lhs(i,j)
        end do
      end do
      return
      end
c---------------------------------------------------------------------
```

```
c In computing matrix A, this is used with the nonlinear terms,
c where we have a multiplication of a vector at time t and some
c derivative of the vector at the new time t+dt.
      subroutine multderiv(n,lmat,d,lout)
      implicit none
      integer i,j,n
      complex*16 d(3*n*(n+1)+1),lmat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 lout(3*n*(n+1)+1,3*n*(n+1)+1)
      do i=1,3*n*(n+1)+1
        do j=1,3*n*(n+1)+1
          lout(i,j) = 0.d0
        end do
      end do
      do i=1,3*n*(n+1)+1
        do j=1,3*n*(n+1)+1
          lout(i,j) = lmat(i,j) * d(j)
        end do
      end do
      return
      end
c————————————————————————————————————————
c These next few subroutines are just differentiation in Fourier space.
c Differentiation of Fourier arrays.
      subroutine diffx(n,u,ux,waveno)
      implicit none
      integer n
      complex*16 u(3*n*(n+1)+1),ux(3*n*(n+1)+1)
      integer i
      real*8 waveno(3*n*(n+1)+1,3)
      real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
```

```
      common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      do i= 1,3*n*(n+1)+1
         ux(i) = (0.d0,1.d0)*q*waveno(i,2)*u(i)
      end do
      return
      end
c————————————————————————————————————
c Differentiation of Fourier arrays.
      subroutine diff2x(n,u,uxx,waveno)
      implicit none
      integer n
      complex*16 u(3*n*(n+1)+1),uxx(3*n*(n+1)+1)
      integer i
      real*8 waveno(3*n*(n+1)+1,3)
      real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      do i= 1,3*n*(n+1)+1
         uxx(i) = -1.d0*q*q*((waveno(i,2))**2)*u(i)
      end do
      return
      end
c————————————————————————————————————
c Differentiation of Fourier arrays.
      subroutine diff3x(n,u,uxxx,waveno)
      implicit none
      integer n
      complex*16 u(3*n*(n+1)+1),uxxx(3*n*(n+1)+1)
      integer i
      real*8 waveno(3*n*(n+1)+1,3)
      real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
```

```
      common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      do i= 1,3*n*(n+1)+1
        uxxx(i) = (0.d0,-1.d0)*q*q*q*((waveno(i,2))**3)*u(i)
      end do
      return
      end
c————————————————————————————————————
c Differentiation of Fourier arrays.
      subroutine diff4x(n,u,uxxxx,waveno)
      implicit none
      integer n
      complex*16 u(3*n*(n+1)+1),uxxxx(3*n*(n+1)+1)
      integer i
      real*8 waveno(3*n*(n+1)+1,3)
      real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      do i= 1,3*n*(n+1)+1
        uxxxx(i) = q*q*q*q*((waveno(i,2))**4)*u(i)
      end do
      return
      end
c————————————————————————————————————
c Differentiation of Fourier arrays.
      subroutine diffy(n,u,uy,waveno)
      implicit none
      integer n
      complex*16 u(3*n*(n+1)+1),uy(3*n*(n+1)+1)
      integer i
      real*8 waveno(3*n*(n+1)+1,3)
      real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
```

```
      common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      do i= 1,3*n*(n+1)+1
         uy(i) = (0.d0,1.d0)*q*waveno(i,3)*u(i)
      end do
      return
      end
c————————————————————————————————————
c Differentiation of Fourier arrays.
      subroutine diff2y(n,u,uyy,waveno)
      implicit none
      integer n
      complex*16 u(3*n*(n+1)+1),uyy(3*n*(n+1)+1)
      integer i
      real*8 waveno(3*n*(n+1)+1,3)
      real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      do i= 1,3*n*(n+1)+1
         uyy(i) = -1.d0*q*q*((waveno(i,3))**2)*u(i)
      end do
      return
      end
c————————————————————————————————————
c Differentiation of Fourier arrays.
      subroutine diff3y(n,u,uyyy,waveno)
      implicit none
      integer n
      complex*16 u(3*n*(n+1)+1),uyyy(3*n*(n+1)+1)
      integer i
      real*8 waveno(3*n*(n+1)+1,3)
      real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
```

```
      common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      do i= 1,3*n*(n+1)+1
        uyyy(i) = (0.d0,-1.d0)*q*q*q*((waveno(i,3))**3)*u(i)
      end do
      return
      end
c------------------------------------------------------------
c Differentiation of Fourier arrays.
      subroutine diff4y(n,u,uyyyy,waveno)
      implicit none
      integer n
      complex*16 u(3*n*(n+1)+1),uyyyy(3*n*(n+1)+1)
      integer i
      real*8 waveno(3*n*(n+1)+1,3)
      real*8 beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      common /params/ beta,mu,lambda,kappa,nu,gamma,g2,g3,q
      do i= 1,3*n*(n+1)+1
        uyyyy(i) = q*q*q*q*((waveno(i,3))**4)*u(i)
      end do
      return
      end
c------------------------------------------------------------
c Does matrix multiplication mat x vectin = vectout.
      subroutine matmult(n,mat,vectin,vectout)
      implicit none
      integer n
      complex*16 mat(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 vectin(3*n*(n+1)+1),vectout(3*n*(n+1)+1)
      integer i,j
      do i = 1,3*n*(n+1)+1
```

```
      vectout(i) = 0.d0
    end do
    do j = 1,3*n*(n+1)+1
      do i = 1,3*n*(n+1)+1
        vectout(j)=vectout(j)+mat(j,i)*vectin(i)
      end do
    end do
    return
    end
c———————————————————————————————
c Given u, gives u**2 (u squared).
    subroutine vectsq(n,u,u2,corners,rows,matplan)
    implicit none
    integer n
    complex*16 u(3*n*(n+1)+1),u2(3*n*(n+1)+1)
    complex*16 umat2(3*n*(n+1)+1,3*n*(n+1)+1)
    integer matplan(3*n*(n+1)+1,3*n*(n+1)+1)
    integer corners(2*n+1,2*n+1,3),rows(2*n+1,2)
    call vecttomat(n,u,umat2,corners,rows,matplan)
    call matmult(n,umat2,u,u2)
    return
    end
c———————————————————————————————
c Given u, gives u**3 (cubed).
    subroutine vectcub(n,u,u3,corners,rows,matplan)
    implicit none
    integer n
    complex*16 u(3*n*(n+1)+1),u2(3*n*(n+1)+1)
    complex*16 u3(3*n*(n+1)+1),umat(3*n*(n+1)+1,3*n*(n+1)+1)
    integer corners(2*n+1,2*n+1,3),rows(2*n+1,2)
```

```
      integer matplan(3*n*(n+1)+1,3*n*(n+1)+1)
      call vectsq(n,u,u2,corners,rows,matplan)
      call vecttomat(n,u,umat,corners,rows,matplan)
      call matmult(n,umat,u2,u3)
      return
      end
c————————————————————————————————————————
c Along with subroutine ic, this is done in the beginning.
c Four arrays are set up.
c Waveno is an array of dimensions ( 3*n*(n+1)+1, 3 ),
c with each row being of the form ( index, xwavenum, ywavenum).
c It is just the wavenumbers to be used for reference.
c Rows is an array with the indices of the leftmost and the rightmost
c element of the solution vector in each row of the hexagonal grid
c in Fourier space.
c Corners and matplan are used to make the matrices corresponding
c to a multiplication in Fourier vector space of the solutions.
c Corners is used to create matplan. It contains the indices of the
c top left element of the smaller matrices that make up the
c multiplication matrix. They are used by along with the makeblock
c and fillblock subroutines to fill in matplan (done in subroutine
c vecttomat).
c Matplan is (3*n*(n+1)+1,3*n*(n+1)+1), and contains the indices
c of the vector u element to be put in place to make the multiplication
c matrix.
      subroutine wavenum(n,waveno,corners,rows,matplan)
      implicit none
      integer n
      integer row,rowtotal,indexgen,i,j
      real*8 rowmin,x,y
```

```
real*8 waveno(3*n*(n+1)+1,3)
integer matplan(3*n*(n+1)+1,3*n*(n+1)+1)
integer corners(2*n+1,2*n+1,3),rows(2*n+1,2)
integer sames,beginning,ending,diaglength
rowtotal = n + 1
indexgen = 1
open(21,file='wavenumbers.dat')
open(22,file='corners.dat')
open(23,file='rows.dat')
y = (n+1)*sqrt(3.d0)/2.d0 ! this loop creates waveno and rows
indexgen = 1
do row = 1, 2*n+1
   y = y - sqrt(3.d0)/2.d0
   if (row.le.(n+1)) then
      rowmin = (-n-row+1)/2.d0
      rowtotal = n + row
   elseif (row.gt.(n+1)) then
      rowmin = -n + (row-n-1)/2.d0
      rowtotal = -row + 3*n + 2
   end if
   rows(row,1) = indexgen ! setting up rows matrix for vecttomat
   do i = 1, rowtotal
      x = rowmin +(i-1)
      waveno(indexgen,1) = indexgen
      waveno(indexgen,2) = x
      waveno(indexgen,3) = y
      write(21,2100) waveno(indexgen,1),waveno(indexgen,2),
*                          waveno(indexgen,3)
      indexgen = indexgen + 1
   end do ! end i loop
```

```
            rows(row,2) = indexgen - 1 !setting up rows matrix for vecttomat
        end do ! end row loop
2100 format(3f10.5)
c These next three loops make corners.
        do i = 1,2*n+1
            do j = 1,2*n+1
                corners(i,j,1) = 0
                corners(i,j,2) = 2*n+1-iabs(i-n-1)
                corners(i,j,3) = 2*n+1-iabs(j-n-1)
            end do
        end do
        do i = 1, 2*n+1
            corners(i,i,1) = (3*n*(n+1)+2)/2
        end do
c below main diag
        sames = n + 1
        beginning = (3*n*(n+1)+2)/2
        do j = 2,n+1
            sames = sames -1
            beginning = beginning + (2*n-j+2)
            ending = beginning +j-1
            diaglength = 2*n+2-j
            do i = 1,diaglength
                if(i.le.sames) then
                    corners(j+i-1,i,1) = beginning
                elseif(i.gt.sames.and.i.lt.(diaglength-sames+1)) then
                    corners(j+i-1,i,1) = beginning +i -sames
                elseif(i.ge.(diaglength-sames+1)) then
                    corners(j+i-1,i,1) = ending
                end if
```

```
      end do
    end do
c above main diag
    sames = n + 1
    beginning = (3*n*(n+1)+2)/2
    do j = 2,n+1
      sames = sames -1
      beginning = beginning - (2*n-j+2)
      ending = beginning -j+1
      diaglength = 2*n+2-j
      do i = 1,diaglength
        if(i.le.sames) then
          corners(i,j+i-1,1) = beginning
        elseif(i.gt.sames.and.i.lt.(diaglength-sames+1)) then
          corners(i,j+i-1,1) = beginning -i +sames
        elseif(i.ge.(diaglength-sames+1)) then
          corners(i,j+i-1,1) = ending
        end if
      end do
    end do ! done making corners
    do i = 1,2*n+1
      do j = 1,2*n+1
        write(22,*) i,j,corners(i,j,1),corners(i,j,2),corners(i,j,3)
      end do
      write(23,*) i,rows(i,1),rows(i,2)
    end do
    call vectmatinit (n,matplan,corners,rows) ! make matplan
    do i = 1,3*n*(n+1)+1
      do j = 1,3*n*(n+1)
        write(41,4101) matplan(i,j)
```

```
c write(41,2101) dreal(matplan(i,j))
          end do
          write(41,4102) matplan(i,j)
c write(41,2102) dreal(matplan(i,j))
        end do
        write(41,*)
2101 format(f5.2,$)
2102 format(f5.2)
4101 format(i8,$)
4102 format(i8)
        close(21)
        close(22)
        close(23)
        return
        end
c———————————————————————————————————————
c This gives the index u(index) of a given wavenumber for number
c of harmonics n.
        subroutine wavetoindex(n,x,y,index)
        implicit none
        integer n,index,row,i,total,rowtotal
        real*8 rowmin,xout,x,y
        row = nint(n-y*2.d0/sqrt(3.d0) + 1)
        if (row.le.n+1) then
          rowmin = (-n-row+1)/2.d0
        elseif (row.gt.(n+1)) then
          rowmin = -n+(row-n-1)/2.d0
        end if
        rowtotal = n
        total = 0
```

```
      do i=1,row-1
         if (i.le.(n+1)) then
            rowtotal = rowtotal + 1
         else
            rowtotal = rowtotal - 1
         end if
         total = total + rowtotal
      end do
      xout = rowmin
   20 if (xout.eq.x) then
         index = total + 1
      elseif (xout.lt.x) then
         total = total+1
         xout = xout + 1.0d0
         go to 20
      end if
      return
      end
c————————————————————————————————————
c Does matrix multiplication mat x mat = matout.
c All matrices are n by n.
      subroutine matmatmult(n,mat1,mat2,matout)
      implicit none
      integer n
      complex*16 mat1(3*n*(n+1)+1,3*n*(n+1)+1),
     * mat2(3*n*(n+1)+1,3*n*(n+1)+1),matout(3*n*(n+1)+1,3*n*(n+1)+1)
      integer i,j,k
      do j = 1,3*n*(n+1)+1
         do i = 1,3*n*(n+1)+1
            matout(i,j) = 0.d0
```

```
      end do
    end do
    do j = 1,3*n*(n+1)+1
      do i = 1,3*n*(n+1)+1
        do k = 1,3*n*(n+1)+1
          matout(i,j)=matout(i,j)+mat1(i,k)*mat2(k,j)
        end do
      end do
    end do
    return
    end
c------------------------------------------------
c Does matrix multiplication mat x vectin = vectout.
    subroutine matvectmult(n,mat,vectin,vectout)
    implicit none
    integer n
    complex*16 mat(3*n*(n+1)+1,3*n*(n+1)+1),vectin(3*n*(n+1)+1)
    complex*16 vectout(3*n*(n+1)+1)
    integer i,j
    do i = 1,3*n*(n+1)+1
      vectout(i) = 0.d0
    end do
    do j = 1,3*n*(n+1)+1
      do i = 1,3*n*(n+1)+1
        vectout(j)=vectout(j)+mat(j,i)*vectin(i)
      end do
    end do
    return
    end
c------------------------------------------------
```

```
c This makes the smaller xdim-by-ydim blocks that are then patched
c together to make the multiplication matrix.
c Each patch made here is made from the same row of the
c modes in Fourier space.
      subroutine makeblock (n,rows,xdim,ydim,corner,block)
      implicit none
      complex*16 u(3*n*(n+1)+1)
      integer block(xdim,ydim)
      integer n,xdim,ydim,diff,corner
      integer i,j,diagmax,diaglength,rowmin,rowmax
      integer rows(2*n+1,2),whichrow
      do i = 1,xdim
        do j = 1,ydim
          block(i,j) = 0
        end do
      end do
      do i = 1,2*n+1
        if(corner.ge.rows(i,1).and.corner.le.rows(i,2)) whichrow = i
      end do
      diagmax = min(xdim,ydim)
      rowmin = rows(whichrow,1)
      rowmax = rows(whichrow,2)
      do i = 1,diagmax
        block(i,i) = corner
      end do
c diags below main diag
      diff = idim(xdim,ydim)
      do j = corner+1,rowmax
        diaglength = diagmax-idim(j-corner+1,diff+1)
        do i = 1,diaglength
```

```
          block(j-corner+i,i)=j
        end do
      end do
c diags above main diag
      diff = idim(ydim,xdim)
      do j = corner-1,rowmin,-1
        diaglength = diagmax-idim(corner-j+1,diff+1)
        do i = 1,diaglength
          block(i,corner-j+i)=j
        end do
      end do
2101 format(f11.4,$)
2102 format(f11.4)
3101 format(i8,$)
3102 format(i8)
      return
      end
c————————————————————————————————————————
c This sets up matplan (here called lmatmult).
      subroutine vectmatinit(n,lmatmult,corners,rows)
      implicit none
      integer n
      integer lmatmult(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 block(2*n+1,2*n+1),tempblock(2*n+1,2*n+1)
      integer i,j
      integer corners(2*n+1,2*n+1,3),rows(2*n+1,2)
      do i = 1,3*n*(n+1)+1
        do j = 1,3*n*(n+1)+1
          lmatmult(i,j) = 0
        end do
```

```
      end do
      do i = 1,2*n+1
        do j = 1,2*n+1
          if(corners(i,j,1).ne.0) then
            call makeblock(n,rows,corners(i,j,2),corners(i,j,3),
*                          corners(i,j,1),block)
            call fillblock(n,corners(i,j,2),corners(i,j,3),
*                    lmatmult,block,
*                    rows(i,1),rows(j,1))
          end if
        end do
      end do
2101 format(f5.2,$)
2102 format(f5.2)
4101 format(i8,$)
4102 format(i8)
      return
      end
c————————————————————————————————————————
c This is the subroutine used at each timestep to set up the
c actual multiplication matrix using matplan, which was made
c with the initial conditions.
c The multiplication matrix is A of the equation A*x=b solved
c at each time step.
      subroutine vecttomat(n,u,lmatmult,corners,rows,matplan)
      implicit none
      integer n
      integer matplan(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 u(3*n*(n+1)+1),lmatmult(3*n*(n+1)+1,3*n*(n+1)+1)
      complex*16 ltemp(3*n*(n+1)+1,3*n*(n+1)+1)
```

```
      complex*16 block(2*n+1,2*n+1),tempblock(2*n+1,2*n+1)
      integer i,j
      integer corners(2*n+1,2*n+1,3),rows(2*n+1,2)
      do i = 1,3*n*(n+1)+1
        do j = 1,3*n*(n+1)+1
          lmatmult(i,j) = 0.d0
        end do
      end do
      do i = 1,3*n*(n+1)+1
        do j = 1,3*n*(n+1)+1
          if (matplan(i,j).gt.0) lmatmult(i,j) = u(matplan(i,j))
        end do
      end do
2101  format(f5.2,$)
2102  format(f5.2)
4101  format(i8,$)
4102  format(i8)
      return
      end
c-----------------------------------------------------------------------
c Used by vectinit to fill in blocks made by makeblock.
      subroutine fillblock(n,xdim,ydim,bigmatrix,block,posi,posj)
      implicit none
      integer n,xdim,ydim,posi,posj
      complex*16 u(3*n*(n+1)+1)
      integer bigmatrix(3*n*(n+1)+1,3*n*(n+1)+1),block(xdim,ydim)
      integer i,j
      do i = 1,xdim
        do j = 1,ydim
          bigmatrix(posi+i-1,posj+j-1) = block(i,j)
```

```
      end do
    end do
    return
    end
c————————————————————————————————————
c Returns eigenvector with largest real part.
    function maxeval(n,wr)
    implicit none
    real*8 wr(3*n*(n+1)+1)
    real*8 maxeval
    integer i,n
    maxeval = wr(1)
    do i = 2, 3*n*(n+1)+1
      if (wr(i).gt.maxeval) maxeval = wr(i)
    end do
    return
    end
c————————————————————————————————————
c Returns eigenvector with least real part.
    function mineval(n,wr)
    implicit none
    real*8 wr(3*n*(n+1)+1)
    real*8 mineval
    integer i,n
    mineval = wr(1)
    do i = 2, 3*n*(n+1)+1
      if (wr(i).lt.mineval) mineval = wr(i)
    end do
    return
    end
```

```
c ---------------------------------------------------------------------------------------
c Random number generator.
      FUNCTION ran0(idum)
      implicit none
      INTEGER idum,IA,IM,IQ,IR,MASK
      REAL*8 ran0, AM
      PARAMETER (IA=16807,IM=2147483647,AM=1.d0/IM,
*                IQ=127773,IR=2836,MASK=123459876)
      INTEGER k
      idum=ieor(idum,MASK)
      k=idum/IQ
      idum=IA*(idum-k*IQ)-IR*k
      if (idum.lt.0) idum=idum+IM
      ran0=AM*idum
      idum=ieor(idum,MASK)
      return
      END
c---------------------------------------------------------------------------------------
      subroutine flush1(iunit)
c A subroutine to write all pending output to a file attached
c to logical unit iunit (integer). David Raitt, February 21, 1994
      character*65 file
      integer iunit
      inquire(iunit,err=10,NAME=file)
      close(iunit)
      open(iunit,file=file,status='old',access='APPEND',err=10)
      return
   10 write(*,*) 'File handling error in subroutine flush1.'
      end
```