Collective Instabilities of Linearly Coupled Parametric Oscillators

Nolan Reilly

under the direction of
Mr. Mason Biamonte
Department of Mathematics
Massachusetts Institute of Technology

Research Science Institute
March 25, 2017

Abstract

The Faraday instability is the formation of waves on the surface of a vibrated liquid. While this phenomenon is well-understood in the setting of a fluid container with uniform depth, little is known about the instability in the presence of variable base topography. Motivated by studies of bouncing drops over non-uniform base terrain, we study a simplified model of the Faraday instability in the presence of non-uniform topography in which a Klein-Gordon type equation reduces to a system of coupled parametric oscillators. We analyze this model by decomposing it into spectral components of two types: eigenfunctions of the Laplacian and of a Schrödinger-like operator. We attempt to deduce the types of wave patterns that would develop on the surface according to our model, and find that these spectral expansions lead to equations that are tractable by analytical as well as numerical methods. Finally, by using Renormalization Group analysis, we are able to predict the surface wave pattern of a general class of systems.

Summary

Imagine the ripples in your cup of coffee. The Faraday instability is precisely that effect: the development of an oscillating wave pattern on fluids vibrated above a certain amplitude. This may seem trivial the ripples in your cup are just circles. Now imagine your two-year old put some Legos in your mug, and its flat interior is replaced by a highly complex topography. What do the waves look like now?

Almost exactly this question underpins a recent series of studies. These have found that non-uniform terrain causes unexpected behavior in droplets bouncing on the surface of a vibrated fluid, including apparent tunneling, where the droplets can escape confinement with some probability like a quantum particle. This behavior, and in particular the nature of the confinement, seems to depend on how the droplets interact with the Faraday surface waves, which are inextricable from the base terrain. Consequently, interest in the Faraday instability has revived dramatically, especially over non-uniform topography.

We attempt to gain some understanding of the instability over non-uniform terrain by studying a simplified model which shares many of the characteristics of the system itself. Specifically, our goal is to determine as much as possible about the driving force at which waves first form, and what wave pattern is elicited. Furthermore, while most systems this complicated require extended calculation by a computer, we find that, in many cases, the equations our model gives rise to yield an approximate general expression for the waves that arise from the Faraday instability.

1 Introduction

A decade ago, millimeter-sized droplets were discovered to self-propel along the surface of a harmonically-driven fluid bath by resonant interaction with the waveforms imprinted on the bath surface by the droplets themselves [1]. This self-interacting fluid system has since been shown to exhibit several features previously thought exclusive to quantum-mechanical systems. In particular, single particle diffraction, tunneling, wave-like statistics in confined geometries, quantized orbits, and Zeeman-type orbital-level splitting have all been observed experimentally with these self-propelling, or "walking", droplets [2, 3]. Current research on these bouncing droplet systems has focused on understanding further the mechanism behind the tunneling behavior and on the emergence of wave-like statistics in confined geometries[4, 5]. In both of these situations, the droplets propagate over a bath with variable depth. Communicated through the wave field they generate on the bath surface, these depth variations indirectly exert effective forces on the droplets, allowing them to be completely confined or to be confined over some large time interval until a tunneling event happens. Thus, both research directions are united under the common goal of understanding droplet dynamics in the presence of variable topography.

The physical mechanism underlying the bouncing droplet system is the Faraday instability, the development of standing waves on the surface of a vibrating liquid [6]. The physics underlying the Faraday instability was first quantified by Benjamin and Ursell in 1954, who established that the time-varying gravitational acceleration turns each spectral mode of the surface into a forced parametric oscillator [7]. Parametric oscillators are well-known to be unstable in certain regions of the parameter space [8], and the Faraday instability results when one of the spectral modes reaches an unstable region, increasing in amplitude until standing waves are visible[7, 9, 10]. Crucially, though, these parametric oscillators are uncoupled only when the bottom of the container is flat; if the bottom topography is nonuniform, the oscillators couple together.

Consequently, most of the analytical research on the Faraday instability focuses on the flatbottomed case. When the fluid's behavior must be known over uneven topography, it is most common to approach this numerically, by direct simulation of the fluid equations[11]. This has undoubted advantages, notably including high accuracy, but it is also computationally intensive, slow, and not necessarily transparent as to the causes of changes in behavior. For all these reasons, we study this system over complicated topography by extending the approach Benjamin and Ursell introduced: spectrally decomposing a linearization of the governing equations.

In this paper, we investigate what insight can be derived from such a spectral decomposition of a simplified model of the Faraday instability over nonuniform bottom topography. We see that there are two natural bases for this decomposition, one in eigenfunctions of the Laplacian, and the other in eigenfunctions of a Schrödinger operator, which have distinct strengths and weaknesses. Furthermore, we see that this technique lends itself admirably to analytical approximations, through perturbation methods and Renormalization Group analysis, as well as to a comparatively transparent and fast form of numerical analysis based on Floquet theory. Finally, we are able to predict analytically, based on Renormalization Group analysis (RG), the surface waves set up in a subset of bottom topographies.

2 Construction of Simplified Model

We consider a simplified model of the Faraday instability, as follows:

$$(\partial_t^2 - c^2(t)\partial_x^2 + \kappa(x))\phi(x,t) = 0, \tag{1}$$

$$\phi(x=0,t) = \phi(x=L,t) = 0 \;\forall t, \tag{2}$$

where $c^2(t) = c_0^2(1+\varepsilon\cos(t))$ is a nondimensionalized form of the wavespeed caused by the gravitational acceleration with the vibrational forcing term, $\kappa(x)$ is a measure of the fluid depth at position x, and $\phi(x,t)$ is the height of the fluid surface at position x and time t.

When we move to decompose this equation into spectral functions, we see that, of the multiple possible bases, two stand out as being particularly useful. Eigenfunctions of the Laplacian, in our particular case sines and cosines, are by far the most familiar basis functions, as well as some of the most straightforward, and thus there is considerable benefit to using them. Using eigenfunctions of the Laplacian, we reach the following form for the time component of the above equations:

$$\ddot{a}_n + \frac{n^2\pi^2}{L^2}c_0^2(1+\varepsilon\cos(t))a_n = \sum_m a_m \int_0^L \kappa(x)\sin(\frac{n\pi x}{L})\sin(\frac{m\pi x}{L})dx, \tag{3}$$

where $a_1 - a_N$ are the weights on the eigenfunctions in the expansion.

However, if we write out the separation of variables formalism explicitly, as

$$\frac{-T''}{T} + \frac{c_0^2 X''}{X} + \kappa(x) = -\varepsilon c_0^2\cos(t)\frac{X''}{X},$$

we see that the choice of eigenfunctions most natural to the problem is of Schrödinger-type eigenfunctions satisfying the relation $c_0^2\psi''(x) + \kappa(x)\psi(x) = \lambda\psi(x)$. Furthermore, as the Schrödinger operator is Hermitian, its eigenfunctions are orthogonal, and we may also meaningfully decompose the PDE into spectral components, whereupon we get

$$\ddot{a}_n + \lambda_n(1+\varepsilon\cos(t))a_n = \varepsilon\cos(t)\sum_m a_n \int_0^L \psi_n(x)\psi_m(x)\kappa(x)dx. \tag{4}$$

The main advantage of using Schrödinger eigenfunctions is that the unperturbed equations with $\varepsilon = 0$ are uncoupled, which simplifies considerably the analytics involved in perturbation and related methods. However, the functions are more complicated, and the eigenvalues are specified by a transcendental equation in all but the simplest of cases.

We next truncate at a finite number of modes N, for computational tractability, and introduce viscosity proportional to the frequency squared (and therefore proportional to the eigenvalues), to mimic the physical system. However, we observe that, for a second-order matrix equation $\ddot{\vec{y}}+$

$\Gamma\dot{\vec{y}}+A(t)\vec{y}=0$, where Γ is a diagonal damping matrix, the change of variables $\vec{x}=e^{\frac{t}{2}\Gamma}\vec{y}$ eliminates the damping terms, leaving us with $\ddot{\vec{y}}+(A(t)-\frac{1}{4}\Gamma^2)\vec{y}=0$. Thus, when we substitute this into the matrix equations for both bases, (3) and (4), we arrive at the following form:

$$\ddot{\vec{x}}+(\Lambda+K-\frac{\gamma^2}{4}\Lambda^2)\vec{x}=-\varepsilon\cos(t)\Lambda\vec{x} \qquad (Laplacian) \tag{5}$$

$$\ddot{\vec{x}}+(\Lambda-\frac{\gamma^2}{4}\Lambda^2)\vec{x}=\varepsilon\cos(t)(K-\Lambda)\vec{x} \qquad (Schrödinger), \tag{6}$$

where γ is the constant of proportionality relating the damping to the eigenvalues.

Finally, we transform the second-order matrix equations above into first order by introducing a new solution vector $\vec{X}(t)=(x_1,...,x_N,\dot{x_1},...,\dot{x_N})$, at which point our final equations become

$$\dot{\vec{X}}+\begin{pmatrix} 0 & I \\ \Lambda-\frac{\gamma^2}{4}\Lambda^2+K & 0 \end{pmatrix}\vec{X}=\varepsilon\cos(t)\begin{pmatrix} 0 & 0 \\ -\Lambda & 0 \end{pmatrix}\vec{X} \tag{7}$$

$$\dot{\vec{X}}+\begin{pmatrix} 0 & I \\ \Lambda-\frac{\gamma^2}{4}\Lambda^2 & 0 \end{pmatrix}\vec{X}=\varepsilon\cos(t)\begin{pmatrix} 0 & 0 \\ K-\Lambda & 0 \end{pmatrix}\vec{X}. \tag{8}$$

3 Numerical Approach

We test our hypotheses and investigate the properties of coupled oscillators by integrating them numerically. The numerical method that we employ to determine stability depends on the Floquet Theorem, which states that, for a linear first-order system of N equations

$$\dot{\vec{y}}(t)=A(t)\vec{y}(t)$$

such that $A(t+T)=A(t)$ for any t, the solution can be expressed as

$$\vec{y}=e^{tF}\vec{p}(t), \tag{9}$$

for some constant matrix F and T-periodic function $\vec{p}(t)$. Therefore,

$$\vec{y}(nT) = e^{nT}\vec{p}(0),$$

so the stability of the system depends on whether the largest eigenvalue of this matrix has modulus less than 1. For later convenience, let us call e^{TF} the time-T map of our system, because

$$e^{TF}\vec{y}(nT) = \vec{y}((n+1)T).$$

Furthermore, observe that

$$\vec{y}(T) = e^{TF}\vec{p}(T) = e^{TF}\vec{p}(0)$$

and

$$\vec{y}(0) = e^{0}\vec{p}(0) = I\vec{p}(0) = \vec{p}(0).$$

Thus, we can obtain a product of the time-T map with the initial condition vector simply by integrating our equations forward from these initial conditions, and given N linearly independent initial condition vectors the desired matrix can be solved for. One particularly simple choice of initial condition vectors is the columns of the identity matrix, as no operations need be performed to solve for the matrix. Performing the time-integration for all initial condition vectors at once is identical to integrating forward the matrix differential equation

$$\dot{\Omega}(t) = A(t)\Omega(t), \quad \Omega(0) = I, \tag{10}$$

and this is the approach we take. To determine overall stability, we integrate the matrix equation forward using the Trapezoid Rule until $t = T$, and then determine if any eigenvalue of the resultant e^{TF} matrix exceeds modulus 1. The Trapezoid Rule is particularly appropriate for this problem because, when integrating over one period of a periodic function, its errors cancel out to higher orders [12], thus allowing more exact integration with fewer timesteps for our periodic-coefficient problem. However, if more detail about the stability properties of the solution is desired, this is fully available. The eigenvectors of the Floquet matrix corresponding to the unstable eigenvalues give the combinations of coefficients that go unstable, and thus specify the asymptotic solution.

The major advantage of this approach is that, unlike more direct simulation, in which the entire solution is generated for an extended length of time, and the asymptotic solution is inferred based on apparent trends, the entire stability properties of the solution may be determined by integrating over only a single period. Thus, even fairly large numbers of basis functions may be included in the simulation while retaining runtimes of less than a few seconds.

4 Analytical Approach

4.1 Instability Tongue-Splitting

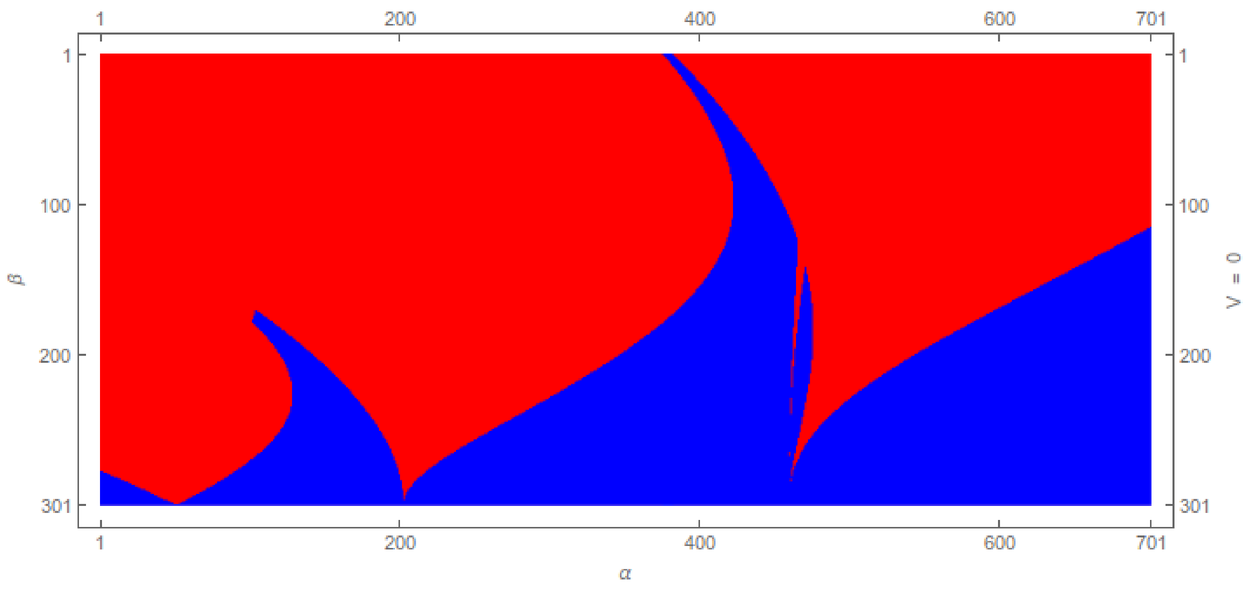

Figure 1: Stability Diagram of Two Uncoupled Oscillators in α^2 ε-space

Figures 5-9 show the effects of coupling on stability in a two-oscillator system using eigen-

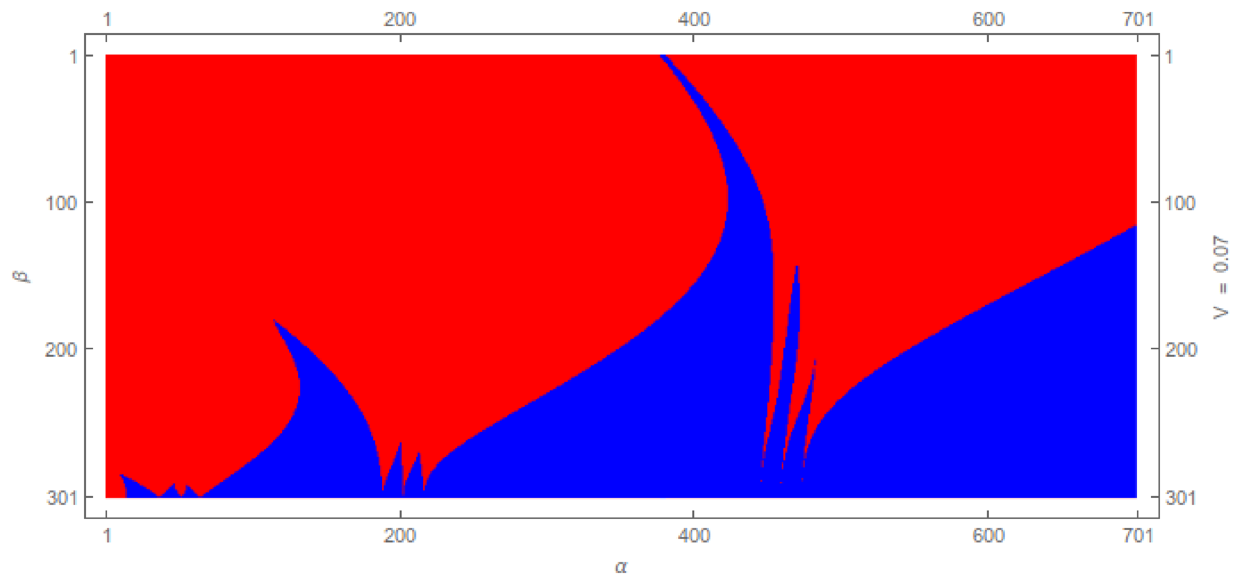

Figure 2: Stability Diagram of Two Weakly Coupled Oscillators in α^2 ε-space

functions of the Laplacian, where the single coupling coefficient is denoted by V. In the stability diagrams (Figures 5-6), red signifies an unstable point, whereas blue indicates a stable solution. Thus, the tongues of instability are the red strips reaching down to graze the axis. Given this, multiple effects of coupling are immediately apparent. For each single intersection in the uncoupled diagram there are three in the coupled diagram, one new intersection on each side of the original one. In the weak-coupling regime, they appear to be equally-spaced around their progenitor, but in the $V = 0.5$ case in Figure 9, they are somewhat asymmetric.

All of these effects for two coupled oscillators can be explained analytically using perturbation analysis. As the dissipative equations can be reduced to the dissipation-free equations, let us express our full equations with no dissipation for two modes as[1]

$$\ddot{x} = -(\alpha^2 + \varepsilon\cos(t))x + V_0 y \tag{11}$$

$$\ddot{y} = -(\alpha^2 + 4\varepsilon\cos(t))y + V_0 x. \tag{12}$$

Next, we expand our solution into a perturbation series. A first order approximation yields the following equations for $\vec{y}_0$ and $\vec{y}_1$:

$$\begin{pmatrix} \ddot{x}_0 \\ \ddot{y}_0 \end{pmatrix} = \begin{pmatrix} -\alpha^2 & V_0 \\ V_0 & -\alpha^2 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \tag{13}$$

$$\begin{pmatrix} \ddot{x}_1 \\ \ddot{y}_1 \end{pmatrix} = \begin{pmatrix} -\alpha^2 & V_0 \\ V_0 & -\alpha^2 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \begin{pmatrix} -\varepsilon\cos(t)x_0 \\ -4\varepsilon\cos(t)y_0 \end{pmatrix}. \tag{14}$$

A combination of α^2, ε, and V_0 yields unstable solutions if the perturbative expansion yields resonance. This occurs in our first-order expansion if the inhomogeneity in the first-order term achieves resonance with the normal modes of the homogeneous system. The normal modes can be calculated as the square roots of the eigenvalues of A_0, where A_0 is the matrix in the zeroth- and first- order equations, and yield

$$\omega = \pm\sqrt{\alpha^2 + V_0}, \quad \pm\sqrt{\alpha^2 - V_0}. \tag{15}$$

Therefore, the solution to our zeroth-order system is some linear combination of $e^{i\sqrt{\alpha^2+V_0}t}$, $e^{-i\sqrt{\alpha^2+V_0}t}$, $e^{i\sqrt{\alpha^2-V_0}t}$, and $e^{-i\sqrt{\alpha^2-V_0}t}$, and we can conclude our system is unstable if $\cos(t) = \frac{1}{2}(e^{it} + e^{-it})$ multiplied by any of the complex exponentials in $\vec{y}_0$ is any of the normal modes. In

[1]Sept. 19, 2016. The author discovered on this date that this formula is missing an n^2 term, and thus the second of these formulae would be $\ddot{y} = -4(\alpha^2 + \varepsilon\cos(t))y + V_0 x$. While the diagrams and results of this section can no longer be considered correct, the results and claims of the paper as a whole, as well as the usefulness of the method, remain valid.

other words, if

$$\pm\sqrt{\alpha^2 \pm V_0} = \pm 1 \pm \sqrt{\alpha^2 \pm V_0}. \tag{16}$$

This yields the following set of solutions for ω_1:

$$\alpha^2 = \left\{ \frac{1}{4} - V_0, \frac{1}{4} + V_0^2, \frac{1}{4} + V_0 \right\}, \tag{17}$$

which are the three solutions we observed on the graph.

First-order perturbation analysis can only reveal the instabilities around $\frac{1}{4}$, but with n^{th}-order perturbation analysis, we can see that there are terms of up to $\cos^n(t)e^{\pm i\sqrt{\alpha^2 \pm V_0 t}}$, which following the same logic as above, are unstable if

$$\pm\sqrt{\alpha \pm V_0} = \pm n \pm \sqrt{\alpha^2 \pm V_0}. \tag{18}$$

This, after some algebra, yields the following formulæ for the n^{th} instability valleys and the values of intersection with the ε-axis:

$$\alpha^2 = \frac{n^2}{4} - V_0, \frac{n^2}{4} + \frac{V_0^2}{n^2}, \frac{n^2}{4} + V_0. \tag{19}$$

When this is compared to the stability diagrams produced by our numerical algorithms, a very precise matching is obtained. Although there is insufficient resolution in the diagrams to see exactly where the intability tongues touch the axis, counting pixels in the diagram allowed us to reach approximate values for the α^2 position of the intersections. The only error observed was that all the values were shifted towards positive alpha by 10 pixels, but this offset was independent of n and V_0, and so is likely inherent in the images, rather than the data. However, the separation of the intersections was exact, with the largest- and smallest- α intersections being separated by $2V_0$ exactly down to the precision of the pixels, or 0.005, and the third intersection deviating from the midpoint of the other two by $\frac{V_0^2}{n^2}$. We confirmed this result on the $n = 2$ and $n = 3$ intersection groupings of the $V_0 = 0.07$, $V_0 = 0.15$, and $V_0 = 0.5$ stability diagrams.

Any formula for N spectral modes (in eigenfunctions of the Laplacian) would have to account for the coupling used between the modes, and for that reason a general formula is not easily obtainable. However, if the eigenvalues of the coupling matrix can be obtained, the rest is just root-finding, because

$$\pm\sqrt{\lambda_i} \pm \sqrt{\lambda_j} = n$$

implies

$$n^4 - 2n^2(\lambda_i + \lambda_j) + (\lambda_i - \lambda_j)^2 = 0,$$

solved for α^2. This form also shows that there are $\frac{N(N+1)}{2}$ roots, since each combination of λ_i and λ_j yields only one solution for α^2, and thus there are $\begin{pmatrix} N \\ 2 \end{pmatrix} = \frac{N(N+1)}{2}$ intersections with the axis.

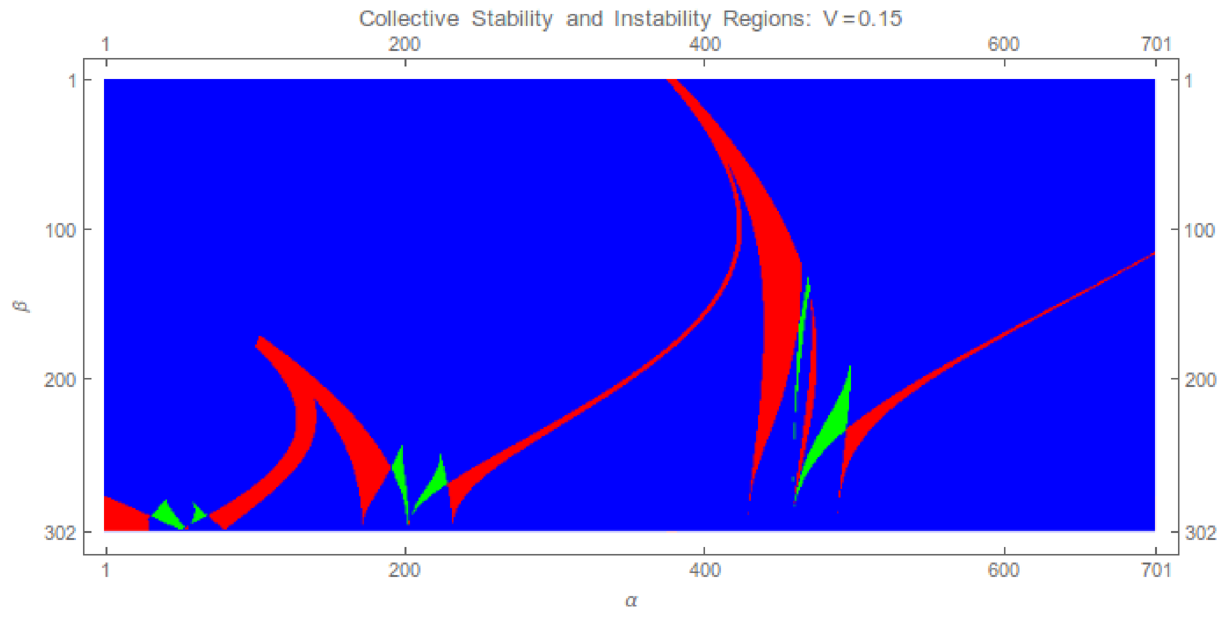

Figure 3: Effects of Coupling: coupled and uncoupled stability diagrams are overlaid, with blue indicating same stability, green indicating greater stability with coupling (collective stability), and red indicating lesser stability with coupling (collective instability.)

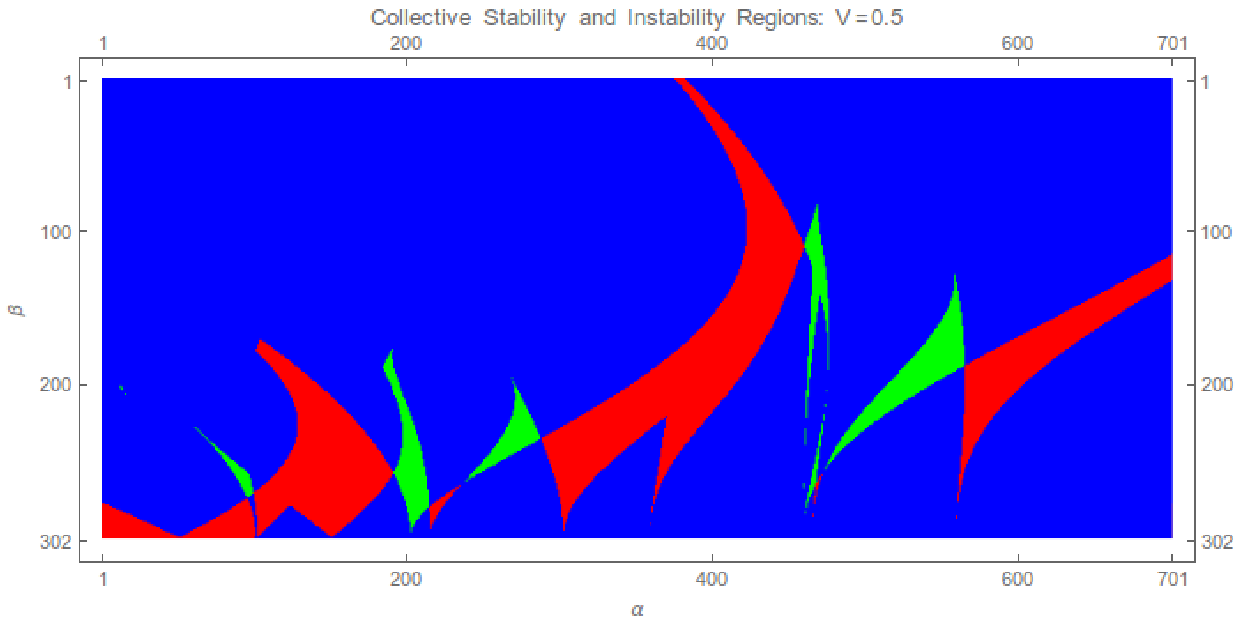

Figure 4: Effects of Coupling: coupled and uncoupled stability diagrams overlaid for a higher value of coupling, V = 0.5

4.2 Characterization of Instabilities using the Renormalization Group

In the following section we will demonstrate the usefulness of the expansion in terms of Schrödinger eigenfunctions by showing that concrete information can be gained about a general system of equations derived from such an expansion by means of the Renormalization Group.

Before applying RG analysis to our equations, we will lay out the general RG procedure (for more detail see [13]). Let us consider a system of linear differential equations (linearity is unnecessary for the procedure, but it simplifies the calculations):

$$\dot{\vec{x}} + (C + \varepsilon D(t))\vec{x} = 0,$$

so that its perturbation expansion has solutions of the form

$$\vec{x} \approx M(t)[(1+\varepsilon)I + \varepsilon(t-t_0)S + \varepsilon P(t)]\vec{A}.$$

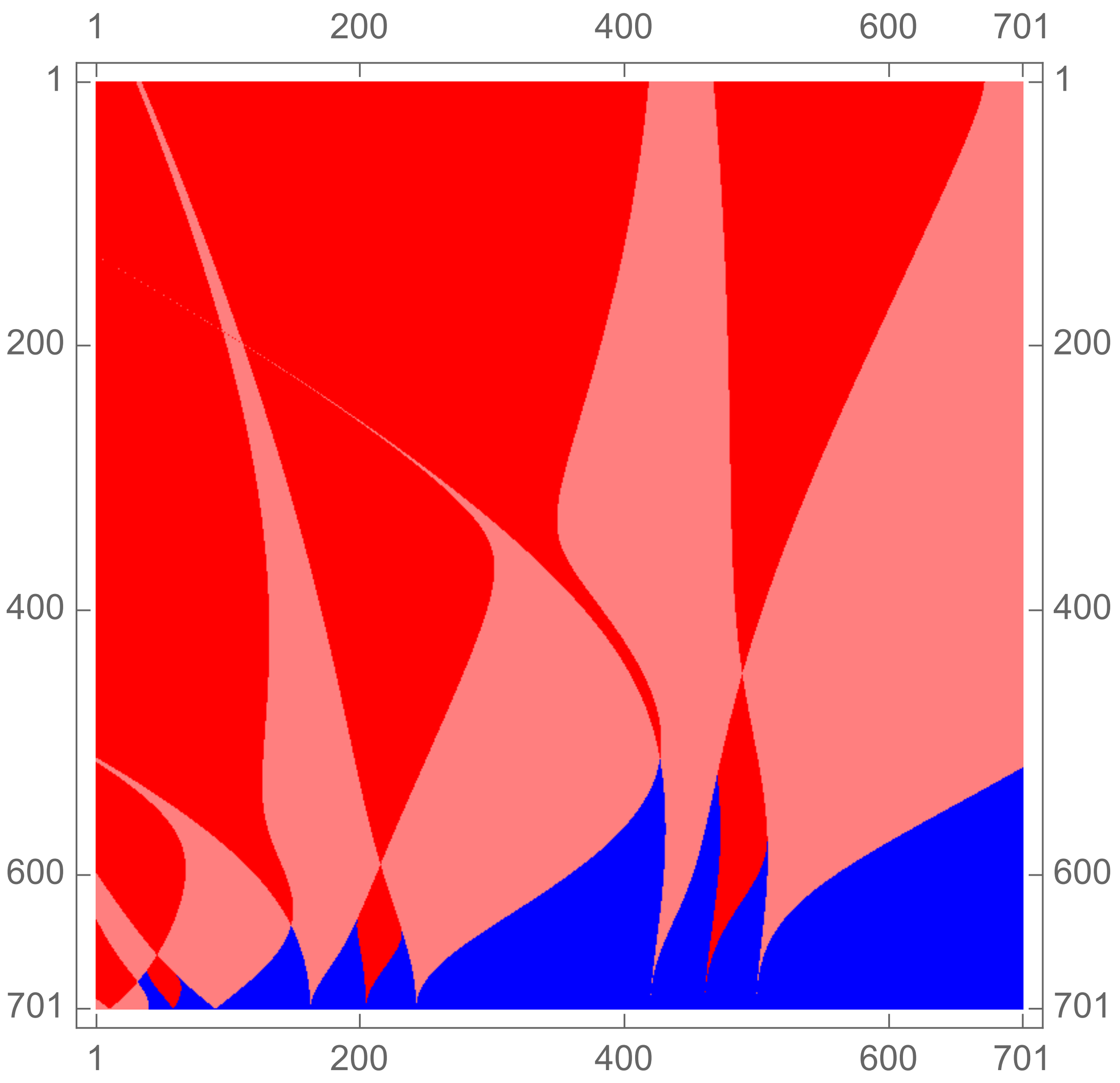

Figure 5: Another look at instability tongues, here in the $V_0 = 0.2$ case. Here we see the number of unstable eigenvalues displayed as color: 0 unstable eigenvalues are blue, 1 unstable eigenvalue is pink, and two are red.

where $M(t)$ is the fundamental matrix of $\dot{\vec{x}} + C\vec{x} = 0$, $(t - t_0)S$ are secular terms, and $P(t)$ are periodic terms. The crucial idea in RG is that there is no reason for which the solution should depend on t_0 especially; we can "renormalize" the solution vector so that $\vec{R} = (I + \varepsilon(\tau - t_0)S)\vec{A}$, where τ is an arbitrary constant, and then the solution, up to first order in ε, is

$$\vec{x} \approx M(t)[(1+\varepsilon)I + \varepsilon(t-\tau)S + \varepsilon P(t)]\vec{R}.$$

But τ is even less intrinsic to the problem that t_0 – in fact, it is completely arbitrary. Consequently, the solution *cannot* depend on τ, and thus

$$\frac{\partial \vec{x}}{\partial \tau} = M(t)[\frac{d\vec{R}}{d\tau} - \varepsilon S\vec{R} + \varepsilon(I+S+P(t))\frac{d\vec{R}}{d\tau}] = 0. \tag{20}$$

The last observation required to simplify this equation into a useful form is that $\frac{d\vec{A}}{d\tau} = 0$, and so, since $\vec{R} = (I+\varepsilon(\tau-t_0)S)\vec{A}$, $\frac{d\vec{R}}{d\tau} = O(\varepsilon)$ [13]. Thus, using the fact that $M(t)$ is composed of linearly independent columns, and is thus invertible, we arrive at the first-order RG equation for a system of linear equations in this form:

$$\frac{d\vec{R}}{d\tau} = \varepsilon S\vec{R}. \tag{21}$$

Finally, once we solve this equation for $\vec{R}(\tau)$, we use the fact that τ is completely arbitrary to set it equal to t, so that our final, renormalized solution is

$$\vec{x} \approx M(t)[(1+\varepsilon)I + \varepsilon P(t)]\vec{R}(t)$$

As we have said before, the major advantage of the Schrödinger eigenfunction expansion is that, to 0^{th} order in ε, the equations are uncoupled. This is particularly useful for finding the secular terms in a set of equations with otherwise arbitrary parameters, since the fundamental matrix of the first-order correction terms is composed of diagonal matrices. Thus, in the variation of parameters for the inhomogeneous solution of the first-order perturbation equation,

$$\varepsilon M(t)\int_{t_0}^{t} M^{-1}(s)D(s)M(s)ds\vec{A},$$

$M^{-1}(s)D(s)M(s)$ can be calculated explicitly.

If Schrödinger eigenfunctions have a drawback, though, it is that, without looking in detail at a particular type of bottom topography, it is nearly impossible to know anything about the eigenvalues. Therefore, it makes most sense to treat the eigenvalues as completely independent one from another and see what we can say for the case with completely arbitrary λ_n.

However, since as in the two-oscillator case we discussed in the instability-tongue-splitting, there are only secular terms on very restrictive combinations of eigenvalues, direct application of RG would only yield information about individual lines or points in what can be considered the "space" of eigenvalue combinations. To get around this, we propose introducing "error" terms[2] into our equations: expressing a point close to a set of eigenvalues $(\lambda_1^*, ...\lambda_N^*)$ as

$$(\lambda_1, ...\lambda_N) = (\lambda_1^*, ...\lambda_N^*) + \varepsilon(E_1, ...E_N), \tag{22}$$

where $(E_1, ...E_N)$ are the error terms. The benefit of this expansion is that, if this expression for a point is inserted into the governing equations, secular terms appear for a combination of eigenvalues that would not otherwise produce them, and thus we can now expand our reach from narrow lines and points to a meaningful fraction of the space as a whole.

If we look at a particular case using the above techniques, namely points surrounding the

[2]This technique is inspired by that used in [13] to calculate the instability boundary $a(\varepsilon)$ of a Mathieu oscillator $\ddot{x} + (a + \varepsilon\cos(t))x = 0$. The procedure there outlined expands $a(\varepsilon) = \frac{1}{4} + a_1\varepsilon + a_2\varepsilon^2 + ...$ and substitutes this in the perturbation expansion. However, as far as I know, the interpretation of the error terms as measures of distance from an unstable solution, as outlined in the text, is original. Furthermore, this interpretation has different implications for the higher-order terms: in [13], a_1 appears at first order, a_2 appears at 2nd order, etc., while the error terms are the same at each order.

$\lambda_m - \lambda_n = \pm 1$ instability, we see that the secular term matrix for this case, with error terms, is:

$$\begin{pmatrix} \frac{-iE_1}{B_1} & \frac{-iK_{mn}}{2\lambda_n} & & & & \\ \frac{-iK_{mn}}{2\lambda_m} & \ddots & & & 0 & \\ & & \frac{-iE_N}{B_N} & & & \\ & & & \frac{iE_1}{B_1} & \frac{iK_{mn}}{2\lambda_n} & \\ & 0 & & \frac{iK_{mn}}{2\lambda_m} & \ddots & \\ & & & & & \frac{iE_N}{B_N} \end{pmatrix},$$

where we have defined $B_i = \sqrt{1 - \frac{\gamma}{4}\lambda_i^2}$ for convenience. Thus, we see by solving the RG equation with this matrix that if a particular set of eigenvalues is close to satisfying $\lambda_m - \lambda_n = \pm 1$, the largest eigenvalue of this matrix will be the dominant growing exponent in the solution to our system of equations, and its eigenvector will then determine what the liquid surface looks like at equilibrium.

The eigenvalues of the matrix can be found to be

$$\lambda = \pm \frac{iE_k}{B_k}, \qquad k \neq m, n \tag{23}$$

$$\lambda = \pm \frac{i}{2}\left(\frac{E_m}{B_m} + \frac{E_n}{B_n}\right) \pm \frac{i}{2}\sqrt{\left(\frac{E_m}{B_m} - \frac{E_n}{B_n}\right)^2 + (-1)^{m+n}\frac{K_{mn}^2}{\lambda_m \lambda_n}}, \qquad \textit{else}. \tag{24}$$

Of particular note is the presence of the $(-1)^{m+n}$ term in the discriminant of the second eigenvalue formula, since it breaks a symmetry between eigenvalue pairs separated by an even number of eigenvalues and those separated by an odd number. In fact, it breaks this symmetry quite dramatically: λ_m and λ_n must be positive to maintain the physically meaningful oscillatory solutions in time, at least for small ε, so we can conclude that $\frac{K_{mn}^2}{\lambda_m \lambda_n} > 0$, and thus there are only complex (i.e. unstable) eigenvalues for $m + n$ odd. Therefore, the existence of any unstable solutions at all for small ε is affected by this symmetry-breaking. We can only speculate about the origin of this asymmetry, but it is perhaps relevant that the eigenvalues are assumed to be ordered in increasing magnitude, and thus that rearranging them (as would be necessary to change $m+n$) would indeed change an essential aspect of the solution.

The error terms in the discriminant show that there are unstable solutions for $|\frac{1}{B_m}E_m - \frac{1}{B_n}E_n| < \frac{|K_{mn}|}{\sqrt{\lambda_m\lambda_n}}$, and thus we can determine two important details about which solutions go unstable near this case. Firstly, basis functions that interact more strongly with the terrain will have a larger effect on the liquid surface, which suggests a heuristic that basis functions with characteristic length scales comparable to the terrain's characteristic length scale will be more significant contributors toward the final waveform. Secondly, larger eigenvalues decrease both the maximum exponent and the size of the region which will be unstable, so smaller eigenvalues will carry a much larger contribution to the instability structure than larger ones.

One last point is highly notable: given the eigenvalues above, we can calculate the eigenvectors corresponding to the growing eigenvalues, and thus determine the linear combination of basis functions forming the surface wave. In particular, we see that the components of the two dominant eigenvectors are:

$$\lambda = \frac{i}{2}(\frac{E_m}{B_m} + \frac{E_n}{B_n}) - \frac{i}{2}\sqrt{(\frac{E_m}{B_m} - \frac{E_n}{B_n})^2 + (-1)^{m+n}\frac{K_{mn}^2}{\lambda_m\lambda_n}} :$$

$$v_j = 0, \qquad j \neq N+m, N+n$$

$$v_{N+m} = \frac{-iK_{mn}}{2\lambda_m}$$

$$v_{N+n} = \frac{i}{2}(\frac{E_m}{B_m} - \frac{E_n}{B_n}) + \frac{i}{2}\sqrt{(\frac{E_m}{B_m} - \frac{E_n}{B_n})^2 - \frac{K_{mn}^2}{\lambda_m\lambda_n}},$$

$$\lambda = \frac{-i}{2}\left(\frac{E_m}{B_m}+\frac{E_n}{B_n}\right)-\frac{i}{2}\sqrt{\left(\frac{E_m}{B_m}-\frac{E_n}{B_n}\right)^2+(-1)^{m+n}\frac{K_{mn}^2}{\lambda_m\lambda_n}}:$$

$$v_j = 0, \qquad j \neq m,n$$

$$v_m = \frac{iK_{mn}}{2\lambda_m}$$

$$v_n = \frac{i}{2}\left(\frac{E_n}{B_n}-\frac{E_m}{B_m}\right)+\frac{i}{2}\sqrt{\left(\frac{E_m}{B_m}-\frac{E_n}{B_n}\right)^2-\frac{K_{mn}^2}{\lambda_m\lambda_n}}$$

To sum up, we can reach very concrete conclusions about systems of equations based on a Schrödinger eigenfunction expansion, even if we know next to no concrete information about the particular values of the eigenvalues. Furthermore, even though this information is based on first-order RG, and is thus confined in validity to close to known unstable solutions, it would not require unduly more effort to extend this procedure to higher orders of RG, and thus to larger distances from the instability tongues. Consequently, it is not inconceivable that the dominant unstable exponents could be determined a priori for most bottom topographies.

5 Discussion and Conclusion

In this paper, we have attempted to show that significant headway can be made in predicting the behavior of a PDE qualitatively similar to that governing the Faraday instability over nonuniform topography by expanding it into spectral components. The techniques we lay out are significant because they offer the possibility of faster prediction of surface waves and of greater analytical understanding of the patterns that arise. Three further components have simplified this analysis dramatically, and allowed us to draw broad-ranging conclusions about individual problems. First is the expansion of the problem into the more natural basis of Schrödinger eigenfunctions, which yield time-component equations uncoupled to 0^{th} order in driving amplitude. This, in turn, allows explicit calculation of the secular terms in the perturbation series – for general eigenvalues – permitting Renormalization Group stability analysis for a general bottom topography. Second is the use of a Floquet Theory-based numerical algorithm, which allows us to reduce the time required to integrate the equations by a large factor. Third is the introduction of the error terms into the perturbation series, which broaden the area of eigenvalue combinations which our RG approximations can reach.

For future research two directions stand out especially. One is a direct extension of the research we lay out in this paper: applying these techniques to predict the shape of the surface wave over a step-function topography according to our toy model of the Faraday instability. This topography function is one of the simplest nontrivial cases, but also one of the most often used in bouncing drops studies, so it could afford an understanding of how close our toy model's predictions are to the observed surface waves in a physically relevant context. Furthermore, a numerical approach would only require calculating the coupling coefficients (and the eigenvalues in the case of a Schrödinger expansion) to reach a good approximation of the solution of the PDE.

An analytical approach would likely also be feasible, since RG yields information about the surface wave through the eigenvector of the largest eigenvalue of the RG matrix. To reach an acceptable approximation, it would be necessary to extend the first-order RG analysis with higher-order terms until the error were small enough over the entire range of possible eigenvalues. However, this would once again be a good test of the practicality of RG analysis of higher orders.

The other area which clearly merits investigation is the application of these techniques to the equations governing the true Faraday instability over uneven terrain. This would introduce several complications, including weak nonlinearity and an implicit relation with the bottom topography. Consequently, the equations thus derived may be most tractable numerically, or through RG. Nevertheless, this is the true goal behind this entire approach: to draw conclusions about the physical system itself.

6 Acknowledgments

Many thanks to Mason Biamonte, who introduced me to this problem and has helped me every step of the way; to Dr. Rickert, who gave much substantive and helpful criticism of my paper; to Dr. Khovanova, for much constructive criticism about how to improve my paper and where to take my project; to Dr. Slava Gerovitch, Dr. David Jerison, and Dr. Ankur Moitra of the Massachusetts Institute of Technology, for watching over my project and making sure nothing went horribly wrong; to MIT, CEE, and RSI, for allowing us to be here in the first place; and to Mr. John H. Downs, Jr., Dr. Pam Krahl, Drs. Joseph and Nell Sedransk, Mr. Jerome H. Powell and Ms. Elissa A. Leonard, Mr. and Mrs. Bhanu Durvasula, Ms. Emily E. Riehl, Dr. Noreen Hynes, and Ms. Carolyn Smith, for making this experience possible for me.

References

[1] S. Protiere, Y. Couder, E. Fort, and A. Boudaoud. Walking and orbiting droplets. *Nature*, 437:208, 2005b.

[2] S. Protiere, A. Boudaoud, and Y. Couder. Particle wave association on a fluid interface. *J.ournal of Fluid Mechanics*, 554:85–108, 2006.

[3] Y. Couder and E. Fort. Single particle diffraction and interference at a macroscopic scale. *Phys. Rev. Lett.*, 97:154101, 2006.

[4] C.Galeanos-Rios, P. Milewski, A. Nachbin, and J. Bush. Faraday pilot-waves: Generation and propagation. In S. A. D. System, editor, *APS Meeting Abstracts*. 2015.

[5] P. A. Milewski, C. A. Galeano-Rios, A. Nachbin, and J. W. M. Bush. Faraday pilot-wave dynamics: modelling and computation. *J. of Fluid Mechanics*, 778(1469-7645):361–388, 9 2015.

[6] M. Faraday. On a peculiar class of acoustical figures; and on certain forms assumed by groups of particles upon vibrating elastic surfaces. *Proc. Royal Soc. London*, 3:49–51, 1830.

[7] T. B. Benjamin and F. Ursell. The stability of the plane free surface of a liquid in vertical periodic motion. *Proceedings of the Royal Society of London. S.eries A, Mathematical and Physical Sciences*, 225:505–515, 1954.

[8] B. V. der Pol and M. J. O. Strutt. On the stability of the solutions of mathieu's equation. *The Philosophical Magazine*, 5:18, 1928.

[9] K. Kumar and L. S. Tuckerman. Parametric instability of the interface between two fluids. *J. of Fluid Mechanics*, 279(1469-7645):49–68, 11 1994.

[10] J. Miles and D. Henderson. Parametrically forced surface waves. *Annu. Rev. Fluid Mech.*, 22:143–165, 1990.

[11] M. Biamonte. Personal conversation, July 2016.

[12] Q. I. Rahman and G. Schmeisser. Characterization of the speed of convergence of the trapezoidal rule. *Numerische Mathematik*, 57:123–138, 1990.

[13] L.-Y. Chen, N. Goldenfeld, and Y. Oono. Renormalization group and singular perturbations: Multiple scales, boundary layers, and reductive perturbation theory. *Physical Review*, 54:376–394, 1996.

Appendix A   Stability Determination Code

```
%--------------------------------------------------------------------------------------------------------
FIRST PROGRAM: floquetChecker.h
%--------------------------------------------------------------------------------------------------------

#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cmath>
#include "eigen/Eigen/Dense"

using namespace std;
using namespace Eigen;

const double PI = 2*acos(0.0);

time_t timer = time(NULL);
struct tm y2k = {0};
unsigned long idum,itemp;
#ifdef vax
static unsigned long jflone = 0x00004080;
static unsigned long jflmsk = 0xffff007f;
#else
static unsigned long jflone = 0x3f800000;
static unsigned long jflmsk = 0x007fffff;
#endif

int floquet(int numOsc, double alpha, double beta, double kappa, double dissipation = 0.0){
/*Faster method of determining whether a point is stable or unstable, by finding the eigenvalues of the primary time-2*pi map. The error over number of

        //ofstream outFile("convergence.txt", ios::app); For the convergence analysis of this algorithm

        int i, j, idx, idx2;

        int steps=28;
        double dt = 2*PI/steps;          //timestep
        double t = 0;
//      double dissipation = 0.3;        //Viscosity/friction constant for simple linear damping

        //Defining constants to get a seed for the random number generator
        y2k.tm_hour = 0;   y2k.tm_min = 0; y2k.tm_sec = 0;
        y2k.tm_year = 100; y2k.tm_mon = 0; y2k.tm_mday = 1;
        idum = (unsigned long) difftime(timer, mktime(&y2k));

        MatrixXd B(2*numOsc, 2*numOsc); //Stores I + dt/2*A(t), where A is the derivative matrix
        MatrixXd C(2*numOsc, 2*numOsc); //Stores I - dt/2*A(t+dt)

        MatrixXd Ilarge = MatrixXd::Identity(2*numOsc, 2*numOsc); //Large identity matrix, b/c will be reused
        MatrixXd Ismall = MatrixXd::Identity(numOsc, numOsc);    //Small identity matrix, b/c will be reused
        MatrixXd zeros = MatrixXd::Zero(numOsc, numOsc);        //Matrix of zeros, b/c will be reused

        MatrixXd fundamental = Ilarge;  //Stores the final time−T map

        MatrixXd Bfun(2*numOsc, 2*numOsc);      //Stores B*fundamental during timestepping

        MatrixXd dissip = Ismall;       //Stores frequency−dependent dissipation
        for (i = 0; i < numOsc; i++)    dissip(i, i) = −dissipation*(i+1)*(i+1);

        //Store the eigenvalues and eigenvectors of fundamental
        VectorXcd eigenvals(2*numOsc);
        MatrixXcd eigenvecs(2*numOsc, 2*numOsc);

        VectorXd y(numOsc);
        VectorXd v(numOsc);

        MatrixXd coupling(numOsc, numOsc);      //Stores the relation between y'' and y, including the

        if (numOsc==2)  coupling << 0, kappa, kappa, 0;

        else{
                for(i = 0; i < numOsc; i++){
                        for(j = i+1; j < numOsc; j++){
                                coupling(i, j) = kappa/abs(i−j);
                                coupling(j, i) = coupling(i, j);
                        }
                }
        }

        for (i = 0; i < numOsc; i++)    coupling(i, i) = −(alpha + (i+1)*(i+1)*beta*cos(t));

        while(t<2*PI){

                B << zeros, Ismall, coupling, dissip;

                B = Ilarge + (dt/2)*B;

                t += dt;

                for (i = 0; i < numOsc; i++)    coupling(i, i) = −(alpha + (i+1)*(i+1)*beta*cos(t));

                C << zeros, Ismall, coupling, dissip;

                C = Ilarge − (dt/2)*C;

                Bfun = B*fundamental;

                //Solves the system C*fundamental(step n+1) = B*fundamental(step n);
                ColPivHouseholderQR<MatrixXd> dec(C);

                for(i = 0; i < 2*numOsc; i++){
                        fundamental.col(i) = dec.solve(Bfun.col(i));
                }
        }

        //Floquet Analysis
        //1st-order linear eqn. system with time-dependent coefficients with period T has solution of the                form y(nT + t) = (time-T map)^n
        //Thus, in the long run, the stability is determined by the largest eigenvalue of the time-T map,                which is an exponential of a m
        //However, the time-T map is the matrix solution to the same initial value problem where the              initial matrix is the identity matrix.
        //Therefore, the system is stable iff the largest-modulus eigenvalue of the time-T map, which we                calculated above, has modulus <

        EigenSolver<MatrixXd> es;

        //cout << fundamental << endl;

        es.compute(fundamental, true);          //true -> does compute the eigenvectors
        eigenvals = es.eigenvalues();
        eigenvecs = es.eigenvectors();
        /*complex<double> logeigen;

        //For convergence testing: outputs eigenvals
        for (i = 0; i < 2*numOsc; i++){
                logeigen = log(eigenvals(i));
                outFile << real(logeigen) << " " << imag(logeigen) << " ";
        }*/
        //complex<double> product = eigenvals(0)*eigenvals(3)*eigenvals(1)*eigenvals(2);
        //outFile << log(real(product)) << " " << dissipation << endl;
        //outFile << endl;

        int unstable = 0;
        double max = 0;

        for (i=0; i<2*numOsc; i++){
                max = 0;
                if (abs(eigenvals(i)) > 1.00001){
                        for (j=0; j<2*numOsc; j++)
                                if (max < abs(eigenvecs(j, i))) max = abs(eigenvecs(j, i));
                        for (j=0; j<2*numOsc; j++){
                                if (max/abs(eigenvecs(j, i)) < 1.0001){
                                        unstable++;
                                }
                        }
                }
        }


        return unstable;
}
%-------------------------------------------------------------------------------------------------------------
SECOND PROGRAM: new_verlet.h
%-------------------------------------------------------------------------------------------------------------

#include <iostream>
#include <fstream>
```

```
#include <cstdlib>
#include <cmath>
#include "eigen/Eigen/Dense"

using namespace std;
using namespace Eigen;

time_t timer = time(NULL);
struct tm y2k = {0};
unsigned long idum,itemp;
#ifdef vax
static unsigned long jflone = 0x00004080;
static unsigned long jflmsk = 0xffff007f;
#else
static unsigned long jflone = 0x3f800000;
static unsigned long jflmsk = 0x007fffff;
#endif

bool param_verlet(int numOsc, double alpha, double beta, double kappa){
/*Tests whether a system of parametric oscillators
                xi" + (alpha + i*i*beta*cos(t))*xi = sum(couplingterm*xj)
is stable*/
        int i, j;

        double dt = .2;         //timestep: largest safe value estimated at 0.2
        double t = 0;
        double E0 = 0.0;        //Energy at t=0, to see whether the energy grows exponentially
        double E;               //Energy at later timesteps



        y2k.tm_hour = 0;   y2k.tm_min = 0; y2k.tm_sec = 0;
        y2k.tm_year = 100; y2k.tm_mon = 0; y2k.tm_mday = 1;
        idum = (unsigned long) difftime(timer, mktime(&y2k));

        VectorXd y(numOsc), v(numOsc), acc(numOsc), acc_last(numOsc);   //Position and velocity, and

        for (i=0; i<numOsc; i++){                         //Slightly random initial conditions
                idum = 1664525L*idum + 1013904223L;
                itemp = jflone | (jflmsk & idum);
                y(i) = 0.01*((*(float *)&itemp)-1.0);

                idum = 1664525L*idum + 1013904223L;
                itemp = jflone | (jflmsk & idum);
                v(i) = 0.01*((*(float *)&itemp)-1.0);

                E0 += y(i)*y(i);
        }

        MatrixXd coupling(numOsc, numOsc);

        if (numOsc==2)  coupling << 0.0, kappa, kappa, 0.0;
```

```
else{
        //More in here later
}


//Definitions for eliminating the clearly stable solutions (e.g., decreasing ones)
double storage;
int nKept = 10;
double maxima[nKept]; //array of the past 3 maxima of the oscillators, det. by y.dot(y)
double past_two[2] = [E0, 0];   //array of the past step (j-1, stored in cell 0) and step j-2                    (in ce
double average;
for(j = 0; j<nKept-1; j++) maxima[j] = 0;
maxima[nKept-1] = y.dot(y);

//cout << "Definitions done!" << E0 << endl;

//Finds the acceleration at t=0, because loop finds acc at the end of the block, not the beginning
for(i=0; i<numOsc; i++)
        acc(i) = -(alpha + (i+1)*(i+1)*beta*cos(t))*y(i);

acc += coupling*y;

//Main loop
for(j=0; j<10000; j++){

        //Start of stable-eliminator code

        E = y.dot(y);                        //Calculating the new "energy"

        //cout << E << "\t" << j << endl;

        if(past_two[0] > storage && past_two[0] > past_two[1])
                for (i=0; i<nKept-1; i++)        maxima[i] = maxima[i+1];

        maxima[nKept-1] = storage;

        average = 0.0;

        for (i = 0; i<nKept-1; i++)      average += maxima[i];

        average /= nKept-1;

        if (maxima[nKept-1]<=average)    return false;

        past_two[1] = past_two[0];
        past_two[0] = storage;
        //If this maximum <= average, it's stable
        //End of stable-solution-eliminator code
        //---------------------------------------------------------------------------

        //cout << y(0) << "  " << y(1) << endl;
```

```
          y += dt*v + 0.5*dt*dt*acc;

          acc_last = acc;          //Makes space for calculating the new acceleration

          for(i=0; i<numOsc; i++)
                  acc(i) = -(alpha + (i+1)*(i+1)*beta*cos(t))*y(i);

          acc += coupling*y;

          v += 0.5*dt*(acc_last + acc);

          if(E>1000*E0)    return true;     //true indicates this parameter combination is unstable

          t += dt;
     }
     return false;                          //false says it is stable
}

%------------------------------------------------------------------------------------------------------------
THIRD PROGRAM: diagrammer.cpp
%------------------------------------------------------------------------------------------------------------

#include "floquetChecker.h"
#include <fstream>
#include <sstream>


int main(int argc, const char* argv[]){
/*Loops over regions of alpha-beta-coupling parameter space and outputs a 0 or a 1
 at each point corresponding to whether the equations at that point are stable or
unstable. The region to be swept is specified in an input file given in the command line*/
     ofstream outFile;
     ostringstream sfilename;

     ifstream inputfile(argv[1]);

     if (argc != 2){
     cout << "./stability, inputfile" << endl;
     return 1;
     }

     int line_number=0;
     int trial_number = 0;    //Number of blocks of input parameters to be swept
     double line_holder[100];

     while(inputfile >> line_holder[line_number]){
          if (line_holder[line_number] == 1010101)         trial_number++;

          line_number++;
     }

     for(int k = 0; k <= trial_number; k++){
```

```
int numOsc = (int)line_holder[9*k];
double alpha = line_holder[1 + 9*k];
int alphasteps = (int)line_holder[2 + 9*k];
double beta = line_holder[3 + 9*k];
int betasteps = (int)line_holder[4 + 9*k];
double kappa = line_holder[5 + 9*k];
int kappasteps = (int)line_holder[6 + 9*k];
double dx = line_holder[7 + 9*k];

bool stable;
int i, j;

double holder;          //Bookkeeping devices to allow parametric_stability to be called on any
double* designated1;    //combination of alpha, beta, and kappa
double* designated2;
int num_steps1;
int num_steps2;

sfilename.str("");

sfilename << numOsc;
sfilename << "Osc";

if (!alphasteps && betasteps && kappasteps){
        designated1 = &kappa;
        designated2 = &beta;
        num_steps1 = kappasteps;
        num_steps2 = betasteps;
        *designated2 += num_steps2*dx;
        holder = kappa;
        sfilename << "Alpha=" << alpha;
} else if (!kappasteps && alphasteps && betasteps){
        designated1 = &alpha;
        designated2 = &beta;
        num_steps1 = alphasteps;
        num_steps2 = betasteps;
        *designated2 += num_steps2*dx;
        holder = alpha;
        sfilename << "V=" << kappa;
} else if (!betasteps && kappasteps && alphasteps){
        designated1 = &alpha;
        designated2 = &kappa;
        num_steps1 = alphasteps;
        num_steps2 = kappasteps;
        *designated2 += num_steps2*dx;
        holder = alpha;
        sfilename << "Beta=" << beta;
} else {
        printf("Can only sweep two dimensions at a time; beta and either alpha or kappa\n");
        cout << alphasteps << " " << betasteps << " " << kappasteps << endl;
        cout << alpha << " " << beta << " " << kappa << " " << dx << endl;
        return 1;
```

```
            }

            sfilename << "dx=" << dx << ".txt";

            string filename = sfilename.str();

            cout << filename << endl;

            outFile.open(filename.c_str());

            for(i = 0; i<=num_steps2; i++){
                    *designated1 = holder;
                    for(j = 0; j<=num_steps1; j++){
                            stable = floquet(numOsc, alpha, beta, kappa, 28);
                            outFile << stable << " ";
                            *designated1 += dx;
                    }//end inner loop over parameter space
                    *designated2 -= dx;
                    outFile << endl;
            }//end outer loop over parameter space
    }//end loop over trials
    return 0;
}//end main

%-----------------------------------------------------------------------------------------------------------
FOURTH PROGRAM: collective_reader.cpp
%-----------------------------------------------------------------------------------------------------------

#include <iostream>
#include <fstream>
#include <cstdlib>
#include <fstream>
#include <sstream>

using namespace std;

int main(int argc, const char* argv[]){
        /*Reads two files containing stability diagrams OF THE SAME SIZE and IN MATRIX FORM
and compares them digit-by-digit, outputting a 0 to another file if corresponding digits in
the two files are identical, a -1 if the higher-coupling file has a stable solution at that
point while the less-coupled file has an unstable solution (collective stability, which has
been observed before), or a 1 if the more-coupled file has an unstable solution where the
less-coupled file has a stable solution (collective instability, the objective of our analysis)*/

        ofstream outFile;
        ostringstream sfilename;

        ifstream lowCoupling(argv[1]);
        ifstream highCoupling(argv[2]);

        if (argc != 4){
        cout << "You need four inputs on this line: the run command, a low-coupling file name,
a high−coupling file name, and the value of the large coupling, in that order." << endl;
        return 1;
        }

        sfilename << "XORredV=" << argv[3] << "and0.txt";

        string filename = sfilename.str();

        cout << filename << endl;

        outFile.open(filename.c_str());

        int lowVdigit, highVdigit;
        int row = 0, col = 0;

        string line1, line2;

        while(!lowCoupling.eof() && !highCoupling.eof()){

                getline(lowCoupling, line1);
                getline(highCoupling, line2);

                istringstream lowVline(line1);
                istringstream highVline(line2);

                col = 0;

                while (lowVline >> lowVdigit && highVline >> highVdigit){

                        /*if(lowVdigit != highVdigit){
                                cout << "A discrepancy! " << row << " " << col << " ";
                                cout << lowVdigit << " " << highVdigit << endl;
                        }*/

                        if (lowVdigit == highVdigit){
                                outFile << 0 << " ";      //nothing happened

                        } else if (lowVdigit != 0 && highVdigit == 0){
                                outFile << 2 << " ";      //collective stability; boring

                        } else if (lowVdigit == 0 && highVdigit != 0){
                                outFile << 1 << " ";      //Yes! Yes! Yes! Collective instability!!

                        }
                        col++;
                }
        outFile << endl;
        row++;
        }

}
```